TinyIBAK: Design and Prototype Implementation of An Identity-based Authenticated Key Agreement Scheme for Large Scale Sensor Networks

Lijun Yang[1], Chao Ding[1] and Meng Wu[2]
[1] College of Computer Science, Nanjing University of Posts and Telecommunications
Nanjing, Jiangsu 210003- China
[e-mail: yanglijun1119@gmail.com, dingchao1984@gmail.com]
[2] Key Lab of "Broadband Wireless Communication and Sensor Network Technology" of Ministry of Education
Nanjing, Jiangsu 210003- China
[e-mail: wum@njupt.edu.cn]
*Corresponding author: Meng Wu

Received July 15, 2013; revised October 1, 2013; accepted October 24, 2013; published November 29, 2013

Abstract

In this paper, we propose an authenticated key agreement scheme, TinyIBAK, based on the identity-based cryptography and bilinear paring, for large scale sensor networks. We prove the security of our proposal in the random oracle model. According to the formal security validation using AVISPA, the proposed scheme is strongly secure against the passive and active attacks, such as replay, man-in-the middle and node compromise attacks, etc. We implemented our proposal for TinyOS-2.1, analyzed the memory occupation, and evaluated the time and energy performance on the MICAz motes using the Avrora toolkits. Moreover, we deployed our proposal within the TOSSIM simulation framework, and investigated the effect of node density on the performance of our scheme. Experimental results indicate that our proposal consumes an acceptable amount of resources, and is feasible for infrequent key distribution and rekeying in large scale sensor networks. Compared with other ID-based key agreement approaches, TinyIBAK is much more efficient or comparable in performance but provides rekeying. Compared with the traditional key pre-distribution schemes, TinyIBAK achieves significant improvements in terms of security strength, key connectivity, scalability, communication and storage overhead, and enables efficient secure rekeying.

Keywords: Large scale sensor network, security, key agreement, identity-based cryptography, prototype implementation

A preliminary version of this paper appeared in ICITMI 2013, July 23-24, Zhuhai, China. This version includes a concrete analysis and supporting implementation results on MICAz motes and TOSSIM. This work is financially supported by National Basic Research Program of China (973 Program) under Grants 2011CB302903, the National Natural Science Foundation of China (Grant No. 61100213), the Key Program of Natural Science for Universities of Jiangsu Province (Grant No.10KJA510035) , the Science and Technology Program of Nanjing (201103003) and the Postgraduate Innovation Project Foundation of Jiangsu Province (Grant No. CXLX11_0411).

http://dx.doi.org/10.3837/tiis.2013.11.013

1. Introduction

Wireless sensor networks (WSNs) are comprised of a large number of limited sensor nodes which are spatially distributed across the field of interest. WSNs have been appiled in many areas including military, healthcare, environment, and manufacturing, etc. WSNs are vulnerable to various malicious attacks such as eavesdropping, message replay and node compromise, due to its nature of open wireless medium, restricted node resources, and unattended operation manners. Thus security is very significant for wireless sensor networks deployed in hostile environments.

In order to prevent malicious nodes from violating the security of sensor networks, cryptographic technologies should be used to achieve the data confidentiality, integrity, and authentication between communicating nodes. Symmetric key cryptography (SKC) is a popular choice for securing sensor networks due to its efficiency. There have been some implementations of SKC-based mechanisms designed for sensor networks, but the security of the key management mechanism in these implementations is very poor. Take TinySec [1], a link layer security mechanism for the MICA2 implemented in TinyOS, as an example. TinySec relies on a single global key all over the network, and doesn't support securely rekeying itself. Meanwhile, TinySec's 4-byte initialization vector (IV) only allows secure communication for 2^{16} times, which may be insufficient for WSNs whose lifespans demand longer lasting security. For instance, in the Great Duck Island Project [2], each node sends one message every 70 seconds, thus 2^{16} packets' secure transmission can only last for approximately 45 days. On the other hand, the reliance on a single global key is vulnerable to the compromise of a single node. Hence, more effective key distribution and rekeying schemes are needed to enhance the security strength of the SKC-based mechanisms.

Key management becomes a major challenge in resource constrained sensor networks. Existing key management schemes for sensor networks can be classified into the key pre-distribution approaches and the public key cryptography (PKC) based approaches. In the key pre-distribution approaches, a subset of keys from the same key pool is preloaded to sensor nodes. Any two nodes search for the common keys to establish the pairwise key. However, this kind of approaches has a high memory and communication costs, and is vulnerable to impersonating and node compromise attack. On the other hand, the traditional PKC based approaches are energy consuming and usually need public key infrastructure (PKI), which is not affordable for sensor networks.

The main idea of identity-based cryptography (IBC) is that any information that uniquely identifies entities (e.g. name or email address) can be used as a public key, thus the PKI is unnecessary. In this work, we argue that IBC is ideal for WSNs. In the context of WSNs, the network deployer can be considered as a trusted entity, thus can play the role of Private Key Generator (PKG) perfectly which takes charge of generating and escrowing nodes' private keys. On the other hand, the private keys can be preloaded into nodes offline in a secure way.

Inspired by the synergy between IBC and WSNs, we proposed an identity-based authenticated key agreement scheme named TinyIBAK based on bilinear paring, for large scale sensor networks. The major contributions of this work are as follows:

1) We performed the formal security proof for the proposed scheme, and heuristically analyzed the security properties which the formal security proof does not cover, and the

resilience against some attacks on routing protocols. Furthermore, we presented the formal security validation using AVISPA tool [3].

2) We presented a prototype implemented of our proposal on the TinyOS2.1 platform based on RELIC cryptographic toolkit [4] to evaluate the feasibility and performance of our proposal in the practical network deployment.

3) We compared our scheme with other ID-based approaches and traditional key pre-distribution schemes in terms of computation, communication and memory overhead, as well as security strength, key connectivity and scalability.

The rest of this paper is organized as follows. In section 2, we discuss the related works. In section 3, we introduce related basic mathematical concepts and the security model. In section 4, we describe the proposed TinyIBAK scheme in details. In section 5, we present the detailed security analysis of TinyIBAK. The Prototype implementation issues and the performance evaluation are discussed in section 6 and section 7 respectively. In Section 8, we compare our proposal with some well-known key agreement schemes for sensor networks. Finally, we conclude our work in section 9.

2. Related Works

Recently, a number of key management schemes have been proposed for large scale sensor networks [5-12], these schemes may be classified into the key pre-distribution and public key cryptographic based approaches.

Key pre-distribution is a promising scheme for key management in sensor networks. In this kind of schemes, keying materials are preloaded into sensor nodes before deployment. Eschenauer and Gligor [5] propose the first random key pre-distribution scheme for pairwise key establishment. The main idea behind the scheme is that each node randomly picks a set of keys from a key pool before deployment so that any two sensor nodes have a certain probability to share at least one common key. Subsequent extensions were proposed in [6, 7], in which two nodes can establish a secure link if they share q keys. Liu and Ning [8] provide further extension to enhance scalability and resilience to attacks by using key polynomials. This set of schemes is static since they do not support rekeying and new node addition after deployment.

Dynamic key pre-distribution approaches are more flexible in rekeying and deployment than the static ones. Moharrum et al. [9] presents a comparative study between the static and dynamic key pre-distribution schemes. Eltoweissy et al. [10] proposed a dynamic key management scheme called the Localized Combinatorial Keying (LOCK) which is based on EBS-based dynamic key management scheme. LOCK significantly enhances network resilience to collusion at the low cost of using key polynomials as compared to that for the Liu et al.'s polynomial-pool-based scheme [8]. Chorzempa et al. [11] proposed another dynamic key management scheme called the Survivable and Efficient Clustered Keying (SECK) for hierarchical wireless sensor networks, which is resilient against node capture as well as key capture to a certain extent while maintaining low levels of communication and computation overheads. Das [12] propose a dynamic random key establishment scheme, which is called multi-phase deployment key establishment scheme (MPDKE). The basic idea is to divide the deployment into multiple phases, and ask the already deployed nodes refresh their own keys in key rings before the next deployment phase occurs. This scheme provides a good tradeoff between network connectivity, overheads, and network resilience against node capture attack.

However, the common shortcomings of key pre-distribution schemes cannot be neglected. To ensure the key connectivity, most pre-distribution schemes require a large number of keys to be preloaded. With the increase of the node number, the key storage occupation and communication costs increase significantly, which makes the pre-distribution schemes inapplicable to large scale sensor networks.

To address the problems aforementioned, researchers have been investigating the possibility of making use of public key cryptography (PKC). Several works [13-15] have demonstrated the feasibility of PKC on the resource constrained sensor nodes. Malan et al. [13] implement elliptic curve Diffie-Hellman (ECDH) key exchange on the MICA2 motes to derive a mutual key between pairs of nodes. But it is not authenticated, and hence is subject to the man-in-the-middle attack. Watro et al. [14] propose a scheme named TinyPK based on RSA cryptosystem and Diffie-Hellman algorithms, allowing authentication and key agreement between a sensor network and a third party as well as between two sensor networks. However, TinyPK requires a public key infrastructure (PKI) for public key authentication, which is not affordable in WSNs. Yang et al. [16] propose an approach based on identity-based encryption and Diffie-Hellman Algorithms, providing authenticated key agreement between pairs of sensor nodes. But its computation and memory overhead are too high to apply practically. Oliveira et al. [15] implement SOK protocol for sensor networks, in which no interaction is required. However, the shared secret derived through SOK is static, which means that SOK doesn't support rekeying.

