Encyclopedia on Ad Hoc and Ubiquitous Computing, edited by Dharma Agrawal and Bin Xie, World Scientific, 2009

Chapter 15

Effective Multi-user Broadcast Authentication in Wireless Sensor Networks

Kui Ren[1], Wenjing Lou[2], and Yanchao Zhang[3]
[1]Dept. of ECE, Illinois Institute of Technology, Chicago, IL 60616
kren@ece.iit.edu
[2]Dept. of ECE, Worcester Polytechnic Institute, Worcester, MA 01609
wjlou@ece.wpi.edu
[3]Dept. of ECE, New Jersey Institute of Technology, Newark, NJ 07102
yczhang@njit.edu

Abstract—Broadcast authentication is a critical security service in wireless sensor networks (WSNs), as it allows the mobile users of WSNs to broadcast messages to multiple sensor nodes in a secure way. Previous solutions on broadcast authentication are mostly symmetric-key-based solutions such as µTESLA and multilevel µTESLA. These schemes are usually efficient; however, they all suffer from severe energy-depletion attacks resulted from the nature of delayed message authentication. Being aware of the security vulnerability inherent to existing solutions, we present several efficient public-key-based schemes in this chapter to achieve immediate broadcast authentication with significantly improved security strength. Our schemes are built upon the unique integration of several cryptographic techniques, including the Bloom filter, the partial message recovery signature scheme and the Merkle hash tree. We prove the effectiveness and efficiency of the proposed schemes by a comprehensive quantitative analysis of their energy consumption regarding both computation and communication.

Index: Security, Wireless Sensor Networks, Broadcast Authentication, Multi-user

15.1 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 [2], [3], [31]. In these applications, WSNs are expected to process, store, and provide the sensed data to the network users upon their demands [20]. As the most common communication paradigm, the network users are expected to issue the queries to the network in order to obtain the information of their interest. Furthermore, in wireless sensor and actuator networks [3], the network users may need to issue their commands to the network (probably based on the information they received from the network). In both cases, there could be a large number of users in the WSNs, which might be either mobile or static; and the users may use their mobile clients to query or command the sensor nodes from anywhere in the WSN. Obviously, broadcast/multicast 1 operations are fundamental to the realization of these network functions. Hence, it is also highly important to ensure broadcast authentication for the security purpose.

Broadcast authentication in WSNs was first addressed by μTESLA [27]. In μTESLA, users of WSNs are 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 I_1. During time interval I_2, the sink broadcasts K_1, and sensor nodes verify $h(K_1) = K_0$.The authenticity of messages received during time interval I_1 are 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 [17] to overcome the length limit of the hash chain. Most recently, μTESLA is also extended in [18] to support multiuser scenario but the scheme assumes that each sensor node only interacts with a very limited number of users.

It is generally held that μTESLA-like schemes have the following shortcomings when applied to multi-hop large scale WSNs 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 [12], where messages

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

could be forged due to the propagation delay of the disclosed keys. However, here we point out a much more 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 food the whole network arbitrarily. All the adversary has to do is to claim that the flooding messages belong to the current time interval which should be buffered for authentication until the next time interval. Since wireless transmission is very expensive in WSNs, and WSNs are extremely energy constrained, the ability to food the network arbitrarily could cause devastating Denial of Service (DoS) attacks. Moreover, this type of energy-depletion DoS attacks become more devastating in multiuser scenario as the adversary now can have more targets and hence more chances to generate bogus messages without being detected. Obviously, all these attacks are due to delayed authentication of the broadcast messages. In [12], 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, multiuser broadcast authentication still remains a wide open problem in WSNs.

Observing that symmetric-key-based broadcast authentication schemes such as μTESLA are insufficient for WSNs, we resort to public key cryptography (PKC) for more effective solutions in this chapter. We address multiuser broadcast authentication problem in WSNs by designing PKC-based solutions with minimized computational and communication costs.

Objectives of the chapter: We focus on providing multi-user broadcast authentication in WSNs, where the broadcast messages are initiated by a number of network users. Please note that the network users in this chapter refer to personnel or devices that use the WSN; they are not sensor nodes. On the one hand, we aim to achieve immediate message authentication and resist DoS attacks in the presence of both user revocation and node compromise. On the other hand, we want to optimize both computational and communication costs.

Overview of the chapter: In this chapter, we propose four different public-key-based approaches and provide in-depth analysis of their advantages and disadvantages. In all the four approaches, the users are always authenticated through their public keys. We first propose a straightforward certificate-based approach and point out its high energy inefficiency with respect to both communication and computation costs. We then propose a direct storage based scheme, which has high efficiency but suffers from the scalability problem. A

Bloom filter based scheme is further proposed to improve the memory efficiency over the direct storage based scheme. Further techniques are also developed to increase the security strength of the proposed scheme. Lastly, we propose a hybrid scheme to support a larger number of network users by employing the Merkle hash tree technique. We give an in-depth quantitative analysis of the proposed schemes and demonstrate their effectiveness and efficiency in WSNs in terms of energy consumption.

Organization of the chapter: The remaining part of this chapter is as follows: In Section II, we introduce the cryptographic mechanisms to be used. Section III presents the system assumption, adversary model, and security objectives. In Section IV, we introduce two basic schemes. We next propose two advanced schemes and detail the underlying design logic in Section V. In Section VI, we analyze the performance and security strength of the proposed schemes. Section VII then discusses further enhancements of the proposed schemes. Finally, Section VIII is the future work and Section IX is the conclusion.

15. 2 Background

15.2.1 Digital Signature

A digital signature algorithm is a cryptographic tool for generating non-repudiation evidence, authenticating the integrity as well as the origin of a signed message. In a digital signature algorithm, a signer keeps a private key secret and publishes the corresponding public key. The private key is used by the signer to generate digital signatures on messages and the public key is used by anyone to verify signatures on messages. The digital signature algorithms mostly used are RSA [32] and DSA [26]. ECDSA is referred to Elliptic Curve Digital Signature Algorithm [11]. While RSA with 1024-bit keys (RSA-1024) provides the currently accepted security level, it is equivalent in security strength to ECC with 160-bit keys (ECC-160). Hence, for the same level of security strength, ECDSA uses a much short key size and hence has a short signature size (320-bit).

15.2.2 The Bloom Filter and Counting Bloom Filter

A Bloom filter is a simple space-efficient randomized data structure for representing a set in order to support membership queries [23]. A Bloom filter for representing a set $S = S_1, S_2, \ldots, S_n$ of n elements is described by a vector υ of m bits, initially all set to 0. A Bloom filter uses k independent hash functions $h_1, \ldots, h_k$ with range $0, \ldots, m\text{-}1$ which map each item in the universe to a random number uniform over $[0, \ldots, \mathrm{m} - 1]$. For each element $s \in S$, the bits $h_i(s)$ are set to 1 for $1 \le i \le k$. Note that a bit of υ can be set to 1 multiple times. To check if an item x is in S, we check whether all bits $h_i(x)$ are set to 1. If not, x is not a member of S for certain, that is, no false negative error. If yes, x is assumed to be in S. A Bloom filter may yield a false positive. It may suggest that an element x is in S even though it is not. The probability of a false positive for an element not in the set can be calculated as follows. After all the elements of S are hashed into the Bloom filter, the probability that a specific bit is still 0 is $(1 - 1/m)^{kn} \approx e^{-kn/m}$. The probability of a false positive f is then $(1-(1-1/m)^{kn})^k \approx (1 - e^{-kn/m})^k$ .We let $f = (1-p)^k$. From now on, for convenience, we use the asymptotic approximations p and f to represent, respectively, the probability that a bit in the Bloom filter is 0 and the probability of a false positive. Let $p = e^{-kn/m}$.

