

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & MANAGEMENT DISSEMINATION SECURITY OF UPDATING CODE IN WIRELESS SENSOR NETWORKS

Unes, Bazband

Islamic Azad University of Kuhdasht

ABSTRACT

Code dissemination is the process of propagating a new program image or relevant commands to sensor nodes via the wireless medium. It has become necessary in Wireless Sensor Networks (WSNs) because program image updates may be required for bug fixes or to provide new functionalities after a WSN has been deployed. Code dissemination protocols provide a convenient way to update program images via wireless communication. Due to the open environment in which Wireless Sensor Networks (WSNs) are typically deployed, it is important that a code dissemination protocol ensures that a program image update can be authenticated as coming from a trusted source. In some applications it is also required that the data be kept confidential in spite of the possibility of message interception. Authentication and confidentiality are implemented through cryptographic operations which may be expensive in power consumption, making a protocol with these features vulnerable to attack by an adversary who transmits forged data, forcing nodes to waste energy in identifying it as invalid (i.e., a signature-based DOS attack).

Keywords- *Wireless sensor networks, Code Dissemination Protocol, Security.*

I. INTRODUCTION

A Wireless Sensor Network (WSN) is composed of small and highly resource-constrained sensor nodes that monitor some measurable phenomenon in the environment, e.g., light humidity, or temperature [1]. Their long-life and large-scale design, various deployment fields, and changing environments necessitate the feasibility of remote maintenance and in-situ reprogramming of sensor nodes using a so-called Over-The-Air Programming (OTAP) protocol. In particular, if sensor nodes are inaccessible after deployment, a reliable OTAP is crucial. The basic concepts of security in WSN consist of authentication Privacy/Confidentiality, Integrity and Subsections Availability [2]. These key factors can be used to determine if a protocol for remote reprogramming scheme is secure and authorized in this work, we analyze the existing security models used in over-the-air dissemination of code updates for possible vulnerabilities, and then, we provide a set of countermeasures, correspondingly named Security Model Requirements. Based on the analysis, we concentrate on Selug. Security wise, it is one of the most promising OTAP protocols.

II. RELATED WORK

Munivel&Ajit represents that MPKI as a reliable data dissemination protocol for propagating large amounts of data from one or more source nodes to other nodes over a multi hop in a network. This protocol takes benefits of using Public Key Cryptography in combination of Symmetric Cryptography, so that it covers the mentioned issues with storing symmetric key in nodes and generally consists of two handshakes which generally shapes the infrastructure of this scheme; one is node to node handshake and the other one is base station to node handshake. The most significant disadvantages in this scheme is an initial overhead caused by creation of a temporary shared key using PKI to be used for symmetric cryptosystem and not protected against DoS attacks. [3] Patrick E. Lanigan presents a new protocol Sluice, which is an extension aiming for the progressive, resource-sensitive verification of updates within sensor networks by exploiting a single digital signature per update, along with a hash-chain construction over pages of the update. It provides enhanced security preventing malicious nodes from propagating or installing malicious updates on uncompromised nodes within the system. [4] The downside of this algorithm is that it does not preserve confidentiality of updates and also it is not resistant to the following attacks: Replay attack, DoS attack and Battery-Drain attack. Hyun et al. have proposed Seluge, where they have used a new, efficient pattern for constructing hash values out of the data packets. Figure 1 illustrates their protocol and describes the pattern this scheme uses. The hash value of each packet in page P is inserted to its corresponding packet in the previous page (P-1). This procedure takes place, until all the hash values of the packets from the last page to the first page are created and placed in their relative packets. [4] Consequently, Seluge has most mitigation against weaknesses but it also has a very high

overhead and is susceptible to Battery-Drain attacks and its DoS resilience is not sufficient for mission-critical applications of WSNs.

III. PROTECTED MULTI AUTHORIZING REPROGRAMMING PROTOCOL

3.1 Seluge Reprogramming Protocol

The key contribution of Seluge is a novel way to organize the packets used to distribute new code images. By carefully arranging code dissemination data items and their hash images in packets, Seluge provides immediate authentication of each packet upon receipt, without disrupting the efficient propagation mechanisms used by Deluge. Thus, it can defeat the DOS attacks exploiting authentication delays. This protocol proposes to attach all the hash values of page 1 sequentially to create Hash Values = $H(\text{Pkt1},1) \dots H(\text{Pkt1},N)$, and then, these values will be split into fragments, where K will be the minimum value that fits in [Equation \(1\)](#). Note that resulting M is the number of leaf nodes of the Merkle hash trees the block length of the hash algorithm in bytes. [4]

