# $H^2BSAP$: A Hop-by-Hop Broadcast Source Authentication Protocol for WSN to mitigate DoS Attacks

Chakib BEKARA and Maryline LAURENT-MAKNAVICIUS and Kheira BEKARA
INSTITUT TELECOM, TELECOM&MANEGEMENT Sud-Paris
CNRS Samovar UMR 5157, 9 rue Charles Fourier, 91000 Evry, FRANCE
{chakib.bekara, maryline.maknavicius, kheira.bekara}@it-sudparis.eu

***Abstract*—Broadcast communication is a dominant communication pattern in WSN. As a major security concern, the broadcast source authentication is needed to mitigate impersonation of a broadcast source, modifications of its broadcasted data, or depletion of the limited energy of sensors caused by an attacker injecting useless broadcast traffic. Several Broadcast Source Authentication Protocols (BSAPs) were proposed in the literature. One class of them is time asymmetry-based BSAPs like $\mu$TESLA [1] protocol. These BSAPs operate delayed key-disclosure to secure broadcast communications, but they suffer from a kind of DoS attack, called *resource-draining* attack, in which an attacker floods the network with fake messages that all sensors of the network buffer and forward, then later verify, thus causing buffer overflow and batteries depletion. In this paper we propose the $H^2BSAP$ protocol, to overcome this kind of DoS attacks, by achieving a hop-by-hop authentication of broadcasted messages, thus limiting the damage of an attacker to its one-hop neighbors only, instead of the entire network.**

## I. Introduction

In wireless sensors networks (WSN), broadcast communications are common usage to collect data from sensors, distribute software update or routing update information, etc. As an example, once nodes are deployed, the base station (BS) can proceed to wireless network programming (ex: Deluge [2]) to broadcast the updated software so sensors of the WSN self update. In addition, network flooding by the BS is needed when requesting information from the WSN, especially in WSN implementing data centric routing protocols where no global identification space exists for sensors [3]. However, broadcast communications need to be carefully secured, otherwise disastrous consequences will arise if an attacker takes advantage from broadcast communications to launch a local or a global attack against the network. For instance, an attacker can use wireless network programming to compromise all nodes of the network by broadcasting a malicious code as an updated software sent by the BS, or it can broadcast useless messages in order to deplete the batteries of sensors. To prevent the network from such attacks, a broadcast source must authenticate each message, by broadcasting both the message and an authenticator (i.e. MAC, signature) computed over the message with some key materials. Receivers verify the authenticity of both the broadcast source and the message, using some authentication procedure, before accepting it as valid. Likely, receivers are ensured that the identity of the broadcast source is the claimed identity, and that received messages are authentic.

Several broadcast source authentication protocols (BSAPs) for WSN were proposed in the literature. These protocols can be divided in two main classes: BSAPs providing delayed authentication [1] [4] [5], and BSAPs providing immediate authentication [6] [7] [8]. The first class achieves delayed authentication of broadcasted data, and is based on delayed key disclosure, where the source uses a key $K$ to authenticate its data at instant $t$ and later discloses $K$ at instant $t + \delta t$ to allow receivers verify data. The second class achieves immediate authentication of broadcasted data and is based on key asymmetry between source and receivers, where both source and receivers use a distinct set of keys to authenticate and verify data, respectively. BSAPs providing delayed authentication are known for their low computation overhead, and low transmission overhead (small-size authenticator per-packet), but due to the key disclosure delay, they are target to resource-draining DoS attacks, in which an attacker aims to cause sensors' buffer overflow, and deplete sensors batteries by making them forwarding useless data. In the other hand, key-asymmetry based BSAPs are known for their heavy computation overhead (especially those based on digital signatures) and a high transmission overhead (big-size authenticator per-packet), but are more immune to resource-draining DoS attack, where only few-hop neighbors of the attacker are affected by the attack and not the entire network as in the previous class.

In this paper, we investigate resource-draining DoS attacks in time-asymmetry based BSAPs, and propose $H^2BSAP$, an efficient hop-by-hop time-asymmetry based BSAP, that limits the effects of resource-draining DoS attack to the one-hop neighbors of the attacker only, while introducing an acceptable extra computation and transmission overheads on the network. Unlike other time-asymmetry BSAPs, in which sensors buffer/forward data, then later verify them, which let them an easy target to resource-draining DoS attacks, in $H^2BSAP$, sensors buffer data, then later verify them, and only if data is authentic, forward them.

The remainder of the paper is organized as follows. In section II, we overview some time asymmetry based BSAPs, and see the impact of resource-draining DoS attacks on them. In section III, we describe our assumptions, our network model, the assumed adversary model and the adopted notations. In section IV and section V, we describe our proposed protocol, and we give a detailed security analysis of it in section VI. In section VII, we give the computation, storage and transmission overheads of our protocol. In section VIII we give the limits and propose improvement of our protocol, and we conclude our work in section IX.

## II. Overview of time-asymmetry based BSAPs

This class of protocols uses only efficient symmetric key cryptography to achieve secure broadcast source authentication. It introduces time asymmetry in order to prevent source impersonation due to usage of shared symmetric keys, that the source uses to authenticate messages it broadcasts, and receivers use to verify the authenticity of the messages. Time-asymmetry based BSAPs assume that receivers are loosely time synchronized with the sender. Each key the source uses to authenticate data, is valid only during a predetermined time interval, after which it is no longer valid. A key $K_i$ the source uses to authenticate data it sends during time interval $I_i$, is later divulged after a time $\delta t$ which is equal to or greater than the maximum propagation delay in the network. Below, we describe $\mu$TESLA protocol [1], which is the most known time-asymmetry based BSAP for WSN.

### A. Overview of $\mu$TESLA protocol

The BS, which is the main source in the network, generates a one way key chain $\{K_n\}$ of *n+1* elements $(K_0, ..., K_n)$, by recursively applying a hash function $H$ on a random secret key $K_n$, where $K_i = H(K_{i+1})$, and $K_0$ is the commitment key. The particularity of a key chain, is that having any authentic key $K_j$, $j \geq 0$, we can easily verify any future key $K_i$, $i > j$, without being able to predict it. Then, the BS divides time into *n* time intervals $I_1, I_2, ..., I_n$ of equal duration $T_{int}$, where $T_i = T_0 + i \times T_{int}$, is the beginning of time interval $I_i = [T_i, T_{i+1}[$. All nodes of the network are initially pre-loaded with the key-chain parameters $(K_0, T_0, T_{int}, \delta, ...)$.