The counting Bloom filter is a variation of the Bloom filter, which allows member deletion. In the counting Bloom filter, each entry in the Bloom filter is not a single bit but a small counter that tracks the number of elements that have hashed to that location [10]. When an element is deleted, the corresponding counters are decremented. To avoid overflow, counters must be chosen large enough [10].

15.2.3 The Merkle Hash Tree

A Merkle Tree is a construction introduced by Merkle in 1979 to build secure authentication schemes from hash functions [22]. It is a tree of hashes where the leaves in the tree are hashes of the authentic data values $n_1, n_2, \ldots, n_w$. Nodes further up in the tree are the hashes of their respective children. For instance, assuming that $w = 4$ in Fig. 15.1, the values of the four leaf nodes are the hashes of the data values, $h(n_i)$, $i = 1, 2, 3, 4$, respectively, under a one-way hash function $h()$ (e.g., SHA-1 [25]). The value of an internal node A is $h_a = h(h(n_1) \| h(n_2))$, and the value of the root node is $h_r = h(h_a \| h_b)$. h_r 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 where N is the number of leaf nodes). For example, a receiver with the authentic hr requests for n_3 and requires the authentication of the received n_3. The source sends the $\text{AAI} :< h_a, h(n_4) >$ to the receiver. The receiver can then verify n_3 by first computing $h(n_3), h_b = h(h(n_3) \| h(n_4))$ 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. The Merkle hash tree can prevent an adversary from sending bogus data to deceive the client. In the earlier example, an adversary impersonating can not send a bogus n_3 to the client without being detected. This is because he can not find h_a and $h(n_4)$ such that $h(h_a \| h(h(n_3) \| h(n_4))) = h_r$ as $h()$ is one-way.

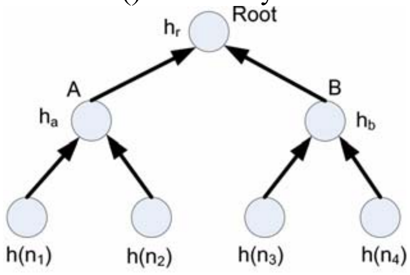

Fig. 15.1 An example of Merkle hash tree

15.3 System Model, Adversary Model, and Design Goals

System Model: In this chapter, we consider a large spatially distributed WSN, consisting of a fixed sink(s) and a large number of sensor nodes. The sensor nodes are usually resource-constrained with respect to memory space, computation capability, bandwidth, and power supply. The WSN is aimed to offer information services to many network users that roam in the network, in addition to the fixed sink(s) [20]. The network users may 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 consist of a number of doctors, nurses, medical equipment (acting as actuators) and so on, in the case of CodeBlue [19], where the WSN is used for emergency medical response. These network users broadcast queries/commands through sensor nodes in the vicinity, and expect the replies that reflect the latest network information. The network users can also communicate with the sink or the backend server directly without going through the WSN if necessary. 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 changes or compromise, and the revocation pattern is not restricted. We also assume that the WSN is loosely synchronized.

Adversary Model: In this chapter, 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, Denial of Service (DoS) attacks such as bogus message flooding, aiming at exhausting constrained network resources, is another important focus of the chapter. 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. However, we do assume that adversary cannot compromise an unlimited number of sensor nodes.

Design Motivation: When μTESLA was proposed, sensor nodes were assumed to be extremely resource-constrained, especially with respect to computation capability, bandwidth availability, and energy supply [27]. Therefore, PKC was thought to be too computationally expensive for WSNs, though it could provide much simpler solutions with much stronger security resilience. At the same time, the computationally efficient onetime signature schemes are also considered unsuitable for WSNs, as they usually involve intense communications [27]. However, recent studies [9], [29], [35] showed that, contrary to widely held beliefs, PKC with even software implementations only is very viable on sensor nodes. For example [35], Elliptic Curve Cryptography (ECC) signature verification takes 1.61s with 160-bit keys on ATmega128 8MHz, a processor used in current Crossbow motes platform [8]. Furthermore, the computational cost is expected to fall faster than the cost to transmit and receive. For example, ultra-low-power microcontrollers such as the 16-bit Texas Instruments MSP430 [34] can execute the same number of instructions at less than half the power required by the 8-bit ATmega128L. The benefits of transmitting shorter ECC keys and hence shorter messages/signatures generation sensor nodes are expected to combine ultra-low power circuitry with so-called power scavengers such as Heliomote [15], which allow continuous energy supply to the nodes. At least 8-20μW of power can be generated using MEMS-based power scavengers [4]. Other solar-based systems are even able to deliver power up to 100mW for the MICA Motes [15], [16]. These results indicate that, with the advance of fast growing technology, PKC is no longer impractical for WSNs, though still expensive for the current generation sensor nodes, and its wide acceptance is expected in the near future [9].

Design Goals: Our security goal is straightforward: all messages broadcasted by the network users of the WSN should be authenticated so that the bogus ones inserted by the illegitimate users and/or compromised sensor nodes can be efficiently rejected/filtered. We also focus on minimizing the overheads of the security design. Especially, energy efficiency (with respect to both communication and computation) and storage overhead are given priority to cope with the resource-constrained nature of WSNs.

15.4 The Basic Schemes

We explore the PKC domain for the possible solutions to multiuser broadcast authentication in WSNs. The PKC-based solutions realize immediate message authentication and thus can overcome the delayed message authentication problem present in μTESLA-like schemes.

15.4.1 The Certificate-Based Authentication Scheme (CAS)

CAS works as follows. Each user (not a sensor) 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 ECDSA [11]. Note that in all our designs, we do not require sensors to have public/private key pairs for themselves. To prove the user's ownership over his public key, the sink[2] is also equipped with a public/private key pair and serves as the certification authority (CA). The sink issues each user a public key certificate, which, to its simplest form, consists of the following content $Cert_{U_{ID}} = U_{ID}, PK_{U_{ID}}, ExpT, SIG_{SK_{Sink}}\{h(U_{ID}||ExpT||PK_{U_{ID}})\}$, where U_{ID} denotes the user's ID, $PK_{U_{ID}}$ denotes its public key, $ExpT$ denotes certificate expiration time, and SIG is a signature over $h(U_{ID}||ExpT||PK_{U_{ID}})\}$ with SK_{Sink}. Hence, a broadcast message is now of the form as follows:

$$< M, tt, SIG_{SK_{U_{ID}}} || \{h(U_{ID} || tt || M)\}, Cert_{U_{ID}} > \qquad \text{(I)}$$

Here, M denotes the broadcast message and tt denotes the current time. For the purpose of message authentication, sensor nodes are preloaded with PK_{Sink} before the network deployment; and message verification contains two steps: the user certificate verification and the message signature verification.