3. Preliminaries

3.1 Basic Concepts

Let E/F_q be an elliptic curve over a finite field F_q, and $E(\mathrm{F}_q)$ be the group of points on this curve, where q is a large prime. Let point $P \in E(\mathrm{F}_q)$ be a generator of order n, and group $G_1 = \langle P \rangle$, where n is a large prime.

Bilinear Pairing. Let G_1 be an additively-written group of order n with identity o, and let G_T be a multiplicatively-written group of order n with identity 1. A *bilinear pairing* is a computable, non-degenerate function $\hat{e}: G_1 \times G_1 \to G_T$. The most important property of pairings in cryptographic constructions is the *bilinearity*, namely, for any $U, V \in G_1$, and any $a, b \in \mathbb{Z}^*$, we have

$$\hat{e}([a]U,[b]V) = \hat{e}(U,[b]V)^a = \hat{e}([a]U,V)^b = \hat{e}(U,V)^{ab}.$$

Bilinear Diffie-Hellman Problem (BDHP). Given P, $[a]P$, $[b]P$, and $[c]P$ for some $a, b, c \in \mathbb{Z}^*$, compute $\hat{e}(P,P)^{abc}$.

BDH assumption. There exists no probabilistic polynomial time (PPT) algorithm which can solve the BDH problem in G_1, G_T with non-negligible probabiliity.

Embedding Degree. A subgroup G of $E(\mathrm{F}_q)$ is said to have an *embedding degree* k with respect to n if k is the smallest integer such that $n \mid q^k - 1$.

3.2 Security Model

In this work, we shall use a modified Blake-Wilson key exchange security model [17] to analyze the protocol security. We adapt the model to the identity-based setting. In the model, a protocol is modeled as a set of oracles $\Pi_{i,j}^{n}$, which means the *n*-th protocol running between participants *i* and *j*. Oracles keep transcripts which record messages they have sent or received as a result of queries they have answered.

Each participant has a pair of ID-based long-term asymmetric keys, where the public key is created using the participant's ID and the private key is computed and issued by a PKG. We assume there is a setup algorithm *Setup* which produces a description of the groups G_1, G_T and the bilinear map $\hat{e}$, assigns random tapes and oracles as necessary, and distributes a long-term master key to the PKG.

The model also includes an adversary *E* who is modeled by a PPT Turing Machine and has access to all the participants' oracles in the game. *E* can reply, modify, delay, interleave or delete messages. *E* is called the benign adversary if she simply passes messages to and fro between participants. We note that all communications go through the adversary. Participant oracles only respond to queries by the adversary and do not communicate directly amongst themselves. It is assumed that *E* is allowed to make the following types of queries of existing oracles as defined in [17]:

- *Send:* this allows *E* to send a message of her choice to an oracle, say $\Pi_{i,j}^{n}$, in which case participant *i* assumes the message has been sent by participant *j*. *E* may also make a special Send query to an oracle $\Pi_{i,j}^{n}$ which instructs *i* to initiate a protocol run with *j*. An oracle is an *initiator oracle* if the first message it has received is a λ. If an oracle has not received a message λ as its first message, then it is a *responder oracle*.
- *Reveal:* this allows *E* to ask a particular oracle to reveal the session key (if any) it currently holds to *E*.
- *Corrupt:* this allows *E* to ask a particular oracle to reveal its long-term private key.

The security of protocol is defined by the game between *E* and a challenger. In the game, *E* has access to a set of oracles $\Pi_{i,j}^{n}$, and is allowed to make a polynomial number of queries including *Send, Reveal, and Corrupt* to any oracle in any order. Then at some point, *E* has to make a *Test* query to a fresh oracle $\Pi_{i,j}^{n}$. Then the challenger chooses at random $b \in \{0,1\}$ and responds to *E* with the session key of $\Pi_{i,j}^{n}$ if $b = 0$, and otherwise a random sample from $\{0,1\}^{k}$. After this point, *E* can continue querying oracles except that *E* cannot reveal the test oracle $\Pi_{i,j}^{n}$ or its partner oracle $\Pi_{j,i}^{n'}$ (if exists), and that *E* cannot corrupt its partner j. Finally, *E* output its guess b' for *b*. *E*'s advantage is defined as the probability that *E* can distinguish the session key held by the queried oracle from a random string, and is denoted as: $Adv^{E}(k) = \left|\Pr[b' = b] - 1/2\right|$.

Definition 1. An oracle $\Pi_{i,j}^{n}$ is fresh if: (1) $\Pi_{i,j}^{n}$ has accepted holding a session key; (2) $\Pi_{i,j}^{n}$ is unopened; (3) the participant $j \neq i$ is not corrupted; (4) there is no opened oracle $\Pi_{j,i}^{n'}$ which has a matching conversation to $\Pi_{i,j}^{n}$.

The above fresh oracle definition is particularly defined to cover the key-compromise

impersonation resilience property since it implies that the user i could have been issued a Corrupt query.

Definition 2. A MAC is a secure MAC if for every PPT adversary E of the MAC, the function $\varepsilon(k)$ defined by $\varepsilon(k)=\Pr\left[k' \leftarrow \{0,1\}^k; (m,a) \leftarrow E : (m,a)=MAC_{k'}(m)\right]$ is negligible.

Definition 3. A protocol is a secure authenticated key agreement protocol with key confirmation (AKC) if: (1) in the presence of the benign adversary $\Pi^n_{i,j}$ and $\Pi^{n'}_{j,i}$, both oracles always accept holding the same session key, and this key is distributed uniformly at random on $\{0,1\}^k$; and if for every adversary E: (2) if uncorrupted oracles $\Pi^n_{i,j}$ and $\Pi^{n'}_{j,i}$ having matching conversations then both oracles accept and hold the same session key; (3) the probability of $No-Matching^E(k)$ is negligible; (4) $Adv^E(k)$ is negligible.

The above security model and definitions imply the implicit key authentication(IKA), known key security (KKS), key compromise impersonation resilience (KCIR) and unknown key share resilience (UKSR), but do not cover the forward secrecy property [17]. We discuss this property of our protocol heuristically later in the paper.

4. Proposed TinyIBAK Scheme

4.1 TinyIBAK

In this section, we describe our proposal TinyIBAK. TinyIBAK is involved with three kinds of entities, i.e. any two nodes intent to agree a secret key, represented as node A and node B, and a management system owned by the network deployer (plays the role of PKG). TinyIBAK is comprised of three steps, including Setup, Extract, and Key Agreement. Before deployment, the management system performs Setup and Extract steps offline in a secure environment, and preloads corresponding information including private key, public key, and public system parameters into each node. After deployment, each node performs neighbor discovery, broadcasts its ID within its neighborhood, and performs the Key Agreement step of TinyIBAK as shown in **Fig. 1**. We assume that an adversary cannot compromise a node in the bootstrapping phase.

Setup. takes a security parameter k as input, and returns the master key and system parameters. For a given security parameter k, the management system does the following,

(1) Chooses a pairing-friendly supersingular elliptic curve $E/\mathbb{F}_q$, and runs the IBC parameter generator to generate the cyclic groups $(G_1,+)$ and $(G_T,\cdot)$ of the prime order n, an arbitrary generator P of G_1, and a bilinear pairing $\hat{e}: G_1 \times G_1 \rightarrow G_T$;

(2) Chooses the master key $s \in_R Z_n^*$, and computes the system public key $P_{pub}=sP$;

(3) Chooses cryptographic secure hash functions $H_1: \{0,1\}^* \rightarrow G_1$ and $H_2: G_T \rightarrow \{0,1\}^k$;

(4) Publishes the tuple $\Pi=(q, G_1, G_T, \hat{e}, P, P_{pub}, H_1, H_2)$ as system parameters and keeps the master key s secret.

Extract. takes the system parameters, master key, and a node identifier as input and returns the node's ID-based long-term key. For each node *i* with identifier ID_i, the management system

(1) Computes $Q_i = H_1(ID_i)$;

(2) Computes the private key $d_i = sQ_i$;

(3) Preloads $\langle ID_i, d_i, Q_i, \Pi \rangle$ into every node of the network.

Key Agreement: For two nodes A and B to establish an authenticated session key, they should do as follows:

(1) To start a key agreement session with the intended responder B, the initiator A chooses a random $a \in_R Z_n^*$, computes the key token $W_A = aQ_A$, and sends ID_A and W_A to B, $\mathrm{A} \to \mathrm{B}: ID_A \| W_A$.

(2) On receiving the initiation message from node A, node B does the followings:

a) Chooses a random $b \in_R Z_n^*$, and computes the key token $W_B = bQ_B$;

b) Computes $K_{BA} = \hat{e}(W_A + bQ_A, d_B)$, and derives k_1, k_2 using the key derive functions (KDFs), $(k_1, k_2) \leftarrow KDF(K_{BA})$;

c) Computes the message authentication code (MAC) with the secret k_1, $t_B = MAC_{k_1}(ID_B, ID_A, W_B, W_A)$;

d) Sends ID_B, W_B and t_B to node A, $\mathrm{B} \to \mathrm{A}: ID_B \| W_B \| t_B$.

