On Broadcast Authentication in Wireless Sensor Networks

Kui Ren[1], Kai Zeng[1], Wenjing Lou[1], and Patrick J. Moran[2]

[1] Worcester Polytechnic Institute, Worcester, MA 01609
{kren, wjlou, kzeng}@ece.wpi.edu
[2] AirSprite Technologies, Inc., Northboro, MA 01532
pmoran@airsprite.com

Abstract. Broadcast authentication is a critical security service in wireless sensor networks (WSNs), since it enables users to broadcast the WSN in an authenticated way. Symmetric key based schemes such as μTESLA and multilevel μTESLA have been proposed to provide such services for WSNs; however, these schemes all suffer from serious DoS attacks because of the delayed message authentication. This paper presents several effective public key based schemes to achieve immediate broadcast authentication and thus overcome the vulnerability presented in the μTESLA-like schemes. Several cryptographic building blocks, including Merkle hash tree and ID-based signature scheme, are adopted to minimize the scheme overhead regarding the costs in both computation and communication. A quantitative analysis on energy consumption of the proposed schemes are given in detail. We believe that this paper can serve as the start point towards fully solving the important multisender broadcast authentication problem in WSNs.

1 Introduction

Wireless sensor networks (WSNs) have enabled data gathering from a vast geographical region, and present unprecedented opportunities for a wide range of tracking and monitoring applications from both civilian and military domains [1, 2, 8, 14]. In these applications, WSNs are expected to process, store and provide the sensed data to the network users upon their demands. As the most common communication paradigm, the network users are expected to issue the queries to the network before obtaining the information of their interest. Furthermore, in wireless sensor and actuator networks (WSANs) [2], the network users may even need to issue their commands to the network (probably based on the information he received from the network). In both cases, there could be a large number of users in the WSNs, which could be either mobile or static. And the users may use their mobile clients to query or command the WSNs from anywhere in the network. Obviously, broadcast/multicast[3] operations are fundamental to the realization of these network functions. Hence, it is also highly important to ensure broadcast authentication for the security purposes.

[3] For our purpose, we do not distinguish multicast from broadcast in this paper.

Broadcast authentication in WSNs has been first addressed by μTESLA in [3]. In μTESLA, the user of WSNs is assumed to be one or a few fixed sinks, which are always assumed to be trustworthy. The scheme adopts a one-way hash function $h()$ and uses the hash preimages as keys in a Message Authentication Code (MAC) algorithm. Initially, sensor nodes are preloaded with $K_0 = h^n(x)$, where x is the secret held by the sink. Then, $K_1 = h^{n-1}(x)$ is used to generate MACs for all the broadcast messages sent within time interval 1. At time interval 2, the sink broadcasts K_1, and sensor nodes verify $h(K_1) = K_0$. The authenticity of messages received during time interval 1 is then verified using K_1. This delayed disclosure technique is used for the entire hash chain and thus demands loosely synchronized clocks between the sink and sensor nodes. μTESLA is later enhanced in [4, 5] to overcome the length limit of the hash chain. Most recently, μTESLA is also extended in [6] to support multiuser scenario at the cost of higher communication overhead per message.

It is generally held that μTESLA-like schemes have the following shortcomings even in the single-user scenario: 1) all the receivers have to buffer all the messages received within one time interval; 2) they are subject to Wormhole attacks [7], where messages could be forged due to the propagation delay of the disclosed keys. However, here we point out a serious vulnerability of μTESLA-like schemes when they are applied in multi-hop WSNs. Since sensor nodes buffer all the messages received within one time interval, an adversary can hence flood the whole network with arbitrary messages all the time and this can be easily achieved by an outsider. All he has to do is to claim that the sending messages belong to the current time interval which should be buffered for authentication until next time interval. Since wireless transmission is very expensive in WSNs[4], and WSNs are extremely energy constrained, the ability to flood the network arbitrarily could cause devastating DoS attacks. Moreover, this type of DoS attacks become even more devastating in multiuser scenario, since the adversary can easily generate more bogus messages without being detected. Obviously, all these attacks are due to authentication delay of the broadcast messages. In [7], TIK is proposed to achieve immediate key disclosure and hence immediate message authentication based on precise time synchronization between the sink and receiving nodes. However, this technique is not applicable in WSNs as pointed out by the authors. Therefore, the problem of broadcast authentication still remains wide open in WSNs.

At the time when μTESLA was proposed, sensor nodes are assumed to be extremely resource constrained, especially with respect to computation capability, bandwidth availability, and energy supply [3]. Therefore, public key cryptography (PKC) is thought to be too computationally expensive, although it could provide much simplified solutions with much stronger security strengths. However, recent studies [9, 10] showed that, contrary to widely held beliefs, PKC even with software implementations is very viable on sensor nodes. For example [9], Elliptic Curve Cryptography (ECC) signature verification takes 1.61s

[4] Wireless transmission of a bit can require over 1000 times more energy than a single 32-bit computation, as shown in [14].

