Fast Response PKC-Based Broadcast Authentication in Wireless Sensor Networks

Panoat Chuchaisri and Richard Newman
Department of Computer and Information Science and Engineering
University of Florida
Gainseville, Florida 32611-6120
Email: {pchuchai,nemo}@cise.ufl.edu

Abstract—**Due to simpler protocol operations, e.g. no synchronization and higher tolerance to node capture attack compared to symmetric key-based approaches, public key-based (PKC) approaches have gained popularity in wireless sensor network (WSN) broadcast authentication. With PKC's security strength, a sensor node that authenticates messages before forwarding them can detect a bogus message within the first hop. While this prevents forged traffic from wasting the sensor nodes' energy, performing PKC operations in the computing-power-limited sensor node can result in undesirably long message propagation time. At the other extreme, the sensor node can forward the messages to other nodes prior to authenticating them. This approach diminishes propagation time with the trade-off of allowing forged messages to propagate through the network. To achieve swift and energy efficient broadcast operation, sensor nodes need to decide wisely when to forward first and when to authenticate first.**

In this paper, we present two new broadcast authentication schemes, called the key pool scheme and the key chain scheme, to solve this dilemma without any synchronization or periodic key redistribution. Both schemes utilize a Bloom filter and distribution of secret keys among sensor nodes to create fast and capture-resistant PKC-based broadcast authentication protocols. Our NS-2 simulation results confirm that our protocols' broadcast delay is only 50% slower than the forwarding-first scheme and ten times faster than the authentication-first scheme for a 3,000-node WSN. The key pool scheme also contains forged message propagation to the minimum even when the majority of nodes have been captured by the attacker.

I. Introduction

In a typical WSN, a large number of nodes are deployed over a target area with a few nodes acting as data sinks called access points, power nodes, or base stations. If the access point wants to gather data from the sensors, it needs to broadcast one or more query commands to all sensor nodes in the network. To avoid unnecessary radio transmission among resource-constrained sensor nodes, each broadcasting packet must be verified by a broadcast authentication protocol prior to forwarding.

Broadcast authentication in a regular network can be performed by either using a public key based digital signature to sign the broadcast packet, thus allowing intermediate nodes to verify its authenticity, or by using a message authentication code (MAC) generated from a shared secret key between nodes. Since WSNs are usually deployed in hostile or unmonitored locations, sensor nodes are highly vulnerable to capture and tampering; we cannot guarantee that locally stored secret keys will stay secure. In addition, using a PKC-based digital signature also poses some challenges. As stated in [1], if PKC is the only mechanism used for WSN's broadcast authentication, sensor nodes will have to choose between forwarding each broadcast message before verifying its authenticity (forwarding-first approach) or verifying the message's digital signature before spreading it further (authentication-first approach). The forwarding-first (FF) approach is susceptible to Denial of Service (DoS) attacks, a scenario in which an attacker impersonates an access point and broadcasts forged messages. Messages will successfully reach every node in the network thus draining each node's short-supplied battery power. On the other hand, the authentication-first (AF) approach can prevent the DoS attacks because any forged messages will be stopped at the first hop. However, due to sensor nodes' limited computing power, signature verification at every hop will impose long delays [2], which in turn will increase the overall delay of the broadcast operations.

Current attempts to solve the broadcast authentication problem can be categorized into either hardware or protocol approaches. The hardware approach prevents adversaries from learning any sensitive information from captured nodes by equipping them with tamper-resistance memory [3]. Securing all secret keys stored inside the sensor nodes removes any risk that can jeopardize broadcast authentication protocols, thus allowing the MAC approach to be used securely in a WSN [4]. Due to the higher cost of tamper-resistant hardware, this approach will be limited to critical applications or small-size WSNs [5].

For the protocol approach, there are several research efforts focusing on creating new protocols that can withstand node capturing. μTESLA [6] and its various extensions [7]–[9] use a delayed key disclosure technique, which message verifications must be delayed for some periods of time, to protect the keys' freshness. Unfortunately, μTESLA's symmetric key-only approach, which aims to create low computational cost protocol, is unsuitable for quick response applications due to its lack of the immediate authentication feature. In addition, it also requires some level of synchronization between all nodes in the network which must be achieved by periodic broadcasting.

In consequence of several emerging PKC applications, es-

COLLABORATECOM 2010, October 9-12, Chicago, USA
Copyright © 2011 ICST
DOI 10.4108/icst.collaboratecom.2010.55

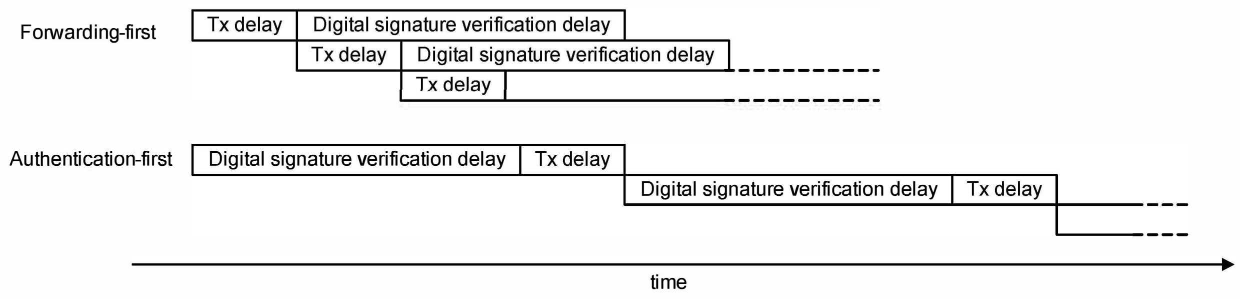

Forwarding-first
Tx delay
Digital signature verification delay
Tx delay
Digital signature verification delay
Tx delay
Authentication-first
Digital signature verification delay
Tx delay
Digital signature verification delay
Tx delay
time

Fig. 1: Forwarding-first and authentication-first broadcast timing diagram.

pecially Elliptic Curve Cryptography (ECC), in sensor nodes [10], [11], a number of researchers have been developing broadcast protocols that incorporate PKC as a primary mean for source authentication [1], [12]–[14]. Because the PKC-based approach eliminates the verification delay problem, the main focus of these research efforts is to speed up the time-consuming PKC operations. However, these protocols still rely on local key pair establishment or periodic key redistribution to filter out bogus messages.

