Secured Packet Level Authentication Scheme for Code Update in Multihop WSN

Md Aleemuddin Ghorı, Syed Abdul Sattar

Abstract: Wireless sensor network is an imminent technology and is getting Popularity quickly and a lot of attention because of their low cost solutions and capable to implement in military as well as for civilians. This technology has many applications as well as several environmental monitoring target tracking scientific exploration patient monitoring and data acquisition in hazardous environments. In Wireless Sensor Networks These tiny sensor nodes are deployed randomly in a hostile environment to collect sensor data and hence they are susceptible to outsider attacks therefore security is an important issue. Several security schemes have been proposed to provide the authenticity and integrity for network programming applications but they are either lacks the data confidentiality or they are not energy inefficient as they are based on digital signature. So still there is a need to design a security Scheme to Enhanced the existing security mechanism for providing the authenticity and integrity of program updates in existing network programming protocols.

Keywords: Wireless, Networks, imminent technology, program updates in existing

I. INTRODUCTION

Wireless Sensor Networks (WSNs) is an advanced and emerging technology which has its application across a wide range of industrial scientific and military as well as for the civilian sector example oil and gas-pipelines power distribution networks water distribution networks and nuclear power plants. In multi-hop wireless sensor networks the remote reprogramming is created by changing application necessities and software errors [1]. In this context resource-awareness time-efficiency and the combination of appropriate security solutions are the keys to success and acceptance of a code update mechanism. To satisfy this demand a dependable data dissemination protocol for time-efficient and secure code updates in wireless sensor networks is utmost required. A desirable feature is resistance to node capture and even if a node is compromised and its key material is exposed an adversary should not be able to gain control of the other parts of the network by using this material. Therefore, securing sensor networks against these threats becomes challenging [2] for the researchers and an authentication scheme for network programming is needed to ensure that the adversary should not be able to locate program image.

II. PROBLEM DEFINITION

The existing network programming protocols [3-7] focus on reliable efficient application image dissemination with the minimal end-to-end update latency with the prevalent appearances but provide no security mechanism. The absence of security mechanism on broadcast of application image imposes a vulnerability of installation with arbitrary application image in WSNs. Attacker could just capture one sensor node in WSNs and inject the malicious code image the same way as these network programming protocols operate. Without any appropriate security mechanism the entire WSN could be taken control by the attacker with slight effort by compromising only one sensor node[8].

A. Paper Contribution

Our main aim through this paper is to design a security scheme for network programming protocols by using one-way multiple key chains to secure code update in a multi-hop sensor networks. Our scheme does not involve any asymmetric cryptographic techniques resulting in energy efficient mechanism. In addition the selections of design parameters in the scheme are discussed to maximize the difficulties of different types of attacks against this scheme. This paper establishes the association between the network topology and security Mechanism.

III. DESIGN AND IMPLEMENTATION

This section describes the design of our secure network programming scheme where the Sensor nodes are grouped according to their hop distance to base station in one-way multiple key chains. The sensor nodes in different hop group share different one-way key chain with the base station for packet authentication and integrity check and hence prevent the malicious code update from the nodes in the upstream hop group. Our scheme works at the packet level and consists of two phases. First phase is initialization and key pre-distribution and the second is packet pre-processing and verification phase.

3.1. Initialization and key pre-distribution phase.

Our scheme employs one-way multiple key chains to secure the Deluge protocol. Key chains are based on a function $H$ with the property that its computation is easy while its inverse $H^{-1}$ is extremely difficult to compute. A hash chain with length L is generated by applying $H$ to an initial element repeatedly for L times. (i.e. $0 \leq i \leq L, 1 \leq j \leq S, K_{i,j} = H(K_{i+1,j})$ shown in Fig : 3.1) The last value after $H$ has been applied L times is called the committed value of the hash chain (Fig : 3.1). Before the sensor nodes are deployed the base station constructs S hash chains. It generates S distinct random seed numbers and computes a one-way hash chain with length of $L+1$ starting

Revised Version Manuscript Received on August 28, 2016.

Dr. Md Aleemuddin Ghorı, M. Tech, PhD, Professor, Department of Computer Science & Engineering, Royal Institute of Technology and Science, Chevella (Telangan) -501503, India.