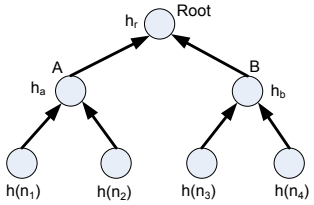

Fig. 1. An example of Merkle hash tree

with 160-bit keys on ATmega128 8MHz processor, a processor used for current Crossbow motes platform [11]. Hence, with the advance of fast growing technology, PKC is no longer impractical for WSNs, although still expensive for the current generation sensor nodes. And its wide acceptance is expected in the near future [10].

Having this observation and knowing that symmetric-key based solutions such as μTESLA are insufficient for broadcast authentication in WSNs, we resort to public key cryptography for effective solutions.

Organization of the paper: The remaining part of this paper is organized as follows: In Section 2, we introduce some preliminary background about the cryptography mechanisms. Section 3 presents the system assumptions, adversary model and security objectives of this paper. Then in Section 4, we introduce our proposed schemes and detail the underlying design logic. Section 5 is the scheme analysis. We conclude our paper in Section 6.

2 Preliminaries

2.1 Merkle hash tree technique: We illustrate the construction and application of the Merkle hash tree [13] through an example. To authenticate data values $n_1, n_2, ..., n_w$, the data source constructs the Merkle hash tree as depicted in Fig. 1, assuming that $w = 4$. The values of the four leaf nodes are the message hashes, $h(n_i), i = 1, 2, 3, 4$, respectively, of the data values under a one-way hash function $h()$ (e.g., SHA-1 [16]). The value of each internal node is derived from its child nodes. For example, the value of node A is $h_a = h(h(n_1)|h(n_2))$. The data source completes the levels of the tree recursively from the leaf nodes to the root node. The value of the root node is $h_r = h(h_a|h_b)$, which is used to commit to the entire tree to authenticate any subset of the data values n_1, n_2, n_3, and n_4 in conjunction with a small amount of auxiliary authentication information `AAI` (i.e., $\log_2 N$ hash values with N as the number of leaf nodes). For example, a user, who is assumed to have the authentic root value h_r, requests for n_3 and requires the authentication of the received n_3. Besides n_3, the source sends the `AAI` $:< h_a, h(n_4) >$ to the user. The user can then check the authenticity of the received n_3 by first computing $h(n_3)$, $h_b = h(h(n_3)|h(n4))$ and $h_r = h(h_a|h_b)$, and then checking if the calculated h_r is the same as the authentic root value h_r. Only if this check is positive, the user accepts n_3.

2.2 ID-based cryptography: Identity-based cryptography (IBC) is receiving extensive attention as a powerful alternative to traditional certificate-based cryptography. Its main idea is to make an entity's public key directly derivable from its publicly known identity information. Although the idea of IBC dates back to 1984 [15], only recently has its rapid development taken place due to the application of the *pairing* technique outlined below.

Let p, q be two large primes and $\mathbb{E}/\mathbb{F}_p$ indicate an elliptic curve $y^2 = x^3+ax+b$ over the finite field $\mathbb{F}_p$. We denote by $\mathbb{G}_1$ a q-order subgroup of the additive group of points of $\mathbb{E}/\mathbb{F}_p$, and by $\mathbb{G}_2$ a q-order subgroup of the multiplicative group of the finite field $\mathbb{F}^*_{p^i}$ $(i = 2, 3, 6)$. The Discrete Logarithm Problem (DLP) is required to be hard[5] in both $\mathbb{G}_1$ and $\mathbb{G}_2$. For us, a pairing is a mapping $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ with the following properties:

1. *Bilinear*: For $\forall\ P, Q, R, S \in \mathbb{G}_1$, $\hat{e}(P+Q, R+S) = \hat{e}(P, R)\hat{e}(P, S)\hat{e}(Q, R)\hat{e}(Q, S)$. Consequently, for $\forall\ c, d \in \mathbb{Z}^*_q$, we have $\hat{e}(cP, dQ) = \hat{e}(cP, Q)^d = \hat{e}(P, dQ)^c = \hat{e}(P, Q)^{cd}$, etc.
2. *Non-degenerate*: If P is a generator of $\mathbb{G}_1$, then $\hat{e}(P, P) \in \mathbb{F}^*_{p^2}$ is a generator of $\mathbb{G}_2$.
3. *Computable*: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for all $P, Q \in \mathbb{G}_1$.