CAS suffers from two main drawbacks. First and foremost, it is not efficient in communication, as the certificate has to be transmitted along with the message across every hop as the message propagates in the WSN. A large per message overhead will result in more energy consumption on every single sensor node. In

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

CAS, the per message overhead is as high as $|tt| + |SIG_{SK_{U_{ID}}}\{h(U_{ID} \| M)\}| + |Cert_{U_{ID}}| = 128$ bytes. As in [35], the user certificate is at least 86 bytes, when ECDSA-160 [11] is used. Here, we assume that tt and U_{ID} are both two bytes, in which case the scheme supports up to 65,535 network users. Moreover, $|SIG_{SK_{U_{ID}}}\{h(U_{ID} \| M)\}| = 40$ bytes, when ECDSA-160 [11] is assumed. Second, to authenticate each message, it always takes two expensive signature verification operations. This is because the certificate should always be authenticated in the first place.

15.4.2 The Direct Storage Based Authentication Scheme (DAS)

One way to reduce the per message overhead and the computational cost is to eliminate the existence of the certificate. A straightforward approach is then to let sensor nodes simply store all the current users' ID information and their corresponding public keys. In this way, a broadcast message now only contains the following contents:

$$< M, tt, SIG_{SK_{U_{ID}}}\{h(U_{ID} \| tt \| M)\}, U_{ID}, PK_{U_{ID}} > \qquad \textit{(II)}$$

Verifying the authenticity of a user public key is reduced to finding out whether or not the attached user/public key pair is contained in the local memory. Upon user revocation, the sink simply sends out ID information of the revoked user, and every sensor node deletes the corresponding user/public key pair in its memory.

The drawbacks of DAS are obvious. Given a storage limit of 5 KB, only 232 users can be supported at most; even with a memory space of 19.5 KB, DAS can only support up to 1, 000 users. At the same time, CAS can support up to 2, 560 users given the same storage limit 5 KB. The reason is that in CAS only the ID information of the revoked users is stored by the sensor nodes. Therefore, DAS is neither memory efficient nor scalable. However, the advantage of DAS is also significant as compared to CAS. It successfully reduces the per message overhead down to $|tt| + |SIG_{SK_{U_{ID}}}\{h(U_{ID} \| M)\}| + |U_{ID}| + |PK_{U_{ID}}| = 64$ bytes. The above analysis clearly shows that more advanced schemes are needed other than DAS and CAS. And the direction to seek is to improve storage efficiency while retaining or further reducing the per message overhead.

15.5 The Advanced Schemes

In this section, advanced schemes are proposed to achieve both storage efficiency and communication efficiency simultaneously. The proposed schemes significantly outperform the previous basic schemes through a novel integration of several cryptographic techniques.

15.5.1 The Bloom Filter Based Authentication Scheme (BAS)

System Preparation: The sink generates the public keys for all network users, and constructs the set:

$$S = \{< U_{ID_1}, PK_{U_{ID_1}} >, < U_{ID_2}, PK_{U_{ID_2}} >, \ldots\},$$

Where $\#\{S\} = N$, and $\#\{\}$ denotes the cardinality of the set. Using the Bloom filter, the sink can apply k system-wide hash functions (cf. Section II.B) to map the elements of S (each with $L+2$ bytes, that is, $|U_{ID}| = 2$ bytes, and $|PK_{U_{ID}}| = L$ bytes to an m -bit vector υ with $\upsilon = v_0 v_1 \ldots v_{m-1}$ where we have $m < N(L = 2)$ to reduce the filter size and $m > kN$ to retain a small probability of a false positive. These k hash functions are known by every node and the sink. For each v_i, $i \in [0, m-1]$, we have

$$v_i = \begin{cases} 1, & \text{if } \exists l \in [1,k], j \in [1,N], \\ & \text{s.t. } h_l(U_{IDj} || PK_{U_{IDj}}) = i \\ 0, & \text{otherwise} \end{cases}$$

Additionally, the sink constructs a counting Bloom filter υ of $m * c$ bits with $\overline{\upsilon} = \overline{v}_0 \overline{v}_1 \ldots \overline{v}_{m-1}$, where each $\overline{v}_i$, $i \in [0, m-1]$ is a c-bit counter, i.e., $|\overline{v}_i| = c$ bits. The value of $\overline{v}_i$ is determined as follows:

$$v_i = \#\{(ID_j, PK_{U_{IDj}}) \mid h_l(U_{IDj} \,||\, PK_{U_{IDj}}) = i, \\ \text{for } \exists l \in [1,k], j \in [1,N]\}.$$

And $c = \lceil \log_2(\max(\overline{v}_i, i \in [0, m-1])) \rceil$ bits, which is usually of 4 bits for most applications [10]. The above operations are illustrated in Fig. 15.2 The sink finally preloads each sensor node with υ (not including $\overline{\upsilon}$), as well as the sink's public key and the common domain parameters of the ECDSA signature scheme.

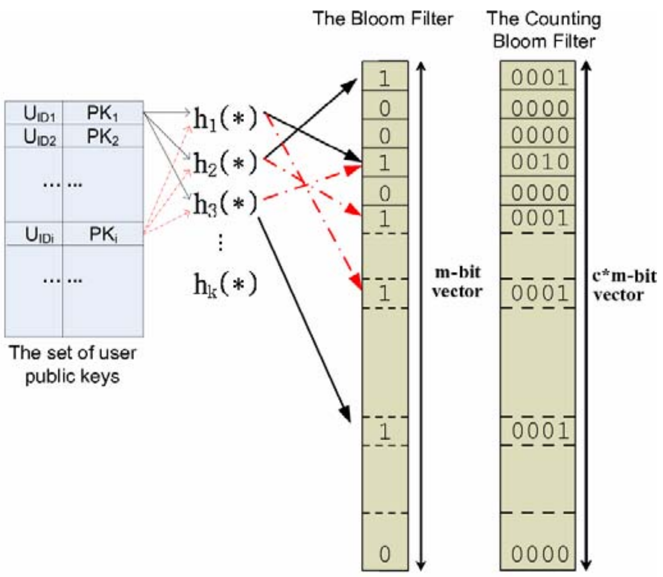

Fig. 15.2 An example of the Bloom filter and Counting Bloom Filter

Message Signing and Authentication: Let $PK_{U_{ID}} = sG$, be the public key of user U_{ID}, where s is the private key of the signer, and G is the generator of a subgroup of an elliptic curve group of order r. Let $S_K(.)$ be a symmetric key cipher such as AES. To broadcast a message M ($|M| \geq 10$ bytes), U_{ID} takes the steps below following [24], a variant of ECDSA with the partial message recovery property:

- Concatenate $< M \| tt \| U_{ID} >$, and break it into two parts M_1 and M_2, where $|M| \leq 10$ bytes.
- Generate a random key pair $\{u, V\}$, where $u \in [1, r-1]$, $V = uG = (x_{1,} y_1)$, and $(x_1 \bmod r) \neq 0$.
- Encode-and-hash V into an integer I [24].
- Form F_1 from M_1 by adding the proper redundancy [1].
- Compute $c = (I + F_1) \bmod r$, and make sure that $c \neq 0$ or repeat the above steps otherwise.
- Compute $F_2 = h(M_2)$, and $D = u^{-1}(F_2 + sC) \bmod r$
- Repeat all the above steps if $D = 0$; Output the signature as $< C, D >$ otherwise.