The BS uses key $K_i$, $i \geq 1$, to authenticate (generate MACs over) packets it broadcasts in time interval $I_i$. Each broadcast packet carries the index of the interval in which it was sent. At the beginning of time interval $i+\delta$ ($I_{i+\delta}$), the BS divulges $K_i$ to allow receivers verify the buffered packets, where $\delta \times T_{int}$ is greater than the maximum propagation delay in the network.

Upon reception of a packet $P_k$ broadcasted in time interval $I_i$, a receiver $R$ checks if $P_k$ verifies the weak security condition in order to buffer it. The weak security condition to verify, is that the BS is not in a time interval in which key $K_i$ used during $I_i$ is already divulged, in order to ensure that $P_k$ could not be spoofed by an adversary. If $t_c$ is the local time at which packet $P_k$ was received, and $\varepsilon_{BS,R}$ is the time synchronization error between R and the BS, the weak security condition to be verified is that $\lfloor \frac{t_c+\varepsilon_{BS,R}-T_0}{T_{int}} \rfloor < i + \delta$, in order to buffer $P_k$, otherwise $P_k$ is dropped.

Once the BS discloses $K_i$, a receiver verifies the key by checking if $K_j = H^{i-j}(K_i)$, where $K_j$, $j < i$, is the latest verified key the receiver posses (initially $K_0$). Once the key is verified, the receiver stores $K_j$ instead of $K_i$ and verifies the buffered packets sent by the BS in interval $I_i$ using $K_i$.

Due to the key disclosure delay $\delta$, which is the same for all sensors, $\mu$TESLA is vulnerable to resource-draining DoS attacks. Indeed, during time interval $I_i$, an attacker can send an important amount of bogus packets, claiming that they were sent by the BS in interval $I_i$. Sensors check that key $K_i$ is not yet disclosed, then buffer the fake packets for $\delta$ time intervals, which cause buffer overflow, where legitimate packets sent by the BS are erased by fake packets. More dangerously, because sensors act as routers, and because they *first forward data, then later verify them*, most nodes of the network deplete their energy on forwarding fake packets. Thus, just a single attacker can generate a resource-draining DoS attack, causing a severe damage to the entire network. Moreover, detecting the faulty node which generated the fake packets is not trivial. Indeed, because each node forwards data, then later verifies it, each node in the network forwarding fake packets, claims that it receives the fake packets from its upstream parents, thus an attacker can never be localized and clearly identified.

## III. Background

### A. Assumptions and Network model

First, we assume that the dimension of the deployment area is known in advance, and is for instance $L \times L$ square. In addition, we assume that each node in the network has a fixed communication range of $R_{max}$. For simplicity, we assume that the BS is in the middle of the deployment area.

Second, we assume that the BS is static and that sensors are static once deployed, whereas attackers can be mobile.

Third, we assume that the network depth is at most *l*-hop (see figure 1). This means that the farthest node from the BS, is *l*-hop away far from the BS. Estimating the maximum depth of the network, can be deduced from estimating the maximum propagation delay in the network.

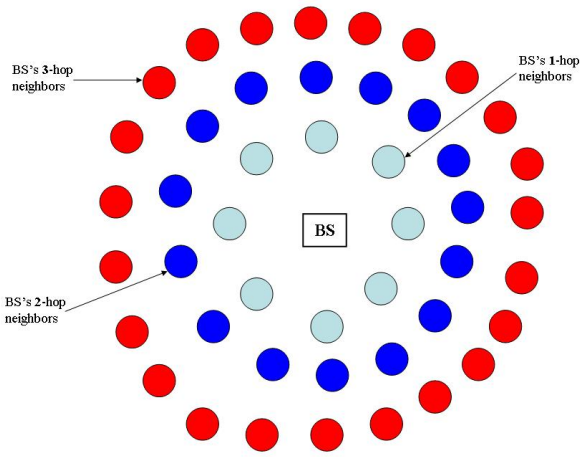


Fig. 1. A WSN of depth *l*=3

Fourth we do not assume any pre-knowledge or post-knowledge of sensors deployment coordinates. However, if sensors deployment coordinates were available, our protocol will achieve better performances.

Fifth, we assume that each node in the network participates in the broadcast process, by rebroadcasting a freshly received packets to its one-hop neighbors.

Finally, as in $\mu$TESLA, we assume that the BS is the main broadcast source in the network, and that sensors are loosely time synchronized with the BS. All sensors of the network trust the BS.

### B. Adversary model and security objectives

The main objective of an attacker is to launch a resource-draining DoS attack, where it broadcasts fake packets that the *maximum* number of sensors in the network *buffer* and *forward*. Damage is maximum when all sensors of the network buffer and forward the fake packets sent by a single attacker in the network. In $\mu$TESLA, a single attacker can cause a resource-draining DoS attack on the entire network. We assume that several attackers, which can be mobile, can exist in the network, and that attackers can collude to launch distributed resource-draining DoS attacks. Attackers can communicate directly if they are in the communication range of each others, or can use a low latency out-of-band communication channel (wired communication channel) to communicate if they are multi-hop away from each others. It is clear that launching a resource-draining DoS attack on the entire network with the minimum set of attackers (ideally a single attacker) is the objective of attackers.

Our main objective, is to secure the WSN against broadcast source impersonation, broadcast data source modification/fabrication and *resource-draining DoS* attack. While all time-asymmetry based BSAPs provide the two first security services, all of them fail to thwart efficiently resource-draining DoS attacks, where a single attacker can generate a resource-draining DoS attack on the entire network. So our main contribution, regarding other time-asymmetry based BSAPs, is to efficiently defend against resource-draining DoS attacks, by limiting the impact of the attack to the *one-hop neighbors* of the attacker only.