Using the PKC approach, we propose two new broadcast authentication schemes using a digital signature as a main authentication mechanism together with a Bloom filter [15] to screen out bogus broadcast messages. Our schemes make use of WSN's large number of nodes to limit the amount of information that adversaries can gain when capturing them. This technique relies on diffusing a WSN's secrets into sensor nodes, with each node possessing only a portion of the keys required for successfully creating an authentic broadcast message. Both schemes use a probabilistic approach, so they do not require synchronization, local key pairs, or periodic key redistribution.

The first proposed scheme, the key pool scheme, separates all sensor nodes into different groups, and each group possesses a partition of the network's global key pool. An authentic broadcast message contains a subset of the global key pool, which can be quickly verified by each intermediate node before the message is forwarded. The second proposed scheme, the key chain scheme, relies on hash chains that are stored on each node to pre-verify the packet.

The rest of this paper is organized as follows. First, section II presents our assumptions about the system and threat model. Next, sections III and IV explain both proposed protocols in detail together with their theoretical performance analysis in filtering bogus packets. Simulation results and discussion are included in Section V. Then, section VI reviews related work on WSN broadcast authentication. Finally, section VII concludes this paper.

II. System and Threat Model

In this work, we assume sensor nodes to be low-cost devices without tamper-resistant hardware. A node is capable of performing basic cryptographic operations (i.e. hash, MAC) and also public key operations (i.e. signature verification) although with considerably more delay compared to the former. A regular broadcast protocol is used for network-wide message flooding. The protocol's packet size is assumed to be able to accommodate both digital signature and Bloom filter vector (BFV). Our WSN's access point works as a network's data sink and cannot be compromised. It also has enough CPU power to generate digital signatures and BFVs with reasonable delay. Sensor nodes passively gather data most of the time until they receive a broadcast message from the access point that they must verify. Our goals are 1) to minimize broadcast delay and 2) to prevent successful broadcast authentication DoS with acceptable communication and computing overhead.

Attackers, typically possessing significantly more powerful computing power than sensor nodes, are capable of performing both symmetric key and public key operation with ease. We also assume that the PKC scheme such as Elliptic Curve Digital Signature Algorithm (ECDSA) and hash are secure enough to hinder any attacker's attempt to circumvent it. However, an attacker can capture sensor nodes and learn all secret information inside them thus allowing them to create an authentic-looking Bloom filter vector of the learned key set. The attacker's objective is to fool sensor nodes into forwarding forged access point broadcast messages throughout the network and wasting sensor node battery power.

III. Key Pool Scheme

In the key pool scheme, the access point possesses all the secret keys, while each sensor node only knows a subset of them. This key partitioning helps limit the amount of secrets the adversaries can learn from node capturing. This scheme assumes that a global key pool of N keys is known to the access point while each sensor node has memory large enough to store k keys locally.

Our key pool scheme comprises of three phases: pre-deployment, signature generation, and message verification and forwarding. In the first phase, each sensor node is loaded with local keys prior to deployment. The second phase describes how the broadcast message and the signature can be generated. The last phase deals with each node's verification and forwarding decision, when a broadcast message arrives.

A. Protocol Description

Pre-deployment Phase:

1) A global key pool $\mathbb{K}$ of size N with keys $\{K_l \mid 1 \leq l \leq N\}$ must be generated and partitioned into n non-overlapping equal-sized sets $S_1, S_2, \ldots, S_n$ thus each set has $m = N/n$ keys. For example, we can partition $\mathbb{K}$ into $S_i = \{K_j | (i-1)m < j \leq im\}$.
2) Next, each sensor node randomly chooses a key set and then arbitrarily picks k keys, where $k \leq m$, from that set. It then stores keys and the corresponding key indices which will be used for BFV verification in the next phase.
3) Every node must be loaded with the access point's public key (E_{pub}), a network-wide hash function $H(\cdot)$ and q independent hash functions $\{H_1, H_2, \ldots, H_q\}$ for the r-bit Bloom filter.
4) Additionally, each sensor node calculates a *key intersection threshold* (τ) as follows:

$$\tau = \begin{cases} h + k - m & \text{if } h > m - k; \\ 1 & \text{otherwise.} \end{cases}$$

where h is the number of keys to be selected from each key set for BFV generation (explained in more detail in step 2 of the signature generation phase).
5) Lastly, the access point is loaded with all the keys in the key pool.

Signature generation Phase:

1) After deployment, when an access point wants to broadcast a message to all nodes, it constructs a packet from a message body M, a time stamp tt and a digital signature

$$DS = E_{prv}(H(M \parallel tt)) \qquad (1)$$

where E_{prv} is signing with the sink's private key and $\parallel$ denotes concatenation.
2) The access point creates a BFV from DS, which will be used by sensor nodes to pre-verify the signature. To create a BFV, the access point randomly picks h keys, where $h \leq m$, from each key set and computes an r-bit BFV using the following algorithm:

```
$BFV \leftarrow 0$
for $i = 1$ to $n$ do
  $\mathbb{R}_i \leftarrow$ randomly picked $h$ keys from $S_i$
  for all keys $K_l \in \mathbb{R}_i$ do
    for all $H_j \mid 1 \leq j \leq q$ do
      $b \leftarrow$ first $r$ bit of $H_j(K_l \parallel tt \parallel DS)$
      $b^{th}$ bit of $BFV \leftarrow 1$
    end for
  end for
end for
```

which has been adapted from [16] (The diagram is shown in Fig 2). Because each key only turns on a single bit in the BFV, this algorithm ensures that the maximum number of 1-bit in the vector will not exceed hnq.
3) Finally, the message is broadcast as

$$\{M, tt, DS, \mathbb{I}, BFV\}$$

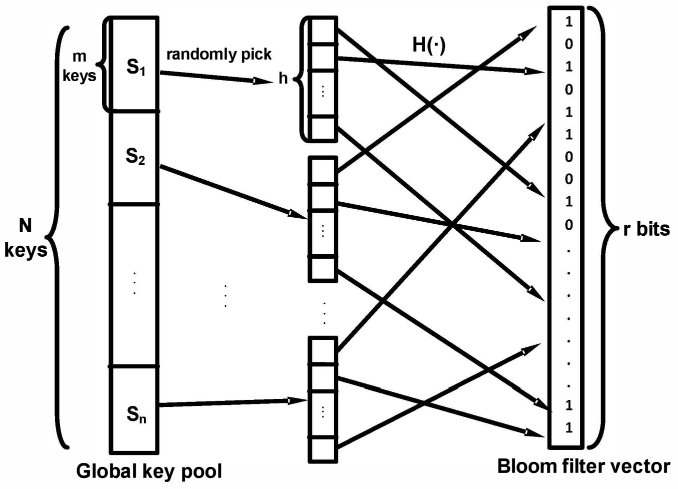


Fig. 2: Key pool scheme's Bloom filter vector generation for $q = 1$

where $\mathbb{I}$ is the set of all key indices that have been included in the BFV defined as $\mathbb{I} = \{l \mid \forall K_l \in \mathbb{C}\}$ and $\mathbb{C} = \bigcup_{i=1}^{n} \mathbb{R}_i$.