(3) On receiving the responding message from node B, node A :

a) Computes $K_{AB} = \hat{e}(d_A, W_B + aQ_B)$, and derives k_1, k_2 using the KDFs, $(k_1, k_2) \leftarrow KDF(K_{AB})$;

b) Checks the received MAC tag $t_B = MAC_{k_1}(ID_B, ID_A, W_B, W_A)$;

c) Computes the MAC $t_A = MAC_{k_1}(ID_A, ID_B, W_A, W_B)$, and sends it to node B, $A \to B: t_A$.

(4) On receiving the confirmation message from node A, node B checks the received $t_A = MAC_{k_1}(ID_A, ID_B, W_A, W_B)$.

(5) The shared session key between node A and node B is k_2, i.e. $k_{AB} = k_2$.

It is easy to see that thanks to the bilinear property,

$$K_{AB} = \hat{e}(d_A, W_B + aQ_B) = \hat{e}(Q_A, Q_B)^{s(a+b)}$$

$$K_{BA} = \hat{e}(W_A + bQ_A, d_B) = \hat{e}(Q_A, Q_B)^{s(a+b)}$$

thus $K_{AB} = K_{BA}$.

Therefore, the shared secret between node A and B agrees.

The shared secret is dynamic, because it uses the number *a* and *b*, which are randomly chosen for every round of key agreement. In TinyIBAK, We use two different secure hash functions $H_2: G_T \to \{0,1\}^k$ and $H_3: G_T \to \{0,1\}^k$ as key derivation functions to generate two

symmetric keys, in which k_1 is used to compute the message authentication code (MAC) and k_2 is the shared session key used for the secrecy communications between node A and B.

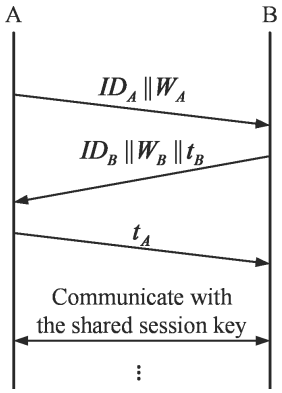


Fig.1. The Key Agreement of TinyIBAK

4.2 Node Addition, Rekeying and Key Revocation

Whenever the new nodes enter the existing network, or failure nodes need replacement, TinyIBAK does not require any new keying material for existing nodes. The management system runs the Setup and Extract steps of TinyIBAK, and preloads $\langle ID_i, d_i, Q_i, \Pi \rangle$ into the new nodes. After deployment, the new nodes run the Key Agreement step to establish shared secret keys with its neighbor nodes.

Whenever rekeying is needed, node A sends rekeying request $Enc_{k_{AB}}(rekeying, nonce)$ to its neighbor node B, and then they both run the Key Agreement step to agree new shared secret key.

Node addition/replacement and rekeying only incur interactions within the neighborhood, and do not affect other nodes of the WSN.

When a sensor node is compromised by an adversary, all the keys shared with this node need to be revoked. Assume that the node compromise is detected by some scheme and is reported to the base station (BS). After detection, BS disseminates a *Revocation* message containing a list of malicious nodes IDs, appending the signature calculated using its private key. When sensor nodes receive the *Revocation* message, they can check the integrity of the message by verifying the digital signature. If the *Revocation* is valid, sensor nodes delete the shared session keys with the compromised nodes, and blacklist the corresponding IDs.

5. Security Analysis

In this section, we formally analyze the security of our protocol, and heuristically discuss the additional security properties that the security proof does not cover. Furthermore, we validite the protocol using the AVISPA tool.

5.1 Security Proof

The inherent security of our protocol relies on the difficulty of BDH Problem.With the security model descripted in Section 3.2, we now state:

Theorem 1. Protocol TinyIBAK is a secure AKC protocol, assuming that the adversary does

not make any *Reveal* queries, the BDH problem (for the pair of groups G_1 and G_T) is hard, the MAC is secure and provided that H_1, H_2 and H_3 are random oracles.

Proof. Conditions 1 and 2 of Definition 2 follow from the assumption that the two oracles follow the protocol and in both cases have matching conversations. Therefore both oracles accept (since they both receive correctly formatted messages from the other oracle) holding the same key *SK* (since $K_{AB} = K_{BA}$ by the bilinearity of the pairing and the matching conversation). Since H_2 is a random oracle, *SK* is distributed uniformly at random on $\{0,1\}^k$. Condition 3 of Definition 2 follows from the Theorem 8 in [17], and the detailed proof is omitted due to the page limitation.

Condition 4. By the way of contradiction, we suppose that there exist a PPT adversary *E* who can win the above game with non-negligible advantage $\varepsilon(k)$. We assume that the number of participants is q_1, and the maximum number of sessions for each participant is q_0.

We construct from *E* an algorithm *F* which solves the BDH problem with non-negligible probability. Let $\langle P, xP, yP, zP \rangle$ be the BDH instance given to *F*, whose task is to compute $e(P,P)^{xyz} \in G_T$. *F* chooses distinct random values *I* and *J* from $\{1,\cdots$, and a value $l \in \{1,\cdots$. *F* simulates the *Setup* algorithm and sets the PKG's master public key to be *xP*. *F* will also simulate all oracles required during the game. *F* then starts *E*, and answers all *E*'s queries as follows.

$H_1(ID_i)$: Algorithm *F* maintains an initially empty list H_1-list with the form $\langle ID_i, Q_i \rangle$. F responds to the query in the following way.

- If ID_i is already on the H_1-list in the tuple $\langle ID_i, Q_i \rangle$, then F outputs Q_i.
- Otherwise, if ID_i is the *J*-th unique identifier query, then F outputs $Q = yP$ and adds the tuple $\langle ID_i, Q_i \rangle$ to the H_1-list.
- Otherwise, F selects a random $r_i \in Z_q^*$, outputs $Q_i = r_i P$ and adds the tuple $\langle ID_i, Q_i \rangle$ to the H_1-list.

F simulates the H_2 oracle in the same way, keeping an H_2-list and answers all H_2 oracle queries at random.

Corrupt query: *F* answers *Corrupt* queries as specified by a normal oracle, i.e., revealing the long-term private key of the related participant, except that if *E* asks *J* a *Corrupt* query, *F* gives up.

Send query: *F* answers all *Send* queries as specified for a normal oracle, i.e., for the first *Send* query to an oracle, *F* takes a random value in Z_q^* to form its contribution, except that if *E* asks $\Pi_{I,J}^n$ for any *n* its first *Send* query, *F* chooses a random $s_n \in Z_q^*$ and answers $s_n r_I zP = s_n zQ_I$. If the oracle is an initiator oracle, the first *Send* query will be a special initiating query, and if it is a responder oracle, the first *Send* query will contain an ephemeral input to $\Pi_{I,J}^n$.

Reveal query: *F* does not need to answer *Reveal* queries since we do not allow the adversary to make *Reveal* queries.

Test query: At some point in the simulation, *E* will ask a *Test* query to a fresh oracle. If *E*

does not choose one of the oracles $\Pi_{I,J}^n$ for some n to ask the *Test* query, then F aborts. However, if E does pick $\Pi_{I,J}^n$ for the *Test* query, then $\Pi_{I,J}^n$ must be fresh, otherwise F aborts. Assuming that $\Pi_{I,J}^n$ obtained some value jQ_J as its input prior to accepting, the oracle should hold a session key of the form $H_2\left(\hat{e}\left(xQ_I, s_n zQ_J + jQ_J\right)\right)$. However, F cannot compute this key. Instead, F simply outputs a random value.

If F does not abort, then E must have computed $K_{I,J}^n = \hat{e}\left(xQ_I, s_n zQ_J + jQ_J\right)$ with the non-negligible probability $\varepsilon(k)$, and queried the H_2 oracle on the corresponding value. At the end of E's attack, if E has not made at least l queries to the H_2 oracle, then F aborts. Otherwise F picks E's l-th query (on some value h) to the H_2 oracle, which F guesses to be the value $\hat{e}\left(xQ_I, s_n zQ_J + jQ_J\right) = \hat{e}\left(xP, yP\right)^{r_I(s_n z+j)} = \delta\hat{e}\left(P,P\right)^{xyz\gamma}$, with $\delta = \hat{e}\left(r_I xP, jQ_J\right)$ and $\gamma = r_I s_n$. F outputs $\left(h/\delta\right)^{1/\gamma}$ as the response to the BDH challenge.

So the probability that F does not abort during the game and produces the correct output is at least $\frac{\varepsilon(k)}{q_1^2 q_0}$, which is non-negligible. This contradicts the BDH assumption. Hence, there exists no PPT adversary against our protocol that has non-negligible advantage. We conclude that our protocol is a secure AKC protocol. □

If a protocol is secure regarding the security model described in section 3.2, it achieves implicit mutual key authentication and the following general security properties: known key security (KKS), key compromise impersonation resilience (KCIR) and unknown key share resilience (UKSR) [17]. Since we assume that the adversary cannot make *Reveal* queries, our proof of security does not guarantee the KKS property.