Then, U_{ID} broadcasts

$$< M_2, C, D, PK_{U_{ID}} >, \tag{III}$$

where tt and U_{ID} are parts of M_2. And this is the known simplest message format that can be achieved using PKC[3]. Now, upon receiving a broadcast message (not from the sink), a sensor node checks the authenticity of the message in two steps. First, it checks the authenticity of the corresponding public key by verifying its membership in S. To do so, the sensor node checks whether $\upsilon[h_l(U_{ID} \| PK_{U_{ID}})] \overset{?}{=} 1$, $l \in [1,k]$, and a negative result will lead to the discarding of the message. We note that here a false positive may happen due to the probabilistic nature of the Bloom filter, but only with a very small (negligible) probability when appropriate parameters are chosen as we will analyze later. Second, it verifies the attached signature as follows:

- Discard the message if $C \notin [1, r-1]$ or $D \notin [1, r-1]$.
- Compute $F_2 = h(M_2)$, $H = D^{-1} \bmod r$, and $H_1 = F_2 H \bmod r$
- Compute $H_2 = CH \bmod r$, and $P = H_1 G + H_2 PK_{U_{ID}}$.
- Discard the message if $P = O$.
- Encode-and-hash P into an integer I [24] and compute $F_1 = C - I \bmod r$.
- Discard the message if the redundancy of F_1 is incorrect.
- Otherwise accept M_1 (obtained from F_1) and the signature and reconstruct $M \| tt \| U_{ID} = M_1 \| M_2$.

User Revocation/Addition: To revoke a user, say U_{ID_j}, the sink follows the steps below:

- First, it hashes $h_l(U_{ID_j}) \| PK_{U_{ID_j}}) = i$ and decreases $\overline{v}_i$ by 1. It repeats this operation for all $h_l, l \in [1,k]$.
- From the updated counting Bloom filter $\overline{\upsilon}$, the sink obtains the corresponding updated Bloom filter υ' with $\upsilon' = v'_0 v'_1 ... v'_{m-1}$. Here, $v'_i = 1$ only when $\overline{v}_i \geq 1$, and $v_i' = 0$ otherwise.
- The sink further calculates $\upsilon_\Delta = \upsilon' \oplus \upsilon$ and deletes υ afterwards. Here $\oplus$ denotes bitwise exclusive OR operation. Obviously, υ_Δ is an m-bit vector with at most k bits set to 1. Hence υ_Δ can be simply represented by enumerating its 1-valued bits, requiring $\overline{k}\lceil \log_2 m \rceil$ bits for indexing ($\overline{k} \leq k$). This representation is efficient for a small $\overline{k}$ as will be analyzed in Section VI.B.

[3] The claim is true only when ID-based cryptography [33] is excluded from consideration, in which case the user's ID is also his public key. Furthermore, the shortest signature size possibly obtained from pairing is around 22 bytes [7], which is shorter than 40 bytes obtained from ECDSA. However, to apply a pairing-based scheme (i.e., an ID-based signature or short signature) on sensor nodes, the known reachable signature size has to be 84 bytes, even when a 32bit microprocessor can be used [36]. And the energy cost is also multiple times higher than that of an ECDSA-160 signature.

- The sink finally broadcasts υ_Δ after signing it. The message format follows (III) but with the sink's public key omitted, as every sensor already has it.
- Upon receiving and successfully authenticating the broadcast message, every sensor node updates its own Bloom filter accordingly, that is, if $v_{\Delta,i} = 1$ then $v_i = 0$, $i \in [0, m-1]$.

BAS also supports simultaneous multiuser revocation. Suppose that N_{rev} users are revoked simultaneously. The sink follows the same manner to construct υ_Δ with $\bar{k}$ bits set 1. Now we have $\bar{k} \le kN_{rev}$. Furthermore, the compressed message for representing υ_Δ now could achieve $mH(p)$ bits theoretically, where $H(p) = -p\log_2 p - (1-p)\log_2(1-p)$ is the entropy function and $p = (1 - 1/m)^{\bar{k}}$ is the probability of each bit being 0 in υ_Δ . As pointed out in [23], using arithmetic coding technique can efficiently approach this lower bound.

BAS supports dynamic user addition in two ways. First, it enables a later binding of network users and their (ID, public key) pairs. In this approach, the sink may generate more (ID, public key) pairs than needed during system preparation. When a new network user joins the WSN, it will be assigned an unused ID and public key pair by the sink. Second, BAS could add new network users after the revocation of old members. This approach, however, could only add the same number of new users as that of the revoked. This requirement ensures that the probability of a false positive never increases in BAS. To do so, the sink updates its counting Bloom filter by hashing the new user's information into the current Bloom filter. The sink then obtains a υ_Δ in the same way as in the revocation case, and broadcasts it after compression. This time, if $v_{\Delta,i} = 1$, sensor nodes will set $v_i = 1, i \in [0, m-1]$ to update their current Bloom filters.

15.5.2 Minimize the Probability of a False Positive

Since the Bloom filter provides probabilistic membership verification only, it is important to make sure that the probability of a false positive is as small as possible.

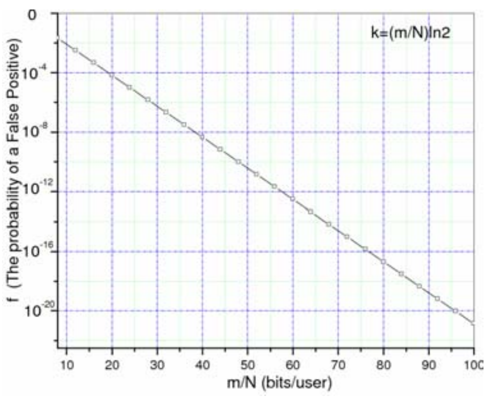

Fig. 15.3 The minimum probability of a false positive regarding $\frac{m}{N}$

Theorem 1: Given the number of network users N and the storage space m bits for a single Bloom filter, the minimum probability of a false positive f that can be achieved is 2^{-k} with $k = \frac{m}{N}\ln 2$, that is,

$$f = (0.6185)^{\frac{m}{N}}.$$

Proof: since $f = (1-(1-\frac{1}{m})^{kN})^k \approx (1-e^{-kN/m})^k$, we then have $f = e^{k\ln(1-e^{-kN/m})}$. Let $g = k\ln(1-e^{-kN/m})$. Hence, minimizing f is equivalent to minimizing g with respect to k. We find

$$\frac{dg}{dk} = \ln(1-e^{-kN/m}) + \frac{kN}{m}\frac{e^{-kN/m}}{1-e^{-kN/m}}$$

It is easy to check that the derivative is 0 when $k = \frac{m}{N}\ln 2$. And it is not hard to show that this is a global minimum [23]. Note that in practice, k must be an integer.

Fig. 15.3 shows the probability of a false positive f as a function of $\frac{m}{N}$ i.e., bits per element. We see that f decreases sharply as $\frac{m}{N}$ increases. When $\frac{m}{N}$ increases from 8 to 96 bits, f decreases from $2.1*10^{-2}$ to $9.3*10^{-21}$. Obviously, f determines the security strength of our design. For example, when $\frac{m}{N} = 92$ bits, the adversary has to generate around $2^{63.8}$ public/private key pairs on average before finding a valid one to pass the Bloom filter. This is almost computationally infeasible, at least within the lifetime of the WSN (usually at most several years). However, when $m / N = 64$ bits, the adversary is now expected to generate around $2^{44.4}$ public/private key pairs before finding a valid pair. The analysis below shows the time and cost of the attack. To generate a public/private key pair in ECDSA-160, a point multiplication operation has to be performed, for which the fastest known implementation speed is 0.21ms through a specialized FPGA design [14]. Suppose the adversary could afford 100,000 such FPGAs, which would cost no less than one million dollars. Then, by executing 100,000 FPGAs simultaneously, to generate one valid key pair still takes 13.2 hours roughly. With the above analysis, we suggest to select the value of f carefully according to the security requirements of the different types of applications. Given a highly security sensitive military application, we suggest that f should be no larger than $6.36*10^{-20}$, i.e., $m / N \geq 92$ bits. On the other hand, when the targeted applications are less security sensitive as in the civilian scenario, we can tolerate a larger f . This is because the adversary is now generally much less resourceful as compared to the former case.

