Remote Attestation of Sensor Nodes on Node Personalization: One Case Study

Jinhui Yuan¹, Hongwei Zhou² and Zhixin Wang³

¹ Zhongyuan University of Technology, Zhengzhou, China
² Information Engineering University, Zhengzhou, China
³ Sias International University, Zhengzhou, China

{jcyjh, hong_wen_zhou, zhi_xin_wang}@126.com

Abstract. For the verifier has to identify the attester which introduces more computation, homogenous node is obstacle for remote attestation of sensor nodes. In this paper, we firstly propose node personalization, which means that every node is different from others in the sensor network, to save the computation energy. Then, we discuss the principle of node personalization. Lastly, we present a practical case. More details, we develop node personalization with obfuscated code, and verify code integrity with cascading hashing. Our discussion shows the solution we present is feasible.

1. Introduction

Due to the limited cost of sensor node, it is vulnerable to attack[1,2]. Sensor node is a small device, and its power supply usually comes from batteries. So the node do not capacity of carrying out large-scale computation. This limits the deployment of complex and reliable security mechanisms on sensor nodes to protect them. Furthermore, the nodes are often deployed in hostile environments in which managers are not fully controlled. It enables attackers to easily gain physical control of sensor nodes.

How to overcome the above problem? One possible solution is remote attestation. Before using the sensing data, we first check whether the node is normal. If the node is compromised, the sensing data may be modified and should be discarded. In this way, we do not need to significantly increase the computational power of nodes to avoid the threats. As a result, nodes do not spend too much energy to complete complex calculations. Thus, we extend the life time of sensor nodes.

However, existing attestation solutions, such as SWATT [3], incur the impersonation attack. In the solutions, the verifier firstly ask some question to the attester. Secondly, the attester answer the question. Lastly, the verifier check the answer to verify that the attester is trusted. Although the questions and answers are random, it is possible to circle the attestation. Because all nodes are same, the verifier fails to identify the attester. For example, the verifier ask a question to attester A, and attester A ask the same question to attester B by playing the role of the verifier. At last, the attester A responses to the verifier by using of the answer of attester B.

To defeat against the threat, we present remote attestation of sensor node on node personalization. In our solution, we hope that each sensor node is unique in the sensor network, that we call it node personalization. Even if an attacker captures a sensor node and analyzes it, the attacker only gets the information of the sensor node and can not affect the security of other sensor nodes. Thus, even if the
questions are the same, the responses of the nodes are not the same. The compromised node can not fake other nodes with their answers.

To show the feasibility of our solution, we proposed obfuscated code for node personalization and cascading hashing for code integrity measurement. Obfuscated code is the method that can change the program instructions without changing the semantic. Using this approach, each node has different instructions, but does the same thing. We use cascading hashing to construct the question for the attestation. More details, cascading hashing is randomly selecting a number of instructions to compute the finally summary value. Our discussion shows that our scheme is feasible, and is capable of defeating the impersonation attack.

The rest of the paper is organized as follows. Section 2 presents our observation. Section 3 presents our solution as one case study. Section 4 is our discussion. Related work is in section 5. At last, we conclude this paper.

2. Our Observation
The attestation process is shown in the Fig.1(a). As shown in the fig.1, node V is the verifier, and node B is the attester. In normal process, node V proposes one question to node B, and node B responses with the answer. However, in most sensor network, the sensor nodes are same to each other. This left the window for the attacker. As shown in the Fig.1(b), node B is the compromised node, and it receives the question form node V. Now, because node B is compromised, it fails to pose the correct answer. So it transmits the same question to node C. Node C gives the answer to node B because it considers that node B is a relay node to transmit the question to node V. Now node B gets the correct answer, and responses to node V. Node V can not identify the origin of the answer, and it thinks the node B is trusted.

![Figure 1. Remote attestation of sensor node](image)

3. Our Solution
3.1. Node Personalization
In this paper, we propose the approach, called as obfuscated code, to personalize the node. Obfuscated code is an existing technology, and it can change the instructions without change the semantics. There are many approaches to make the instruction confused. A direct approach is to add some NOP instructions into the program. Noted that the NOP instructions do not affect the processor execution.
So the program is changed without change the semantics. Equivalent instruction replacement is another interesting approach. For example, the instruction ADD 1 is able to be replaced with the instruction SUB -1. There are more approaches for obfuscated code, and we consider that they are out of the scope of our paper.

