Advances in Engineering Research, volume 118

2nd International Conference on Automation, Mechanical Control and Computational Engineering (AMCCE 2017)

Detecting Compromised Sensor Nodes with Group Verification

Jinhui Yuan[1, a] and Hongwei Zhou [2, b,*]

[1]Information Engineering University, Zhengzhou, 450001, China;

[a]jcyjh@126.com, [b]hong_wei_zhou@126.com, *corresponding author

Keywords: Sensor network, Group verification, secret sharing.

Abstract. To detect compromised sensor node, current solutions fail to resist the collusion attack. To address it, in this paper, we propose a novel solution called group verification. The key idea of group verification is that neighbor nodes and sink node cooperate to detect the compromised node. If one sink node want to check one suspicious node, every neighbor node contribute to pose a question for the suspicious node with secret sharing scheme, and sink node introduces its random number into the question. Thus one compromised neighbor node fail to disturb the question distribution. To expose illegal question, every question is accompanying with one checking token which is generated with one private checking key. The suspicious node recovers the question by collecting enough shares. After that, the suspicious node floods the answer, and its neighbor nodes all verify the answer with the help of sink node. Our discussion shows our solution is capable of detecting the collusion attack.

1.Introduction

Due to the nature that sensor nodes are easy to be physically captured, an adversary may disturb the sensor networks by controlling several sensor nodes. For example, we deploy lots of sensor nodes to monitor the moving things. An adversary physically captures one sensor node, and inserts the malicious code into the node. Then the compromised node sends some false sensing data into the network. Thus the network may continue to report the alarm, or fail to catch the moving attacker. To protect the sensor network, it is necessary to detect these compromised nodes.

There are some work to detect compromised node. However, some solutions often fail to resist the collusion attack. As an example, [1] propose a software-based attestation to detect the compromised node. The key idea is to design a pseudorandom memory traversal as a trusting anchor. If the node is compromised, it takes more time to obtain the checksum with the above pseudorandom memory traversal. Thus, the compromised nodes are exposed on the abnormally increase of time. However, in sensor network, the network delay is unpredictable. If one routing node is compromised, it is easy to disturb the detection. As another example, [2] proposes another solution. In this solution, all neighbor nodes cooperate to produce the question and verify the answer. However, in our opinion, it is easy to be circled. If a neighbor node is also compromised node, it can answer to the suspicious node. Thus other nodes fail to detect the suspicious node.

To address the collusion attack, we propose a novel solution called group verification in this paper. The question is simple if sink node can communicate directly with the suspicious node. However, it is unusual in most situation. Usually, its neighbor nodes have to take part in the detection. So one compromised neighbor node is able to disturb the detection. To overcome this, in our group verification, all neighbor nodes contribute to pose a question for the suspicious node with secret sharing scheme[5], and the suspicious node recover the question by collecting enough shares. After that, the suspicious node flood the answer to its neighbor nodes, and neighbor nodes all verify the answer with the help of sink node. However, there is one chance to cheat sink node. Suppose one compromised neighbor node poses the same question to other normal node, gets the right answer, and transmit it to the compromised node. So the compromised node obtains the answer from other normal node, and circle the detection. To improve our solution, each node shares one private checking key with sink node, and every question is accompanied with one checking token which is generated with the checking key. If sink node wants to check one suspicious node, it produces one question token for this question. One node would refuse to answer the question without one legitimate question token. Thus our group verification resist the collusion attack.

2.Our motivation

In our opinion, there are two main kinds of collusion attacks. The first attack is shown in fig.1(a). In the figure, the shadow circles are compromised nodes, and the square is sink node. Now, sink node wants to check node O, and sends the question to it. However, node A is also compromised node, and it can drop the message. Besides that, node A may ask the same question to other node to get the answer, for example node C. Thus node A is able to cheat sink node with the right answer.