Message verification and forwarding phase:

1) We define a per-message *key intersection set* $\mathbb{X}$ as a set intersection between node's local keys and all the keys that have been included in the message's BFV. A sensor node can derive $\mathbb{X}$ from $\mathbb{I}$ in the broadcast message.
2) When a sensor node receives a broadcast message, it first confirms that the time stamp is fresh and the message has at most hnq "1" bits in the BFV. This can prevent the attacker from deliberately marking all bits in BFV to circumvent our scheme.
3) If the BFV is plausible, a sensor node then finds the message's $\mathbb{X}$ and compares $|\mathbb{X}|$ with τ. If $|\mathbb{X}| < \tau$, a sensor node will forward the message only when the DS verification is successful. Consequently, a sensor node will operate in the authentication-first mode whenever it does not share enough keys with the message.
4) On the other hand, if the threshold is achieved ($|\mathbb{X}| \geq \tau$), each node will compute a BFV of DS with all the keys in $\mathbb{X}$and then checks whether each "1" bit in the BFV has a corresponding "1" in the BFV from the message or not. It only forwards the message when BFV verification is successful.
5) The sensor node performs digital signature verification before accepting the message.

B. Protocol Analysis

1) Security Consideration: One of the main objectives of this protocol is to make WSNs more resistant to node capturing, thus we will analyze the effectiveness of our protocol in the situation that an attacker has learned some of the keys from node capturing. We assume that a out of n key sets has been compromised. When the attacker forges a new message, he can correctly mark haq bits and randomly mark $h(n-a)q$ bits to fully exploit hnq-bit limit of the BFV. Since the location of "1" from a key in the r-bit BFV is uniformly distributed, the probability of the attacker correctly marks the BFV for any unlearned key is

$$P_k = \begin{cases} \left(\dfrac{hnq}{r}\right)^q & \text{if } hnq < r; \\ 1 & \text{otherwise.} \end{cases} \tag{2}$$

Our protocol requires at least τ common keys between local keys and message's keys and all of them must have correct markings in the BFV before a sensor node will forward a message without checking the digital signature. Thus, the probability that the attacker successfully fools a sensor node that possesses keys from a secure key set to forward a forged message is

$$P_a = \frac{\sum_{i=\tau}^{min\{h,k\}} \binom{h}{i}\binom{m-h}{k-i}(P_k)^i}{\binom{m}{k}} \tag{3}$$

For a compromised key set, a forged message still can be rejected if $|\mathbb{X}|$ is less than τ. Therefore, the probability of a forged message successfully passing BFV verification on any node in the network is

$$\begin{aligned} P_b &= \frac{a}{n}\left[\frac{\sum_{i=\tau}^{min\{h,k\}} \binom{h}{i}\binom{m-h}{k-i}}{\binom{m}{k}}\right] + \left(\frac{n-a}{n}\right)P_a \\ &= \frac{a}{n}\left[\frac{\sum_{i=\tau}^{min\{h,k\}} \binom{h}{i}\binom{m-h}{k-i}(1-(P_k)^i)}{\binom{m}{k}}\right] + P_a \end{aligned} \tag{4}$$

Equation 4 shows that P_b grows linearly to the proportion of compromised keys (a/n) with y-intercept at P_a which is later confirmed in Fig 9a in the simulation result section. We now have the average number of nodes forwarding the forged message equal to P_bN_t where N_t is the total number of nodes in the network. If we consider a broadcast path as a tree with the access point as the root, the estimated number can be further reduced because we do not consider the case where a whole branch has been cut off by a node that successfully detects and drops a forged packet.

2) Reducing Maximum Bloom Filter Marking Limit: From Equations 2 and 3, we notice that the higher the hnq-bit limit, the easier the forged broadcast message can pass through the BFV verification. The naive solution can be done by simply reducing h,n and/or q. However, reducing n will increase the chance that each sensor node will share the same key set thus making the WSN more vulnerable to node capturing.

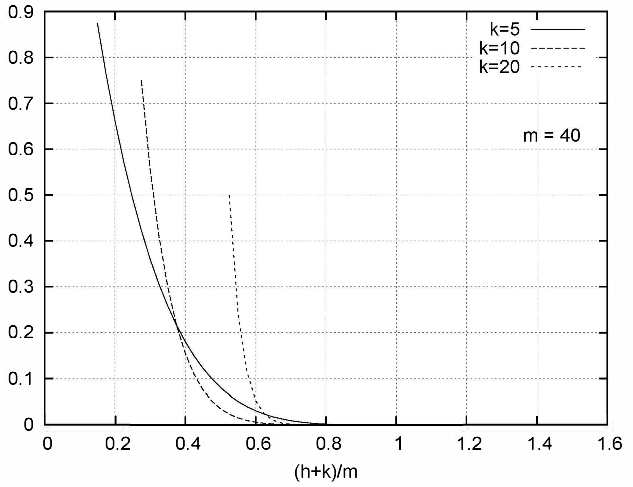


Fig. 3: Probability that a sensor node need to perform DS verification due to unsatisfied τ requirement.

On the other hand, reducing h will decrease the $h+k$ to m key intersection ratio, which will increase the chance a sensor node needs to perform DS verification as shown in Fig 3. To maintain the key intersection ratio, k needs to be increased, which also increases the number of keys the attacker can learn from capturing a node. In this section, we will describe a technique to reduce the limit without altering those parameters although with a minimal trade-off in communication overhead and initial sending delay.

From the uniformly distributed characteristic of the marked bits over the BFV, the probability of all or most of the hnq bits are set to "1" is small, hence we can reduce the maximum bit limit with the trade-off that a legitimate broadcast message may go over the new limit. In order to make an efficient decision on how much we are going to reduce the maximum bit limit, we have to determine $P(x,y,r)$, the probability of x number of "1" bits occur in the process of marking y bits out of r-bit BFV. If all y markings are uniformly distributed, we have

$$P(x,y,r) = \frac{\binom{r}{x}\Gamma_y^x}{r^y} \tag{5}$$

where

$$\Gamma_y^x = \sum_{\text{all possible } \mathcal{A}} \binom{y}{\mathcal{A}}\binom{x}{\mathcal{B}} \tag{6}$$

$$\begin{aligned} \mathcal{A} &= \{a_1, a_2, \ldots, a_x | \sum_{i=1}^{x} a_i = y, 1 \le a_i \le y, \\ &\qquad a_i \le a_j \iff i < j\} \end{aligned}$$

$$\mathcal{B} = \{|\mathcal{B}_1|, |\mathcal{B}_2|, \ldots, |\mathcal{B}_y|\}, \mathcal{B}_i = \{\forall a_j \in \mathcal{A} | a_j = i\}$$

.