Dr. Syed Abdul Sattar, BE, MTech, PhD (ECE), PhD (CS), Professor & Dean of Academics, Royal Institute of Technology and Science, Chevella (Telangan) -501503, India.
from each seed which is the \((L - i + 1)^{th}\) output of hash function derived from \(j^{th}\) random seed number and is denoted as \(K_{i,j}\). Sensor nodes are divided into \(S\) groups according to their hop distance to the base station (see Fig. 3.2). The committed value of the \(i^{th}\) key chain \(K_{0,i}\) corresponding to the hop distance is pre-distributed to nodes in the \(i^{th}\) hop group before the keys are deployed.

Table 3.1: The notations in Section 3.3 and Section 3.4

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>(H(B))</td>
<td>(H(B)) Hash of (B)</td>
</tr>
<tr>
<td>(A</td>
<td></td>
</tr>
<tr>
<td>(K_a)</td>
<td>(K_a) Shared key between node (a) and base station</td>
</tr>
<tr>
<td>(E_{K_a}(M))</td>
<td>(E_{K_a}(M)) Encryption of message (M) with symmetric key (K_a)</td>
</tr>
<tr>
<td>(D_{K_a}(M))</td>
<td>(D_{K_a}(M)) Decryption of (M) with symmetric key (K_a)</td>
</tr>
<tr>
<td>(A \rightarrow B : P)</td>
<td>(A \rightarrow B : P) Node (A) unicasts a packet</td>
</tr>
</tbody>
</table>

**Figure 3.1: Establishment of multiple one-way key chains.**

3.2. **Phase 2: packet pre-processing and verification**

After the node deployment once the base station has a program update to broadcast the program image is divided into fixed-size packets like \(P_0, P_1, \ldots, P_{L-1}\) (shown in Fig. 3.4). The base station pre-processes these packets so that the sensor nodes can verify the packets accordingly.

3.3. **Packet pre-processing**

First we describe the packet pre-processing of the very first packet of program image \((P_0)\). The committed value of the first key chain \((K_{0,1}\) in Fig. 3.1) is used to encrypt the next key element in the order of key dissemination \((K_{1,1}\) in Fig. 3.1). The encrypted result \((E_{K_{0,1}}(K_{1,1}))\) is the key update segment for the first hop group. Then \(K_{1,1}\) is concatenated with \(P_0\) and the result is hashed yielding the packet authentication segment \((H(P_0||K_{1,1}))\). The key update segments and packet authentication segments the successive hop groups are generated in the same way using their corresponding key chains. Finally all these segments are concatenated with \(P_0\) as shown in Fig. 3.3, giving the first packet to be transmitted. The way in which the key update and packet authentication segments are concatenated with the data packet is used in a countermeasure against tunnel attack and this packet pre-processing procedure is repeated for successive packets in the image to be broadcast.
Figure 3.2: Grouping of nodes when S = 2, A₁,₂… are nodes in the one-hop group while B₁,₂… are nodes in the two-hop group.

The packet pre-processing result (P′ᵢ) for the iᵗʰ (∀ i, 0 ≤ i < L) packet (Pᵢ) propagated to S hops is

\[ P′ᵢ = Pᵢ || Tᵢ, s || Uᵢ, s || … || Tᵢ, l || Uᵢ, l, \]

Where

\[ Tᵢ, j = H M A C (Kᵢ₊₁, j, Pᵢ) \quad (3.1) \]

And

\[ Uᵢ, j = E_{Kᵢ, j} (Kᵢ₊₁, j) \quad (3.2) \]

The complete mechanism of packet pre-processing is illustrated in Fig. 3.4, while the complete verification of iᵗʰ packet in node A (A ∈ Gᵢ) is shown in Algorithm

3.2.2. Packet verification

The packet verification for the first data packet destined to the first hop group will be described. The verification of subsequent packets in the other hop groups use the same procedure with keys corresponding to those that were used in the packet pre-processing. After the pre-processing of a packet and the respective concatenation (i.e., \( P'_0 = P_0 || T_{0,1} || U_{0,1} \)), \( P'_0 \) is transmitted to nodes in the first hop group. After retrieving the correct group information from \( P'_0 \) (i.e., the sensor node parses the right fields \( T_{0,1} \) and \( U_{0,1} \)), the sensor nodes verify the key update segment (\( U_{0,1} \)) and packet authentication (\( T_{0,1} \)) segments as follows.