3.2. The Limitations of Seluge's Merkle Tree

Referring to [Equation \(1\)](#) defined in Section A, in Seluge, to identify a value of M , we have to fill the equation with the default values of $N = 48$, $|H(0)| = 8$ and payload size = 72. Authors of Seluge have used a truncated 64-bit SHA-1, which is more vulnerable to collision attacks compared to 160-bit SHA-1. However using Seluge's default values, the value of M will be equal to eight ($k = 3$). Unfortunately, for situations where security must be at the highest possible level (e.g., battlefields, healthcare systems), using 64-bit SHA-1 is not recommended. Furthermore as shown in [Equation \(2\)](#), Seluge cannot afford to use 160-bit SHA-1, because no value of integer number (K) will satisfy [Equation \(1\)](#) when $|H(0)| = 20(160 \text{ bit})$. This statement remains true, even if we use a maximum payload size instead of Seluge's default value of 72:

The only solution for this problem is to decrease Seluge's default value of the number of packets per page from $N = 48$ to a possible maximum of $N = 17$, which then will result in the K to satisfy [Equation \(1\)](#) with a value of four, consequently creating a Merkle hash tree with = 16 leaf nodes. This value for N is an awkward selection, because it will highly degrade the performance of Deluge as the underlying dissemination protocol.

IV. THREAT MODEL

An attacker has a powerful set of devices, both in terms of computation and communication medium. As node communications are based on a wireless medium, it is assumed that there is an untrusted and insecure channel in which an attacker can eavesdrop, inject, delay, modify or remove any packet transmitted. The base station cannot be compromised by attackers; therefore, the public/private key pair is always safe. Jamming attacks are possible on the network, and since a jamming attack is out of the scope of this study, it is assumed that the adversary cannot perform a permanent jamming without being detected and removed. The attacker might try to disseminate malicious code updates into a network or also replay previously sent packets (replay attack). The attacker also might perform denial of service (DoS) attack on the network.

V. PROPOSED APPROACH

Studies made by authors shows that a successful way for authenticating image code has to be in packet-level. This requires assigning hash value of each packet in previous packets using a pre-defined pattern so that each packet being received in nodes can be immediately verified by its previously received hash value. To achieve this, similar to Seluge the hash value of each packet in last page P is inserted to its corresponding packet in previous page ($P-1$). This procedure takes place until all the hash values of packets from the last page through first page are hashed and placed in their relative packet. This procedure takes place until all the hash values of packets from the last page through first page are hashed and placed in their relative packet. Hash values of Page 1 which do not have any corresponding packet for storage will be then called Page 0. this method will only be effective on verification of data from Page 1 through P , but packets in Page 0 must also be disseminated in a way that they can be immediately verified. In section III, it is shown that using Merkle Hash Tree does not comply with our assumption of a mission-critical network therefore unlike Seluge, A new method for verification of packets in Page 0 is introduced. This method is taking benefit of a Message Authentication Code use symmetric Cryptography for both Integrity preservation and packet verification (using MAC) of hash packets in Page 0.

5.1. Key Agreement

Importance of having a temporary session key per each communication link is highlighted, therefore a key exchange mechanism has to be chosen to securely exchange a symmetric key in the beginning of propagation phase. Since it is needed to make sure this key generation is triggered by an authenticated source, the key exchange algorithm must be integrated into asymmetric signature verification process. The best method which suits these conditions and also has an acceptable security level is Diffie-Hellman Key Agreement Algorithm


5.2 Code dissemination Process

After a code update is available, Base Station will start to send advertisement packets until a node requests for update code. Since session key is needed prior to propagation phase in order to encrypt data before hash calculation of packets in Data Verification Method, Diffie Hellman key agreement process will be the next step that node and BS will take. The node after receiving an advertisement packet will choose its sender and ask for signature packet; then BS will generate a very large prime number P , G (a primitive root mod P and $G < P$) and will select a random number $X_S < P$ and a random 160-bit number (I) , then BS will calculate. Server will sign packet containing (I, G, P, X_S) (To keep freshness), $I, G, P,$ and X_S with his and will send to node.