We divide the instructions of the node into two parts including base part and confusing part. In extreme case, all instructions of the node can be confused. However, we consider that it is not a good method. The software of the node includes operated system and application. Confusing operated system is a challenge for the system administrator. Moreover, it is inconvenience to the installation of software systems for sensor node. On the above considerations, the base section should hold the instructions which are not confused, while confusing section hold the instructions which are confused.

The image of every node is delivered as shown in the fig.2. With the existing tools, the administrators are able to generate the unique image with some random factor for every node, and these images are different to each other. Every node, that wants to install the software system, send a random factor to the sink node. Sink node is a node which has enough energy to finish complex computation. Sink nodes generate the unique image for the nodes and send the image to them. The base section of node has basic functions that include receiving the image and installing the software on the image, etc. On a security point of view, the image and network communications should be protected. However, to simple the solution, we advice the nodes should receive the image in a secure environment, and there are not attackers in the network. For example, before the deployment, every node should receive and install the image in the secure room. After the installation, these personalized nodes are deployed out of the door.

![Figure 2. Scatter unique image for every node](image)

There are other ways for node personalization. We consider that ASLR (Address Space Layout Randomization) is one. ASLR is designed to defeat against buffer overflow. But we think it also be used to change the layout of the instructions. Some instructions can be implemented as position independent code, and they are loaded randomly at the different addresses. Thus, in the entity view, the instructions of node are different to each other with the same semantics. In this paper, we do not discuss more methods for node personalization, and we consider that this is our future work.

### 3.2. Node Attestation

In this paper, we present cascading hashing for node attestation. Since we develop node personalization with obfuscated code, a direct way of attestation is hashing. Summary calculations can be performed on all instructions in node memory to ensure the integrity of node. Noted that the integrity of instruction only indicate that the node is trusted to a certain extent. However this above way seems reliable, but it is not easy to circle by an attacker. An attacker can obtain the summary before he tampers with the instruction of the node, and circles the checking with the summary. So every hashing involves a random factor to ensure that each question and answer is different, thus making such above attack ineffective.

We present cascading hashing with the Fig.3. The first relevant parameter is called as initial factor including the address and the length which is shown as addr1 and len1 in the fig 3. Initial factor is the input parameter of cascading hashing, and it means that the computed section is starting from the address addr1 with the length of len1. The second relevant parameter is called as cascading factor. Similar to initial factor, cascading factor is also two-tuples including one address and one length. In the fig.3, (addr2, len2) and (addr3, len3) are also cascading factors. Cascading factor come from the result of the previous hashing. For example, we use the SHA1 algorithm to calculate the summary value with 160 bit length. So we can define the values of the first to the fourth bits as the address and the fifth to the eighth bits as the length to obtain cascading factor. With the fig.3, after the first round
of calculation, we get the next cascading parameters, denoted as (addr2, len2), from the results of the calculation. Similarly, we can obtain (addr3, len3) from the second calculation. Repeat it until the maximum address of the storage space.

![Figure 3. Cascading hashing](image)

There is a balance between the size of the computed range and amount of calculation. From an integrity perspective, we should cover as much space as possible. From a computational point of view, we should reduce the amount of space we cover. If the range is too large, the node needs more computation, which consumes too much energy. If the range is too small, then there may be a risk of malicious instruction evasion of checking. Dynamic adjustment of coverage may be a better choice. When the node battery can provide sufficient energy, the coverage can be increased.

Changing the parameters is able to change the coverage. If the user sets the initial factor too large, the low address range is skipped over a large area. For example, if the address in the initial factor is 1MB, 1MB space is skipped. If the cascading parameter is set too large, it will also cause a large area of space to be skipped. For example, if we set 8 bits of the summary value as the next hop address, then, on average, 128 bytes of space are skipped. In this way, the attester and verifier can control the scope of the inspection by negotiating parameters.

3.3. Attestation Protocol

With the support of node personalization, the attestation protocol is simplified. Node personalization means that each attester gives a different and correct answer to the same question of the verifier. With this feature, it is not necessary to complex the messages with cryptographic algorithm to defeat the attacker. This also brings an additional benefit that the computation reduced.