5.2 Other Properties of the Protocol

Since the security model which we used to prove Theorem 1 doesn't cover some known active attacks, we now heuristically discuss some of the security properties related to these attacks.

Known key security (KKS). Each run of the TinyIBAK between the node A and B produces a different session key that depends on the random selection of *a* and *b*. The adverary, who learned some other session keys, can predict neighter new or subsequent session keys, nor any earlier session keys either under the BDH assumption. Therefore, TinyIBAK achieves known key security.

Partial forward secrecy(P-FS). Compromise of long-term private keys d_A or d_B does not seem to lead to the compromise of past communications. But compromising both d_A and d_B does lead to the compromise of past communications, because the shared secret K can be computed as $K = \hat{e}\left(d_A, W_B\right) \cdot \hat{e}\left(W_A, d_B\right)$. So TinyIBAK only achieves partial forward secrecy, but not offer perfect forward secrecy.

Key confirmation. If the tag t_A and t_B are validated, both parties can be sure that the other party does calculate the shared secret $K_{AB} = K_{BA} = \hat{e}\left(Q_A, Q_B\right)^{s(a+b)}$, the other party knows the identity of the entity that it is communicating with, and the communication process of both sides has not been tampered with. Therefore, TinyIBAK achieves mutual authentication of the

communication, and confirmation of the shared key.

Explicit key authentication. Our protocol achieves both implicit key authentication and key confirmation, so we can say it achieves explicit key authentication.

Resilience against node capture attack. In TinyIBAK, each sensor is preloaded with one unique private key. After the session key established, each pair of nodes has a different shared key. Thus, compromising some sensors does not affect the security of communication among other pairs of nodes. Therefore, TinyIBAK achieves strong resilience against the node compromise attacks.

Resilience against replay attack. In the bootstrapping phase, an attack can replay the old pairwise key establishment messages. This attack is prevented as each sensor has a secret random number which allows it to generate the key. So, even if an adversary replays an old message trying to establish a pairwise key with a valid node, she will not be able to do this as she must solve a DLP to get the random number.

Resilience against node fabrication attack. Each node has a pair of ID-based long-term asymmetric keys, where the public key is created using the node's ID and the privacy key is computed and issued by a trusted PKG. Therefore, even an adversary compromise a node, she still couldn't fabricate another legitimate node.

Resilience against man-in-the-middle attack. Assume that the adversary E tries to launch the man-in-the-middle attack, as the follow steps.

(i) E intercepts the message A→B: $ID_A \| \mathrm{W}_A$, computes $W_A^* = a'Q_A$, and sends $ID_A \| \mathrm{W}_A^*$ to node B;

(ii) Node B computes the shared key $K_B = e\left(W_A^* + bQ_A, s_B\right) = e\left(Q_A, Q_B\right)^{(a'+b)s}$, and the authentication tag $t_B = MAC_{k_{B1}}\left(ID_B, ID_A, W_B, W_A^*\right)$;

(iii) E intercepts the message B→A： $ID_B \| W_B \| t_B$, computes $W_B^* = b'Q_B$, and sends $ID_B \| W_B^* \| t_B$ to node A;

(iv) Node A computes the shared key $K_A = e\left(s_A, W_B^* + aQ_B\right) = e\left(Q_A, Q_B\right)^{(b'+a)s}$, and the tag $t_B^{'} = MAC_{k_{A1}}\left(ID_B, ID_A, W_B^*, W_A\right)$, then node A finds $t_B^{'} \neq t_B$. The key agreement fails.

E does not know the private keys d_A and d_B, so she can calculate neither K_A nor K_B. Therefore, our protocol is resistant to the man-in-the-middle attack. We validate it using AVISPA tool in section 5.3

Resilience against attacks on routing protocols. Many attacks on routing protocols are based on the node capture attack. Our proposed protocol can efficiently defend against several of them such that the sybil attack, the sinkhole attack and the wormhole attack. More details about these attacks can be found in [18].

After deployment, the proposed protocol goes through the neighborhood discovery phases. At the end of the phase, each node would keep a list of its immediate neighbors. Given the assumptions mentioned before: an adversary cannot compromise a node in the bootstrapping phase. Since after key agreement each node knows the set of valid nodes with which it may communicate, an adversary cannot convince another node, which is not really in its neighborhood, that it is a near one. Therefore, a sinkhole attack or a wormhole attack can be detected. Moreover, the sybil attack is prevented as valid IDs and ID-based private keys are

assigned by the trusted PKG. Hence the adversary cannot present as multiple IDs.

5.3 Formal Security Validation Using AVISPA Tool

In this paper, we choose AVISPA [3], an automated formal security validation tool, to analyze the security properties of the proposed TinyIBAK protocol. The reason for the choice of AVISPA lies in that (1) it is able to specify the security protocols in a very fine-grained manner, so that we can model the security properties like secrecy of keys, authentication, resilience against the man-in-the-middle and replay attacks, etc., (2) it integrates four different back-end model checkers (i.e. the OFMC [19], CL-AtSe [20], SATMC [21], and TA4SP [22]) that implement a variety of state-of-the-art automatic analysis techniques, so that we can choose the appropriate back-ends according to the scenario, and (3) it is widely-accepted by academic researchers to analyze the potential threats on the security protocols.

Specifying the Protocol. In order to validate the security properties of the proposed scheme with the AVISPA tool, we have implemented the key agreement phase of this scheme in the HLPSL language [23]. In the implementation, we assume that the public keys can be derived from the node ID via the hash function $H_1(\cdot)$. We model the operations of pairing, point addition and point multiplication as hash functions, denoted as $Ebilinear(\cdot)$, $Pred(\cdot)$, and $Union(\cdot)$, respectively, since the inverse of these operations is hard to compute for intruders. In addition, the proposed protocol is simulated under the Delov-Yao intruder model [24]. In this model, we assume that the intruder fully controls the whole network, namely the intruder can forward, modify, replay, suppress and synthesize all the messages sent by participants and send them anywhere. Besides, an intruder can play the legitimate participants, and gain knowledge from compromised participants, but he cannot violate the cryptography.

Our implementation includes two basic roles: alice and bob, which represent the initiator A and responder B in the proposed protocol, respectively. **Fig. 2** shows the role specification of the initiator A. When the initiator A firstly receives the *start* signal, it changes its initial state 0 to 2, and sends the message $\langle A, W_a' \rangle$ to the responder B with the $Snd(\cdot)$ operation, where W_a' is the key token derived from the point multiplication of the random number *RndA* and the public key Q_a. After receiving the message $\langle B, W_b', T_b' \rangle$ from B, A changes its state 2 to 4, computes the shared secret K_{ab}' using *RndA*, W_b' and Q_b', and derives the authentication tag T_a' using the key deriving function $KDF(\cdot)$ and the message authentication function $MAC(\cdot)$. Finally, A sends the message $\langle T_a' \rangle$ to *B*.

Fig. 3 shows the specification for the role of responder B. After receiving the message $\langle A, W_a' \rangle$ from A, B changes its initial state 1 to 3, computes the key token W_b', and sends it to A using the $Snd(\cdot)$ operation. Then the shared secret K_{ab}' and the authentication tag T_b' are obtained with the same method as that of the initiator A, and T_b' is sent to A. After receiving the authentication message $\langle T_a' \rangle$, B changes its state 3 to 5, and finally verifies the source of the message exchanged in this session.

We have specified the top-level role "environment" to describe the execution context of our scheme. A typical top-level role contains the global constants, the intruder knowledge, and a composition of one or more sessions. The intruder in our protocol has the knowledge of all the cryptographic operations, system parameters, and the public key of each node in the network. The intruder plays some roles as legitimate users and participates in the execution of protocol.

Due to the page limitation, the top-level role specification is omitted from this paper. A similar example is presented in [25]. The following four secrecy and two authentication goals are verified in TinyIBAK:

- secrecy_of ra: It represents that *RndA* is kept secret to A only.
- secrecy_of rb: It represents that *RndB* is kept secret to B only.
- secrecy_of kab: It represents that K_{ab} is kept secret to A and B only.
- secrecy_of k1: It represents that K_1 is kept secret to A and B only.
- authentication_on alice_bob_tb: Since K_1 is only known to A and B, if A verifies the message authentication code T_b, then A authenticates B.
- authentication_on bob_alice_ta: Since K_1 is only known to A and B, if B verifies the message authentication code T_a, then B authenticates A.

```
role alice ( A, B: agent,
             IDa, IDb: text,
             Qa: public_key,
             H1, Pred, Union, Ebilinear: hash_func,
             KDF, MAC: hash_func,
             SND, RCV: channel(dy))

played_by A  def=

  local  State: nat,
         RndA, Wa, Wb, Tb, Ta, Kab, K1: text,
         Qb: public_key

  const  bob_alice_ta, alice_bob_tb, kab, ra: protocol_id

  init State := 0

  transition

     0. State = 0 ∧ RCV(start)  =|>
        State' := 2 ∧ RndA' := new()
                    ∧ Wa' := Pred(RndA', Qa)
                    ∧ SND(A.Wa')
                    ∧ secret(RndA', ra, A)

     2. State = 2 ∧ RCV(B.Wb'.Tb')  =|>
        State' := 4 ∧ Qb' := H1(IDb)
                    ∧ Kab' := Ebilinear(inv(Qa),
                      Union(Wb', Pred(RndA, Qb')))
                    ∧ secret(Kab', kab, {A, B})
                    ∧ K1' := KDF(Kab')
                    ∧ secret(K1', k1, {A, B})
                    ∧ Ta' := MAC(A, B, Wa, Wb', K1')
                    ∧ SND(Ta')
                    ∧ witness(A, B, bob_alice_ta, Ta')
                    ∧ request(A, B, alice_bob_tb, Tb')
endrole
```