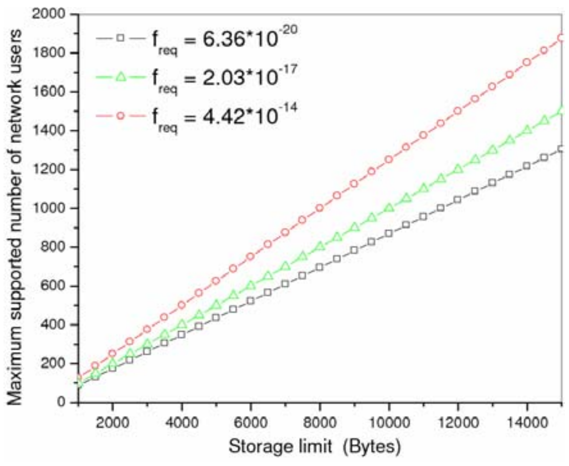

Fig. 15.4 Maximum supported number of network users with respect to storage limit

15.5.3 Maximum Number of Network Users Supported

It is important to know how many network users can be supported in BAS so that the WSN can be well planned. The following theorem provides the answer.

Theorem 2: Given the storage space m bits for a single Bloom filter and the required probability of a false positive $f_{req}(f_{req} \in (0,1))$, the maximum number of network users that can be supported is $\frac{-m(\ln 2)^2}{\ln f_{req}}$, that is,

$$N \le \frac{-0.4805m}{\ln f_{req}}$$

Proof: Since the minimal probability of a false positive $f = 2^{-k}$ is achieved with $k = \frac{m}{N}\ln 2$, we have $f_{req} = 2^{-\frac{m}{N}\ln 2}$. Then, we can easily get $N = \frac{-m(\ln 2)^2}{\ln f_{req}}$ in this case; and this is the maximum number of users that can be supported given f_{req} and m. □

Fig. 15.4 illustrates the maximum supported number of network users as a function of the storage limit. Fig. 15.4 shows that BAS supports up to 1,250 users when $f_{req} = 4.42 * 10^{-14}$, 1,000 users when $f_{req} = 2.03 * 10^{-17}$ and 869 users when $f_{req} = 6.36 * 10^{-20}$, for a storage space of 9.8 KB. Obviously, BAS also allows tradeoff between the maximum supported number of network users and the probability of a false positive given a fixed storage limit.

15.5.4 Supporting More Users using the Merkle Hash Tree: The Hybrid Authentication Scheme (HAS)

Through the above analysis, we know that the maximum supported number of network users is usually limited given the storage limit and the probability of a false positive. For example, if $f_{req} = 6.36 * 10^{-20}$ and the storage limit is 4.9 KB, the maximum number of users supported by BAS is 434. Therefore, an additional mechanism has to be employed to support more users when necessary. HAS achieves this goal by employing the Merkle hash tree technique, which trades the message length for the storage space. That is, by increasing the per message overhead, HAS can support more network users. Specifically, HAS works as follows.

The sink first calculates the maximum number of users supported in case of BAS according to the given storage limit and the desired probability of a false positive. It then collects all the public keys of the current network users and constructs a Merkel hash tree. In fact, the sink constructs N leaves with each leaf corresponding to a current user of the WSN. For our problem, each leaf node contains the binding between the corresponding user ID and his public key, that is, $h(U_{ID}, PK_{U_{ID}})$. The values of the internal nodes are determined by the method introduced in Section II.C. The sink further prunes the Merkle hash tree into a set of equal-sized smaller trees. We denote the value of the root node of a small hash tree as h_r^i, $i = 1,...,|S|$, where $|S|$ equals the maximum number of supported users the sink calculates in BAS.

Next, the sink constructs a Bloom filter υ following the same way as described in the last section. The difference is that now the member set $S = \{h_r^1, h_r^2, ..., h_r^{|S|}\}$. Then, the sink preloads each sensor node with υ. At the same time, each user should obtain its AAI according to his corresponding leaf node's location in the smaller Merkle hash tree. Let T denote all the nodes along the path from a leaf node to the root (not including the root), and A be the set of nodes corresponding to the siblings of the nodes in T. Then, AAI further corresponds to the values associated with the nodes in A. Obviously, AAI is of size $(\overline{L} * \log_2 \frac{N}{|S|})$ bytes, where $\overline{L}$ is the length of the hash values. Upon user revocation, the sink simply updates all the sensor nodes with the ID information of the revoked users. And each node directly stores the revoked IDs as described earlier. Now a message sent by a user U_{ID} is of form

$$< M_2, C, D, PK_{U_{ID}}, AAI_{U_{ID}} > . \qquad \textit{(IV)}$$

Each node verifies the authenticity of a user public key in two steps. First, it calculates the corresponding root node value h_r^i using $AAI_{U_{ID}}$ attached in the message. Second, it checks whether or not the calculated h_r^i is a member of υ stored by itself. By checking Message (IV), we can easily find that HAS doubles the maximum supported number of users as compared to BAS at the cost of 20 more bytes per message overhead, assuming SHA-1 is used [25]. And the number can be further doubled with 40 more bytes per message overhead.

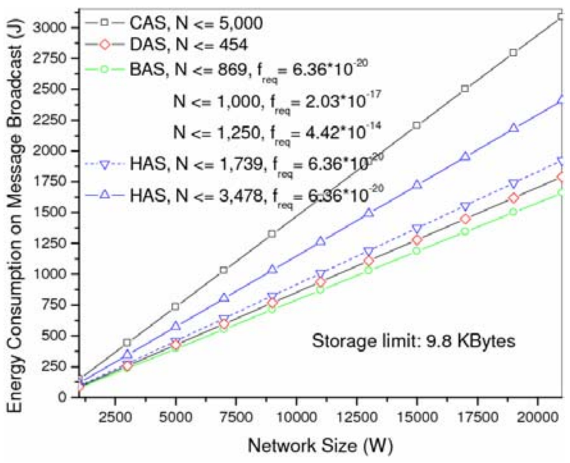

Fig. 15.5 Energy consumption in communication regarding different schemes

15.6 Performance Analysis

In this section, we analyze the performance of BAS and HAS with respect to communicational and computational overheads (in terms of energy consumption), and security strength. We give a quantitative analysis of the schemes and compare them with the other two basic schemes.

15.6.1 Communication Overhead

We study how the message size affects the energy consumption in communication in a WSN. We investigate the energy consumption as the function of the size of the WSN (denoted as W). We denote by E_{tr} the hop-wise energy consumption for transmitting and receiving one byte. As reported in [35], a Chipcon CC1000 radio used in Crossbow MICA2DOT motes consumes 28.6 and $59.2\,\mu\text{J}$ 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 bytes for the payload and 9 bytes for the header [35]. The header, ensuring an 8-byte preamble, consists of source, destination, length, packet ID, CRC, and a control byte [35]. We also assume that $|M| = 20$ bytes.

