I.J. Wireless and Microwave Technologies **2012, 1, 1-9**
Published Online February 2012 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijwmt.2012.01.01

Available online at http://www.mecs-press.net/ijwmt

Efficient Homomorphic Hashing Approach for Secure Reprogramming in Wireless Sensor Networks

[a]Yu Zhang ,[a] Xing She Zhou, [b]Yee Wei Law ,[b] Marimuthu Palaniswami

[a] *School of Computer Science Northwestern Polytechnical University Xi'an, China*
[b] *Department of Electrical and Electronic Engineering The University of Melbourne Melbourne, Australia*

Abstract

While existing solutions can provide authentication services, they are insufficient for a new generation of network coding-based reprogramming protocols in wireless sensor networks. We present a security approach that is able to defend pollution attack against reprogramming protocols based on network coding. It employs a homomorphic hashing function and an identity-based aggregate signature to allow sensor nodes to check packets on-the-fly before they accept incoming encoded packets, and introduces an efficient mechanism to reduce the computation overhead at each node and to eliminate bad packets quickly. Castalia simulations show that when the 5% of the nodes in a network of 100 nodes are rogue, using our approach, the efficiency of the secure reprogramming protocol based on network coding improves almost ten-fold for a checking probability of 2%.

Index Terms: Sensor Network Security; Reprogramming Protocols; Network Coding; Pollution Attacks; Homomorphic Hashing.

1. Introduction

Reprogramming over wireless links has become crucial service for wireless sensor networks (WSNs) due to the need of fixing bugs or providing new functionalities after a WSN is deployed [1]. Several reprogramming protocols [2], [3], [4] have been proposed for WSNs. Deluge [2] is currently the benchmark reprogramming protocol since it has been included in TinyOS distributions, making it easily accessible by the WSN community.

In recent years, the technique of network coding has been applied to improve the performance of existing reprogramming protocols, resulting in such protocols as Rateless Deluge [5], AdapCode [6] and Synapse [7]. Like Deluge, these protocols still divide the new code image into pages, and divide a page into packets. The advantage of network coding based protocols is that they achieve significant savings of energy and communication, especially when packet loss rate or network density increases. There is hence much incentive in using network coding-based protocols for reprogramming in WSNs.

Corresponding author:
E-mail address: [a] {zhangyu, zhouxs}@nwpu.edu.cn, [b] { ywlaw , palani}@unimelb.edu.au

However, the network coding-based reprogramming protocols have a potential problem in hostile environments, where an adversary may attack the sensor nodes by launching famous pollution attack [8], in which a malicious node can send bad encoded packets that consist of bogus data, which leads to erroneous decoding of a large part of the original data upon retrieval. For reprogramming protocols that are not based on network coding, several security approaches [9], [10], [11], [12] have been proposed. All these approaches are extensions to Deluge. The technique common to them is the use of a single digital signature and off-line hash chains [13]. This technique is not sufficient for network coding-based reprogramming protocols because each sensor node produces unique encoded packets which cannot be signed by a base station.

The goal of this paper is to propose a new security approach that is able to defend pollution attack against reprogramming protocols based on network coding. To achieve this goal, we employ a homomorphic hashing function and an identity-based aggregate signature to allow sensor nodes to check packets on-the-fly before they accept incoming encoded packets, and introduce an efficient mechanism to reduce the computation overhead at each node and to eliminate bad packets quickly.

This paper is organized as follows. Section II clarifies our assumptions and the threats which this proposal based. Section III describes our approach in detail. Section IV details efficient mechanism. Section V presents the simulation results of our schemes. Section VI summarizes related work. Section VII concludes the paper.

2. ASSUMPTIONS AND THREAT MODEL

2.1. Assumptions

We assume the source of the code images, i.e., the base station, is a powerful node (e.g., a laptop) with sufficient energy supply and the only trusted entity in the network. We assume while each sensor node is resource constrained, it has sufficient memory to store the security mechanisms our approach adds. We assume the packet size is large enough to hold a signature and other information required by a signature packet. This can be satisfied on sensor platforms with IEEE 802.15.4 compliant radios [14], where the maximum payload size is 102 bytes.