The multinomial coefficient $\binom{n}{N}$ where $N = n_1, n_2, \ldots, n_m$ is equivalent to $\frac{n!}{n_1!n_2!\cdots n_m!}$ given that

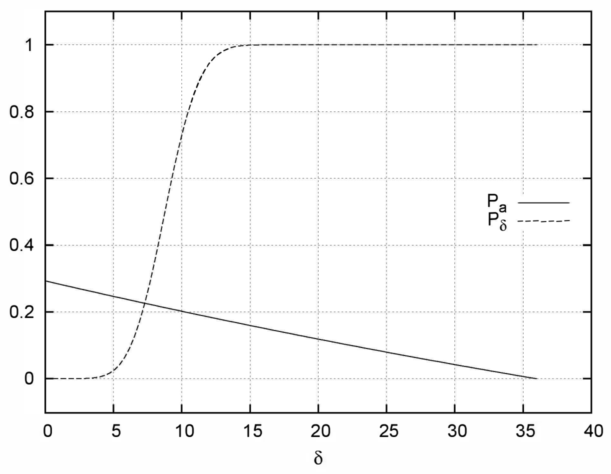


Fig. 4: P_a and P_δ with r = 64 bit, hnq/r = 0.56 and $(h{+}k)/m$ = 0.325

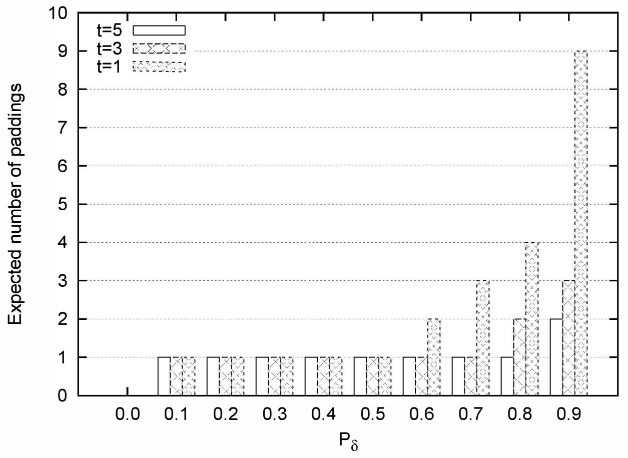


Fig. 5: Average number of paddings for different P_δ and t

$\sum_{i=1}^{m} n_i = n$. In Equation 6, each $\mathcal{A}$ represents one possible integer partition of y with exactly x addends and $\mathcal{B}$ is a set that contains the counting of each distinct addend of the corresponding set $\mathcal{A}$. If we reduce the hnq-bit limit by δ bits, $\delta < hnq$, the probability that a legitimate r-bit BFV contains more than $hnq - \delta$ marking can be calculated by

$$P_\delta = \sum_{i=hnq-\delta+1}^{hnq} P(i, hnq, r). \tag{7}$$

By reducing hnq-bit limit to $hnq - \delta$, we improve the protocol by decreasing P_a because P_k now becomes $\left(\frac{hnq-\delta}{r}\right)^q$. Fig 4 shows how δ affects P_a and P_δ without changing any number of key requirement for both broadcast messages and sensor nodes.

Nevertheless, a legitimate message can still be rejected, with probability P_δ, for violating the limit which is considered unacceptable in some scenarios. To counter this false negative situation, we can either choose δ such that P_δ is small enough to be negligible, or we can utilize the access point's computing power to guarantee that the limit will not be exceeded.

The access point, who initially sends out a message, needs to add random bits to the message whenever its BFV goes over the $hnq - \delta$ limit. First, the access point appends w bits of all zeros to the end of the message before calculating the BFV. If the $(hnq - \delta)$-bit limit is exceeded, the access point recalculates the BFV using random w bits instead. After t tries of BFV generation without any success, the access point changes all w-bit *padding* to ones and appends additional randomly chosen w bits and starts the process again. Theoretically, the number of extra bits can grow up to infinity. However, the actual number of paddings follows a geometric distribution with the expected number equal to $\left\lceil \frac{P_\delta}{t(1-P_\delta)} \right\rceil$. Fig 5 shows the average number of w-bit paddings which can be considered causing only minimal communication overhead. The optimum choice of w for any chosen t will be $\lceil \log_2(t+1) \rceil$.

3) Protocol Overheads: The key pool scheme adds both computational overhead from hash operations and communication overhead from sending additional bits in the broadcast message.

Let the length of the original broadcast message, which includes message body, time stamp and digital signature, be λ and every key index is $\lambda_{ind} = \lceil \log_2 N \rceil$ bits long, then the length of our broadcast message will be

$$\lambda_b = \lambda + hn\lambda_{ind} + r \tag{8}$$

If we use the technique from the previous section to reduce our maximum BFV marking by δ bit with w-bit padding, the expected final message length will be

$$\lambda_b = \lambda + hn\lambda_i + w \left\lceil \frac{P_\delta}{t(1-P_\delta)} \right\rceil + r \tag{9}$$

Next, the computational overhead for each sensor node depends on how many keys a broadcast message and a node share, i.e., $|\mathbb{X}|$. Given any sensor node with k local keys and a message that incorporates h keys from each set into its BFV, the probability that a sensor node and a broadcast message share exactly x number of keys ($P(|\mathbb{X}| = x)$) is:

$$P(|\mathbb{X}| = x) = \frac{\binom{h}{x}\binom{m-h}{k-x}}{\binom{m}{k}}$$

If $\mathcal{M} = \min\{k, h\}$, then the expected number of keys each sensor node has to verify will be:

$$E_k = \sum_{i=\tau}^{\mathcal{M}} i \cdot P(|\mathbb{X}| = i)$$

Hence, a sensor node is expected to perform on average qE_k and at most $q\mathcal{M}$ hash operations per message.