TABLE I
USED NOTATIONS

| Notation | Significance |
|---|---|
| BS | Base station |
| $u, v$ | Two nodes of the WSN |
| $K_{u,v}$ | A shared secret key established between nodes $u$ and $v$ |
| $l$ | The maximum depth (in terms of hop-count) of the network |
| $T_{int}$ | A duration of one time interval |
| $\epsilon_{u,v}$ | Upper bound of time synchronization error between $u$ and $v$ |
| $I_i$ | The $i^{th}$ time interval, $i = 1...n$ |
| $T_i$ | The beginning of the time interval $I_i$ |
| $\{K_n^i\}$ | A one way key-chain of length $n+1$ used for the $i$-hop neighbors of the BS |
| $\delta_i$ | Disclosure delay of key-chain $\{K_n^i\}$ |
| $MAC_K(M)$ | An 8-byte message authentication code generated over *M* using key *K* |
| $MAC_{op}$ | Time needed to generate/verify a MAC |
| $H$ | A one way hash function, with an output length of 8 bytes |
| $Hash_{op}$ | Time needed to perform a hash operation |
| $A.b$ | A field b of a structure A |
| $A \parallel B$ | A concatented to B |

### C. Notations

For clarity, the symbols and notations used throughout the paper are listed in table I.

## IV. MOTIVATION AND MAIN IDEA OF $H^2BSAP$

The development of $H^2BSAP$ is mainly motivated by the observation we made about time-asymmetry based BSAPs. Most of the time-asymmetry BSAPs we found in literature, have a low computation and transmission overhead for authentication/verification and transport of broadcasted data. However, they all suffer from resource-draining DoS attacks. A single attacker, can cause buffer overflow and rapid energy depletion on all sensors of the network, by continuously sending fake data. This attack is made possible because of the following observations:

- Nodes adopt the principle of "First buffer data and forward them, then *later* verify them". As a consequence, if the received packets verify the weak security condition (the used authentication key is not yet disclosed) and are first time received, nodes must buffer/forward them, regardless whether they are really authentic or not. For an attacker, it is easy to broadcast fake packets verifying the weak security condition, knowing the key chain-parameters (disclosure delay $\delta$, time interval duration $T_{int}$, time reference $T_0$) of the source.
- The broadcast source uses a single key-chain to authenticate its data, where the disclosure delay $\delta$ of the key-chain is generally chosen to be greater than the maximum propagation delay in the network. Because sensors must buffer packets of one time interval at least during one disclosure delay ($\delta$ time intervals),the longer is the time interval $T_{int}$, the more data sensors will buffer, and the more severe are the damages caused by a resource-draining DoS attack.

To defend against resource-draining DoS attacks on time-asymmetry based BSAPs, nodes must adopt the principle of "First buffer data, then later verify them, and only if authentic, forward them". In other words, we must achieve a hop-by-hop verification of broadcasted data, to limit the damage of resource-draining DoS attack to the one-hop neighbors of the attacker, as do BSAPs achieving immediate authentication [9] [6], which are based on digital signatures. However, using a single key-chain only, sensors can not adopt this principle, because there will be one key-disclosure delay for the entire network. Thus, we need several independent key-chains, with distinct key-disclosure delays, where the number of the used key-chains is equal to the maximum depth of the network in number of hops. Assuming the depth of the network is $l$ hops, the BS uses $l$ distinct key-chains $\{K_n^1\}$, ..., $\{K_n^l\}$, to authenticate its data, where each key chain $\{K_n^r\}_{r=1,...,l}$, is used for the $r$-hop neighbors of the BS. The BS appends $l$ distinct MACs to each packet sent in time interval $I_i$, using the current key $K_i^r$, of each key-chain $\{K_n^r\}_{r=1,...,l}$. Upon reception of packets, each node $u$, depending on its hop-distance $r$ from the BS, buffers the packets and waits for the disclosure of the appropriate key $K_i^r$. Once the key $K_i^r$ is disclosed, $u$ verifies the key and checks the buffered data authenticity, and forwards only authentic data to its neighbors, which are $(r+1)$-hop away far from the BS. In this way, data are verified then propagated in a hop-by-hop way, and resource-draining DoS attack affects only the one-hop neighbors of the attacker. Moreover, nodes need to buffer data during a duration less than the duration of a key disclosure delay like in $\mu$TESLA. In the next section, we detail our solution

## V. $H^2BSAP$ PROTOCOL

Our protocol involves three phases: initialization phase, data broadcast phase and data buffering/verification phase. In the initialization phase, the BS generates the necessary key-chains, and loads sensors with the key-chains parameters. In the data broadcast phase, the BS authenticates data it sends in the current time interval using the current key of each key-chain, and broadcasts the data, then later discloses the keys. In data buffering/verification phase, sensors of level $r$, that are $r$-hop away far from the BS buffer received data until the associated authentication key is disclosed, and only if the data is authentic, they forward them to the $(r+1)$-hop neighbors of the BS.

### A. Initialization phase

The BS divides time into $n$ intervals of equal duration $T_{int}$, where $T_i = T_0 + i \times T_{int}$ is the beginning of time interval $I_i$, and $T_0$ is a time counting reference. Assuming that the depth (i n hops number) of the network is at most $l$, the BS generates $l$ independent one way key-chains $\{K_n^1\}$, $\{K_n^2\}$, ..., $\{K_n^l\}$ of $n+1$ elements each.

A key-chain $\{K_n^r\}_{r=1,...,l}$=($K_0^r$, $K_1^r$, ..., $K_n^r$), is generated from a secret random value $K_n^r$, using a hash function $H$, where $K_{i-1}^r = H(K_i^r)_{i=1,...,n}$. Key-chain $\{K_n^r\}_{r=1,...,l}$ is associated to the $r$-hop neighbors of the BS, where the BS uses $\{K_n^r\}$ to authenticate its data to sensors which are $r$-hop away far from it. To each key-chain $\{K_n^r\}$, is associated a key-disclosure delay $\delta_r$ ($\delta_r$ time intervals), where $\delta_1 < \delta_2 < ... < \delta_l$ (see figure 2).

Sensors are initially preloaded with the parameters of all key-chains $(T_0, T_{int}, \{K_0^r, \delta_r\}_{r=1,...,l})$.