2.2. Threat Model

We assume that individual sensor-nodes are untrusted. An adversary may introduce its own nodes in a network, or it may capture, compromise and re-introduce existing nodes in the network, as sensor nodes are generally not tamper-resistant. With those capabilities, the adversary attempts to launch pollution attacks to reduce the performance of the network and to consume the limited resources (e.g., battery power, memory) on sensor nodes.

3. PROPOSED APPROACH

We choose the homomorphic hashing function based on the hardness of discrete logarithm [15], the message space is F_p^d, where p is a prime number and $p \approx 2^{160}$ for 80-bit security. To check incoming encoded packets on the fly, sensor node i computes a raw hash value $h_i = H(pkt_i)$, where pkt_i is a raw data packet, and uses a method to sign h_i, which will be described later, in a way that allows verifiers to verify the authenticity of h_i. When a verifier receives an incoming packet $pkt = \sum_{i=1}^{j} \alpha_i pkt_i$ along with j pairs of (raw hashing value, weight) (h_i, α_i), it first determines whether the hashing values are valid, and then verifies the incoming packet's integrity by checking if

$$\prod_{i=1}^{j} h_i^{\alpha_i} \stackrel{?}{=} H(pkt) \tag{1}$$

This work was supported by the National Natural Science Foundation of China under Grant No.60573161 and National Key Technology R&D Program of China under Grant No.2007BAD79B02.

In the following subsection, we describe a method to authenticate h_i.

3. 1 Identity-Based Aggregate Signature

To verify the authenticity of raw hash values at all intermediate sensor nodes in an energy-efficient manner, the best method providing such a property is an identity-based aggregate signature (IBAS), in which different raw hash values can be authenticated by one single aggregate signature.

As far as we know, there is an IBAS scheme [16]which is suitable for secure reprogramming in WSNs although it has one too strong assumption, i.e., all signers must use a same unique string when signing. However, this assumption can be satisfied in the WSN secure reprogramming service because a unique *Imgid* or every code image is known to all nodes.

Let G_1 and G_2 be two cyclic groups of some large prime order q that efficiently support a bilinear mapping $\hat{e}: G_1 \times G_1 \to G_2 : \hat{e}(aQ, bR) = \hat{e}(Q, R)^{ab}, Q, R \in G_1$,

$a, b \in Z$. The IBAS scheme of our approach works as table I.

Any sensor node can verify the signature by checking if

$$\hat{e}(S, P) \stackrel{?}{=} \hat{e}(T, P_{\mathrm{Im}gid})\hat{e}(Q, \sum_{i=0}^{j} P_{i,0} \sum_{i=0}^{j} c_i P_{i,1}) \quad (2)$$

TABLE I. THE IBAS OF OUR APPROACH

Setup: a private key generator (PKG)
1. enerates groups G1 and G2 of prime order q and an admissible pairing $\hat{e}: G_1 \times G_2 \to G_2$.
2. chooses an arbitrary generator $P \in G_1$.
3. picks a random $s \in$ Z/qZ as the master key of PKG and sets $Q = sP$.
4. chooses three cryptographic hash functions H_1, H_2:{0,1}* × G_1 and H_3:{0,1}* × Z/ qZ.
Private key generation:
1. node *i* receives from the PKG the values of $sP_{i,\Rightarrow}$ as its private key for $\Rightarrow \in \{0,1\}$, where $P_{i,\Rightarrow} = H_1$ $(id_i, \Rightarrow) \in G_1$.
Signing: To sigh h*i*, node *i*
1. computes $P_{Imgid} = H_2(\mathrm{Imgid}) \in G_1$.
2. computes $ci = H_3(hi, idi, \mathrm{Imgid}) \in$ Z/qZ.
3. generates random $ri \in$ Z/qZ.
4. computes signature (Si, Ti), where $Si = riP_{Imgid} + sP_{i,0} + cisP_{i,1}$ and $Ti = riP$.
Signature Aggregation:
1. signatures $(Si; Ti)$ for $1 \le i \le j$ can be aggregated into (S,T), where $S = \sum_{i=1}^{j} S_i$ and $T = \sum_{i=1}^{j} T_i$.