3 System, Adversary Model, and Security Objectives

System model: In this paper, we consider a very large spatially distributed WSN, consisting of a fixed sink and a large amount of sensor nodes. The sensor nodes are not necessarily homogenous in their functionalities and capabilities. The WSN under consideration is aimed to offer information services to a large number of network users that roam in the network, in addition to the fixed sink. These WSN users include mobile sinks, vehicles, and people with mobile clients, and they are assumed to be more powerful than sensor nodes in terms of computation and communication abilities. For example, the network users could include a number of doctors, nurses, medical equipments (acting as actuators) and so on, in the case of CodeBlue [22], where the WSN is used for emergency medical response. These network users broadcast queries/commands through sensor nodes at their vicinity, and expect the replies that reflect the latest sensed results. The network users also directly communicate with sink or the backend server if needed. We assume that the sink is always trustworthy but the sensor nodes are subject to compromise. At the same time, the users of the WSN may be dynamically revoked due to either membership changing or compromise, and the revocation pattern is not restricted. As the μTESLA-like schemes, we also assume that the WSN time is loosely synchronized.

Adversary model: In this paper, we assume that the adversary's goal is to inject bogus messages into the network, attempt to deceive sensor nodes, and obtain the information of his interest. Additionally, Deny of Service (DoS) attacks

[5] It is computationally infeasible to extract the integer $x \in \mathbb{Z}^*_q = \{a | 1 \leq a \leq q-1\}$, given $P, Q \in \mathbb{G}_1$ (respectively, $P, Q \in \mathbb{G}_2$) such that $Q = xP$ (respectively, $Q = P^x$).

such as bogus message flooding, aiming at exhausting scarce network resources, is another important focus of the paper. We assume that the adversary is able to compromise both network users and the sensor nodes. The adversary hence could exploit the compromised users/nodes for such attacks. More specifically, we consider the following types of attacks: 1) The adversary may directly broadcast bogus messages to the WSN by itself; 2) The adversary may use one or more compromised nodes to propagate bogus messages to the WSN by pretending that the messages are initiated by legitimate network users; 3) The adversary may use one or more compromised users to broadcast messages to the WSN.

Security objectives: Given the adversary model above, our security objective is straightforward: First, user authentication is needed so that illegitimate users will be excluded from injecting bogus messages. Second, user revocation mechanisms have to be implemented so that sensor nodes could deal with user revocations. Third, the authenticity of any message broadcast by a user should be able to be verified by every receiving node. In summary, all messages being broadcast to the WSN should be authenticated so that any bogus ones issued by the illegitimate users and/or compromised sensor nodes can be efficiently and deterministically rejected/filtered.

4 The Proposed Schemes

PKC-based solutions can realize immediate message authentication and thus overcome the delayed authentication problem present in μTESLA-like schemes. However, the straightforward solutions such as certificate-based approach can not be directly applied in WSNs due to their high scheme overhead as we analyze below. More advanced techniques have to be adopted to achieve a desirable scheme performance.

4.1 The Certificate-Based Authentication Scheme

The scheme: Each user of the WSN is equipped with a public/private key pair (PK/SK), and signs every message he broadcasts with his SK using a digital signature scheme such as RSA or DSA [17, 18]. To prove the user's ownership over his public key, the sink[6] is also equipped with a public/private key pair and serves as the certificate authority (CA). The sink issues each user a public key certificate, and such a certificate, to its simplest form, consists of the following contents: $\mathtt{Cert}_{U_{ID}} = U_{ID}, \mathsf{PK}_{U_{ID}}, \mathtt{ExpT}, \mathrm{SIG}_{\mathsf{SK}_{Sink}}\{h(U_{ID}||\mathtt{ExpT}||\mathsf{PK}_{U_{ID}})\}, \mathtt{para}$, where U_{ID} denotes the user's ID, $\mathsf{PK}_{U_{ID}}$ denotes its public key, $\mathtt{ExpT}$ denotes certificate expiration time and $\mathrm{SIG}_{\mathsf{SK}_{Sink}}\{h(U_{ID}||\mathtt{ExpT}||\mathsf{PK}_{U_{ID}})\}$ is a signature signed over $h(U_{ID}||\mathtt{ExpT}||\mathsf{PK}_{U_{ID}})$ with SK_{Sink}. Hence, a broadcast message is now of the form as follows:

$$< M, tt, \mathrm{SIG}_{\mathsf{SK}_{U_{ID}}}\{h(U_{ID}||tt||M)\}, \mathtt{Cert}_{U_{ID}} > \qquad (I)$$

Here, M denotes the broadcast message and tt denotes the current time. Then, sensor nodes are enabled to verify the authenticity of the received messages

[6] We assume that the sink represents the network planner.

by preloading PK_{Sink} before the network deployment. The verification contains two steps: the certificate verification and the signature verification.