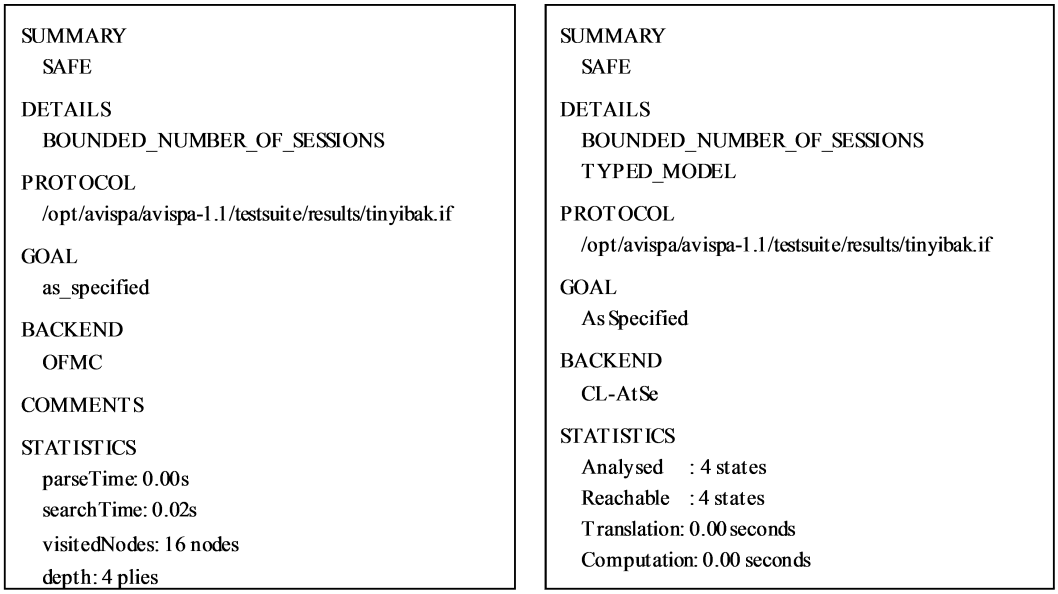
Fig. 2. Role specification for the initiator A

```
role bob (A, B: agent,
          IDa, IDb: text,
          Qb: public_key,
          H1, Pred, Union, Ebilinear: hash_func,
          KDF, MAC: hash_func,
          SND, RCV: channel(dy))

played_by B def=
  local  State: nat,
         RndB, Wb, Wa, Kab, K1, Ta, Tb: text,
         Qa: public_key

  const  bob_alice_ta, alice_bob_tb, kab, rb: protocol_id

  init State := 1

  transition

     1. State = 1 ∧ RCV(A.Wa') =|>
        State' := 3 ∧ RndB' := new()
                    ∧ secret(RndB', rb, B)
                    ∧ Wb' := Pred(RndB', Qb)
                    ∧ Qa' := H1(IDa)
                    ∧ Kab' := Ebilinear(Union(Wa', Pred(b,
                              Qa')), inv(Qb))
                    ∧ secret(Kab', kab, {A, B})
                    ∧ K1' := KDF(Kab')
                    ∧ secret(K1', k1, {A, B})
                    ∧ Tb' := MAC(B, A, Wb', Wa', K1')
                    ∧ SND(B.Wb'.Tb')
                    ∧ witness(B, A, alice_bob_tb, Tb')

  3. State = 3 ∧ RCV(Ta') =|>
       State' := 5 ∧ request(B, A, bob_alice_ta, Ta')
endrole
```

Fig. 3. Role specification for responder B

Analysis of the Results. We choose the OFMC [19] and CL-AtSe [20] back-ends and a

bounded number of sessions model checking to analyze the security properties of our scheme. According to the summary result for the formal security verification analysis depicted in **Fig. 4**, the proposed TinyIBAK is secure against the passive attacks and active attacks such as man-in-the-middle attack and replay attack. Both the OFMC and CL-AtSe report that the protocol is SAFE.



(i) (ii)

Fig. 4. (i) Summary of analysis results from OFMC back-end and (ii) summary of CL-AtSe back-end

6. The Prototype Implementation of TinyIBAK

In order to evaluate the feasibility and performance of our proposal in the practical network deployments, we have implemented a prototype of TinyIBAK on the TinyOS 2.1 platform. Current implementation involves the offline operations performed on the management system (a PC with TinyOS installed) and the online operations performed on the sensor nodes**.** The management system performs the Setup and Extract steps, and preloads the system parameters and node's private key into each sensor node. The sensor nodes perform the Key Agreement step.

All the programs running on the sensor nodes have been developed in NesC, a C-like language for developing applications in TinyOS. The implementation of TinyIBAK is based on the RELIC cryptographic toolkit [4], which provides the necessary tools to perform operations on elliptic curves. RELIC is a publicly available and open source library which is specifically designed for resource-constrained devices.

As recommended by NIST [26], we adopt an 80-bit security level (RSA-1024 equivalent), which can be achieved by choosing $n \geq 2^{160}$ and $q^k \geq 2^{1024}$ for pairing-based cryptography. It means that we need to work with values that have more than 1024 bits. This fact makes the pairing computation the most expensive operation in TinyIBAK. For the pairing implementation we employed the η_T algorithm [27], which is demonstrated to achieve a good efficiency on the embedded sensor devices [28]. The scalar point multiplication is another

major operation in our scheme. We used the Right-to-Left window nonadjacent form (R-wNAF) method [29] for general point multiplication, and the Left-to-Right window nonadjacent form (L-wNAF) method [29] for fixed point multiplication. Both methods we chose are constructed based on the slide window techniques, and specially designed for memory-constrained platforms. In accordance with the required security level we choose the supersingular elliptic curve $y^2 + y = x^3 + x$ with the embedding degree $k = 4$ defined over the binary field $\mathbb{F}_{2^{271}}$ for our implementation.

As shown in **Fig. 5**, the core functions of TinyIBAK are modeled as three TinyOS components: *SecPrimitives*, *KeyAgree*, and *IdentityVerify*. The *SecPrimitives* component encapsulates the necessary security primitives from RELIC library, and provides the calling interface for other components. The *KeyAgree* component takes the responsibility for generating the key token and computing the shared secret key. The *IdentityVerify* component generates and verifies the authentication tags of the IDs and key tokens of both sides, using a keyed-hash based message authentication code (HMAC) algorithm. TinyIBAK uses Timer and Random interfaces (provided by TimerC and RandomLFSR components, respectively) to handle the randomly delay transmission during the *Key Agreement* phase. On receiving the key agreement request from the initiator, the neighbors start a timer to wait a random period of time before transmission, to avoid channel contention and message collisions.

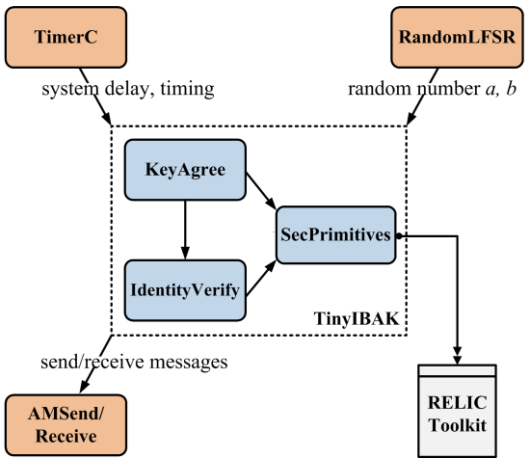


Fig. 5. The TinyOS components used in the implementation of TinyIBAK

7. Experiment and Performance Evaluation

We conduct two experiments to explore the feasibility and performance of our proposal. The first experiment focuses on the performance of TinyIBAK deployed on a pair of sensor nodes, and the second experiment investigates the effect of node density on the performance of our proposal.

7.1 Node-Level Experiment

The goal of our first experiment is to evaluate the feasibility and performance of TinyIBAK on the MICAz sensor motes, a popular choice among the research community. The MICAz is based on the low-power 8-bit microcontroller ATmega128L, which operates at 7.3828MHz and offers 4KB of RAM memory and 128KB of program space. The MICAz runs TinyOS and embeds an IEEE 802.15.4 compliant CC2420 transceiver with a claimed data rate of 250 kbps.

