Reliability Analysis of Wireless Sensor Networks with Distributed Code Attestation

Ing-Ray Chen, *Member, IEEE*, and Yating Wang

***Abstract*—In this paper we analyze the reliability of a homogeneous wireless sensor network executing a distributed code attestation protocol with neighbor sensor nodes serving as code verifiers. By considering the tradeoff between energy exhaustion vs. security vulnerability for causing sensor node failures, we identify how often distributed code attestation should be performed as well as how many neighbor sensors should serve as code verifiers per attestation event to maximize the system lifetime without compromising performance. Sensitivity analysis of the results with respect to critical model parameters is presented with physical interpretations given.**

***Index Terms*—Wireless sensor networks, distributed code attestation, reliability, security, intrusion detection.**

I. Introduction

WIRELESS sensor networks (WSNs) are broadly deployed in many safety-critical applications, such as health, construction and military, as well as diverse environments, such as highways, forests and battlefields [5]. Power is supplied by battery so that energy conservation is crucial in sensor node (SN) design to prolong the WSN lifetime. A WSN is frequently deployed in unattended environments making SNs vulnerable to capture attacks by the adversary (humans or robots). A captured SN is subject to physical sabotage turning it into a compromised SN capable of performing attacks.

Among the many methods to cope with inside attackers in WSNs, software code attestation [1], [2], [6] has received high attention because of its verifiable and provable semantics. The basic idea is that the code of a compromised SN would be different from that of a normal SN. Hence by inspecting if the code is still the same as what originally was put in, the system can detect whether the SN has been compromised.

In the literature, existing work on code attestation mostly focused on the protocol design for performing code attestation, and verification of the protocol design, e.g., by correctness proof. In particular, [1] assumed the existence of a trusted third party capable of verifying if a SN is compromised through a challenge-response mechanism. To avoid a single point of failure (the trusted base station), [2] extended centralized code attestation to distributed code attestation by using designated servers or just neighbor SNs to a target SN. The focus was again on protocol design and correctness proof. Performance issues, especially in energy conservation to best tradeoff security failure and prolong lifetime of WSNs with code attestation, are relatively unattended. To the best of our knowledge, only [3] previously addressed performance issues of centralized code attestation and no prior work has been done to address performance issues of distributed code attestation.

This paper concerns the effect of distributed code attestation on the reliability and performance of WSNs, taking into account both security failure and energy exhaustion failure. We adopt the Byzantine failure definition that the embedded WSN fails when more than one third of the SNs are bad nodes (reporting incorrect sensor data) beyond which there is no way to have any form of information consistency. Compared with existing work, the contribution of this work is that we address reliability and performance issues of distributed code attestation by identifying operational settings to execute distributed code attestation such that the WSN lifetime is maximized without compromising performance.

II. System and Performance Models

A. System Model and Assumptions

We consider a homogeneous WSN in which SNs are deployed randomly and distributed according to a homogeneous spatial Poisson process with intensity $n/\pi r_{SN}^2$ where n is the average neighbor size and r_{SN} is the SN radio range. The main function of a SN is to report sensing data periodically to the sink node with the interval time of T through multihop routing. To cope with compromised nodes, SNs also act as verifiers if selected to perform distributed code attestation to their neighbors. With respect to a target SN, only a subset of its neighbors up to n_v SNs will be selected randomly as the verifiers to conserve energy and to make the approach scalable. We adopt a randomization strategy in selecting n_v neighbors as the verifiers every time code attestation is performed, so the adversaries will not have specific targets to perform attacks. Due to potential imperfection of code attestation protocol design [6] or software bugs, a verifier may misdiagnose a good SN as a bad SN with a false positive probability P_{fp}, and misidentify a bad SN as a good node with a false negative probability P_{fn}.

The code attestation event overlaps with the sensing and reporting event with probability q. That is, every SN will be attested with probability q in a sensing and reporting cycle. By this way, we control how often code attestation is to be performed for the purpose of energy conservation. We consider a delay reporting design, that is, at the beginning of each interval, a SN first allows code attestation to perform (if it is a target node) before it reports its current sensing readings. After n_v verifiers each independently perform code attestation toward a target SN, they perform majority voting to determine if the target SN is compromised. We assume persistent attacks by a compromised node. That is, when acting as a verifier, a

Manuscript received July 2, 2012. The associate editor coordinating the review of this letter and approving it for publication was A. Shami.

The authors are with the Department of Computer Science, Virginia Polytechnic Institute and State University, Falls Church, VA, 22043 (e-mail: {irchen, yatingw}@vt.edu).