Node will receive that packet and will decrypt its content using K which is configured in nodes in priori. The value of $I, G, P,$ and X_S will temporarily be kept in RAM storage of node. Before node takes any other action it will compare the value of X_S with the value in previous disseminated code, if the new value is less or equal to old value, it means that a reply attack is ongoing; therefore node will stop the current process and will notify the BS about this attack. Otherwise, if new value of X_S is greater than old value, node will continue. In this phase node needs to do the rest of Diffie-Hellman key agreement procedure by selecting a random number of X_N less than P and calculate, and a random 160-bit number (I) where K will behave as like as Session Key. Then node will create packet data in following format and will send to server: $\{X_N, I\}$

Upon receiving previous packet, server will first calculate, and then checks if the generated Session Key (K) is authentic using equation (3) and if it was a valid session key then it will run calculation (4) to extract X_N ; if generated session key found to be bogus then it means an attacker is trying to send forged packets to BS, since BS is assumed to be powerful it is very unlikely for an attacker to perform a

$\{X_N, I\}$ (3)



(

DOS attack by sending many reply packets from a malicious node. Now server has the session key therefore it can start preparation of code update packets BS will fragment code updates in comply with Figure 1, and then default behavior of original Seluge will be applied on generation of hash packets. The key difference is that in design of Proposed model, to preserve confidentiality it is proposed to use a temporary session key to encrypt data packets to prevent malicious nodes from understating behavior of network by disassembling code update In next step, bs sending hash packets of page 0 from 1st through Ith to the node using following format:

}

Node then can verify the integrity of receiving hash values using MAC and authenticity of them using the fact that session key used in MAC is generated by help of server signature and Diffie Hellman key agreement technique. When all the packets are delivered to the node, it will put all the pages together and will decrypt the data file to extract actual update image.

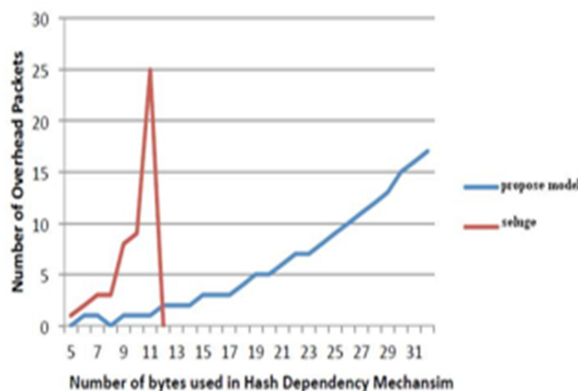


Fig. 1. Overhead packets using different hash block size

VI. Analysis

6.1. Overhead Improvement

mechanism employed firstly merges all the hash values of page 1 to create packets of Page 0; Number of packets in Page 0 (NP0) in absence of required bytes for MAC can be calculated using equation (5)

$$(5)$$

Then each packet in page 0 needs to be merged with a number of MAC bytes for authentication and integrity check purposes. This will create the initial overhead of proposed mechanism and can be calculated using equation(6):

Considering above facts, two major differences will be identified so that with increase to hash strengthen used in these algorithms, original Seluge will have an exponential increase in number of overhead packets (O) while proposed approach will have a linear increase (Figure 1).

Original Seluge will have 25 packet overhead for only hash block size with 11 bytes while with the same value proposed model has only 1 byte overhead.

VII. CONCLUSION

The proposed scheme reduces the DoS and wormhole attacks in network, also it supported algorithms hashed from the larger block sizes. Compared with the previous version of the model is less processing overload.

REFERENCES

- [1] Li B., Batten L., Doss R. *Lightweight Authentication for Recovery in Wireless Sensor Networks. Proceedings of 2009 5th International Conference on Mobile Ad-Hoc and Sensor Networks; Fujian, China. 14–16 December 2009;*
- [2] Chien T.V., Chan H.N., Huu T.H. *A Comparative Study on Operating System for Wireless Sensor Networks. Proceedings of 2011 International Conference on Advanced Computer Science and Information System (ICACSIS); Jakarta, Indonesia. 17–18 December 2011.*
- [3] Munivel E., Ajit G.M. *Efficient Public Key Infrastructure Implementation in Wireless Sensor Networks. Proceedings of 2010 International Conference on Wireless Communication and Sensor Computing (ICWCSC); Chennai, India. 2–4 January 2010; pp. 1–6K. Elissa, "Title of paper if known," unpublished.*
- [4] Hyun S., Ning P., Liu A., Du W. *Seluge: Secure and DoS-Resistant Code Dissemination in Wireless Sensor Networks. Proceedings of 2008 International Conference on Information Processing in Sensor Networks; St. Louis, MO. 22–24 April 2008; pp. 445–456.*

- [5] *Deng J., Han R., Mishra S., Dengcoloradoedu J., Hancoloradoedu R. Secure Code Distribution in Dynamically Programmable Wireless Sensor Networks. Proceedings of the 5th International Conference on Information Processing in Sensor Networks; Nashville, TN, USA. 19–21 April 2006; pp. 292–300.*
- [6] *Handschuh H., Knudsen L.R., Robshaw M.J. Analysis of SHA-1 in Encryption Mode. Proceedings of the Cryptographers' Track at RSA Conference; San Francisco, CA, USA. 8–12 April 2001; pp. 70–83.*