The attestation protocol includes the following steps.

- Step 1: verifier generates a random factor as the question message, and sends it to attester.
- Step 2: attester receive the random factor, and Complete the calculation associated with its specific properties, and the result is the response message which is sent to verifier.
- Step 3: verifier judges the current status of a node based on response message.

The above protocol is an abstract protocol on the node personalization, and more detail is needed to be discussed. For example, the verifier may be a sink node, and the attester is another sensor node, and they interact with each other through intermediate nodes. So the routing protocol has to add into the above protocol as the attachment. In our case, the random factor is the initial parameter, and verifier check the instruction integrity on the cascading hashing. To perform the check, the verifier has to store the images of all nodes. Fortunately, the verifier is usually sink node or other computer, and they have sufficient energy, large storage capacity and strong computing power. They are capable of checking the integrity of the instructions.

4. Discussion

4.1. The Guide of Node Personalization

We believe that there are many methods to implement node personalization. In this paper, we just present one feasible approach. How to choose the appropriate methods for node personalization? In our opinion, the following conditions must be met for implementing the method of node personalization.

First, the nature of personalization must be closely related to node security. In our case, the factor that implements node personalization is the instruction. Since the attacker often tamper with the code of the node to launch the attack, the code integrity can defeat this threat. Of course, it is possible to
circle the code integrity. For example, ROP attack is developed without inserting any code in the node. In any case, code integrity is an important and not negligible factor in ensuring node security.

Second, node personalization must ensure that the node is somewhat different from other nodes. In our case, we use obfuscated code to personalize nodes. As mentioned earlier, obfuscated code can not change fully the instruction of the sensor node. In fact, fully node personalization is unrealistic. Moreover, it is usually necessary for the attestation. On the other side, the node must be different to other nodes in node personalization. In summary, obfuscated code can make the node different, and that difference is enough to support the attestation.

Last, the attester’s response should be different even if with the same factor. The target of node personalization is to make the response different. In our case, even if the verifier gives the same initial factor, the response message returned by each node is different. In other words, we have to find a checking method that is closely related to the node differentiation factor.

4.2. The Feasibility of Our Case

In our opinion, there are two factors that affect the feasibility of our case, and we discuss them as following.

First factor is the degree that obfuscated code changes the node instruction. In our opinion, entire node instructions can be change with the support of obfuscated code. However, we consider that it is necessary. In most common views, the base section, we mentioned it in previous section, should not include the operated system such as TinyOS. So the modified code may only account for 1/5 of all software on the node. However, we believe that it is enough to find the compromised node. Even with only a small number of key code changes, the node can be personalized to meet the checking requirements. Of course, integrity measuring methods must support this architecture, and this leads to a second factor.

Second factor is the range that the cascading hashing covers. In the extreme case, cascading hashing is able to cover all instructions. In this case, the initial factor can be set as (0, len) where len is the length of the entire instructions. We don’t recommend it because that the node has to perform a great deal of calculation. In other extreme case, cascading hashing may skip the entire instruction. This is obviously ineffective. We want to get the necessary coverage by adjusting the initial parameters and cascading parameters. The effect of the initial parameters is obvious, and we do not discuss them in the detail. To simplify the discussion, we suppose the initial parameter as (0,0). Furthermore because parameters can only be affected by setting widths, we discuss the impact on coverage in terms of cascading parameter widths. We suppose that the value of each cascade parameter is averaged. With the above assumption, we obtain the formula (1)

$$\eta = \frac{2^{m-1} \times \left\lfloor \frac{L}{2^{n-1} + 2^{m-1}} \right\rfloor}{L}$$  \hspace{1cm} (1)$$

In formula (1), n is the size of the address and m is the size of the length, and L is the size of all instructions. To simplify the formula (1), we suppose L is enough large. Thus we obtain the formula(2)

$$\eta = \frac{2^{m-1}}{2^{n-1} + 2^{m-1}}$$ \hspace{1cm} (2)$$

With formula (2), it is easy to obtain fig.4. In the fig.4, the vertical axis is the coverage rate, while the horizontal axis represents the length of two factors in the address and length of the parameter. For example 1:2 means that n=1 and m=2. With the fig.4, it is evident that the greater m, the higher the coverage. We can adjust the parameters to change the coverage.