For simplicity, we deployed the programs of TinyIBAK on a pair of MICAz motes. Here we only focus on the operations carried out by the sensor nodes during the Key Agreement phase. We used Avrora [30], an instruction-level AVR-microcontroller-oriented analysis tool, to evaluate the time and energy performance on the MICAz motes. We measured the memory occupation by examining the binary image of the programs of TinyIBAK. In the rest of this subsection, we discuss the performance of our proposal in terms of computation, communication and storage.

Computation Overhead. The TinyIBAK scheme involves the computations of one bilinear pairing, two 271-bit scalar point multiplications, one scalar point addition, two hashing functions and one MAC. The empirical computation overhead of TinyIBAK is shown in **Table 1**. *Initialization* comprises the parameter configuration of RELIC and the loading of the system parameters. *Parameter Computation* comprises the random chosen of a (b) and the computation of key token W_A (W_B). *Shared Key Computation* comprises the computation of the shared secret and the derivation of the shared key.

Table 1. Computation overhead of TinyIBAK

	Initialization	Parameter Computation	Shared Key Computation	MAC Computation	MAC Verification	Total
CPU Cycles	9,614,652	121,582	16,115,641	156,815	157,058	26,165,748
Energy (mJ)	29.60	0.37	49.91	0.48	0.49	80.85
Time (s)	1.304	0.016	2.186	0.021	0.021	3.55

As shown in the **Table 1**, the TinyIBAK scheme takes 3.55s and consumes 80.85mJ in total to execute on ATmega128L micro-controller. Since the *Initialization* is only executed once at the bootstrap of sensor nodes, the actual time and energy consumption for the key agreement are 2.25s and 51.25mJ respectively. These data show that TinyIBAK consumes an acceptable amount of resources, given that these operations are performed very rarely and mainly at the beginning of the network operation. After the key agreement phase nodes will use much cheaper symmetric key encryption methods.

Memory Overhead. The memory overhead of TinyIBAK includes the RELIC codes, TinyOS codes, node's id, private key, public key, system parameters and the established shared keys with its neighbors. A private key (public key) requires a point on an elliptic curve, which is represented by coordinates (x, y) from a finite field with 271-bit elements. Given x and a single bit of y, one can easily derive y. Thus, a private key (public key) can be compressed to 34 bytes.

Table 2. Memory overhead of TinyIBAK

Scheme	RAM (Byte)		ROM (Byte)
	.data	*.bss*	*.text*
TinyIBAK	1,096	1,168	57,146

Table 2 depicts the memory occupation of TinyIBAK scheme, where the *.bss* and *.data* segments consume RAM while the *.text* segment consumes ROM. The RAM utilization may seem high for MICAz sensor motes but most memory is reserved only for the duration of key agreement. The values showed in **Table 2** present only the peak numbers for stack usage during the program execution.

Communication Overhead. In our TinyIBAK scheme, three messages are required to be exchanged during the key agreement, which can be expressed in the form of (1).

$$\begin{aligned} &\mathrm{A} \to \mathrm{B}: ID_A \| W_A \\ &\mathrm{B} \to \mathrm{A}: ID_B \| W_B \| t_B \\ &\mathrm{A} \to \mathrm{B}: t_A \end{aligned} \tag{1}$$

The key token W_A (W_B) is a point on $E\left(\mathrm{F}_{2^{271}}\right)$, thus can be compressed to 34 bytes. Each node identifier consists of 2 bytes. The MAC function is implemented with SHA1, thus the MAC tag occupies 16 bytes. Therefore, the initiator and the responder are required to transmit 52 bytes, respectively. Here we do not take the extra overhead arose from TinyOS packet encapsulation into consideration.

The communication overhead of TinyIBAK scheme is fixed for each node pair and independent of the network size. This feature allows our scheme to scale gracefully with the number of nodes in the network.

7.2 Network-Level Experiment

The goal of our second experiment is to explore the effects of different node densities on the performance of TinyIBAK. We consider the case that a new sensor node is added into the network. After the neighborhood discovery, it needs to establish pairwise keys with all of its neighbors. We measured the time, energy and memory required by the newly added node to establish pairwise key with all of its neighbors, whose number is increasing with the node density.

We conduct the experiment within the TOSSIM simulation framework [31] for its accurate representation of real-world sensor networks. In the simulation, we randomly deployed 100 sensor nodes in the area of 200 m * 200 m, and chose a set of parameters to describe a scenario with a low-level asymmetric radio channel, as depicted in **Table 3**. The neighbors of the new comer are assigned by setting appropriate link qualities between the new comer and other nodes in the network.

Table 3. Simulation parameters

Parameter	Value
Area Size	200 m * 200 m
Quantity of Nodes	100
New Comer Position	(100, 100)
Packet size	36 Bytes
Maximum data rate	250 Kbps
Maximum backoff times	Infinity
Average radio noise floor	-110 dbm
Standard deviation for WGN	4.0dB
Receiving Sensitivity	-105dBm

We repeated our experiment for 100 times and generated a new set of inter-node link qualities for each run of the experiment, in order to prevent the result from being unduly affected by particularly good or bad connectivity for certain nodes in a generated topology. We chose to configure the media access protocol to make the sensor node keep backing off until it gets a chance to transmission. The choice greatly reduces retransmission at the cost of increasing latency.

In the second experiment, we evaluated the time, energy and memory consumed by a newly added node to establish pairwise keys with its neighbors. Since TOSSIM itself does not provide the energy model or MCU executing time model, we used the experiment data

measured in section 7.1 to model the MCU executing time and energy consumption. We adopted the communication energy model constructed based on the CC2420 transceiver which transmits at a maximum data rate of 250 Kbps and an output power of 0 dBm, and receives at a constant reception power of 35.28 mW [32]. The parameters of time and energy models are listed in **Table 4**.

Table 4. Parameters of time and energy model

Behavior	Time	Energy
System Initialization	1.304 s	29.6 mJ
Parameter Generation	0.016 s	0.37 mJ
Shared Key Computation	2.186 s	49.91 mJ
MAC Computation	0.021 s	0.48 mJ
MAC Verification	0.021 s	0.49 mJ
Data Transmission (@ 250 kbps, 0dBm)	N/A	122 nJ/bit
Data Reception (@ 34.28 mW)	N/A	140 nJ/bit

Time for New Node Addition. We conducted 100 repeated experiments to measure the time taken by a newcomer to accomplish the key agreement with its neighbors. The experiment results depicted in **Table 5** indicates that when the number of neighbors *d* varies from 2 to 20, the empirical time shows a trend of increase in general, whereas the rising rate remains stable within an acceptable range, which is not affected by the network density. When *d* reaches 20, the newcomer needs about 47 seconds to complete the key agreement with all its neighbors.

Table 5. Time for a sensor node to establish pairwise keys with its ***d*** neighbors

d	2	4	6	8	10	12	14	16	18	20
Time (s)	6.48	10.85	16.29	21.54	25.78	30.69	35.47	38.70	44.48	47.03

Energy Consumption for New Node Addition. We recorded the energy dissipation of each operation of the newly added node in accordance with the model depicted in **Table 4**, and added up the records to achieve the total value at the end of each round of simulation. **Fig. 6** shows the energy consumption of a newly added node to accomplish the key agreement with all of its neighbors, which ranges from 2 to 20. The total energy consumption rises slowly with the increase of neighbors, which mainly due to the growth of traffic and the time for listening channel. The total energy consumption reaches around 1.0 joule when the number of neighbors is 20.

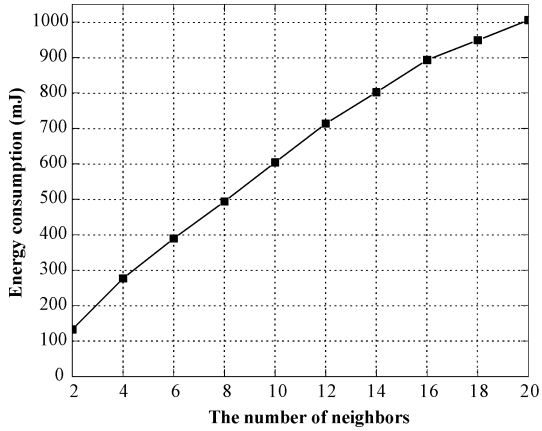


Fig. 6. Energy consumption for a sensor node to establish pairwise keys with its neighbors

RAM Occupation. As the node density increased, the newly added node needs to establish more pairwise keys with its neighbors, which means that the newcomer has to cache more key tokens from its neighbors. The relationship between the RAM occupation and the number of neighbors is shown in **Table 6**. Note that the memory allocated for the key tokens are reusable, which means once the pairwise key is established, the memory is available for other operations. Thus, the memory efficiency of our scheme is much higher than the key pre-distribution schemes like E-G, in which the key ring always resides in RAM.

Table 6. RAM occupation of TinyIBAK for d neighbors

d	2	5	10	15	20
RAM (bytes)	2298	2400	2570	2740	2910

The results from the two experiments demonstrate that our proposal consumes an acceptable amount of resources, and is feasible for infrequent key agreement and rekeying in large scale sensor networks.

8. Comparison with Other Schemes

In this section, we compare TinyIBAK with some well-known key distribution protocols, including the ID-based key distribution approach proposed by Yang et al. [16] (referred to as Yang's scheme), Oliveira et al. [15] (referred to as TinyPBC), and the key pre-distribution scheme proposed by Eschenauer and Gligor [5] (referred to as E-G scheme).