### B. Data broadcast phase

The BS divides data it sends during time interval $I_i$ in several messages $M_{i,j}$, where each message fits in one packet, and where $j$ is the message index in the interval. To broadcast a message $M_{i,j}$ the BS proceeds it as follows:

- First, the BS computes a MAC $MAC_{K_i^l}(i \parallel j \parallel M_{i,j})$ using the current key $K_i^l$ of the last key-chain $\{K_n^l\}$. Then it sets packet $P_{i,j} = i \parallel j \parallel M_{i,j} \parallel MAC_{K_i^l}(i \parallel j \parallel M_{i,j})$.
- Second, the BS recursively authenticates $P_{i,j}$ using the current key $K_i^r$ of key-chain $\{K_n^r\}$, by computing $P_{i,j} \leftarrow P_{i,j} \parallel$

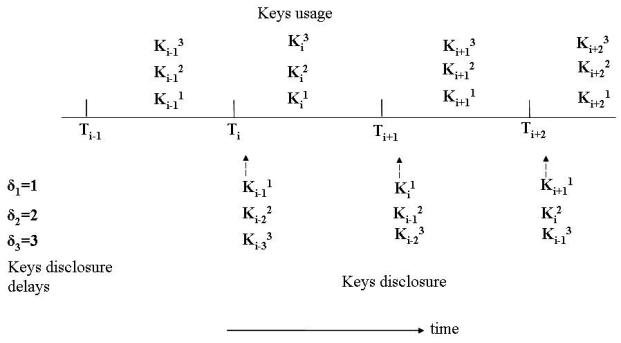


Fig. 2. The H2BSAP protocol, with *l*=3

$MAC_{K_i^r}(P_{i,j})$, for $r = l-1$ down to 1 in this order. Then, the BS sets $P_{i,j} \leftarrow P_{i,j}$ ||Hop-count=1, where Hop-count is used to determine the hop-distance between receivers and the BS.

- Finally, the BS broadcasts $P_{i,j}$ in the network, where $P_{i,j}$ fits in one packet.

Later, the BS discloses keys $K_i^r$, $r = 1, ..., l$, at the beginning of interval $I_{i+\delta_r}$, for $r = 1, ..., l$, to allow the $r$-hop neighbors of the BS verify buffered data sent in interval $I_i$. Key $K_i^1$ is first divulged, then key $K_i^2$, until key $K_i^l$ is the last key to be divulged. Each node receiving a key $K_i^r$, $r = 1, ..., l$, first verifies it and, if authentic, buffers it and forwards it. Likely, all sensors of the network have an up-to-date key, of each key-chain.

### C. Data buffering/verification phase

A node $u$, receiving at instant $t_c$ a packet $P_{i,j}$ (sent on interval $I_i$) proceeds it as follows:

1. If $P_{i,j}$ was already received, $u$ drops it. $P_{i,j}$ is considered to be already received if it is already buffered, or if packets sent on time interval $I_i$ were already *received and verified* by $u$ with a smaller Hop-count value than the Hop-count value contained in the received packet $P_{i,j}$.
2. If $P_{i,j}$ was first received, $u$ checks if $P_{i,j}$ verifies the weak security condition in order to buffer it. To do this, $u$ sets $r = P_{i,j}$.Hop-count, where $r$ indicates the (*claimed*) hop-distance of node $u$ from the BS. Then, $u$ ensures that key $K_i^r$ used in time interval $I_i$ and associated to the $r$-hop neighbors of the BS was not yet disclosed, but key $K_i^{r-1}$ used in time interval $I_i$ and associated to the $(r-1)$-hop neighbors of the BS was disclosed. Weak security condition can be checked by verifying that the following inequalities hold: $T_i + \delta_{r-1} \times T_{int} < t_c + \epsilon_{u,BS} < T_i + \delta_r \times T_{int}$. If the weak security condition is verified, $u$ buffers $P_{i,j}$. Due to the delayed hop-by-hop authentication/forwarding, $r$-hop neighbors of the BS receive packets only and only if $(r-1)$-hop neighbors of the BS have *received* and *verified* the packets. As a consequence, at the time of reception of a packet $P_{i,j}$, with $P_{i,j}$.Hop-count=$r$, $(r-1)$-hop neighbors of the BS are assumed to have received the key $K_i^{r-1}$ and verified the packets, but the BS has not yet disclosed key $K_i^r$.
3. When the BS discloses key $K_i^r$, node $u$ verifies it using the latest disclosed key $K_{i'}^r$ it possesses, where $i' < i$. If $K_i^r$ is authentic, $u$ stores it instead of $K_{i'}^r$, forwards it, then verifies the authenticity of buffered packets $P_{i,j}$, and discards any fake packets. Finally, $u$ updates the authentic packets by deleting the last MAC of the packets (corresponding to the $r$-hop neighbors of the BS), and increasing the Hop-count field of $P_{i,j}$ by 1 (Hop-count ← Hop-count + 1), then forwards them to its one-hop neighbors which are $(r+1)$-hop away far from the BS.

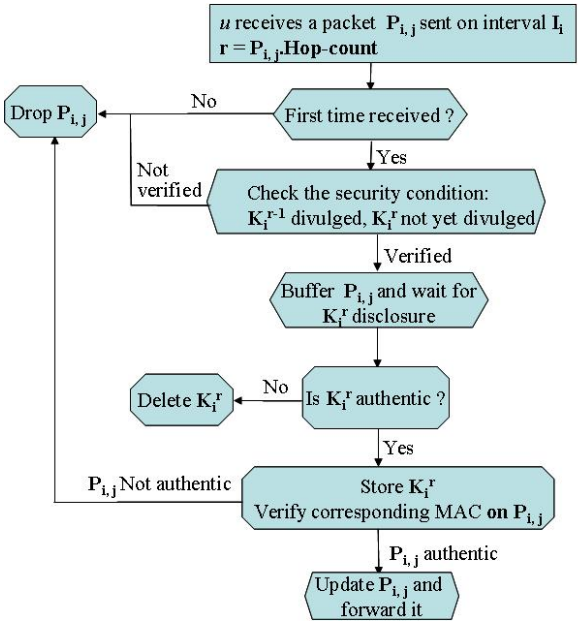


Fig. 3. Data buffering/verification phase

In this way, packets sent by the BS in time interval $I_i$, $i = 1, ..., n$, reach all nodes of the network, in a delayed hop-by-hop way, where $r$-hop neighbors of the BS, first buffer data then later authenticate them, and, only if authentic, forward them to the $(r+1)$-hop neighbors of the BS.