4. EFFICIENT MECHANISM

To reduce the cryptographic work at each sensor node while preventing malicious packets from infecting large portions of the network, we introduce cooperative mechanism where nodes cooperate in checking malicious incoming packets. By having a number nodes checking at every point in time and making them cooperate, expensive homomorphic hashing can be applied less frequently, hence, improving the performance of the network. Next we describe the details of how such efficient mechanism works.

We assume that nodes check blocks with probability *Pr*. Encoded packets that pass the check are marked as safe, while encoded packets that have not yet been checked are kept in an insecure window. Encoded packets are checked in batches. The batch window is equal to the insecure window. Whenever a node verifies its insecure window, valid encoded packets are marked as safe and the insecure window is reset.

One possible solution to improve computation time is to verify encoded packets in batches, either probabilistically or periodically. In such solution nodes do not check every encoded packet, but they check a window of encoded packets all at once. Batching is possible thanks to the homomorphic property of the hashing functions. Let's assume we have a window of *J* encoded packets to verify all together. We can build a batched encoded packets *pktr* as a linear combination of all *J* encoded packets. Let *J* = (<*pkt*1*; c*1>,…, <*pktJ, cJ*>), and let the resulting batched packet $pkt_r = \sum_{i=1}^{J} pkt_i$, with combined coefficient vector $c_r = \sum_{i=1}^{J} c_i$. The advantages of using batching is that nodes only need to check one *pktr* packet rather than *L* packets. However, this batching scheme is exposed to a specific byzantine attack, which is called the pairwise byzantine attack [17]. To solve this problem, we create a batched encoded vector $pkt_r = \sum_{i=1}^{J} a_i pkt_i$, using a random set of coefficients, where $c_r = \sum_{i=1}^{J} a_i c_i$. Doing this, it's difficult for an attacker to launch such pairwise byzantine attack.

In our efficient mechanism, sensor nodes do not rely on other nodes to mark encoded packets as safe. However, they actively cooperate with other nodes to detect malicious encoded packets. Whenever a sensor node detects a malicious incoming encoded packets, it sends an alert message to all its neighbors. To prevent sensor nodes that have not been infected from processing the alert message, a given node keeps an insecure-activity table with the ID of i) those nodes that downloaded incoming packets encoded with insecure window encoded packets, and ii) those nodes that delivered the encoded packets inside the insecure window. Alert messages are propagated from one node to another until all infected nodes are informed. If the insecure window is empty, alert messages are not processed. Alert messages are processed as soon as they are received. However, alert messages are only propagated after the node is convinced that a malicious encoded packet exists. Duplicated alert messages can be received for the same malicious packet since overlays often contain loops. However, such duplicated messages will be discarded when i) the insecure window is empty, or ii) the duplicate message comes from a sensor node that is not in the insecure activity table. In addition to alerting its neighbors, a sensor node takes the following actions: i) it puts encoded packets in the insecure window in isolation to be checked and cleaned in the background, ii) it stops using encoded packets in the insecure window for network coding, and iii) it starts checking encoded packets with probability one until the insecure window is secured and cleaned, thus, preventing new malicious incoming packets from infecting the reprogramming system.

5. PERFORMANCE EVALUATION

In this section we show that the performance improvement of efficient mechanism in detecting malicious encoded packets when network coding is used for reprogramming.

We have built a simulator by Castalia [18] that allows us to study the damage that malicious nodes can cause in reprogramming system in WSNs under different settings (e.g. new generation programming protocol based on network coding, classic reprogramming protocol, or non-cooperative and cooperative environments).

We let simulator of Castalia generate the overlay topology of the network, where each sensor node has a constant number of neighbors *k* and the population size was fixed to 50-100 nodes. We randomly choose a

portion of malicious nodes that generate malicious encoded packets to attack the reprogramming system. The percentage of the malicious nodes varied between 5-20% in our experiments. We also varied the percentage of bad packets injected by a particular malicious sensor node.

The normal sensor nodes cooperate to disseminate a new code image that has a large number of packets. The reprogramming system uses either protocol based on network coding, or classic protocol (e.g. Deluge) without coding. The simulator uses a page-by-page dissemination strategy and in each page-round each sensor node can receive and decode at most one page packets. In each page-round, each sensor node with probability p verifies the validity of the encoded packets and, if one or more of them are corrupted, then the node removes them. If cooperative protection is used, then, sensor nodes behave as described in Section IV to alert other infected sensor nodes, so that they also check their content. We measure the number of transmissions of correct encoded packets and the number of corrupted encoded packets.