Analysis: This straightforward scheme suffers from many severe drawbacks. Firstly and most importantly, it is highly inefficient to support user revocation in this scheme. In order to support user revocation and hence certificate revocation, sensor nodes have to receive and store a certificate revocation list (CRL). Clearly, the CRL requires a storage space linear to the total number of revoked certificates over the whole network operation period at each sensor node. However, this is practically impossible due to the stringent storage limitations of sensor nodes, especially given a large number of users or a highly dynamic membership changing scenario. For example, assuming that a public key is 20-byte long, a CRL containing only $1,000$ revoked certificates is at least of size 19.5 KB even in the simplest format. At the same time, resorting to the sink on-demandingly for CRL verification is obviously impossible either, because this could introduce too much communication cost. Embedding validity interval into the certificate does not really help reduce the storage overhead much, since the revocation pattern is not available priori. Secondly, to authenticate each message, it always takes two signature verification operations, instead of one. This is because the certificate should always be authenticated in the first place.

4.2 The Basic Merkle Hash Tree Based Authentication Scheme

Observing the CRL problem inherent to the first scheme, we next propose a Merkle hash tree based authentication scheme, which is highly storage efficient.

Scheme initialization: The sink collects all the public keys of the current network users and constructs a merkel hash tree. Specifically, we construct N leaves with each leaf corresponding to a current user of the WSN. For our problem, each leaf node contains the bindings between the corresponding user ID and the public key of the user, that is, $h(U_{ID}, \mathsf{PK}_{U_{ID}})$. The values of the internal nodes are determined with the same method as in Section 2.1. We denote the value of the final root node of the hash tree as h_r. Then, the sink preloads/broadcasts each sensor node with this value either before network deployment or during the network operation time. However, if h_r is broadcast during the network operation time, h_r should be signed by the sink to prove its authenticity. Of course, in this case, sensor nodes should be preloaded with the sink's public key. At the same time, each user should obtain its `AAI` according to his corresponding leaf node's location in the Merkle hash tree. Let T denote all the nodes along the path from a leaf node to the root (not including the root). Then A is defined as the set of nodes corresponding to the siblings of the nodes in T; and `AAI` further corresponds to the values associated with the nodes in A. Obviously, `AAI` is $(L * \log_2 N)$ bytes, with the hash value equal to L bytes.

Message authentication: Now a message sent by a user U_{ID} is of form

$$< M, tt, \mathrm{SIG}_{\mathsf{SK}_{U_{ID}}}\{h(U_{ID}||tt||M)\}, U_{ID}, \mathsf{PK}_{U_{ID}}, \mathtt{AAI}_{U_{ID}} > \quad (II)$$

Each node verifies such a message in two steps. First, it verifies $\mathsf{PK}_{U_{ID}}$ using $\mathtt{AAI}_{U_{ID}}$ attached in the message and h_r stored by itself. The verification operation is a chain of hash operations with the final value equal to h_r as the way we demonstrated in section 2.1. A different value suggests the invalidity of the cor-

responding public key. Second, the sensor node verifies $\mathrm{SIG}_{\mathsf{SK}_{U_{ID}}}\{h(U_{ID}||M)\}$ using $\mathsf{PK}_{U_{ID}}$. Upon user revocation and/or addition, the sink updates the Merkle hash tree and obtains a new h_r. This new h_r is then signed by the sink using SK_{Sink} and broadcast to sensor nodes immediately. Furthermore, each current user also obtains his updated $\mathtt{AAI}_{U_{ID}}$ from the sink.

Analysis: In this scheme, a user does not need a certificate to prove the binding to his public key. Instead, a Merkle hash tree technique is used. A revoked or invalid user public key will never pass the verification, as long as the user holds the up-to-date root node value h_r. Hence, in this scheme, certificates are no longer necessary and can be eliminated. Furthermore, the user revocation problem (i.e., certificate revocation problem) is now reduced to the problem of updating sensor nodes a single hash value h_r, which requires a storage space of only L bytes. Assuming that SHA-1 [16] is used, $L = 20$ bytes. However, the scheme is communication inefficient when N becomes large. This is because the size of AAI grows logarithmically as N grows. Since $L = 20$ bytes, AAI alone is of size 200 bytes, given number of users N reaches $1,024$; and $|\mathtt{AAI}| = 260$ bytes, when $N = 8,192$.

4.3 The Enhanced Merkle Hash Tree Based Authentication Scheme

In the above scheme, the storage overhead is only one hash value, i.e., L bytes, but the communication overhead is no less than $L * \log_2 N$ bytes. We hence, want to make a compromise between the storage and communication overheads. That is, we increase the number of stored hash values to reduce the size of AAI.