Then, for BAS, the signature size is still the same as that of ECDSA, but only part of the message now has to be transmitted, with the saving of up to 10 bytes. Therefore, the per message overhead of BAS is 54 bytes, which is 10 bytes less than that of DAS. As Message (III) is 74 bytes, there should be 3 packets in total, among which two of them are 41 bytes, and one is 19 bytes. Therefore, there should be $41*2+19*1+8*3=125$ bytes for transmission (including 8-byte preamble per packet). Hence, the hop-wise energy consumption of message transmission is $125*59.2\,\mu\text{J} = 7.40\,\text{mJ}$; and the energy consumption of message reception is $125*28.6\,\mu\text{J} = 3.58\,\text{mJ}$. For each message broadcast, every sensor node should retransmit the message once and receive ω' times of the same message assuming the blind flooding is *used* [4] Here, ω' denotes node density in terms of the total number of sensor nodes within one unit disc, where a unit disc is a circle area with radius equal to the transmission range of sensor nodes[5]. Hence, the total energy consumption in communication will be $W*(7.4+3.58*\omega')\,\text{mJ}$.

Fig. 15.5 illustrates the energy consumption in communication as a function of W with $\omega'=20$. Clearly, BAS consumes a much lower energy as compared to others. For example, when W=15,000, CAS always costs 2.20 KJ, while BAS costs only 1.18 KJ. The energy saving for a single broadcast can be more than 1,000 J between BAS and CAS. Note that although DAS also consumes much less energy than CAS, DAS only supports up to $1000/22 \approx 454$ users. At the same time, BAS can handle 869 users even when $f_{req} = 6.36*10^{-20}$ CAS handles more users than BAS and DAS, however, at the cost of much higher energy consumption. Moreover, HAS can handle a large number of users but with a much lower energy consumption when compared to CAS. In summary, BAS demonstrates the highest communication efficiency, as well as desirable storage efficiency. From Fig. 15.5, we also find that the energy consumption in communication is the critical cost for WSNs, as a single broadcast of a message of only 20 bytes in length could cost energy on the order of KJ. This also exposes the severe vulnerability of the µTESLA-like schemes, as they allow the adversary to flood the WSN arbitrarily.

15.6.2 User Revocation/Addition Traffic Overhead

Another important performance metric for the broadcast authentication schemes

[4]In an idealized lossless network, blind flooding, i.e., every node always retransmits exactly once every unique message it receives, is wasteful, as individual nodes are likely to receive the same broadcast multiple times. In practice, however, blind flooding is a commonly used technique, as its inherent redundancy provides some protection from unreliable (lossy) wireless networks [21].

[5]We assume a uniform transmission range for all sensor nodes.

is the overhead of the user revocation/addition traffic. As analyzed in Section V.A, BAS requires the sink to broadcast υ_{Δ} upon user revocation/addition. We have shown that in the single user case, υ_{Δ} can be efficiently represented by simply enumerating all its 1-valued bits, the length of which is bounded by $\bar{k}\lceil \log_2 m \rceil$ bits. That is, the per user revocation traffic overhead is upper bounded by $\bar{k}\lceil \log_2 m \rceil$ bits. And the theoretical lower bound obtained from the entropy function is $mH(p)$ bits with $H(p) = -p\log_2 p - (1-p)\log_2(1-p)$ and $p = (1 - \frac{1}{m})^k$. It is not hard to see that the expectation value of $\bar{k}$ is around $k/2$, where $k = \frac{m}{N}\ln 2$. Our simulation shows that $\bar{k}$ is always around $k/2$. Hence, for a given $f_{req} = 6.36 * 10^{-20}$, we will have $\bar{k}\lceil \log_2 m \rceil = 68$ bytes, and $mH(p) \approx 52$ bytes, for $N = 1{,}000$. This implies that the per user revocation traffic υ_{Δ} only ranges from 52 to 68 bytes on average for $N = 1{,}000$, depending on the used coding method[6]. And for $N \le 11{,}000$, υ_{Δ} is at most 80 bytes on average. This overhead is much lower as comparable to that of the µTESLA-like scheme proposed for supporting multiple users [18]. In [18], the per user revocation traffic (i.e., a revocation certificate) is no less than $1 + \lceil \log_2 N \rceil$ hash values, which is 220 bytes for N =1,000, and 300 bytes for N = 11,000, assuming the same hash length of 20 bytes. We further note that in contrast to µTESLA-like schemes, BAS does not require periodic key chain update (for running out of available keys) among users and sensor nodes. This is the advantage inherent to the PKC-based schemes.

15.6.3 Computational Overhead

It was previously widely held that PKC is not suitable in WSNs, as sensor nodes are extremely computation constrained. However, recent studies [9], [35] showed that PKC with only software implementations is very viable on sensor nodes. For example in [35], an ECC signature verification takes 1.61s with 160-bit keys on ATmega128 8MHz processor used in a Crossbow mote. We analyze the computation cost of the proposed schemes to further justify the suitability of PKC-based schemes in WSNs. In all our proposed schemes, the major computational cost is due to the signature verification operation. In the following

[6] We assume that the number of simultaneous network users is always around N.

analysis we omit the cost of other operations such as hash operations and table lookup, as they are negligible as compared to the signature verification operation [35].

In CAS, two ECDSA signature verifications are needed for each broadcast message. In BAS, to verify a message takes $k = \frac{m}{N}\ln 2$ hash operations and one ECDSA signature verification. It was reported in [35] that an ECDSA-160 signature verification operation costs 45.09 mJ on a 8-bit ATmega128L processor running at 4 MHz. If we assume that the sensor CPU is a low-power high-performance 32-bit Intel PXA255 processor, the energy cost can be further minimized. Note that the PXA255 has been widely used in many sensor products such as Sensoria WINS 3.0 and Crossbow Stargate running at 400MHz. According to [13], the typical power consumption of PXA255 in active and idle modes are 411 and 121 mW, respectively. It was reported in [5] that it takes 92.4 ms to verify an ECDSA-160 signature with the similar parameters on a 32-bit ARM microprocessor at 80 MHz. Therefore, the same computation on PXA255 roughly needs $80/400 \times 92.4 \approx 18.48$ ms, and the energy cost is hence around 7.6 mJ. Therefore, we can obtain the computational costs of the proposed CAS and BAS schemes on different sensor platforms[7]. The results are summarized below.

Scheme	ATmega128L	PXA255
CAS	90.18 mJ	15.4 mJ
BAS	45.09 mJ	7.6 mJ

BAS is obviously also more computationally efficient than CAS. Furthermore, when we compare the computational cost with the communication cost on hop-wise message transmission, we can find that both are on the same order, which justifies the suitability of PKC-based schemes in WSNs.

15.6.4 Security Strength

The Bloom filter based public key verification ensures the security strength of the proposed scheme by enabling immediate message authentication. That is, there is no authentication delay on messages being broadcast. Therefore, it is very hard for the adversary to perform network wide flooding in the WSN. As we analyzed

[7] DAS and HAS consume similar amount of energy as BAS does, as they both require one signature verification.

above, by appropriately choosing a suitable value of f_{req}, such as $6.36*10^{-20}$ in military applications, it is infeasible to forge a valid public/private key pair during the lifetime of the WSN. Furthermore, by embedding a time stamp into the message, the message replay attack is also effectively prevented, as WSN is assumed to be loosely synchronized [28]. Therefore, the immediate message authentication capability provided by the proposed schemes can effectively protect the WSN from network wide flooding attacks. This is the most significant security strength over the μTESLA-like schemes, in which network wide flooding attacks are always possible.