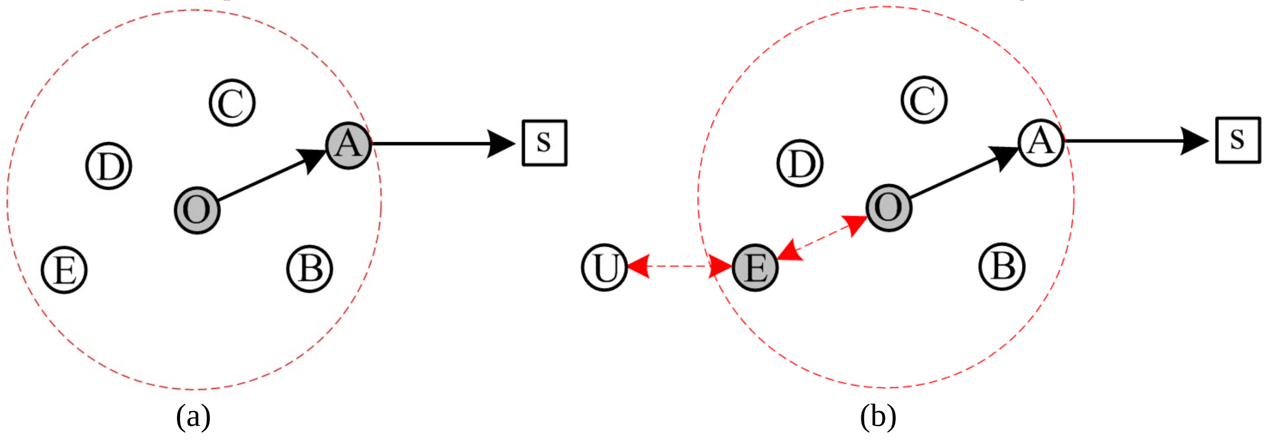


Fig. 1 Two kinds of collusion attack

The second attack is shown in fig.1(b). Sink node wants to check node O, and the problem is sent to it. Node O does not know the answer, but it knows that node E is its conspirator. So node O delivers the same question to node E. After receiving the question, node E delivers it to node U and asks node U to answer the question. Node U fails to identify that the question is designed for node O, so it gives the answer to node E. Then node E delivers the answer to node O. At last, node O delivers the answer to sink node. Moreover, sink node still takes node U's answer as node O's, and considers that node O is not compromised.

In this paper, we propose the solution that is able to resist the above collusion attack. As pointed earlier, in most situation, sink node cannot communicate directly to suspicious node. So our solution has to detect the compromised node with some compromised nodes. For example, though node A is compromised, our solution is able to expose node O. Our solution is called group verification, and the key idea is that neighbor nodes and sink node cooperate to detect the compromised node. Thus a small fraction of compromised neighbor nodes fail to disturb our solution.

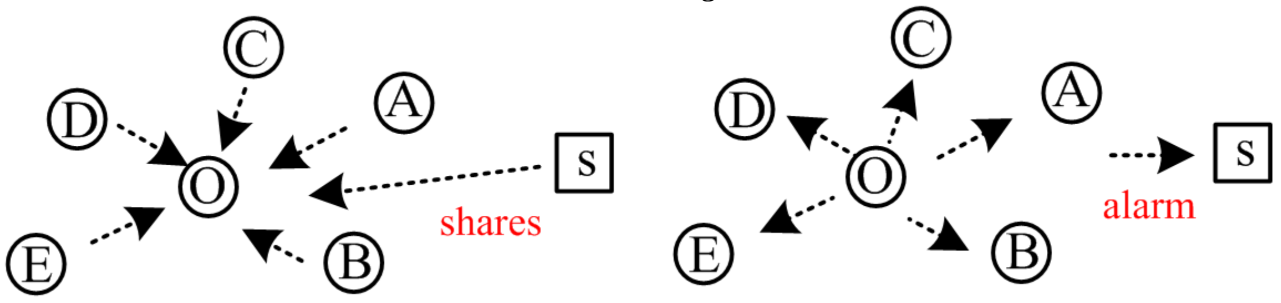
3.Attack Model and Assumption

We suppose that an adversary can capture few sensor nodes, but fail to control most of nodes in the network. In some time, an adversary may capture the nodes that are geographically close to each other. In a local perspective, the adversary may control one node and a small fraction of neighbor nodes. There are most of neighbor nodes that are not be controlled. In a extreme case, all neighbor nodes are all captured, our solution fail to detect the compromised node.

We also assume that the attacker does not physically change the node. For example, it can enlarge the memory of nodes. The assumption is similar to [3-4]. In this paper, we does not want to design one novel pseudorandom memory traversal or others. Some existing schemes[1-4], in our opinion, can be directly used in our solution. In other words, these schemes are used as the solid ground to develop the question. In this paper, we focus on how to build the question on secret share scheme and verify the answer.