We illustrate how to do it through an example. In Fig.1, h_r is made public and stored by the authenticator. Hence, the user corresponding to leaf node n_3 must have $\mathtt{AAI} :< h_a, h(n_4) >$. However, if both h_a and h_b are made public and stored by the authenticator, the corresponding AAI now contains $h(n_4)$ only. Therefore, by trimming down the Merkle hash tree constructed in the above scheme, we can have a set of smaller Merkle hash trees. If each sensor node is loaded with all the values of the root nodes corresponding to these smaller trees, then the size of AAI can be reduced to the height of the smaller trees multiplying L bytes. In fact, if we remove k levels of the original Merkle tree, the communication overhead is reduced by $k * L$ bytes. However, the storage cost increases to $2^k * L$ bytes. Note that if we require sensor nodes to store all the leaf values, the scheme is reduced to the trivial memorize-all-keys case, which demands $N * L$ bytes storage space.

Analysis: Since sensor nodes are storage constrained, the value of k is obviously limited. Given that $m = 2^k$ hash values can be stored by each sensor node, the size of AAI is now $(L * \log_2 \frac{N}{m})$ bytes. If $N = 1,024$ and $m = 32$, this is 100 bytes; and if N is increased to $8,192$, this is 160 bytes. If m is made to be 64, then the size of AAI will be 80 bytes, give $N = 1,024$, and 140 bytes, given $N = 8,192$. This result is much improved as compared to the above basic scheme. When $N = 8,192$, the message overhead in this optimized scheme is 120 bytes less than that of the basic Merkle hash tree based scheme. This gain comes at the cost of increased storage overhead, which is now $64 * 20 = 1,280$ bytes $= 1.25$ KB. Therefore, this scheme is still communication inefficient when N is large.

However, when N is on the order of hundred, the proposed enhanced scheme can behave fairly well. We defer the detailed analysis to Section 5.

4.4 ID-Based Authentication Scheme

In this section, we propose an ID-based authentication scheme. In contrast to the Merkle hash tree based schemes, the proposed ID-Based authentication scheme requires sensor nodes to memorize the revoked user IDs only, and adopts an automatic public key update technique.

In our ID-based authentication scheme, the time is divided into consecutive time intervals, denoted by v_1, v_2,..., and we assume that sensor nodes and users are loosely synchronized. We then adopts $U_{ID}||\overline{v_i}$ as user U_{ID}'s public key under an ID-based signature scheme [19]. In this way, before a user wants to authenticate itself to the sensor nodes, he has to firstly obtain its private key from the sink. And since each obtained private key is valid only within the current time interval, every user has to obtain a new private key from the sink at the beginning of each time interval. Now upon user revocation, the sink only needs to broadcast the corresponding user IDs to the sensor nodes. Each sensor node stores a local copy of such revoked IDs only within the current interval and dumps them afterwards. The scheme works as follows.

Scheme initialization: Prior to network deployment, we assume that the sink does the following operations:

1. Generate the pairing parameters $(p, q, \mathbb{E}/\mathbb{F}_p, \mathbb{G}_1, \mathbb{G}_2, \hat{e})$, as described in Section 2.2. Select an arbitrary generator P of $\mathbb{G}_1$.
2. Choose two cryptographic hash functions: H, mapping strings to non-zero elements in $\mathbb{G}_1$, and h, mapping arbitrary inputs to fixed-length outputs, e.g., SHA-1 [16].
3. Pick a random $\kappa \in \mathbb{Z}_q^*$ as the network master secret and set $P_{pub} = \kappa P$.
4. Preload each sensor node with the public system parameters $(p, q, \mathbb{E}/\mathbb{F}_p, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, H, h, P, P_{pub})$.
5. Preload each user U_{ID} with the private key $\mathsf{SK}_{U_{ID}} = kH(U_{ID}||v_1)$

Message broadcast authentication: Assume that user U_{ID} wants to broadcast a message M. He first obtains its private key from the sink as $\mathsf{SK}_{U_{ID}} = \kappa H(U_{ID}||\overline{v_i})$, where v_i is the current time interval. U_{ID} then picks a random $\alpha \in \mathbb{Z}_q^*$ and computes $\theta = \hat{e}(P, P)^\alpha$. U_{ID} further computes $U_{x,y} = h(M \parallel tt \parallel \theta)\mathsf{SK}_{U_{ID}}$, and $\sigma_{x,y} = U_{x,y} + \alpha P$. $< \sigma_{x,y}, h(M \parallel tt \parallel \theta) >$ is the signature on message M. And the broadcast message is now of form

$$< U_{ID}, tt, M, \sigma_{x,y}, h(M \parallel tt \parallel \theta) > \qquad (III)$$

Upon receiving Message (III), each sensor node verifies its authenticity in the following way: It checks the current time $\overline{tt}$ and determines whether or not the received message is fresh. Assume δ is the predefined message propagation time limit. Then, we should have $\overline{tt} - tt \leq \delta$. If so, the sensor node further computes, $\theta' = \hat{e}(\sigma_{x,y}, P)\hat{e}(H(U_{ID}||\overline{v_i}), -P_{pub})^{h(M\|tt\|\theta)}$, using the current time interval $\overline{v_i}$.