Figure 3 summarizes the data buffering/verification phase for a node $u$

### D. Determining the key-chains disclosure delays

Security condition of time asymmetry based BSAPs, is based on *delayed key-disclosure* and *loosely time synchronization* between the source and the receivers. The key disclosure delay $\delta$ influences receivers performances. A large value of $\delta$ implies that receivers will buffer broadcasted packets for a long duration, resulting on an important storage overhead and a risk of buffer overflow on receivers. A small value of $\delta$, may cause receivers rejecting part or most of the legitimate packets sent by the BS$\delta$ is usually chosen to be equal to or greater than the maximum transporting delay in the network, needed for packets of time interval $I_i$ to travel from the BS to all nodes of the network through. Transporting delay includes the time needed for data to propagate into the entire network through, and any processing delay which applies to the transmitted data en-route through intermediate nodes.

As depicted before, in our protocol each key chain serves a subset of nodes of the network that are far away from the BS by the same number of hops. Thus $\{K_n^r\}_{r=1,...,l}$, is used to authenticate data to the $r$-hop neighbors of the BS. For each key-chain $\{K_n^r\}_{r=1,...,l}$, a disclosure delay $\delta_r$ is associated, where $\delta_1 < \delta_2 < ... < \delta_i < ... < \delta_l$. We'll choose each $\delta_r$ value in such a way that $\delta_r \times T_{int}$ is equal to or greater than the maximum transporting delay of a network of depth $r$. In what follows, we determine the appropriate value of key disclosure delay $\delta_r$ for each key-chain $\{K_n^r\}_{r=1,...,l}$.

Assume that all nodes have the same communication range, which is equal to $R_{max}$ meters (maximum transmission range), and assume

that all nodes have the same output data rate which is equal to $Out_{max}$ bps (maximum data output). $R_{max}$ and $Out_{max}$ values are given by sensors' constructors. For TelosB and MicaZ sensors, $R_{max}$=100 m and $Out_{max}$ =250 Kbps = 250 000 bps. As a consequence, 1-hop neighbors of the BS are at most at distance $R_{max}$ from it, 2-hop neighbors of the BS are at most at distance $2 \times R_{max}$ from it, , and $l$-hop neighbors of the BS are at most at distance $l \times R_{max}$ from it.

Knowing an upperbound of the propagation speed of our wireless signal (bounded by the speed of light in air), we can deduce the maximum propagation delay over 1-hop. Given a speed of light $s_{light} = 2 \times 10^8 m.s^{-1}$, the propagation time of a wireless signal over a one-hop communication range is $t_{R_{max}} = \frac{R_{max}}{2 \times 10^8} s$. This means that a bit needs a time $t_{R_{max}}$, which is *negligeable*, to be transmitted over a one-hop of a distance $R_{max}$. As a consequence, data sent by the BS in time interval $I_i$ (at most the BS sends data during the entire interval), using key $K_i^1$ needs a time $T_{int} + t_{R_{max}}$ to reach the BS's 1-hop neighbors. Thus the disclosure delay can be set to $\delta_1 = [\frac{T_{int}+t_{R_{max}}}{T_{int}} \approx 1]$, Where [X] represents the smallest integer greater or equal to X. However, we can put $\delta_1 = 1$, although $\frac{T_{int}+t_{R_{max}}}{T_{int}} > 1$, because $t_{R_{max}}$ is so extremely small, that the time needed for an attacker, upon reception of the key, to modify some packets of the BS and then send them to the BS's 1-hop neighbors, is greater than the time needed for the original packets to reach all the 1-hop neighbors of the BS.

The other keys disclosure delays $\delta_r$, $r = 2, ..., l$, are computed based on the following parameters:

- $\delta_{r-1}$: the disclosure delay of key chain $\{K_n^{r-1}\}$, relative to the $(r-1)$-hop neighbors of the BS.
- $T_{\{K_n^{r-1}\}}$: the time needed for the current disclosed key of key-chain $\{K_n^{r-1}\}$ to reach the $(r-1)$-hop neighbors of the BS.
- $TV_{r-1}$: the time needed by the $(r-1)$-hop neighbors of the BS to verify the buffered packets of one time interval, once the corresponding key is disclosed.
- $TS_{r-1 \rightarrow r}$: the time needed by the $(r-1)$-hop neighbors of the BS to transmit data to $r$-hop neighbors of the BS.

Indeed, to disclose a key $K_i^r$ the BS used in interval $I_i$ for its $r$-hop neighbors, the BS must have already disclosed key $K_i^{r-1}$ of key-chain $\{K_n^{r-1}\}$ (which corresponds to $\delta_{r-1}$), and the $(r-1)$-hop neighbors of the BS must have *received* key $K_i^{r-1}$ (which corresponds to $T_{\{K_n^{r-1}\}}$) and *verified* buffered packets (which corresponds to $TV_{r-1}$), and *forwarded* valid packets to the $r$-hop neighbors of the BS (which corresponds to $TS_{r-1 \rightarrow r}$). Thus, the transporting delay of a network of depth $r$, can be set as $\delta_{r-1} \times T_{int} + T_{\{K_n^{r-1}\}} + TV_{r-1} + TS_{r-1 \rightarrow r}$, and consequently we can put $\delta_r = [\frac{\delta_{r-1} \times T_{int} + T_{\{K_n^{r-1}\}} + TV_{r-1} + TS_{r-1 \rightarrow r}}{T_{int}}]$, with $\delta_1 = 1$.