The E-G scheme doesn't involve any PKC operations, so its computation cost is negligible compared with other schemes. In **Table 7**, we compare the computation cost of the other two ID-based schemes with our proposal. We ignore the time taken by conventional hash operations and point addition operations as they are much more efficient compared with pairings, scalar point multiplications, and modular exponentiation operations. As shown in **Table 7**, TinyIBAK need two more scalar point multiplications than TinyPBC.

To compare the practical performance of the three schemes, we also implement TinyPBC and Yang's scheme on the MICAz motes. For the implementation of TinyPBC, we adopt the same parameters as in our proposal. For the implementation of the Yang's scheme, we employ a modulus of 1024 bits and an exponent of 160 bits for Diffie-Hellman, as recommended by NIST. All the performance data are the average results of 20 repeated experiments. The comparison of the performance evaluation results for the TinyIBAK, TinyPBC and Yang's scheme is shown in **Table 8**.

Table 7. Comparisons of computation cost

Scheme	Bilinear Pairing	Scalar point multiplication	Modular exponentiation	Exp. on G_T
TinyIBAK	1	2	0	0
Yang's	2	1	2	1
TinyPBC	1	0	0	0

Table 8. Performance comparisons

Scheme	CPU Cycles	Time(s)	Energy(mJ)	ROM(Byte)	RAM(Byte)
TinyIBAK	26,165,748	3.55	80.85	57,146	2,264
Yang's	46,566,670	6.30	149.73	78,004	4,967
TinyPBC	24,584,034	3.33	75.69	57,096	1,726

In respect of communication overhead, Yang's scheme requires each node to transmit 168 bytes due to the utilization of 1024-bit modular in Diffie-Hellman key exchange. In TinyPBC, the SOK protocol only needs to exchange the node IDs, which only introduces 2 bytes communication for each node. In the E-G scheme, each sensor broadcasts the list of identifiers of keys on its key ring. When the key pool $S = 10000$, the number of preloaded keys m of each node needs to be larger than 150 to achieve a high key sharing probability of 0.9 [5]. Each key identifier requires at least 14 bits and the resulting message have a size more than 263 bytes. The comparison of the communication overhead of the three key distribution schemes with TinyIBAK is shown in **Fig. 7**. Here we do not take the extra overhead arose from TinyOS packet encapsulation into consideration.

According to the results in **Table 8** and **Fig. 7**, TinyIBAK is much more efficient than Yang's scheme in the terms of computation, communication and memory overhead. The performance of our TinyIBAK is comparable with TinyPBC, except that in terms of the communication cost TinyIBAK is slightly less efficient than TinyPBC. However, TinyPBC, in which the shared secret is static, does not provide rekeying and key confirmation. Compared with the E-G scheme, TinyIBAK has many advantages in terms of security strength, key connectivity, scalability, communication and storage overhead.

The communication overhead of the TinyIBAK scheme is fixed for each node pair, which is independent of the network size. Whereas in case of the key pre-distribution schemes (e.g. E-G), the communication overhead increases significantly with the number of preloaded keys and the key pool size. If two nodes have to resort to a third node that might be one or multiple hops away, for helping establishing a pairwise key, larger communication overhead will be incurred. When the network expands to a large scale, this kind of scheme is no longer practicable.

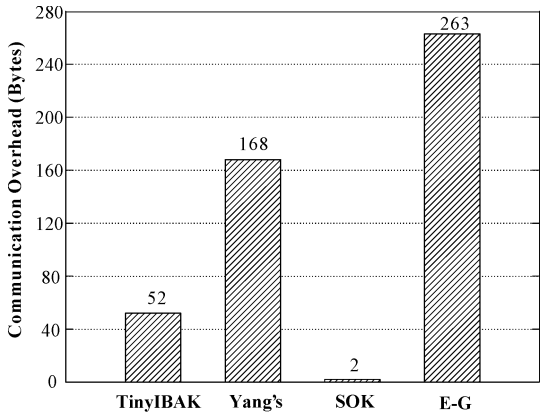


Fig. 7. Comparisons of communication overhead

One of the biggest advantages of TinyIBAK when compared with the E-G scheme is the significant storage saving in terms of cryptographic keys. Assume that the number of sensors in a WSN is N. In TinyIBAK, each node is preloaded with a pair of keys (node's private key and public key, 34 bytes each), thus the total number of preloaded keys in the network is $2N$. In the E-G scheme, each node is preloaded with m keys (e.g. each key 80-bit). Thus the total number of preloaded keys in the network equals to mN. The value of m depends on the key pool size (S) and the probability of sharing at least one key between two nodes (p).

Fig. 8 depicts the total number of keys preloaded in the whole network for E-G and TinyIBAK. The parameters of E-G were set as $S = 5000$, $p = 0.87$, and two different values of m ($m = 20, 50$). As shown in **Fig. 8**, the storage savings of TinyIBAK increase drastically with the number of sensor nodes in the network. In TinyIBAK, the number of keys preloaded in

each node is independent of the number of nodes. This feature allows our scheme to scale gracefully with the number of nodes in the network.

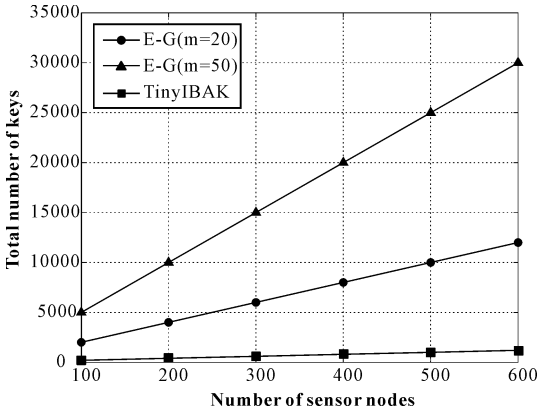


Fig. 8. Comparison of the preloaded key storage space

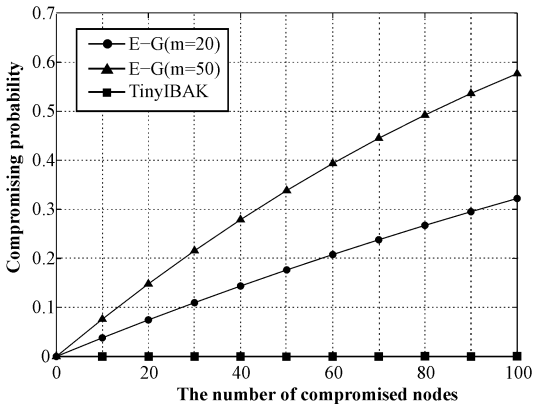


Fig. 9. Comparison of the compromising probability

As referred to the security strength, we first define the *compromising probability*, which is the probability that the attacker can decrypt the communication between any two nodes which are not compromised after capturing c nodes. In TinyIBAK, each sensor is preloaded with one unique private key. After the session key established, each pair of nodes has a different shared key. Thus, compromising c sensors does not affect the security of communication among other pairs of nodes. Chan et al. [7] presents the formula to calculate the probability that two sensors have exactly j common keys in the E-G scheme $p(j)=\binom{S}{j}\binom{S-j}{2(m-j)}\binom{2(m-j)}{m-j}\Big/\binom{S}{m}^2$, where S is the key pool size and m is the number of preloaded keys in each node. The compromising probability can be calculated as (2).

$$c(m)=\sum_{j=1}^{m}\left(1-\left(1-\frac{m}{S}\right)^{c}\right)^{j}p(j)\Big/\sum_{j=1}^{m}p(j) \tag{2}$$

In **Fig. 9** we compare the compromising probability of TinyIBAK and E-G. We set the parameters S as 5000 and m as 20 and 50 in two cases for the E-G scheme, respectively. As **Fig. 9** shows, for the E-G scheme the compromising probability increases significantly with the number of preloaded keys, while for TinyIBAK the compromising probability is always zero for all c values. Therefore, TinyIBAK provides deterministic security, and strong resilience to the node compromise attack. Whereas, the E-G scheme only provides probabilistic security, and is vulnerable to the node compromise attack.

9. Conclusions

In this paper, we proposed an identity-based authenticated key agreement scheme, TinyIBAK, based on the bilinear paring, for large scale sensor networks. Through the formal security proof, the heuristical security analysis, and the formal security validation using AVISPA, we can see that our scheme is strongly secure against the passive and active attacks, such as replay attacks, man-in-the middle attacks and node compromise attacks, etc.

To evaluate the feasibility and performance of our proposal, we implemented it for

TinyOS-2.1 based on the MICAz motes, analyzed the memory occupation, and evaluated the time and energy performance with Avrora. Moreover, we evaluated the effect of the node density on the performance of our scheme within the TOSSIM simulation framework. Experimental results indicate that our proposal consumes an acceptable amount of resources, and is feasible for infrequent key distribution and rekeying in large scale sensor networks.

Compared with other ID-based key agreement approaches, TinyIBAK is much more efficient, or comparable in performance but provides rekeying. Compared with the traditional key pre-distribution scheme, TinyIBAK achieves many advantages in terms of security strength, key connectivity, scalability, communication and storage overhead, and enables efficient secure rekeying.