The access point is expected to perform $\lceil \frac{P_\delta}{1-P_\delta} \rceil$ signature and BFV generations before sending each message when using BFV with the maximum marking bit reduction, otherwise

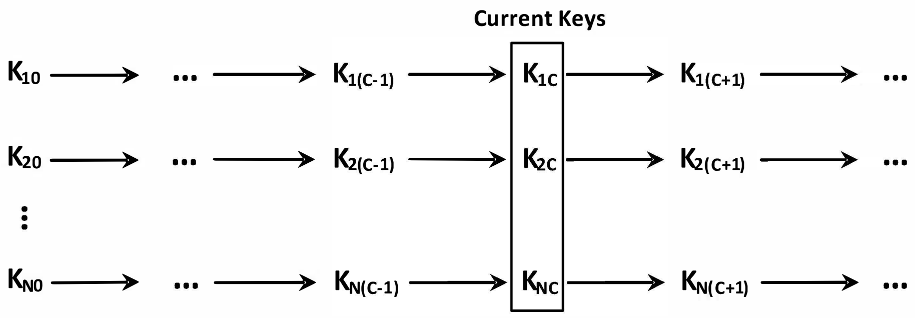


Fig. 6: A pool of N key chains with current keys enclosed in a box (key index c). Each arrow represents left-to-right hash function application.

it only performs one signature generation. It also needs to perform hnq hash operations to generate the BFV.

IV. Key Chain Scheme

The key chain scheme aims to eliminate the communication overhead in the key pool scheme by using multiple one-way hash chains. The randomness of each hash value on the chain prevents key prediction and at the same time eliminates the need to include key indices in the broadcast messages. In this scheme, a global key pool of size N is used to store N independent key chains without any partitioning. Given a network-wide hash function $H(\cdot)$ and starting key K_0, we can find a key at position i in the chain (K_i) as $K_i = H(K_{i-1})$.

Normally, the one-way hash chain must be generated in advance and the last key (the key with the highest position) is the first key to be used for authentication. Typically, the key chain is consumed "backwards" until reaching K_0 before a new chain is generated and redistributed. Our scheme, on the other hand, uses the one-way key chain in the "forward" manner where every node starts from key index zero and progresses toward higher key indices. With this approach, our protocol does not require key redistribution and does not need to include any key index in the broadcast message. However, once a key is compromised, the whole key chain is also compromised. The key chain scheme, which consists of three different phases, similar to the key pool scheme, will be described in the following section.

A. Protocol Description

Pre-deployment phase:

1) First, a global key pool, which contains N starting keys (denoted by 0 in the subscript) $K_{10}, K_{20}, \ldots, K_{N0}$ of N independent key chains, must be generated (shown in Fig 6).
2) Each sensor node randomly picks k starting keys, $1 \leq k \leq N$, and stores them in the node's memory. Similar to the key pool scheme, every node must be loaded with the access point's public key, a network-wide hash function $H(\cdot)$ and Bloom filter parameters including q independent hash functions $\{H_1, H_2, \ldots, H_q\}$.

Signature generation Phase:

1) The access point generates DS from a message body M, a time stamp tt and E_{prv} by using Equation 1.
2) Every key chain in the global key pool must be advanced from K_{ij} to $K_{i(j+1)}; \forall i \in [1, N]$.
3) The access point then inserts all the new keys into BFV using the algorithm:

```
$BFV \leftarrow 0$
**for** $i = 1$ to $N$ **do**
    **for all** $H_j \mid 1 \leq j \leq q$ **do**
        $c \leftarrow$ current key chain number
        $b \leftarrow$ first $r$ bit of $H_j(K_{ic} \parallel tt \parallel DS)$
        $b^{th}$ bit of $BFV \leftarrow 1$
    **end for**
**end for**
```

which guarantees that the number of "1" bit in the BFV will not exceed Nq.
4) The final message to be broadcast is

$$\{M, tt, DS, c, BFV\}$$

where c is the current key index in the chain.

Message verification and forwarding phase:

1) Similar to the key pool scheme, when each sensor node receives a new message, it makes sure the time stamp is fresh and then checks for BFV forgery by confirming that the number of "1" does not exceed Nq.
2) If the BFV passes the test and the key index c has not been used, the sensor node advances all its k local key chains to that index.
3) If all the corresponding bits in BFV for the new local keys are verified, the sensor node forwards the message further. If the verification fails, the packet is dropped.
4) Lastly, each sensor node must verify the digital signature before accepting the packet. If the verification fails at this stage, a sensor node must revert c and all local keys back to their previous values.

B. Protocol Analysis

1) Security Considerations: The main security concern for this scheme is how well it can resist node capture and how easily the attacker can circumvent our Bloom filter checking. We will also answer what will happen if the attacker manipulates the value of c in the broadcast message.

With Nq-bit limit on the key pool scheme's BFV, we assume that the attacker always utilizes it to assure his maximum chance of deceiving the key checking mechanism. For a uniformly distributed BFV, the attacker can correctly mark r-bit BFV for an unknown key with probability

$$P_k = \begin{cases} \left(\frac{Nq}{r}\right)^q & \text{if } Nq < r; \\ 1 & \text{otherwise.} \end{cases} \quad (10)$$

If the attacker has learned A keys out of N possible keys from node capturing, the probability that a forged broadcast message can bypass our BFV checking on any single sensor node is

$$\begin{aligned} P_b &= \left(\frac{A}{N} + \left(\frac{N-A}{N}\right) P_k\right)^k \\ &= \left(\frac{A}{N}(1 - P_k) + P_k\right)^k \end{aligned}$$

It is also worth mentioning that due to independent verification of each broadcast message at each node, the same P_b analysis still applies for both key pool and key chain scheme even when the attacker's wireless signal is strong enough to cover an entire WSN's deployment area, which can be problematic in some schemes such as [1].

We can apply the same technique from the key pool scheme to reduce P_b by lowering the Nq-bit limit by δ bits. The false negative probability P_δ can be calculated by simply replacing hnq with Nq in Equation 7. Also, the probability of correctly marking an unknown key P_k is changed to $\left(\frac{Nq-\delta}{r}\right)^q$.

Another issue that needs to be discussed is how the attacker can manipulate key index c to his advantage. If the attacker modified c to be smaller than the original one, i.e., $c' < c$, this will result in sensor nodes treating it as an obsolete message and rejecting it. The attacker cannot benefit much from this case, since he cannot manipulate sensor nodes into forwarding his forged message. On the other hand, if c' is more than c, a sensor node will be forced to perform $k(c' - c)$ hash operations before it can verify the BFV. However, the probability of successfully passing the BFV verification (P_b) remains minimal due to the randomness of key value in the key chain.

To deal with the c' exploitation, we can limit the maximum window size ω, the maximum allowance of $c' - c$, in which the sensor node still performs the BFV verification. If the ω limit is exceeded, sensor nodes will verify the digital signature directly. With this extension, the attacker can inflict at most $k\omega$ hash operations per broadcast message with trade-off in the flexibility of WSNs in handling lost packets. In other words, ω is the maximum acceptable lost broadcast messages before sensor nodes switch to authentication-first mode. Another benefit of using window size limitation is when the key index start to wrap around or overflow. A properly chosen ω can prevent confusion in that circumstance similar to how the sliding window mechanism accommodates the sequence number wrap-around in TCP [17].