5. 1 Impact of the Probability of Checking

Fig.1 and Fig.2 respectively shows that the percentage of valid encoded packets and the number of infected nodes as a function of the probability of checking. We consider a network of 100 nodes in which 10% of them are attackers and send bad packets at a rate of 10%.

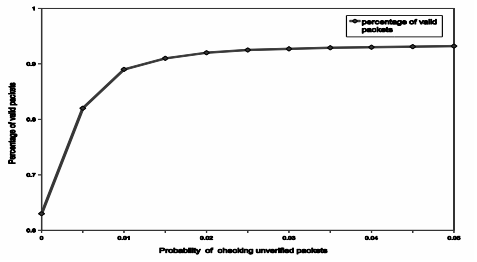

Figure 1 Relationship between percentage of valid packets and the probability of checking

Fig.1 shows that the benefits of collaboration are achieved even for a minimum probability of checking (e.g. 1% provides 89% efficiency). Larger probability of checking results in small increase in the efficiency, hence, not justifying the extra-computational effort.

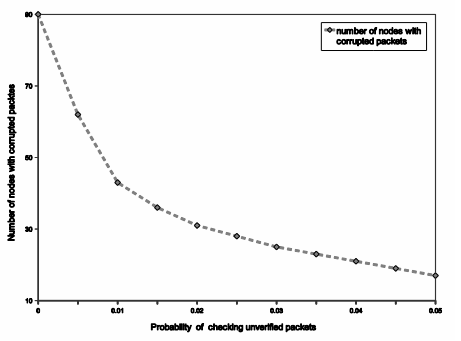

Figure 2 Relationship between infected sensor nodes and the probability of checking

Fig.2 shows that the average number of infected sensor nodes per round drops fast as we increase the probability of checking.

5. 2 Comparison with Different Settings

Table II shows the relationship between bad encoded packets and the probability of checking for a network of 100 nodes in which 5% of them are attackers. We study the case of network coding, and no coding, with a cooperative and a non-cooperative system.

From table II we can see that the classic reprogramming system like Deluge that do not use encoding only suffer from minor damage in the network and is independent of the probability of checking. However, this is not the case for network coding since malicious packets get quickly re-encoded in the network.

TABLE II. RELATIONSHIP BETWEEN PERCENTAGE OF BAD ENCODED PACKETS AND THE PROBABILITY OF CHECKING

p	**Network coding with cooperation**	**Network coding without cooperation**	**No network coding no cooperation**
0.5%	18.3%	88.4%	4.2%
1.0%	11.6%	87.7%	3.6%
1.5%	9.2%	87.3%	4.4%
2.0%	8.7%	87.1%	4.5%
3.0%	7.6%	86.7%	3.8%
4.0%	6.3%	85.7%	4.7%
5.0%	6.0%	84.9%	4.2%
10.0%	5.6%	82.1%	3.7%
20.0%	5.3%	81.2%	4.3%

From Table II we can see that with network coding and no cooperation, the damage of the system decreases linearly with the checking probability, which requires nodes to check almost every incoming encoded packet to have acceptable levels of efficiency. If cooperation is introduced, then the performance of network coding improves. Actually, a cooperative mechanism improves the efficiency of the secure reprogramming system almost ten-fold for a checking probability of 2% (see Table II). Thus, with cooperation the system is able to limit the propagation of bad packets. Moreover, observe that the percentage of bad packets is very close to the minimum, which in this simulation is 5%.

5. 3 Impact of the Rate of Bad Packets

Fig. 3 shows the efficiency of the system for a network of 50 sensor nodes where each node uses network coding and checks with 1% probability as a function of bad packets' rate. We observe that the efficiency of the system largely depends on whether a cooperative mechanism is in place, dropping to less than 20% if this is not the case. We also observe that the efficiency of the system increases linearly as the rate of infection of a malicious node decreases, achieving 90% efficiency for a bad packet rate of 10%.