References

[1] Chris Karlof, Naveen Sastry and David Wagner, "TinySec: a link layer security architecture for wireless sensor networks," in *Proc. of 2nd Int. Conf. on Embedded Networked Sensor Systems*, pp.162-175, November 3-5, 2004. Article (CrossRef Link).

[2] Robert Szewczyk, Alan Mainwaring, Joseph Polastre, John Anderson and David Culler, "An analysis of a large scale habitat monitoring application," in *Proc. of 2nd Int. Conf. on Embedded Networked Sensor Systems*, pp.214-226, November 3-5, 2004. Article (CrossRef Link).

[3] AVISPA Project. Automated Validation of Internet Security Protocols and Applications [Online]. Available: http://www.avispa-project.org/.

[4] Diego F. Aranha. RELIC Cryptographic Toolkit [Online]. Available: http://code.google.com/p/relic-toolkit.

[5] Laurent Eschenauer and Virgil D. Gligor, "A key-management scheme for distributed sensor networks," in *Proc. of 9th ACM Conf. on Computer and Communications Security*, pp.41-47, November 17-21, 2002. Article (CrossRef Link).

[6] Roberto Di Pietro, Luigi V. Mancini and Alessandro Mei, "Random key-assignment for secure Wireless Sensor Networks," in *Proc. of 1st ACM Workshop on Security of Ad hoc and Sensor Networks,* pp. 62-71, November 1-3 , 2003. Article (CrossRef Link).

[7] Haowen Chan, Adrian Perrig and Dawn Song, "Random key predistribution schemes for sensor networks," in *Proc. of 23th IEEE Symposium on Security and Privacy*, pp.197-213, May 11-14, 2003. Article (CrossRef Link).

[8] Donggang Liu, Peng Ning and Rongfang Li, "Establishing pairwise keys in distributed sensor networks," *ACM Transactions on Information and System Security*, vol. 8, no. 1, pp. 41-77, February, 2005. Article (CrossRef Link).

[9] Mohammed A. Moharrum and Mohamed Eltoweissy, "A study of static versus dynamic keying schemes in sensor networks," in *Proc. of 2nd ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*, pp. 122-129, October 13-15, 2005. Article (CrossRef Link).

[10] Mohamed Eltoweissy, Mohammed A. Moharrum, and Ravi Mukkamala, "Dynamic key management in sensor networks," *IEEE Communications Magazine*, vol. 44, no. 4, pp. 122-130, April, 2006. Article (CrossRef Link).

[11] Michael Chorzempa, Park Jung-Min, and Mohamed Eltoweissy, "SECK: survivable and efficient clustered keying for wireless sensor networks," in *Proc. of 24th IEEE International Performance, Computing, and Communications Conference*, pp.453-458, April 7-9, 2005. Article (CrossRef Link).

[12] Ashok Kumar Das, "A random key establishment scheme for multi-phase deployment in large-scale distributed sensor networks," *International Journal of Information Security*, vol. 11, no. 3, pp. 189-211, June, 2012. Article (CrossRef Link).

[13] David J. Malan, Matt Welsh and Michael D. Smith, "A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography," in *Proc. of 1st Annual IEEE*

Communications Society Conference on Sensor and Ad Hoc Communications and Networks, pp.71-80, October 4-7, 2004. Article (CrossRef Link).

[14] Ronald Watro, Derrick Kong, Sue-fen Cuti, Charles Gardiner, Charles Lynn and Peter Kruus, "TinyPK: securing sensor networks with public key technology," in *Proc. of 2nd ACM Workshop on Security of Ad hoc and Sensor Networks*, pp.59-64, November 1-3, 2004. Article (CrossRef Link).

[15] Leonardo B. Oliveira, Diego F. Aranha, Conrado P. L. Gouvêa, Michael Scott, Danilo F. Câmara, Julio López, et al., "TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks," *Computer Communications,* vol. 34, no. 3, pp. 485-493, March, 2011. Article (CrossRef Link).

[16] Geng Yang and Hongbin Cheng, "An efficient key agreement scheme for wireless sensor networks," *Chiness Journal of Electronics*, vol. 36, no. 7, pp. 1389-1395, July, 2008. Article (CrossRef Link).

[17] Simon Blake-Wilson, Don Johnson, and Alfred Menezes, "Key agreement protocols and their security analysis," *Cryptography and Coding,* vol. 1355, pp. 30-45, December, 1997. Article (CrossRef Link).

[18] Chris Karlof and David Wagner, "Secure routing in wireless sensor networks: attacks and countermeasures," in *Proc. of 1st IEEE International Workshop on Sensor Network Protocols and Applications*, pp. 113-127, May 11, 2003. Article (CrossRef Link).

[19] David Basin, Sebastian Modersheim, and Luca Vigano, "OFMC: A symbolic model checker for security protocols," *International Journal of Information Security*, vol. 4, no. 3, pp. 181-208, June, 2005. Article (CrossRef Link).

[20] Mathieu Turuani, "The CL-Atse Protocol Analyser," *Term Rewriting and Applications,* vol. 4098, pp. 277-286, August, 2006. Article (CrossRef Link).

[21] Alessandro Armando and Luca Compagna, "SAT-based Model-Checking for Security Protocol Analysis," *International Journal of Information Security*, vol. 7, no. 1, pp.3-32, January, 2008. Article (CrossRef Link).

[22] Y. Boichut, P. C. Heam, O. Kouchnarenko, and F. Oehl, "Improvements on the Genet and Klay Technic to Automatically Verify," in *Proc. of International Workshop on Automated Verification of Infinite-State System, joint to ETAPS'04*, pp. 1-11, March 27-April 4, 2004. Article (CrossRef Link).

[23] David von Oheimb, "The high-level protocol specification language HLPSL developed in the EU project AVISPA," in *Proc. of APPSEM workshop*, September 13, pp.1-17, 2005. Article (CrossRef Link).

[24] Danny Dolev and Andrew C. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198-208, March, 1983. Article (CrossRef Link).

[25] Ashok Kumar Das, Santanu Chatterjee, and Jamuna Kanta Sing, "Formal Security Verification of a Dynamic Password-Based User Authentication Scheme for Hierarchical Wireless Sensor Networks," *Security in Computing and Communications*, vol. 377, pp. 243-254, August, 2013. Article (CrossRef Link).

[26] NIST. Special Publication 800-57: Recommendation for Key Management [Online]. Available: http://csrc.nist.gov/CryptoToolkit/kms/guideline-1-Jan03.pdf.

[27] Paulo L. M. Barreto, Steven D. Galbraith, ColmÓ' hÉigeartaigh and Michael Scott, "Efficient pairing computation on supersingular Abelian varieties," *Designs, Codes and Cryptography,* vol. 42, no. 3, pp.239-271, March, 2007. Article (CrossRef Link).

[28] Piotr Szczechowiak, Anton Kargl, Michael Scott, and M. Collier, "On the application of pairing based cryptography to wireless sensor networks," in *Proc. of 2nd ACM Conference on Wireless Network Security*, pp.1-12, March 16-18, 2009. Article (CrossRef Link).

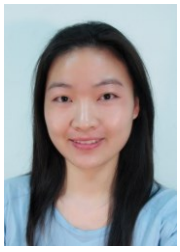
[29] Brain King, "wNAF *, an Efficient Left-to-Right Signed Digit Recoding Algorithm," in *Applied Cryptography and Network Security*, vol. 5037, pp. 429-445, June, 2008. Article (CrossRef Link).

[30] UCLA Compliers Group. Avrora: The AVR Simulation and Analysis Framework [Online]. Available: http://compilers.cs.ucla.edu/avrora/.

[31] Philip Levis, Nelson Lee, Matt Welsh, and David Culler, "TOSSIM: accurate and scalable

simulation of entire TinyOS applications," in *Proc. of 1st Int. Conf. on Embedded Networked Sensor Systems*, pp.126-137, November 5-7, 2003. Article (CrossRef Link).

[32] Agustin Barderis, Leonardo Barboni, and Maurizio Valle, "Evaluating Energy Consumption inWireless Sensor Networks Applications," in *Proc. of 10th Euromicro Conf. on Digital System Design Architecture, Methods and Tools (DSD)*, pp.455-462, August 29-31, 2007. Article (CrossRef Link).



Lijun Yang received the B.S. degree in 2007 from the School of Communication and Information Engineer, Nanjing University of Posts and Telecommunications, Nanjing, China, where she is currently pursuing the Ph.D. degree in information and Network Security. Her research interests include wireless sensor networks, information security, privacy preservation, and public key cryptography.

Chao Ding received the B.S. degree in 2006 from the School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing, China, where he is currently pursuing his Ph.D. degree in Information and Network Security. His current research interests include applied cryptography, anomaly detection in sensor networks and implementation of real sensor platforms.

Meng Wu received his B.S., M.S. and Ph.D. degrees in Communication Engineering and Computer Science from Zhejiang University, Shanghai Jiao Tong University, Southeast University, in 1985, 1990 and 1993, respectively. Currently, he is a professor and the leading person of Information Security doctoral discipline of Nanjing University of Posts and Telecommunications. His main research areas are wireless communication and information security.