If the message is authentic, we will have

$$\begin{aligned}\theta' &= \hat{e}(\sigma_{x,y}, P)\hat{e}(H(U_{ID}||\overline{v_i}), P_{pub})^{-h(M\|tt\|\theta)} \\ &= \hat{e}(h(M \parallel tt \parallel \theta)\mathsf{SK}_{U_{ID}} + \alpha P, P)\hat{e}(H(U_{ID}||\overline{v_i}), \kappa P)^{-h(M\|tt\|\theta)} \\ &= \hat{e}(h(M \parallel tt \parallel \theta)\mathsf{SK}_{U_{ID}} + \alpha P, P)\hat{e}(\kappa H(U_{ID}||\overline{v_i}), P)^{-h(M\|tt\|\theta)} \\ &= \hat{e}(\mathsf{SK}_{U_{ID}}, P)^{h(M\|tt\|\theta)}\hat{e}(P, P)^{\alpha}\hat{e}(\mathsf{SK}_{U_{ID}}, P)^{-h(M\|tt\|\theta)} = \theta.\end{aligned} \tag{1}$$

Therefore, if $h(M \parallel tt \parallel \theta') = h(M \parallel tt \parallel \theta)$, a sensor node considers the message authentic. If the above verification fails, a sensor node thinks of the message a fabricated or replayed one, and simply dumps it. Otherwise, it propagates the message to the next hop.

Analysis: The pros of the ID-based authentication scheme are two-fold: First, it eliminates the existence of certificate or auxiliary authentication information. Therefore, the resulted message size can be reduced. Second, it requires much smaller storage space to support user revocation, since now only the revoked user IDs have to be stored. Assuming a WSN supporting up to 65535 users, then two bytes are enough for the length of a user ID. Hence, accumulating the same 1,000 revoked users, now only 2,000 bytes = 1.95 KB storage space are needed. However, the cons of the ID-based authentication scheme are also obvious, since it has a very high computation cost due to the pairing operation involved.

5 A Quantitative Performance Comparison

Energy consumption on message broadcast: In this section, we study how much are the energy consumptions to broadcast messages of different sizes to the whole WSN. We further study these energy consumptions as the function of the WSN size W. We denote by E_{tr} the hop-wise energy consumption for transmitting and receiving one byte. As reported in [9], a Chipcon CC1000 radio used in Crossbow MICA2DOT motes consumes 28.6 and 59.2 μJ to receive and transmit one byte, respectively, at an effective data rate of 12.4 kb/s. Furthermore, we assume a packet size of 41 bytes, 32 for the payload and 9 bytes for the header [9]. The header, ensuing a 8-byte preamble, consists of source, destination, length, packet ID, CRC, and a control byte [9].

For the certificate-based scheme, $\mathtt{Cert}_{U_{ID}}$ is at least 86 bytes [9], even if ECDSA-160[7] is used. The total message size of form (I) is then 148 bytes, assuming M 20 bytes, tt 2 bytes. Hence, there should be 5 packets in total, among which four of them are of size 41 bytes, and one packet is of size 29 bytes. Therefore, there should be $41 * 4 + 29 * 1 + 8 * 5 = 233$ bytes for transmission (including 8-byte preamble per packet). Hence, the hop-wise energy consumption on transmitting Message (I) equals to $233 * 59.2$ μJ = 13.79mJ; And the

[7] ECDSA is referred to Elliptic Curve Digital Signature Algorithm [**?**]. While RSA with 1024-bit keys (RSA-1024) provides the currently accepted security level, it is equivalent in strength to ECC with 160-bit keys (ECC-160). And hence, for the same level of security strength, ECDSA uses a much small key size and hence has a small signature size (320-bit).