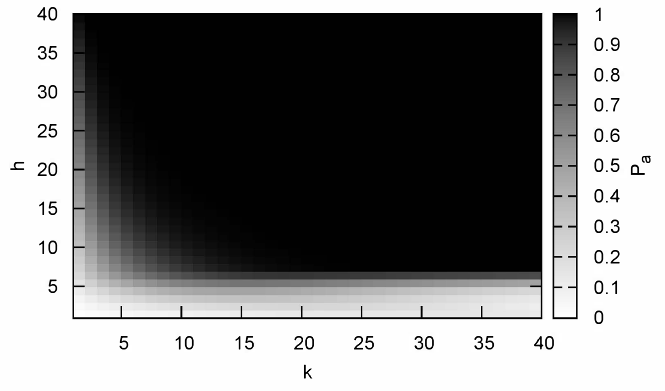


Fig. 7: P_a of all possible h and k values with $m = 40$ (the key pool scheme).

2) Protocol Overheads: Using the same set of notations from the previous protocol overhead analysis, the key chain scheme's message length λ_b will equal that in Equation 8 and 9 but with $hn\lambda_i nd$ replaced by λ_c, where λ_c is the bit length of key index c. The optimal λ_c for a given ω is $\lceil \log_2 \omega \rceil$. These equations display one of the benefits of the key chain scheme, i.e., the message length remains constant while the key pool scheme's message grows proportionally to $hn\lambda_i$.

Since the access point includes all keys into BFV calculation, each sensor node will have to perform exactly $k((c' - c) + q)$ hash operations per broadcast message. On the other hand, the access point itself needs to perform N hash operations for key chain advancement plus a single digital signature and BFV generation. The number of signature and BFV generations will increase to an average of $\lceil \frac{P_\delta}{1-P_\delta} \rceil$ if the maximum bit reduction is used.

V. Simulation and Results

We simulate our protocols in NS-2 with the ManaSim WSN module extension [18]. We randomly deploy 3,000 stationary nodes inside a simulated area of size 200m by 200m with a single access point located at the center. Each sensor node has a transmission range of 7.7459m and the hardware specification of a MICA2 mote [19]. Both protocol schemes are implemented using 64-bit BFV, 1.6s verification time for a 20-byte digital signature [2] and 0.6ms delay per hash operation per one byte of data [20]. Unless stated otherwise, we run our simulation 30 times without maximum BFV bit reduction ($\delta = 0$) for each data point on the graph.

A. Parameter Selection

1) Key pool scheme: Several parameters of the key pool scheme need to be adjusted to create optimum operating conditions for the sensor network. First, the choice of r and N are determined by the network's maximum packet size.

Because WSN's typical packet size is less than 100 bytes, the appropriate value of r is between 32 and 64 bits. The total number of keys (N) and the number of key sets (n) affects the size of key index set $\mathbb{I}$ thus they are limited by the maximum packet size as well. Larger n and N values hinder attackers from learning the complete key set from node capturing but they increase the communication overhead as well. The number of keys in each set (m) is then calculated from N/n.

Next, the $h+k$ to m ratio is chosen such that it balances the forged message detection power with the network's broadcast delay. With low $h+k$ to m ratio, forged messages will be easily detected as each sensor node is more likely to operate in the authentication-first mode. On the other hand, it also increases the broadcast delay which is undesirable for WSN applications that require fast response time. Fig 3 shows that a desirable $(h+k)/m$ value is between 0.5 and 0.6.

After choosing an $h+k$ value, we have to determine how many keys will be locally stored in the sensor nodes' memory (k) and how many keys will be included in the messages (h). There are two main benefits of having a relatively small h value for a given $h+k$. First, low h values have low P_a for almost every possible k values as shown in Fig 7 (the lighter shaded area). Second, it has lower communication overhead according to Equation 8. A k value then can be chosen to match a preferred $h+k$ to m ratio.

The last parameter q can be optimized by finding a q that minimize the P_k in Equation 2. By performing a first-order partial derivative test on P_k with respect to q, we have

$$\left(\ln\left(\frac{hnq}{r}\right)+1\right)\left(\frac{hnq}{r}\right)^q = 0$$

$$\therefore q = \frac{r}{hne}$$

Because q must be a positive integer, the optimum value for q is $max(nint\left(\frac{r}{hne}\right), 1)$, where $nint$ is the nearest integer function.

2) Key chain scheme: The key chain scheme has significantly fewer numbers of parameters to be chosen because its key assignment is less complicated in comparison to the key pool scheme. First, the selection of r is similar to the key pool scheme but a larger r is also possible due to the key chain scheme's lower overhead. The total number of key chains (N) must be selected such that $Nq < r$ to avoid $P_k = 1$ case in Equation 10. A higher N gives a higher false positive probability (P_k) for a forged message to bypass BFV verification. On the contrary, a higher N also impedes the attacker from learning all the key chains. Therefore, a choice of N must balance between these two aspects. Using a similar optimization as the key pool scheme, the ideal choice of q is $max(nint\left(\frac{r}{Ne}\right), 1)$.

Next, the number of locally stored key chains (k) affects the probability of a forged message slip through the BFV verification (P_b) as shown in Fig 8. After choosing desired N and P_b, we then calculate k from

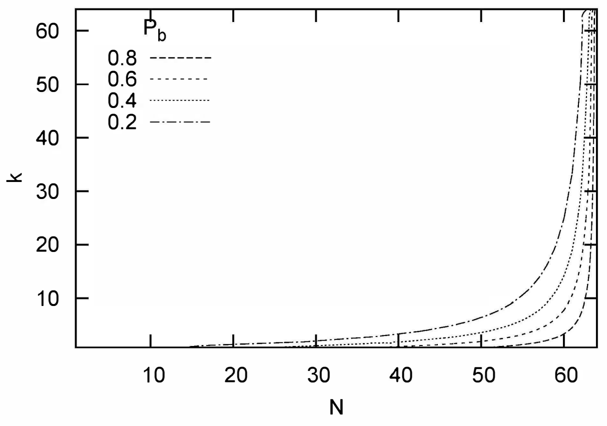


Fig. 8: P_b of all possible k and N values with $r = 64$ and $a = 0$ (the key chain scheme).