4.Our solution

We introduce our solution with figure 2 in summary. Suppose that sink node wants to check node O, and node O has some neighbor nodes. At first, as shown in figure 2.(a), sink node and neighbor nodes contribute to generate the question with secret share scheme. Neighbor nodes flood their shares to each other, and sink node generate one random number as the part of question. Sink node composes the question, and send to the suspicious node. Now node O has known the question, and it easily obtains the answer if it is not compromised. At last, as shown in figure 2.b, node O floods the answer, and its neighbor nodes all obtain and check the answer. If any neighbor node finds the answer is not right, it sends an alarm message to sink node. There are some to be more discussed in detail. First, what is the question. Second, how to develop the question. Third, how to identify the question. At last, how to check the answer. We discuss them as following.



(a) Question distribution (b) Question verification

Fig. 2 Overview of our solution

What is the question? In this paper, we do not propose a novel scheme to build the question, but develop the question along with [2,3]. In fact, a question is a randomly generated number which is considered as the seed of the pseudorandom memory traversal[2,3]. There is two kinds of memory traversal including sequential traversal and pseudorandom traversal. It is evident that pseudorandom memory traversal is more difficult to be circled. For example, the output of block cipher algorithm is treated as the addresses for memory traversal. If taking the old output as the new input of algorithm, it outputs the new addresses each time. Noted that it is difficult to forecast these addresses. So the adversary fails to compute the checksum. In this paper, we denoted the pseudorandom memory traversal as fp=fun_tr(s) where s is the seed. We call s as the question, and fp as the answer. A normal node easily computes fp with s while a compromised node fails to do that.

How to develop the question? In our opinion, all neighbor nodes and sink node should contribute to develop the question. To the end, these nodes cooperate to do that on secret share scheme. For one neighbor node, it constructs a polynomial as $f^1(x)=a^1{}_0+a^1{}_1x+ a^1{}_2x^2+ a^1{}_3x^3+... + a^1{}_kx^k$, and computes the shares as $f^1(1)$, $f^1(2)$,..., $f^1(n)$ where n is the sum of neighbor nodes. Every neighbor node floods the shares, and sink node collect these shares. Noted that every neighbor node constructs a polynomial. On these polynomials, sink node pick some polynomials as the data base to recover the final polynomial, which is denoted as: $f^n(x)=(a^1{}_0+a^2{}_0+...+a^m{}_0)+ (a^1{}_1+a^2{}_1+...+a^m{}_1)x+... + (a^1{}_k+a^2{}_k+...+a^m{}_k)x^k$. Now sink node generates random number r, and seed is $(a^1{}_0+a^2{}_0+...+a^m{}_0+r)$. In this way, the question is developed.

How to identify the question? To the end, sink node shares one checking key with every node. The goal of checking key is to identify the question. When one node receives a question, it should first check the legitimacy of the question. Otherwise one compromised node may take a legal role to request other node to answer the question. Thus the compromised node easily obtain the answer. With checking key, every question is accompanied with a question token. A question token is denoted as $E(seed)_k$ where seed is the question and $E(seed)_k$ means the cryptograph of seed with checking key k. To get seed, sink node recovers the final polynomial, and generate a random number r. Then it encrypts seed to develop checking token denoted as $E(seed)_k$. To distribute the question, sink node sends the message, including checking token and seed, to suspicious node. The suspicious node verifies the source of question with the checking token.

How to check the answer? The checked node should report the answer to sink node. However, it is dangerous to transmit the answer along some routing nodes because the nodes may be compromised. To avoid that, the answer has to be encrypted. However, it takes more energy to do that for every energy-limited node. So we consider that its neighbor nodes can check the answer. Suppose the suspicious node floods the answer, and its neighbor nodes all receive it. If any neighbor node consider that the answer is wrong, it sends an alarm message to sink node. In this scenery, though one neighbor node may be compromised and does not send an alarm, other neighbor nodes are still send the alarm message to sink node. At last, sink node send hash(fp) to every neighbor node, and these nodes check the answer with it.

The group attestation protocol is summarized as following.