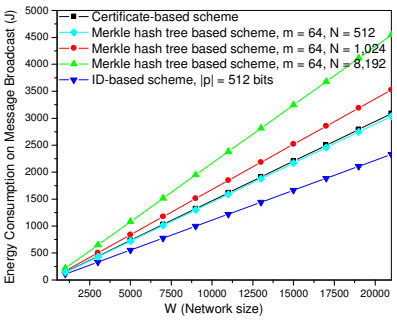

Fig. 2. Energy consumption on Message Broadcast vs. network size

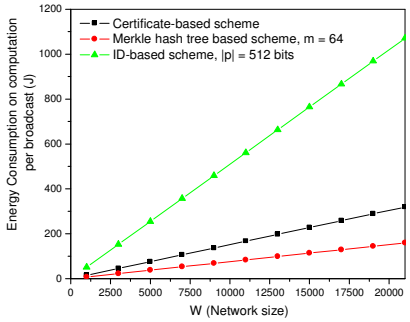

Fig. 3. Energy consumption on computation vs. network size

energy consumption on receiving Message (I) equals to $233 * 28.6\ \mu\text{J} = 6.66\text{mJ}$. To broadcast a message to the whole WSN, every sensor node should at least retransmit once and receive w' times the same message, when the simple flooding technique is used. Here, w' denotes the neighborhood density. Hence, the total energy consumption on message broadcast will be $W * (13.79 + 6.66 * w')\text{mJ}$. The energy consumption on message broadcast for the remaining scheme can also be calculated similarly.

Fig. 2 illustrates these broadcast energy consumptions as a function of network size W, assuming $w' = 20$. Clearly, we see that the ID-based scheme offers a much lower energy consumption as compared to that of the remaining two schemes. For example, when $W = 10,000$, to broadcast Message (II) to the WSN costs 1.45 KJ, give $N = 512$. And as N grows to $8,192$, the broadcast cost quickly increases to 2.17 KJ. At the same time, the energy cost on Message (III) is independent to N and is at most 1.11 KJ, which is less than 50% of the former. On the other hand, we see that the Merkle hash tree based scheme outperforms of the certificate-based scheme, when N is no more than 512.

Energy consumption on computation: In this subsection, we evaluate the computation overhead of the proposed schemes also in terms of energy consumption. In the certificate-based scheme, the computation overhead is mainly due to the verification of two ECDSA signatures. In the Merkle hash tree based scheme, the computation overhead is due to the verification of one ECDSA signature and a number of hash operations. And in the ID-based scheme, the computation cost is due to the verification of the ID-based signature.

We now study the energy consumptions of these operations. Assume $|p| =$ 512-bit, we use the following method to quantify the computation time and energy consumption of the Tate pairing used in verifying the ID-based signature. We assume that the sensor CPU is a low-power high-performance 32-bit Intel PXA255 processor at 400 MHz. The PXA255 has been widely used in many sensor products such as Sensoria WINS 3.0 and Crossbow Stargate. According to [20], the typical power consumption of PXA255 in active and idle modes are 411 and 121 mW, respectively. It was reported in [21] that it takes 752

ms to compute the Tate pairing with the similar parameters as ours on a 32-bit ST22 smartcard microprocessor at 33 MHz. Therefore, the computation of the Tate pairing on PXA255 roughly needs $33/400 \times 752 \approx 62.04$ ms, and the energy consumption E_p is approximately 25.5 mJ. Then, to verify the ID-based signature requires one exponentiation in $\mathbb{G}_2$, one hash function evaluation and two evaluations of the Tate pairing. As noted in [19], the pairing evaluation by far takes the most running time of a signature verification operation. Thus, for the sake of simplicity, we use energy consumed on pairing evaluations to approximate that of the signature verification, which ranges from E_p to $2E_p$. Furthermore, it was reported in [12] that it takes 92.4 ms to verify a ECDSA-160 signature with the similar parameters on a 32-bit ARM microprocessor at 80 MHz. Using the same estimation method, we can obtain the energy consumption roughly as 7.6 mJ. Similarly, we omit the energy cost on the hash operations and use 7.6 mJ as the energy cost regarding verification of an ECDSA-160 signature.

Fig. 3 illustrates the energy consumption on computation when the message is broadcast under different message forms. Several conclusions can be drawn from Fig. 3. First, for message broadcast, energy cost on propagation is much higher than that of computation. For example, when $W = 10,000$, the energy cost on computation is 510 J, while it is 1,110 J on propagation. Second, The ID-based scheme incurs a much higher computation cost as compared to the remaining schemes. When we consider energy cost on both computation and propagation, the ID-based scheme is much more energy inefficient except when N is very large. Also observe that more efficient broadcast techniques other than the simple flooding method are used for message broadcast in practice. This further obsoletes the choice of ID-based scheme. Third, when N is less than 500, the Merkle hash tree based scheme is the overall best choice, considering both communication and computation cost. Fourth, when N is large, it still remains to find a satisfying scheme when is computational and communicational efficient at the same time. We leave this as our future work.