Now let explicitly describe each of the above parameters:

- $T_{\{K_n^{r-1}\}} = (r-1) \times ((\frac{80}{250000}) + t_{R_{max}} + Hash_{op})$. When the BS divulges the current 80-bit key of key-chain $\{K_n^{r-1}\}$, the key needs on average a time $(\frac{80}{250000}) + t_{R_{max}}$ to be sent and propagated from hop to hop. In addition, each node receiving the key applies a hash operation on it to verify it, in time $Hash_{op}$.
- $TV_{r-1}$: $TV_{r-1}$ depends on the number of data sent by the BS in each time interval, and on the maximum buffer size of sensors to buffer the data. Assuming each sensor has a buffer of a maximum size of $t$ packets, each sensor needs at most a time of $t \times MAC_{op}$ in-order to verify the buffered data, thus $TV_{r-1}$=$t \times MAC_{op}$. Assuming the BS uses packets of size (29(payload) + 7(header)) bytes=288 bits as in TinyOS, the BS will send at most $t = \frac{Out_{max} \times T_{int}}{288}$ packets during one time interval of duration $T_{int}$.
- $TS_{r-1 \rightarrow r}$: $(r-1)$-hop neighbors of the BS need at most a time $T_{int} + t_{R_{max}}$ to forward data sent by the BS in one time interval to the $r$-hop neighbors of the BS.

Finally, an appropriate value of a key-disclosure delay $\delta_i$, $i = 2, ..., l$, can be written as $\delta_i = \delta_{i-1} + [\frac{((i-1) \times ((\frac{80}{250000}) + t_{R max} + Hash_{op}) + t \times MAC_{op} + T_{int} + t_{R max})}{T_{int}}]$. This value guarantees, with high probability, that when the BS discloses a key $K_i^r$, all the $r$-hop neighbors of the BS are assumed to have received the packets the BS sent in interval $I_i$.

## VI. Security analysis of $H^2BSAP$

In this section, we analyze the security of $H^2BSAP$ according to the following threats and attacks:

- Broadcast source identity usurpation and data modification: can an attacker successfully spoof the identity of the BS, and thus send data on behalf of it so receivers will accept it as valid? Can the attacker successfully modify the data sent by the BS and make receivers accept them as valid?
- Resources-draining DoS attack: Can an attacker launch a resource-draining DoS attack on the network? and what is the portion of the network affected by the attack (number of nodes buffering and forwarding fake packets)?

### A. Broadcast source identity usurpation and data modification

Upon reception of a packet $P_{i,j}$ (sent on interval $I_i$) at instant $t_c$, a receiver node $u$ first checks the weak security condition of the packet $T_i + \delta_{r-1} \times T_{int} < t_c + \epsilon_{u,BS} < T_i + \delta_r \times T_{int}$ before buffering it, where $r = P_{i,j}$.Hop-count and indicates the claimed hop-distance of $u$ to he BS. Later, when the BS discloses the associated authentication key $K_i^r$, $u$ verifies the key, then verifies the authenticity of buffered packets $P_{i,j}$ sent in interval $I_i$. Assuming an attacker sends a fake packet $P_{i,j}^\star$ on behalf of the BS, that verifies the weak security condition, node $u$ will buffer the packet. $P_{i,j}^\star$ can be either a fake message computed from scratch, or obtained from a currently sent packet $P_{i,j}$, by illegally changing any field of $P_{i,j}$ (mainly the **data** $M_{i,j}$, the MAC and the Hop-count fields).

According to the weak security condition, at the instant of reception of $P_{i,j}^\star$, the BS has not yet disclosed key $K_i^r$, with $r = P_{i,j}^\star$.Hop-count, otherwise the packet will be dropped. Thus, the attacker can not compute the right MAC over $P_{i,j}^\star$. As a consequence, when the BS discloses key $K_i^r$, $u$ will check the authenticity of $P_{i,j}{}^\star$, and finding that the computed MAC is wrong, it discards the packet.

As a result, an attacker can not spoof the identity of the BS, or modify packets sent by it, and make receivers accept fake packets as legitimate packets sent by the BS.

### B. Resources-draining DoS attack

Now, let see the impact on the network of a resource-draining DoS attack. The impact can be estimated by the number of affected nodes, that will buffer/forward fake packets. We distinguish here two cases:

- *Case 1*: a single attacker in the network.
- *Case 2*: a set of at least two attackers in the network. The attackers can be several-hop distant, and can communicate via an out-of-band low latency communication channel to launch a distributed resource-draining DoS attack.

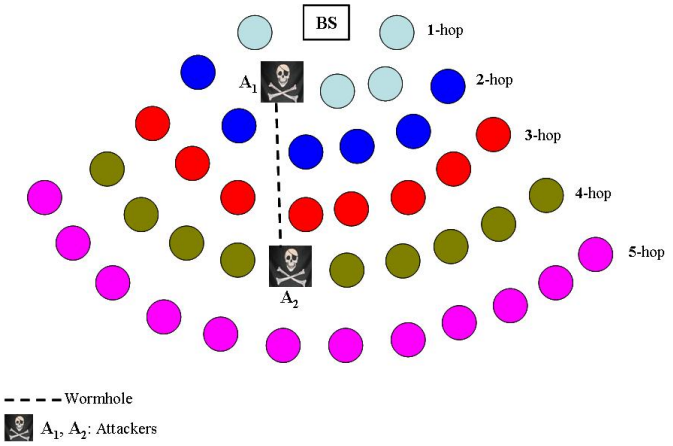


Fig. 4. Resource-draining DoS attack in the presence of two colluding attackers

*1) Resource-draining DoS attack in the presence of a single attacker:* The attacker generates a resource-draining DoS attack on the network, by broadcasting randomly generated fake packets, or modifying/altering legitimate packets sent by the BS.

As described in the previous section VI-A, the one-hop neighbors of the attacker will buffer the fake packets (if they verify the weak security condition, dropped otherwise), and will wait for the disclosure of the corresponding key. Once the key is disclosed, one-hop neighbors of the attacker check its authenticity, then verify the authenticity of all buffered data, discard all the fake packets, and forward only valid packets. As a consequence, resource-draining DoS attack affects just part or all of the attacker's one-hop neighbors, whereas the remaining nodes, which represent the most part of the network, are uninfluenced by the attack. In $\mu$TESLA protocol, a single attacker launching a resource-draining DoS attack, can affect the entire network, where most/all nodes of the network will buffer/forward the fake packets it sends.

*2) Resource-draining DoS attack in the presence of several attackers:* Attackers can operate independently, or can *collaborate and collude* to generate a distributed resource-draining DoS attack on the network. In the latter case, attackers are assumed to have access to a low latency communication channel, allowing attackers which are several-hop far away from each other to directly communicate (wormhole attack, see figure 4).