$$P_k = \left(\frac{Nq}{r}\right)^q, q = \frac{r}{Ne}$$

$$\therefore P_k = e^{-\frac{r}{Ne}}$$

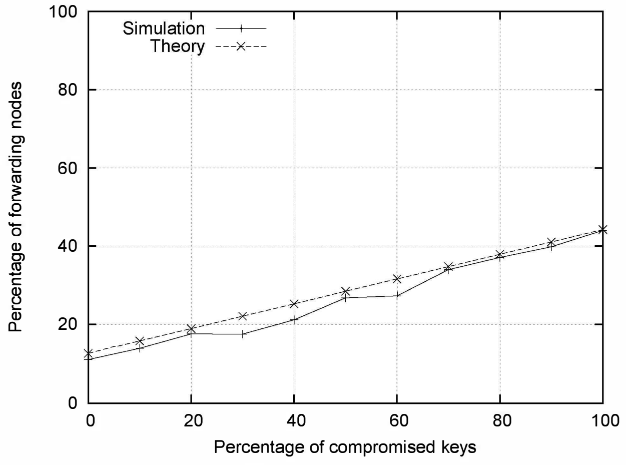
$$\text{For } a = 0, P_b = (P_k)^k = e^{-\frac{rk}{Ne}}$$

$$k = -\frac{Ne}{r}\ln(P_b)$$

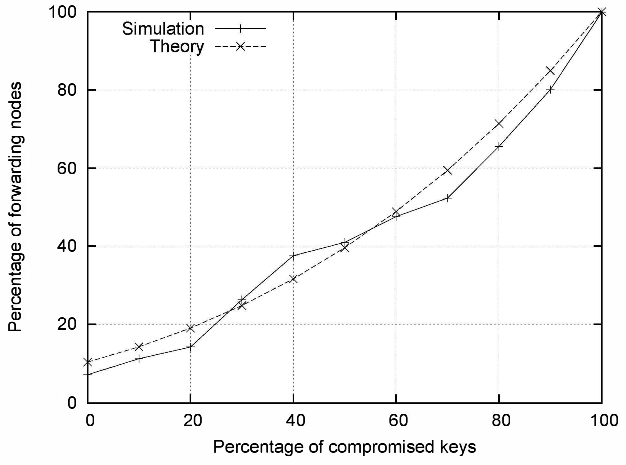
For any given P_b value, an N value which does not exceed the maximum curvature point (the knee of the curve) is preferable. This region creates good capture-resistance (low k) and good forged message detection (moderate to high N) WSNs than the top right region with the same P_b.

B. Attack Resistant

To evaluate our protocol's effectiveness in forged message propagation prevention, we plot the number of sensor nodes that forward a forged message against the number of compromised keys in the message. We simulate the keys that the attacker has learned from node capturing by using authentic keys for those compromised key sets. As the number of compromised keys increase, sensor nodes are more likely to forward the forged broadcast message as shown in Fig 9. For the key pool scheme, we use the parameters h = 2, n = 10, q = 1, m = 40 and k = 10. To tighten the test scenario, we assume the attacker compromises an entire key set at a time which, in a real situation, the attacker will need to capture several nodes of the same set to achieve this. For the key chain scheme, we use the parameter N = 30, q = 1 and k = 3. The result in Fig 9 also shows that the key pool scheme performs better under a higher percentage of compromised keys than the key chain scheme. This advantage comes at a higher cost in packet length and propagation delay.



(a) Key pool scheme



(b) Key chain scheme

Fig. 9: Portion of nodes forwarding a forged message under different number of compromised keys

TABLE I: Average delay of various schemes

	Receiving		Forwarding		Passing to the application	
	Time(sec)	% of FF time	Time(sec)	% of FF time	Time(sec)	% of FF time
Forwarding-first (FF)	0.422	100%	0.422	100%	2.022	100%
Authentication-first (AF)	20.145	4773.7%	21.745	5152.8%	27.745	1372.2%
Key pool (KP)	1.306	309.5%	2.262	536.0%	2.966	146.7%
Key chain (KC)	1.147	271.8%	1.219	288.9%	2.819	139.4%

C. Authentication Delay

Understanding the impact of our protocols on authentication delay is also crucial. We use the same setup as in the previous section to simulate and record the time that a sensor node receives and forwards a message. We also record the time when our protocol passes the message up to the application layer after it finished verifying the digital signature. In this scenario, we only used authentic broadcast message to simulate the delay introduced by our schemes under normal WSN operations. As shown in Table I, both proposed schemes introduce very little propagation delay compared to the authentication-first. The key pool and key chain scheme's application layer timings are 46.7% and 39.4% slower than the forwarding-first scheme. This is a significant improvement over the authentication-first scheme which is more than 10 times slower. The key pool scheme has more delay because its key intersection requirement (τ) causes some of the nodes to act like the authentication-first scheme while the key chain scheme does not.

VI. Related Work

Many researchers have worked on mitigating and preventing denial of service attacks in sensor network environments. Wood et al. provided a taxonomy of WSN denial of service attacks in [21]. Luk et al. pointed out key properties of designing DoS countermeasures in [22].

μTESLA [6] utilizes a one-way hash chain and delayed disclosure time to guarantee a key's secrecy within the undisclosed period. Its main shortcoming, authentication delay, has been addressed in the subsequent TESLA extension [7] by allowing immediate authentication, however it requires sending rate to remain constant.

Wang et al. first proposed a dynamic window scheme [1] using additive increase multiplicative decrease (AIMD) to regulate the window size, which, in turn, allows the node to adaptively switch between authentication-first and forwarding-first behavior. Unlike our proposed method, this scheme is less effective if malicious nodes are equipped with high-power antennae. Moreover, a good strategic placement of multiple malicious nodes can force an entire WSN into an authentication-first mode.

Dong et al. addressed the dynamic window scheme's shortcomings using pre-authentication filters [12]. This scheme relies on establishing a group-key with a node's neighbors and filtering out misbehaving nodes. However, it only focuses on preventing unnecessary signature verifications and does not address the message propagation delay issue. Additionally, the proposed protocol still allows malicious nodes to successfully broadcast forged messages at a certain rate.

Ning et al. provided weak authentication to pre-filter bogus messages using a one-way key chain in [23] which can be used in tandem with any existing digital signature-based broadcast authentication protocol. Like μTESLA, which also uses a one-way hash chain, this approach requires synchronization and periodic broadcasting between the access point and sensor nodes when it is used with signature-based authentication.

Wang et al. later proposed ShortPK [14] to reduce the signature verification time by using multiple short-lived public keys instead of a single key. However, it requires multiple public keys to be stored in sensor nodes and the sink node to periodically broadcast and redistribute public keys to the entire network once the lifetime of the keys has expired.