Moreover, since the public key operation is expensive, it is also important that sensor nodes can be resistant to the local jamming attacks. Under such attacks, the adversary may simply broadcast random bit strings to the sensor nodes within his transmission range. If these neighbor sensors have to perform the expensive signature verification operation for all received messages, it will be a heavy burden on them. CAS obviously suffers from this type of attacks, as the signature verification operation has to be performed for every received message. However, in both BAS and HAS, such an attack can be effectively mitigated. This is because in both schemes, a sensor node first verifies the authenticity of the attached user public key through hash operations, so it performs signature verification operation for a bogus public key only with a negligible probability (e.g., $6.36*10^{-20}$). As reported in [35], the energy cost of SHA-1 is only 5.9 μJ/byte on a 8-bit ATmega128L processor, while ECDSA-160 could consume 45.09 mJ on signature verification. An adversary may also flood the sensor nodes with forged messages but containing valid user public keys, which can be obtained by eavesdropping the network traffic. In this case, the forged messages can only be discarded after signature verification, and sensor nodes that are physically close to the adversary can thus be abused. We note that this type of attacks is always possible for PKC-based security mechanisms. However, this attack can still be mitigated in BAS by implementing an alert report mechanism. If a sensor node fails to authenticate the received messages multiple times in a row, it will derive that an attack is going on and alert the sink about the attack. The sink further carries out field investigations or other means to detect the adversary and take corresponding remedy actions that are outside the scope of this chapter.

15.7 Further Enhancements

15.7.1 Dealing with Long Messages

The messages broadcast in WSNs are usually short, due to the application specific nature of WSNs. The query or command messages can be less than one hundred bytes. However, there are few cases that long messages may be required to be broadcast in WSNs. For example, the sink may broadcast code images to the sensor nodes for the purpose of retasking WSNs [37]. The size of such code images can be on the order of KB. In this case, it is not desirable to apply the proposed BAS or HAS scheme directly by signing the whole message (i.e., the message hash) only once or signing on every single packet otherwise. This is because of two reasons. First, if we sign the whole message once, then each sensor node can authenticate a message only after it obtains the entire message. That is, the sensor nodes have to buffer a large number of received packets before it can authenticate them. This obviously introduces a severe vulnerability that could result in message flooding attacks. Second, if we sign every packet belonging to the same message, the scheme overheads will increase significantly with respect to both computation and communication. This is because now every packet is attached with a signature, which is 40 bytes in our setting.

Fortunately, several solutions were proposed to solve this problem in the context of code update in WSNs [38], [39]. The first solution is suitable for lossless network environments, which employs off-line hash chain technique to amortize the cost of a single digital signature over multiple packets and allow for incremental message authentication and packet pipelining [38], [40]. The second solution is aimed at tolerating packet losses. This solution makes use of a signed hash tree technique and trades message overhead for potential packet losses [39]. Both solutions can be directly superimposed with BAS and HAS in dealing with long messages. We omit the details of these solutions for space interest.

15.7.2 Reducing the Probability of a False Positive

In [41], a method is introduced to use two families of k hash functions, instead of using one. And an element is in the set if either family gives back all 1s from the filter. The trick is to choose one of the two families of the hash functions adaptively: choose which family of hash functions to use for each element of your set in such a way to keep the number of 1s in the filter as small as possible. In such a way, a smaller false positive probability in the same space can be achieved at the cost of more hashing. This method can reduce the probability of a false positive to the half under certain conditions using the same storage space.

This technique can be exploited by BAS so that we achieve a desirable probability of a false positive with a smaller storage space.

15.7.3 Optimization on Constructing the Merkle Hash Tree

Different types of network users may have different broadcast frequencies in practice. This fact can be exploited by HAS, when supporting a vast number of network users is a must. Instead of pruning the user Merkle hash tree into a set of equal-sized smaller trees, now the tree can be trimmed into the same number but different-sized smaller ones based on user broadcast frequency. The higher the frequency is, the smaller hash tree the user is grouped in. In such a way, the energy efficiency can be improved in the overall sense, as more messages being broadcast containing only smaller AAI sizes. This is similar to the idea introduced in [9].

15.7.4 Using "Fast Forward, Slow React" to Prevent Public Key Forgery Attacks

Since a false positive is still possible though small, the adversary can forge different user public/private key pairs and seek to pass the membership test. In this way, the adversary could possibly pretend to be a valid network user and has its messages authenticated. The sensor nodes, however, will not be able to distinguish the bogus public keys, once a successful guess is made.

On the other hand, a bogus public key can always be deterministically detected by the sink, as the sink keeps the copies of all the user public keys. Furthermore, as the sink is always connected to the WSN as assumed, it also receives the broadcast messages sent by the users. Thus, the sink can always detect a bogus public key that cannot be detected by the sensor nodes through analyzing the received messages. The following "Fast forward, Slow React" policy is designed to leverage this fact and does not affect message propagation efficiency. In "Fast forward, Slow React" policy, each sensor node estimates the round trip time between itself and the sink. We denote this time as Δ_{UID} .The estimation of Δ_{UID} can be obtained from either direct location-based calculation or the previous interactions between the node and the sink. If there are multiple sinks, Δ_{UID} is estimated for the closest one. Then, upon successfully authenticating a received message, each sensor node waits for Δ_{UID} additional time to take the further reactions ("Slow React") except for forwarding the message ("Fast forward"). Suppose the message is a query. If node U_{ID} has the (partial) answer to this query, it delay replying the user with the answer $\Delta_{U_{ID}}$ time. When $\Delta_{U_{ID}}$ is timeout and there's no warning received from the sink, U_{ID} now sends out the answer. And if the sink does broadcast a revocation message, U_{ID} will be able obtain it before any further action is taken. Note that no matter whether U_{ID} is an intended recipient, it always forwards the message without delay, as long as the message is successfully authenticated by itself. Therefore, a public key forgery attack will only result in one successful message broadcast, given the "Fast forward, Slow React" policy enforced. And we note that the "Fast forward, Slow React" policy can be implemented in an on-demand manner. That is, only when the WSN is under attack will sensor nodes implement this policy. The sink can be used to control the starting and ending time.

15.8 Challenges for Future Research

There are many challenges regarding multiuser broadcast authentication in WSNs. One of the foremost important issues is to further reduce the computational overhead of PKCs when applied on resource constrained sensor nodes. Research along this direction can be two-fold: One is to adapt and test more efficient state-of-the-art PKC algorithms on sensor nodes, which also includes design new approaches to allow trade-offs between security strength and computational complexity. The other is to use hardware-software co-design for speeding up PKC computations on sensors. As sensor nodes are usually specially purposed according to the desired application, its functionality can be predetermined, which allows specific hardware design optimized both for the application and security related algorithms. Other challenges include better protocol design according to different user query patterns and data storage mechanisms. It is also useful to make use of the potential heterogeneity among the sensor nodes to design more resilient and efficient mechanisms for broadcast authentication.