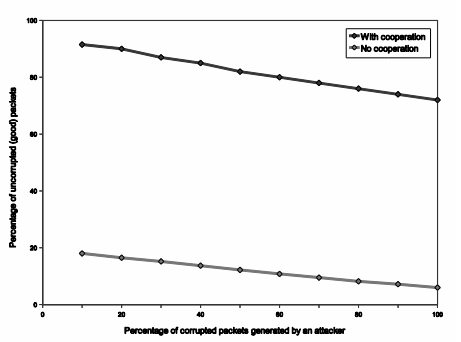

Figure 3 Relationship between good packets and bad packets generated by an attacker

6. RELATED WORK

In recent years, solutions for securing Deluge have been proposed and improved upon [9], [10], [11]. They are mainly distinguished through their structure, granularity and strength of hashing. Hyun et al. [12] derive a hash tree only for the first page. This improvement drastically reduces the amount of overhead. There exists a scheme that uses a different one-way key chain for each hop from the base station [19], with the disadvantage of transmission overhead increasing with hop count. Merkle's one-time signature has been proposed as a ROM-friendly alternative to digital signature [20]. The trade-off is that the nodes need to be stateful.

The preceding solutions only consider the authentication of data packets. A TESLA-like scheme has been proposed for authenticating the 'Inject', 'Reboot', 'Erase' commands [1]. Another scheme [21] has been proposed for authenticating advertisement (ADV) and request (REQ) messages. Despite the progress in secure reprogramming, none of the aforementioned approaches can be used for network coding based reprogramming protocols because in these approaches, the packets of a page are transmitted as is (i.e., not encoded). Alternative approaches are necessary.

The closest solution to the problem in the literature is from Law et al. [22], who propose Sreluge which is resistant to pollution attacks. Sreluge employs a neighbor classification system and a time series forecasting technique to isolate polluters, and a combinatorial technique to decode data packets in the presence of polluters before the isolation is complete. However, to detect the presence of polluted encoded packets in the page, this solution is only by failing to verify a decoded page using its corresponding hash. By the time a bad packet is detected, precious energy might have been wasted.

7. CONCLUSION

In this paper, we presented an approach that is able to defend pollution attack against reprogramming protocols based on network coding. This approach employs a homomorphic hashing function and an identity-based aggregate signature to allow sensor nodes to check packets on-the-fly before they accept incoming encoded packets, and introduces an efficient mechanism to reduce the computation overhead at each node and to eliminate bad packets quickly.

ACKNOWLEDGMENT

We would like to thank the anonymous reviewers for their valuable comments.

References

[1] An Liu and Peng Ning and Wang, C, "Lightweight Remote Image Management for Secure Code Dissemination in Wireless Sensor Networks," in INFOCOM '09: Proceedings of the 28th Annual Joint Conference of the IEEE Computer and Communications Societies, April 2009, pp. 1242–1250.

[2] J. W. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," in SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems. New York, NY, USA: ACM, 2004, pp. 81–94.

[3] S. Kulkarni and L. Wang, "Energy-efficient multihop reprogramming for sensor networks," ACM Trans. Sen. Netw., vol. 5, no. 2, pp. 1–40, 2009.

[4] M. D. Krasniewski, R. K. Panta, S. Bagchi, C.-L. Yang, and W. J. Chappell, "Energy-efficient on-demand reprogramming of large-scale sensor networks," ACM Trans. Sen. Netw., vol. 4, no. 1, pp. 1–38, 2008.

[5] Hagedorn, Andrew and Starobinski, David and Trachtenberg, Ari, "Rateless Deluge: Over-the-Air Programming of Wireless Sensor Networks Using Random Linear Codes," in IPSN '08: Proceedings of the 7th international conference on Information processing in sensor networks. Washington, DC, USA: IEEE Computer Society, 2008, pp. 457–466.

[6] I.-H. Hou, Y.-E. Tsai, T. Abdelzaher, and I. Gupta, "AdapCode: Adaptive Network Coding for Code Updates in Wireless Sensor Networks," in INFOCOM '08: Proceedings of the 27th Annual Joint Conference of the IEEE Computer and Communications Societies, 2008, pp. 1517–1525.