Table 1 Group attestation protocol

Step	Description
1	Neighbor generates a polynomial , and computes shares, and floods the shares
2	Sink node recovers the final polynomial, and gets $a^1{}_0+a^2{}_0+...+a^m{}_0$
3	Sink node generates random number r, develops seed, and generates the checking token
4	Sink node sends seed and checking token to suspicous node
5	Sink node generates the answer fp=fun_tr(seed), and computes hash(fp)
6	Sink node sends hash(fp) to neighbor nodes.
7	Suspicous node recovers seed, computes the answer, and floods the answer
8	Neighbour node checks the answer with hash(fp), and sends an alarm if mismatch

5.Discussion

Our solution is capable of resisting the collusion attack. In the question distribution, every neighbor node contributes to construct the question, but the compromised node fails to know the question in advance. First, the part of question is built with secret share scheme. An adversary cannot recover the secret without enough shares. Second, the other part of question is generated by sink node, and we consider that sink node is trusted. If the suspicious node is compromised, the adversary can obtain the question after the compromised node receives the checking token. At this time, one possible method to cheat sink node is to ask other node to answer the question. In our opinion, this is unfeasible. Any node checks the checking token before answering the question. The adversary cannot forge the checking token without the checking key. In the question verification, a compromised neighbor node still fail to cheat sink node. The checking baseline, which is sent to neighbor nodes by sink node, is the hash value of answer. It means that compromised neighbor node cannot know the answer, and the compromised node also knows it. After the compromised node floods the answer, all neighbor nodes are able to read it, and verify the answer. One possible of the compromised node is that it does not send the alarm message to sink node. However, in our opinion, all neighbor nodes obtain the result, and a small fraction of neighbor nodes are not capable of preventing other neighbor nodes sending the alarm message to sink node.

There are some interesting topics. For example, who launch the group verification? In our opinion, the suspicious node's neighbor nodes are the good choice. On spatial correlation[6-7] and temporal correlation[8], the node's sensing data should be similar to its neighbor node's sensing data in most time. If one node is to be compromised, its neighbor node first know that. As mentioned earlier, if one node is captured, its neighbor node is also easy to be captured. So any node should check its neighbor node at all time, and send an alarm message at once if any compromised node occur. Suppose that an adversary need time t_1 to capture one node, and one node need time t_2 to send an alarm message. If t_1 is bigger than t_2, one node has the chance to send alarm message. Then sink node launches the group verification at once. Thus the compromised nodes are exposed.

6.Summary

In this paper, we propose a novel method to detect compromised sensor node when its neighbor nodes may also be compromised. Considering the nature that sensor node is easy to be physically captured, we present group verification to detect compromised node with the cooperation of neighbor nodes and sink node. To construct the problem, the neighbor nodes negotiate to build the secret data as the part of question, then sink node generate a random number as the other part of question. To expose illegal question, sink node shares a checking key with every sensor node, and sink node builds one checking token for each question. To decrease the danger of verification, all neighbor nodes are responsible for checking the answer. Our discussion shows our group verification are capable of resisting the collusion attack.

7. Acknowledgments

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).

References

[1] A. Seshadri, A. Perrig, L. van Doorn, and P. Khosla. SWATT: SoftWare-based Attestation for embedded devices. In IEEE Symposium on Security and Privacy, 2004.

[2] Yang, Y., Wang, X., Zhu, S., & Cao, G. (2007). Distributed Software-based Attestation for Node Compromise Detection in Sensor Networks. IEEE International Symposium on Reliable Distributed Systems (pp.219-230). IEEE Computer Society.

[3] T. Park and K. G. Shin. Soft tamper-proofing via program integrity verification in wireless sensor networks. IEEE Trans. Mob. Comput., 4(3):297–309, 2005.

[4] M. Shaneck, K. Mahadevan, V. Kher, and Y. Kim. Remote software-based attestation for wireless sensors. In ESAS, July 2005.

[5] A. Shamir. How to share a secret. Commun. ACM, 22(11):612–613, 1979.

[6] M. Ding, F. Liu, A. Thaeler, D. Chen, and X. Cheng, Fault- Tolerant Target Localization in Sensor Networks, EURASIP J. Wireless Comm. and Networking, vol. 2007, no. 1, pp. 1-9, Jan. 2007.

[7] Z. Merhi, M. Elgamel, and M. Bayoumi, A Lightweight Collaborative Fault Tolerant Target Localization System for Wireless Sensor Networks, EURASIP J. Wireless Comm. and Networking, vol. 8, no. 12, pp. 1690-1704, Dec. 2009.

[8] J. Yuan, H. Zhou, and H. Chen. Subjective Logic-Based Anomaly Detection Framework in Wireless Sensor Networks. International Journal of Distributed Sensor Networks. Volume 2012, 2012.