15.9 Conclusion

In this chapter, we studied the problem of multiuser 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 easily lead to severe energy-depletion DoS attacks. We then came up with several effective PKC-based schemes to address the problem. Both computational and communication costs of the schemes are minimized through a novel integration of several cryptographic techniques. A quantitative energy consumption analysis, as well security strength analysis were further given in detail, demonstrating the effectiveness and efficiency of the proposed schemes.

15.10 Problems

1. What is a wireless sensor network?
2. What is a digital signature?
3. What is a bloom filter?
4. What is a Merkle hash tree?
5. How is the False Positive rate of a bloom filter defined?
6. How are broadcast messages authenticated according to the solutions proposed in this chapter?
7. What is the security issue of µTESLA scheme when applied in large scale multihop wireless sensor networks?
8. How to trade off between storage overhead and the false positive rate of the bloom filter in this chapter?
9. What is a partial message recovery digital signature?
10. How does partial message recovery signature technique help improve communication efficiency in this chapter?
11. How is user revocation operation performed in this chapter?

References

[1] IEEE P1363a Standard, "Standard specifications for public key cryptography," (http://grouper.ieee.org/groups/1363/index.html), 2000.

[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks, IEEE Communications Magazine," Vol. 40, No. 8, pp. 102-116, August 2002.

[3] I. Akyildiz and I. Kasimoglu, "Wireless sensor and actor networks: research challenges," Ad Hoc Networks 2(4): 351367, 2004

[4] R. Amirtharajah, A. Chandrakasan, "Self-powered signal processing using vibration-based power generation," IEEE Journal of Solid-State Circuits, Vol. 33, pp. 687-695, 1998

[5] M. Aydos, T. Yanik, and C. Koc. "An high-speed ECC-based wireless authentication protocol on an ARM microprocessor," In Proc. of ACSAC, pp. 401-409, New Orleans, Louisiana, 2000.

[6] J. Baek, J. Newmarch, R. Naini and W. Susilo, "A Survey of Identity-Based Cryptography," AUUG 2004, pp. 95-102, 2004.

[7] D. Boneh, H. Shacham, and B. Lynn, "Short signatures from the Weil pairing," J. of

Cryptology, Vol. 17-4, pp. 297-319, 2004.
[8] Crossbow Technology Inc, Wireless sensor networks, http://www.xbow.com/. 2004.
[9] W. Du, R. Wang, and P. Ning "An Efficient Scheme for Authenticating Public Keys in Sensor Networks," In Proc. MobiHoc'05, pp.58-67, May 25-28, 2005.
[10] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol," IEEE/ACM Transactions on Networking 8:3, 281-293, 2000
[11] D. Hankerson, A. Menezes, S. Vanstone, "Guide to Elliptic Curve Cryptography," ISBN 0-387-95273-X, 2004.
[12] Y. Hu, A. Perrig, and D. Johnson, "Packet Leashes: A Defense against Wormhole Attacks in Wireless Ad Hoc Networks," In proceedings of INFOCOM, 2003.
[13] "Intel PXA255 Processor Electrical, Mechanical, and Thermal Specification," http://www.intel.com/design/pca/applications processors/manuals/278780.htm
[14] T. Itoh and S. Tsujii, " A fast algorithm for computing multiplicative inverse in GF (2^m) using normal bases," Information and Communication, 78:171-177, 1988.
[15] A. Kansal, D. Potter and M. Srivastava, "Performance Aware Tasking for Environmentally Powered Sensor Networks," In Proc. of ACM SIGMETRICS'04, 2004
[16] A. Kansal and M. Srivastava, "An Environmental Energy Harvesting Framework for Sensor Networks," ACM/IEEE ISLPED, 2003.
[17] D. Liu and P. Ning, "Multi-level mTESLA: Broadcast authentication for distributed sensor networks," ACM Transactions in Embedded Computing Systems (TECS), vol. 3, no. 4, 2004.
[18] D. Liu, P. Ning, S. Zhu, and S. Jajodia, "Practical Broadcast Authentication in Sensor Networks," In Proc. of MobiQuitous 2005, July 2005.
[19] K. Lorincz, D. Malan, T. Fulford-Jones, A. Nawoj, A. Clavel,
V. Shnayder, G. Mainland, S. Moulton, and M. Welsh, "Sensor Networks for Emergency Response: Challenges and Opportunities," In IEEE Pervasive Computing, 2004.
[20] C. Lu, G. Xing, O. Chipara, C. Fok, and S. Bhattacharya, "A Spatiotemporal Query Service for Mobile Users in Sensor Networks, In Proc. of ICDCS, Columbus, 2005
[21] J. McCune, E. Shi, A. Perrig, and M. Reiter, "Detection of denial-of-message attacks on sensor network broadcasts," IEEE Symposium on Security and Privacy, pp.64-78, 2005.
[22] R. Merkle, "Protocols for public key cryptosystems," In Proceedings of the IEEE Symposium on Research in Security and Privacy, Apr 1980.
[23] M. Mitzenmacher, "Compressed Bloom Filters," IEEE/ACM Transactions on Networks, 10:5, pp.613-620, October 2002.
[24] D. Naccache and J. Stern, "Signing on a Postcard," In Proc. of Financial Cryptography'00, LNCS vol. 1962, pp.121-135, 2000.
[25] NIST, "Digital hash standard," Federal Information Processing Standards PUBlication 180-1, April 1995.
[26] NIST: Proposed Federal Information Processing Standard for Digital Signature Standard (DSS). Federal Register, vol. 56, no. 169, pp. 42980-42982 (1991)
[27] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and D. Tygar, "SPINS: Security protocols for sensor networks," In Proc. of MobiCom'01, 2001.
[28] A. Perrig, R. Canetti, J. Tygar, and D. Song, "The TESLA Broadcast Authentication Protocol," RSA CryptoBytes, 5, 2002.

[29] K. Ren, K. Zeng, W. Lou, and P. Moran, "On Broadcast Authentication in Wireless Sensor Networks," IEEE Transactions on Wireless Communications, Vol. 6, No. 11, pp.4136-4144, Nov., 2007

[30] K. Ren, W. Lou, and Y. Zhang, "Multi-user Broadcast Authentication in Wireless Sensor Networks," In Proc. of IEEE SECON 2007, San Diego, Jun., 2007

[31] 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, Apr. 23-29, Barcelona, Spain, 2006

[32] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Commun. ACM 21(2), pp.120-126, 1978

[33] A. Shamir, "Identity based cryptosystems and signature schemes," In Proc. of CRYPTO'84, LNCS Vol. 196, pp. 4753, 1984

[34] Texas Instruments Inc., "MSP430 Family of Ultra-lowpower 16-bit RISC Processors," http://www.ti.com

[35] 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.

[36] 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

[37] J. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," In Proc. of ACM SenSys'04, Baltimore, 2004.

[38] P. E. Lanigan, R. Gandhi, and P. Narasimhan, "Secure dissemination of code updates in sensor networks," In Proc. of ACM SenSys'05, 2005.

[39] J. Deng, R. Han, S. Mishra, "Secure Code Distribution in Dynamically Programmable Wireless Sensor Networks," In Proc. of ACM/IEEE IPSN 2006, pp. 292-300.

[40] R. Gennaro and P. Rohatgi, "How to sign digital streams," Information and Computation, 165(1):100-116, 2001.

[41] S. Lumetta and M. Mitzenmacher, "Using the Power of Two Choices to Improve Bloom Filters," Preprint.