[7] M. Rossi, G. Zanca, L. Stabellini, R. Crepaldi, A. Harris, and M. Zorzi, "SYNAPSE: A Network Reprogramming Protocol for Wireless Sensor Networks Using Fountain Codes," in SECON '08: Proceedings of the 5th Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks, 2008, pp. 188–196.

[8] J. Dong, R. Curtmola, R. Sethi, and C. Nita-Rotaru, "Toward secure network coding in wireless networks: Threats and challenges," in NPSec '08: Proceedings of the 4th Workshop on Secure Network Protocols, 2008, pp. 33 – 38.

[9] P. E. Lanigan, R. Gandhi, and P. Narasimhan, "Sluice: Secure dissemination of code updates in sensor networks," in ICDCS'06: Proceedings of the 26th IEEE International Conference on Distributed Computing Systems. Washington, DC, USA: IEEE Computer Society, 2006, p. 53.

[10] P. K. Dutta, J. W. Hui, D. C. Chu, and D. E. Culler, "Securing the deluge network programming system," in IPSN '06: Proceedings of the 5th international conference on Information processing in sensor networks. New York, NY, USA: ACM, 2006, pp. 326–333.

[11] J. Deng, R. Han, and S. Mishra, "Secure code distribution in dynamically programmable wireless sensor networks," in IPSN '06: Proceedings of the 5th international conference on Information processing in sensor networks. New York, NY, USA: ACM, 2006, pp. 292–300.

[12] S. Hyun, P. Ning, A. Liu, and W. Du, "Seluge: Secure and dos-resistant code dissemination in wireless sensor networks," in IPSN '08: Proceedings of the 7th international conference on Information processing in sensor networks. Washington, DC, USA: IEEE Computer Society, 2008, pp. 445–456.

[13] R. Gennaro and P. Rohatgi, How to sign digital streams, ser. CRYPTO '97: Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology, Lecture Notes in Computer Science. Springer Berlin/Heidelberg, pages 180-197, 1997.

[14] IEEE Std 802.15.4-2003, "IEEE standard for information technology C telecommunications and information exchange between systems C local and metropolitan area networks Cspecific requirements C part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs)."

[15] M. Krohn, M. Freedman, and D. Mazieres, "On-the-fly verification of rateless erasure codes for efficient content distribution," in S&P '04: Proceedings of the IEEE Symposium on Security and Privacy, 2004, pp. 226–240.

[16] C. Gentry and Z. Ramzan, Identity-Based Aggregate Signatures, ser. PKC '06: Public Key Cryptography, Lecture Notes in Computer Science. Springer Berlin/Heidelberg, pages 257-273, 2006.

[17] M. Bellare, J. A. Garay, and T. Rabin, Fast Batch Verification for Modular Exponentiation and Digital Signatures, ser. Crypto '98: Proc. Advances in Cryptology, Lecture Notes in Computer Science. Springer Berlin/Heidelberg, pages 236-250, 1998.

[18] "Castalia," http://castalia.npc.nicta.com.au/.

[19] H. Tan, S. Jha, D. Ostry, J. Zic, and V. Sivaraman, "Secure multi-hop network programming with multiple one-way key chains," in WiSec '08: Proceedings of the first ACM conference on Wireless network security. New York, NY, USA: ACM, 2008, pp. 183–193.

[20] O. Ugus, D. Westhoff, and J.-M. Bohli, "A rom-friendly secure code update mechanism for wsns using a statefulverifier ¿-time signature scheme," in WiSec '09: Proceedings of the second ACM conference on Wireless network security. New York, NY, USA: ACM, 2009, pp. 29–40.

[21] Yu Zhang,Xingshe Zhou, Yiming Ji, Zhiyi Fang and Lifang Wang, "Secure and DoS-Resistant Network Reprogramming in Sensor Networks Based on CPK," in WiCOM '08: Proceedings of the 4th IEEE International Conference on Wireless Communication, Networking and Mobile Computing, 2008, pp. 1–5.

[22] Y. W. Law, Y. Zhang, J. Jiong, M. Palaniswami, and P. Havinga., "Secure rateless deluge: Pollution-resistant reprogramming and data dissemination for wireless sensor networks," EURASIP Journal on Wireless Communications and Networking, Special Issue on Security and Resilience for Smart Devices and Applications, vol. 2011, no. 1, pp. 11–33, 2011.