Ren et al. [13] implemented multi-user authentication for WSN by using a Bloom filter to store multiple user IDs and public keys. Because all the information required to construct an authentic-looking Bloom filter is publicly available to all nodes, the Bloom filter itself can be easily forged and thus cannot be used to prevent broadcast DoS attacks.

Fundamentally, our key pool-based technique is similar to what Statistical En-route Filtering (SEF) [16] uses for preventing a bogus message from a malicious node from reaching the access point. However, the extent of damage in terms of resource exhaustion for this attack is more confined in comparison to the network-wide attack in broadcast authentication.

Our proposed protocols can prevent broadcast authentication DoS, and rely on neither node synchronization nor periodic broadcasting. Moreover, they can resist node capture until a considerable number of nodes in the network are compromised. At the same time, the protocol overhead is low enough to be practical while the propagation delay is close to the forwarding first scheme.

VII. Conclusion

We have demonstrated two new broadcast authentication schemes, which utilize Bloom filters to enhance PKC-based broadcast authentication in WSN. The Bloom filter is used by each sensor node to verify quickly the authenticity of the message's digital signature. Because the Bloom filter is computed from multiple secret keys that are randomly distributed to each sensor node, the attacker is required to capture and learn secrets from a large portion of the sensor nodes before he can effectively launch broadcast authentication DoS against the network. This approach also eliminates the time synchronization, local key pair establishment, and key redistribution requirements present in other schemes. Moreover, since each node independently verifies the message, both schemes remain effective even though the malicious node's wireless transmission area covers the entire WSN.

The key pool scheme which is based on key partitioning has improved node capturing resistance with a trade-off in communication overhead. The key chain scheme considerably reduces the overhead but underperforms the key pool scheme when the compromised key proportion is high. The theoretical analysis of both protocols is validated by our NS-2 simulation result.

In our future research, we want to study the effects of different key partitioning techniques and probabilistic key distribution models on the rate of which the attacker can learn the global key pool.

Acknowledgment

The authors would like to thank Prapaporn Rattantamrong and David Wolinski for their helpful feedbacks. We would also like to thank Kittipat Kampa for his comments on our statistical analysis model.

References

[1] R. Wang, W. Du, and P. Ning, "Containing denial-of-service attacks in broadcast authentication in sensor networks," in *MobiHoc '07: Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA: ACM, 2007, pp. 71–79.

[2] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, "Comparing elliptic curve cryptography and rsa on 8-bit cpus," *Cryptographic Hardware and Embedded Systems - CHES 2004*, pp. 119–132, 2004.

[3] P. Pecho, J. Nagy, P. Hanacek, and M. Drahansky, "Secure collection tree protocol for tamper-resistant wireless sensors," *Communications in Computer and Information Science*, vol. 58, pp. 217–224, 2009.

[4] S. Basagni, K. Herrin, D. Bruschi, and E. Rosti, "Secure pebblenets," in *MobiHoc*. ACM, 2001, pp. 156–163.

[5] A. Perrig, J. Stankovic, and D. Wagner, "Security in wireless sensor networks," *Communications of the ACM*, vol. 47, no. 6, pp. 53–57, jun 2004.

[6] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar, "SPINS: security protocols for sensor networks," in *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM, 2001, pp. 189–199.

[7] A. Perrig, R. Canetti, D. Song, and J. D. Tygar, "Efficient and secure source authentication for multicast," in *In Network and Distributed System Security Symposium, NDSS 01*, 2001, pp. 35–46.

[8] D. Liu and P. Ning, "Multi-level μTESLA: A broadcast authentication system for distributed sensor networks," Department of Computer Science, North Carolina State University, Tech. Rep. TR-2003-08, 1 2003, sat, 01 Mar 2003 22:54:54 GMT.

[9] Q. Li and W. Trappe, "Reducing delay and enhancing dos resistance in multicast authentication through multigrade security," *IEEE Journal of Intelligent and Fuzzy Systems*, vol. 1, no. 2, pp. 190–204, 2006.

[10] E. Mykletun, J. Girao, and D. Westhoff, "Public key based cryptoschemes for data concealment in wireless sensor networks," in *ICC2006*, vol. 5, 2006, pp. 2288–2295.

[11] P. Han, Y. Zhu, and Y. Hu, "Design of multi-signature scheme in wireless networks," in *ACIS-ICIS*. IEEE Computer Society, 2007, pp. 247–251.

[12] Q. Dong, D. Liu, and P. Ning, "Pre-authentication filters: providing dos resistance for signature-based broadcast authentication in sensor networks," in *WiSec '08: Proceedings of the first ACM conference on Wireless network security*. New York, NY, USA: ACM, 2008, pp. 2–12.

[13] K. Ren, S. Yu, W. Lou, and Y. Zhang, "Multi-user broadcast authentication in wireless sensor networks," *IEEE Transactions on Vehicular Technology*, vol. PP, no. 99, p. 1, 2009.

[14] W. D. Ronghua Wang and X. Liu, "ShortPK: A short-term public key scheme for broadcast authentication in sensor networks," *ACM Trans. Sensor Netw.*, vol. 6, no. 1, p. 29, December 2009.

[15] B. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, jul 1970.

[16] F. Ye, H. Luo, S. Lu, and L. Zhang, "Statistical en-route filtering of injected false data in sensor networks," in *Proc. INFOCOM 2004.*, vol. 4, 2004, pp. 2446–2457 vol.4.

[17] J. Postel, "Transmission Control Protocol," RFC 793 (Standard), Internet Engineering Task Force, Sep. 1981, updated by RFCs 1122, 3168. [Online]. Available: http://www.ietf.org/rfc/rfc793.txt

[18] Manna Research Group. (2010, May) Mannasim framework. [Online]. Available: http://www.mannasim.dcc.ufmg.br/index.htm

[19] Crossbow Technology. (2010, May) MICA2 mote datasheet. [Online]. Available: http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf

[20] C. Karlof, N. Sastry, and D. Wagner, "TinySec: A link layer security architecture for wireless sensor networks," in *SenSys 2004*, nov 2004.

[21] A. D. Wood and J. A. Stankovic, "Denial of service in sensor networks," *Computer*, vol. 35, no. 10, pp. 54–62, 2002.

[22] M. Luk, A. Perrig, and B. Whillock, "Seven cardinal properties of sensor network broadcast authentication," in *SASN '06*. New York, NY, USA: ACM, 2006, pp. 147–156.

[23] P. Ning, A. Liu, and W. Du, "Mitigating DoS attacks against broadcast authentication in wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 4, no. 1, jan 2008.