In case each attacker operates independently, a resource-draining DoS attack launched by each attacker, only affects its one-hop neighbors. Thus, the impact of each attacker is seen as if it was the alone attacker in the network.

In case attackers collaborate, an attacker $A_1$ which is $r$-hop away far from the BS, upon reception of packets $P_{i,j}$ with Hop-count=$r$, will send them to attacker $A_2$ which is $(r+d)$-hop away far from the BS, through the created wormhole (see fig 4), and $A_2$ simply forwards the packets to its neighbors. Assuming that the BS has not yet disclosed key $K_i^r$, we distinguish two scenarios, depending on if packets were sent intact or modified by the attackers:

- If $P_{i,j}$.Hop-count field was modified (increased or decreased), $A_2$'s neighbors will reject the packets, because they do not satisfy the weak security condition.
- If Hop-count field was not modified, but $M_{i,j}$ data or the MAC were modified, $A_2$'s neighbors will buffer packets, because they satisfy the weak security condition. However, they will later drop the packets, when key $K_i^r$ is divulged, because the packets could not be verified.
- If packets $P_{i,j}$ were sent intact without any modification, $A_2$'s neighbors buffer the packets, and later verify their authenticity once key $K_i^r$ will be disclosed. In this case, attackers just speed the delivery of packets to reach nodes which are $(r+d+1)$-hop away far from the BS, through the wormhole. In this case, the Hop-count field of $P_{i,j}$ does not reflect really the hop-distance of receivers to the BS, however, the intact valid packets sent through the wormhole could be easily verified.

## VII. Computation, storage and transmission overheads

Now, let describe the overheads induced by our protocol. We mainly focus on the following overheads:

- BS's computation overhead to authenticate broadcasted packets.
- Receivers' computation overhead to check the packets authenticity.
- Transmission overhead, which depends on the authenticator size generated by the BS per-packet, and forwarded by receivers.
- Storage overhead on the BS and receivers, to store the key materials (key-chain parameters) generated by the BS.
- Verification delay, which is the time packets of one interval $I_i$ are buffered on receivers before being verified.

Figures 5 and 6, summarize the overheads induced by $H^2BSAP$ and $\mu$TESLA, respectively, where values between brackets [] apply only once per time interval. $\mu$TESLA protocol has lower computation, storage and transmission overheads than $H^2BSAP$, because in $\mu$TESLA the BS uses a single key-chain to authenticate its data, whereas in $H^2BSAP$, the BS uses $l$ distinct key-chains. However, our protocol has a lower verification delay than $\mu$TESLA. In $H^2BSAP$, a receiver buffers packets of a time interval $I_i$, for a duration of $T_{int} + r \times Hash_{op}$, whereas in $\mu$TESLA, all receivers buffer packets of a time interval $I_i$ for a duration of $\delta \times T_{int}$. Knowing that performing a hash operation using SHA-1, took around $Hash_{op}$=**35** ms [10], and that $T_{int}$ is in the order of several seconds, it is clear that nodes in $\mu$TESLA buffer packets for a longer duration than in $H^2BSAP$. Assuming our network depth is **10** hops, $T_{int}$=**60**s, and the disclosure delay $\delta$=**2**, nodes in $H^2BSAP$ need to buffer packets of one time interval during at most **60.35**s, whereas nodes in $\mu$TESLA need to buffer packets during **120**s. Due to the long verification delay on $\mu$TESLA, receivers in $\mu$TESLA need to allocate more buffer space to store received packets than in $H^2BSAP$.

## VIII. Limitation and improvements

$H^2BSAP$ protocol mainly suffers from its restriction to static WSN, where nodes are considered to be static once deployed. In addition, like $\mu$TESLA, $H^2BSAP$ suffers from scalability issue, because it does not efficiently support multiple broadcast sources in the network.

$H^2BSAP$ induces some extra computation and transmission overheads in the BS, greater than the induced overheads in the $\mu$TESLA protocol. However, the BS being more powerful than sensors, it can afford these extra overheads in order to prevent the entire network against resource-draining DoS attacks. Moreover, $H^2BSAP$ introduces some extra transmission overhead on sensors, which forward packets with at most $l$ MACs, whereas in $\mu$TESLA each packet carries a single MAC only. To reduce the transmission overhead per-packet, due to the authenticator size, we suggest to use either MACs of reduced size (**4** bytes instead of **8** bytes), or to use Bloom filters [11].

Using reduced MACs of **4**-byte length, will decrease the maximum transmission overhead per packet from $8 \times l$ bytes to $4 \times l$ bytes.

| | Overhead at the BS | Overhead at receiver $u$ |
|---|---|---|
| Computation overhead per packet | $l \times MAC_{op}$ | $MAC_{op} + \lfloor l \times Hash_{op} \rfloor$ |
| Transmission overhead per packet | $l \times \lvert MAC \rvert + \lfloor l \times \lvert key \rvert \rfloor$ | $(l-r) \times \lvert MAC \rvert + \lfloor l \times \lvert key \rvert \rfloor$ |
| Storage overhead | $l \times (n+1) \times \lvert key \rvert$ | $l \times \lvert key \rvert$ |
| Verification delay | - | $T_{\text{int}} + r \times ((\frac{8 \times \lvert key \rvert}{250000}) + t_{R_{\max}} + Hash_{op}) \approx T_{\text{int}} + r \times Hash_{op}$ |

$MAC_{op}$: time to generate/verify a MAC
$Hash_{op}$: time to perform a hash operation
$|key|$: a key byte-length (ex: 8 bytes)
$|MAC|$: a MAC length (ex: 4-8 bytes)
$r$: the claimed hop-distance of a receiver from the BS
$t_{R_{\max}}$: the propagation delay over a one-hop
$n$: the size of a key-chain
$l$: the hop-number depth of the network

Fig. 5. The induced overheads of $H^2BSP$

| | Overhead at the BS | Overhead at receiver $u$ |
|---|---|---|
| Computation overhead per packet | $MAC_{op}$ | $MAC_{op} + \lfloor Hash_{op} \rfloor$ |
| Transmission overhead per packet | $\lvert MAC \rvert + \lfloor \lvert key \rvert \rfloor$ | $\lvert MAC \rvert + \lfloor \lvert key \rvert \rfloor$ |
| Storage overhead | $(n+1) \times \lvert key \rvert$ | $\lvert key \rvert$ |
| Verification delay | - | $\delta \times T_{\text{int}}$ |

$\delta$: the key disclosure delay of the $\mu$TESLA key-chain
$n$: the key-chain length

Fig. 6. The induced overheads of $\mu$TESLA

Having a maximum data payload of **104** bytes for Zigbee-compliant sensors [12], and using **44** bytes of the payload for data transportation, we can support a large WSN of **15**-hop depth, where the size of authenticator per packet is **15×4 = 60** bytes.