In summary, obfuscated code is able to personalize the node, and cascading hashing is capable of cover most of the instruction. Our presented solution is feasible.
5. Related Work
SWATT[3] is a key related work. SWATT is a solution that is able to check the memory contents of embedded devices. If the attacker changes the memory content, SWATT is capable of detecting it with high probability. To the end, SWATT obtain checksum of the memory content with pseudorandom memory traversal. Since the verifier know the fully memory content, any malicious change on memory content can be detected with the checksum. SCUBA[4] is similar work. However, in our opinion, every node is same to each other, and it is vulnerable to the attacks. On the observation, we propose node personalization.

Group verification is another solution to defeat the above threat [5]. In group verification, the node and its neighbor node cooperate to detect the compromised node. These related nodes all contribute to the generation of the question with secret sharing scheme, and a few nodes can't cheat other nodes. In our opinion, the cryptographic algorithms are used, and introduce high performance overhead.

Most of existing approaches are static including SWATT. For example, they only check the integrity of code when code is loaded in the memory. They are vulnerable to dynamic attack. To overcome it, C-FLAT[7] is presented. It checks the control flow of the application which is run-time attestation. C-FLAT is developed on ARM TrustZone which is considered as the trusted anchor for the measurement of control flow.

In general, remote attestation requires the attester to have a trusted root that is usually a trusted hardware. However, it is not a good assumption for sensor node which are small devices. [6] discuss the feasibility of the attestation without the support of trusted hardware. In our opinion, our solution is the no-trusted-hardware method. Furthermore, we do not judge whether nodes are abnormal based on time delay that is do in SWATT[3].

From our point of view, the above solutions are homogenous for the node is same to each other. Homogenous node brings a challenge for the verifier without the encrypting. On the above observation, we propose the node personalization, and simplify the attestation.

6. Conclusion
Remote attestation is a kind of approach to ensure the security of sensor network. However, existing attestation solutions fail to distinguish the node because they are homogenous. To avoid the attack, some complex computation is necessary for every node. But the sensor node is a small device and its power supply is limited, which comes from batteries. In order to save the energy, the sensor nodes need avoiding too complex computation. So we propose node personalization that every node is unique in the sensor network. In our case study, we present obfuscated code and cascading hashing for the attestation on node personalization. Our discussion shows that our solution is feasible.

7. Acknowledgment
The authors would like to thank the anonymous reviewers for their insightful comments that helped improve the presentation of this paper. The work is supported in part by the National Natural Science Foundation of China (61303074), and Foundation for University Young Key Teacher of Henan.
8. References

[1]. Yuan, J., Zhou, H., & Chen, H. (2012). Subjective Logic-Based Anomaly Detection Framework in Wireless Sensor Networks. International Journal of Distributed Sensor Networks, 2012.

[2]. Chen, X., Makki, K., Kang, Y., & Pissinou, N. (2009). Sensor network security: a survey. IEEE Communications Surveys & Tutorials, 11(2), 52-73.

[3]. A. Seshadri, A. Perrig, L. van Doorn, and P. Khosla. SWATT: SoftWare-based ATTestation for Embedded Devices. In Proceedings of the 2004 IEEE Symposium on Security and Privacy, Oakland, California, May 2004.

[4]. Khosla, P., Khosla, P., Khosla, P., Khosla, P., & Khosla, P. (2006). SCUBA: Secure Code Update By Attestation in sensor networks. ACM Workshop on Wireless Security (Vol.33, pp.85-94). ACM.

[5]. Yuan, J. H., & Zhou, H. (2017). Detecting Compromised Sensor Nodes with Group Verification. International Conference on Automation, Mechanical Control and Computational Engineering.

[6]. Francillon, A., lien, Nguyen, Q., Rasmussen, K. B., & Tsudik, G. (2015). A minimalist approach to remote attestation. Design, Automation and Test in Europe Conference and Exhibition (pp.1-6). IEEE.

[7]. Abera, T., Asokan, N., Ekberg, J. E., Ekberg, J. E., Nyman, T., & Paverd, A., et al. (2016). C-flat: control-flow attestation for embedded systems software. 743-754.