6 Concluding Remarks

In this paper, we first identified the problem of multisender broadcast authentication in WSNs. We pointed out that symmetric-key based solutions such as μTESLA are insufficient for this problem by identifying a serious security vulnerability inherent to these schemes: the delayed authentication of the messages can lead to severe DoS attacks, due to the stringent energy and bandwidth constraints in WSNs. We then came up with several effective PKC-based schemes to address the proposed problem. Both computational and communication costs are minimized. We further analyzed both the performance and security resilience of the proposed schemes. A quantitative energy consumption analysis was given in detail. We believe that this paper can serve as the start point towards fully solving the important multisender broadcast authentication problem in WSNs.

Acknowledgement: This work was supported in part by a research grant from AirSprite Technologies, Inc., Northboro, MA, USA.

References

1. I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks, IEEE Communications Magazine," Vol. 40, No. 8, pp. 102-116, 2002.
2. I. Akyildiz and I. Kasimoglu, "Wireless sensor and actor networks: research challenges," Ad Hoc Networks 2(4): 351-367 (2004)
3. A. Perrig, R. Szewczyk, V. Wen, D. Culler, and D. Tygar, "SPINS: Security protocols for sensor networks," in Proc. of MobiCom'01, July 2001.
4. D. Liu and P. Ning, "Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks," in Proc. of NDSS'03, pp.263-276
5. D. Liu and P. Ning, "Multi-level mTESLA: Broadcast authentication for distributed sensor networks," ACMTransactions in Embedded Computing Systems (TECS), vol.3, no.4, 2004.
6. D. Liu, P. Ning, S. Zhu, and S. Jajodia, "Practical Broadcast Authentication in Sensor Networks," in Proc. of MobiQuitous 05, July 2005.
7. Y. Hu, A. Perrig, and D. Johnson, "Packet Leashes: A Defense against Wormhole Attacks in Wireless Ad Hoc Networks," In proceedings of INFOCOM, 2003.
8. K. Ren, W. Lou, and Y. Zhang, "LEDS: Providing Location-aware End-to-end Data Security in Wireless Sensor Networks," In Proc. of IEEE INFOCOM'06.
9. A. Wander, N. Gura, H. Eberle, V. Gupta, and S. Shantz. "Energy Analysis of Public-Key Cryptography on Small Wireless Devices," IEEE PerCom, March 2005.
10. W. Du, R. Wang, and P. Ning "An Efficient Scheme for Authenticating Public Keys in Sensor Networks," In Proceedings of MobiHoc, pp58-67, 2005.
11. Crossbow Technology Inc, http://www.xbow.com/, 2004.
12. M. Aydos, T. Yanik, and C. K. Koc. "An high-speed ECC-based wireless authentication protocol on an ARM microprocessor," In proceedings of the 16th Annual Computer Security Applications Conference, pp.401-409, 2000.
13. R. Merkle, "Protocols for public key cryptosystems," in Proceedings of the IEEE Symposium on Research in Security and Privacy, Apr 1980.
14. Y. Zhang, W. Liu, W. Lou, and Y. Fang, "Location based security mechanisms in wireless sensor networks," IEEE JSAC, Special Issue on Security in Wireless Ad Hoc Networks, vol. 24, no. 2, pp. 247-260, Feb. 2006.
15. A. Shamir, "Identity based cryptosystems and signature schemes," in Proc. CRYPTO'84, ser. LNCS, vol. 196. Springer-Verlag, 1984, pp. 47.53.
16. NIST, "Digital hash standard," Federal Information Processing Standards PUBlication 180-1, April 1995.
17. R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Commun. ACM 21(2), 120C126 (1978)
18. National Institure of Standards and Technology: Proposed Federal Information Processing Standard for Digital Signature Standard (DSS). Federal Register, vol. 56, no. 169, pp. 42980C42982 (1991)
19. F. Hess, "Efficient identity based signature schemes based on pairings," in Proc. SAC'02, St. John's, Newfoundland, Canada, Aug. 2002.
20. "Intel PXA255 Processor Electrical, Mechanical, and Thermal Specification," http://www.intel.com/design/pca/applicationsprocessors/manuals/278780.h
21. G. Bertoni, L. Chen, P. Fragneto, K. Harrison, and G. Pelosi1, "Computing tate pairing on smartcards," White Paper, STMicroelectronics, 2005. Available: http://www.st.com/ stonline/products/families/smartcard/ast ibe.htm
22. K. Lorincz, et al., "Sensor Networks for Emergency Response: Challenges and Opportunities," In IEEE Pervasive Computing, Special Issue on Pervasive Computing for First Response, 2004.