3.4. Key update segment

The sensor node decrypts \( U_{0,1} \) with \( K_{0,1} \), i.e., \( D_{K_{0,1}} (U_{0,1}) \). The sensor node checks if \( H (D_{K_{0,1}} (U_{0,1})) \) is the same as \( K_{0,1} \) (line 7 in Algorithm). If it is, the authenticity of \( U_{0,1} \) is ensured and \( K_{0,1} \) is replaced by \( K_{1,1} \) for the next packet verification (line 8 in Algorithm). Otherwise, this packet is discarded (line 17 in Algorithm).

3.3.1. Packet authentication segment

After the sensor node retrieves the authenticated packet \( K_{1,1} \) from \( U_{0,1} \), it performs the HMAC operations with \( K_{1,1} \) as the key over P₀ to see if it matches \( T_{0,1} \) since a simple hash function has been found problematic in integrity verification (line 10 in Algorithm). If it does then the integrity of packet is assured (line 12 in Algorithm). Otherwise this packet is discarded (line 14 in Algorithm).

Figure 3.4: Packet pre-processing in the base station \( U_{1,j} = E_{K_{1,j}} (K_{i+1,j}) \), \( T_{i,j} = H M A C (K_{i+1,j}, P_i) \).
Secured Packet Level Authentication Scheme for Code Update in Multihop WSN

The $i^{th}$ ($1 \leq i \leq L$) packet ($P_i$) verification in node ($A$) where $A \in G_j$ ($1 \leq j \leq S$)

1. base station: $\text{Key\_Section\_i\_j} \leftarrow E_{K_{i\_j}}$(key链条生成
2. base station: $P_{i\_K_{i\_j}} \leftarrow H(P_{i\_K_{i\_j}})$
3. base station: $P_i \leftarrow P_{i\_K_{i\_j}} || K_i$ [验证的密文组织]

We implemented our scheme as an extension to the Deluge network programming protocol that is distributed with TinyOS [8, 9]. In Deluge the base station programs are distributed in Java and run on a PC. These Java programs are used to construct and broadcast the new code dissemination packets into the WSN [10]. The programs for the sensor nodes are nesC programs. They are used to receive the packets of broadcast program images and update the appropriate node program image slots. On the base station side we used the JCE provider in the Bouncy Castle Crypto Application Program Interface (API) for hash function (SHA-1) key generation and packet encryption (RC5). On the sensor node side we employed the hashing operation from Tiny ECC and packet decryption from Tiny sec. The committed values of the key chains are generated in the base station and pre-distributed on the nodes in the corresponding hop group before deployment. The operations of packet pre-processing are implemented in ImageInjector.java while the operations of packet verification are implemented in delugePageTransferM.nc. The reference codes for the base stations and sensor nodes are available in and respectively. Our evaluation consists of two parts. We first evaluated our scheme using the TOSSIM Then we tested our scheme in a network of Tmotes. We compare our scheme with the original Deluge and Sluice in terms of end-to-end latency.

3.5. Implementation and performance evaluation

We implemented our scheme as an extension to the Deluge network programming protocol that is distributed with TinyOS [8, 9]. In Deluge the base station programs are distributed in Java and run on a PC. These Java programs are used to construct and broadcast the new code dissemination packets into the WSN [10]. The programs for the sensor nodes are nesC programs. They are used to receive the packets of broadcast program images and update the appropriate node program image slots. On the base station side we used the JCE provider in the Bouncy Castle Crypto Application Program Interface (API) for hash function (SHA-1) key generation and packet encryption (RC5). On the sensor node side we employed the hashing operation from Tiny ECC and packet decryption from Tiny sec. The committed values of the key chains are generated in the base station and pre-distributed on the nodes in the corresponding hop group before deployment. The operations of packet pre-processing are implemented in ImageInjector.java while the operations of packet verification are implemented in delugePageTransferM.nc. The reference codes for the base stations and sensor nodes are available in and respectively. Our evaluation consists of two parts. We first evaluated our scheme using the TOSSIM Then we tested our scheme in a network of Tmotes. We compare our scheme with the original Deluge and Sluice in terms of end-to-end latency.

IV. CONCLUSION

Current network programming protocols can propagate and disseminate the application image in sensor networks reliably and efficiently. However security issues did not play an important role in design of these systems in early stages. As a result an attacker could simply propagate and disseminate its own malicious code image in sensor network pretending to be a genuine entity. Although some security schemes have been proposed to defend against the malicious code distribution they are either not power efficient or lack the consideration on code confidentiality and integrity and hence exploiting on the security design itself. Our future work will include the design of security models for network programming to improved security and scalability in multihop wireless sensor networks.

REFERENCES