Using Bloom filter techniques, the BS first computes a set $S_{i,j}$ of $l$ MACs per-packet $P_{i,j}$. However, unlike the original protocol (see section V-B), where each MAC was computed over the data and the previously computed MACs, each MAC now is computed only over the data $i \parallel j \parallel M_{i,j}$. Then, the BS computes an $m$-bit Bloom vector $V_{i,j}$ over $S_{i,j}$, where $m << 8 \times l \times \mid MAC \mid$ (bits). In addition to its reduced size compared to the size of the set $S_{i,j}$, Bloom vector $V_{i,j}$ supports approximate queries membership over $S_{i,j}$. If a candidate MAC value $MAC' \in V_{i,j}$, then $MAC' \in S_{i,j}$, with *high probability*. Now, each packet sent by the BS, carries as authenticator $V_{i,j}$, instead of the set $S_{i,j}$ of $l$ MACs. When the BS discloses key $K_i^r$, an $r$-hop distant sensor buffering a packet $P_{i,j} = i \parallel j \parallel M_{i,j} \parallel V_{i,j}$, first computes a candidate MAC $MAC' = MAC_{K_i^r}(i \parallel j \parallel M_{i,j})$, then checks that $MAC' \in V_{i,j}$, in order to accept $P_{i,j}$ as valid, with high probability, otherwise it drops it.

## IX. Conclusion

In this paper, we described $H^2BSAP$, a time-asymmetry based BSAP, that efficiently thwarts resource-draining DoS attacks in WSN. $H^2BSAP$ limits the impact of a resource-draining DoS attack, to the one-hop neighbors of the attacker only, by achieving a delayed hop-by-hop authentication/forwarding of broadcasted packets. In addition, $H^2BSAP$ considerably reduces the verification delay on sensors, where each receiver needs to buffer packets of one time interval for a small duration only. Moreover, $H^2BSAP$ introduces some extra overheads (computation, storage, transmission), which are acceptable and which can be further reduced using the improvements we are listing in section VIII. Simulation of $H^2BSAP$ is ongoing, and the results will be published in a next work.

As a future work, we plan to implement the improvements we suggested to the original $H^2BSAP$ protocol, and evaluate them through simulations. Mainly, we will implement the Bloom filter technique in order to reduce authenticator's size per packet, and we will implement the MPR-based broadcasting in order to reduce the transmission overhead on the network. We also plan to study the impact of nodes/BS mobility on our protocol.

## Acknowledgements

Authors are thankful to French ANR (Agence Nationale de la Recherche) agency which funded RNRT project CAPTEUR (2005-2008).

## References

[1] A. Perrig, R. Szewczyk, V. Wen, D. Cullar, and J. D. Tygar, "Spins: Security protocols for sensor networks," in *Proc. of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking*. Rome, Italy: IEEE/ACM, Jul. 2001, pp. 189–199.

[2] J. Hui, *Deluge 2.0 - TinyOS Network Programming*. http://www.cs.berkeley.edu/ jwhui/research/deluge/deluge-manual.pdf: CS Berkley, 2005.

[3] W. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking*. Seattle, Washington, US: ACM/IEEE, Aug. 1999, pp. 174–185.

[4] Y. Zhou and Y. Fang, "Babra: Batch-based broadcast authentication in wireless sensor networks," in *Proc. of IEEE GLOBECOM*. San Francisco, USA: IEEE, Nov. 2006, pp. 1–5.

[5] J. Drissi and Q. Gu, "Localized broadcast authentication in large sensor networks," in *Proc. of the IEEE Int. Conf. on Networking and Services*. Silicon Valley, USA: IEEE, Jul. 2006.

[6] K. Ren, W. Lou, K. Zeng, and P. J. Moran, "On broadcast authentication in wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 6, no. 11, pp. 4136–4144, Nov. 2007.

[7] M. Luk, A. Perrig, and B. Whillock, "Seven cardinal properties of sensor network broadcast authentication," in *Proc. of the fourth ACM workshop on Security of ad hoc and sensor networks*. Alexandria, Virginia, USA: ACM, Oct. 2006, pp. 147–156.

[8] T. Wu, Y. Cui, B. Kusy, A. L. J. Sallai, N. Skirvin, J. Werner, and Y. Xue, "A fast and efficient source authentication solution for broadcasting in wireless sensor networks," in *Proc. of NTMS Conference*. Paris, France: Springer-Verlag, May 2007, pp. 1–12.

[9] P. Ning, A. Liu, and W. Du., "Mitigating dos attacks against signature-based broadcast authentication in wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 4, no. 1, pp. 1–18, Jan. 2008.

[10] H. Lee, Y. Choi, and H. Kim, "Implementation of tinyhash based on hash algorithm for sensor network," *Proc. of World Academy of Science, Engineering, and Technology*, vol. 10, no. 1, pp. 135–139, Dec. 2005.

[11] A. Brodery and M. Mitzenmacherz, "Network applications of bloom filters: A survey," *Internet Math*, vol. 1, no. 4, pp. 485–509, Sep. 2003.

[12] Zigbee, *IEEE 802.15.4: IEEE standard for information technology*. http://standards.ieee.org/getieee802/download/ 802.15.4-2003.pdf: IEEE, 2003.

[13] A. Qayyum, L. Viennot, and A. Laouti, "Multipoint relaying for flooding broadcast messages in mobile wireless networks," in *Proc. of the* $35^{th}$ *Hawaii Int. Conf. on System Sciences*. Big Island, Hawaii, US: IEEE, Jan. 2002, pp. 1–10.