Digital Object Identifier 10.1109/LCOMM.2012.12.121454

$$P_{fp}^{DCA}(t) = \sum_{i=0}^{n_v-n_m} \frac{\binom{n_b(t)}{n_m+i} \times \binom{n_g(t)}{n_v-(n_m+i)}}{\binom{n}{n_v}} + \sum_{i=0}^{n_v-n_m} \frac{\binom{n_b(t)}{i} \times \sum_{j=n_m-i}^{n_v-i} [\binom{n_g(t)}{j} \times (P_{fp})^j \times \binom{n_g(t)-j}{n_v-i-j} \times (1-P_{fp})^{n_v-i-j}]}{\binom{n}{n_v}} \quad (1)$$

$$P_{fn}^{DCA}(t) = \sum_{i=0}^{n_v-n_m} \frac{\binom{n_b(t)}{n_m+i} \times \binom{n_g(t)}{n_v-(n_m+i)}}{\binom{n}{n_v}} + \sum_{i=0}^{n_v-n_m} \frac{\binom{n_b(t)}{i} \times \sum_{j=n_m-i}^{n_v-i} [\binom{n_g(t)}{j} \times (P_{fn})^j \times \binom{n_g(t)-j}{n_v-i-j} \times (1-P_{fn})^{n_v-i-j}]}{\binom{n}{n_v}} \quad (2)$$

bad node will perform the following attacks: (a) *Bad-mouthing attacks*: it always votes "no" to a good node to increase the false positive probability of this good node being misidentified as a bad node; and (b) *Good-mouthing attacks*: it always votes "yes" to another bad node to increase the false negative probability of this bad node being undetected by the system.

We assume that a SN is compromised through capture, that is, it is captured by humans/robots after which its code is tampered and it is converted into a bad node. Since all nodes have an equal chance of being captured as they are being deployed randomly in the WSN operational area, the node compromise time may be considered as i.i.d. with a distribution function $F_c(t)$ as input which provides knowledge about the environment hostility. When a SN is diagnosed as compromised (whether a true negative or a false positive), the SN is recovered with probability P_r depending on its accessibility to a sink node with code reload capability.

All SNs each have an initial energy level E. The energy being consumed for sensor reading and reporting (to the sink node), packet routing, running code attestation (as a target SN), sending/receiving challenge/response messages for code verification (as a target SN or a verifier), and recovery (if diagnosed as compromised) are E_s, E_R, E_c, E_v and E_r, respectively, which can be calculated based on knowledge about the protocol design and the wireless bandwidth available.

B. Performance Model

We first derive the false positive probability P_{fp}^{DCA} and false negative probability P_{fn}^{DCA} as a result of distributed code attestation based on voting in Equations 1 and 2. We use the notation n for the number of neighbors per node initially, n_v for the number of verifiers, n_m for the majority of the verifiers, $n_b(t)$ for the number of bad SNs out of n at time t, and $n_g(t)$ for the number of good SNs out of n at time t, with $n_b(t) + n_g(t) = n$. Below we explain Equation 2 for the false negative probability at time t below. Equation 1 for the false positive probability can be explained similarly. A false negative results when more than the majority of the verifiers vote the target node (which is a bad SN) as a good node. The first term in Equation 2 accounts for the case in which more than 1/2 of the verifiers selected from the neighbors are bad SNs who will perform good-mouthing attacks by always voting "yes" to this bad node to increase the chance of this bad node being undetected. Here the denominator is the total number of combinations to select n_v verifiers out of n neighbor nodes, and the numerator is the total number of combinations to select at least n_m bad verifiers out of $n_b(t)$ nodes and the remaining good verifiers out of $n_g(t)$ nodes. The second term accounts for the case in which more than 1/2 of the verifiers selected from the neighbors are good SNs but unfortunately some of these good nodes mistakenly miss the target SN as a good node with probability P_{fn}, resulting in more than 1/2 of the verifiers (some of those may be bad SNs) voting "yes" for the target node. Here the denominator is again the total number of combinations to select n_v verifiers out of n neighbor nodes, and the numerator is the total number of combinations to select i bad verifiers not exceeding the majority n_m, j good verifiers who diagnose incorrectly with $i + j \geq n_m$, and the remaining $n_v - i - j$ good verifiers who diagnose correctly.

To use Equations 1 and 2, we need to know $n_g(t)$ and $n_b(t)$. The probability that a SN is compromised at time t, given that it was a good node at time $t - T$, is given by:

$$\begin{aligned} F_T &= 1 - P\{X > t | X > t - T\} \quad (3) \\ &= 1 - \frac{P\{X > t, X > t - T\}}{P\{X > t - T\}} = 1 - \frac{1 - F_c(t)}{1 - F_c(t - T)} \end{aligned}$$

The number of good neighbor SNs, $n_g(t)$, needed in Equations 1 and 2 is equal to $n_g(t-T)$ minus the number of newly compromised nodes over T, i.e.,

$$n_g(t) = n_g(t - T) - F_T \times n_g(t - T) \quad (4)$$

On the other hand, the number of bad neighbor SNs at time t is given by:

$$n_b(t) = n_b(t - T) + F_T \times n_g(t - T) \quad (5)$$

Note that $n_g(t)$ and $n_b(t)$ obtained above are good and bad neighbor SN populations before code attestation is performed. Thus, plugging $n_g(t)$ and $n_b(t)$ into Equations 1 and 2 will allow us to obtain P_{fp}^{DCA} and P_{fn}^{DCA} at discrete sensing interval time points, i.e., $t = iT$ with $i = 1, 2$, etc. Because code attestation is being performed only with probability q, and nodes recovery is being performed only with probability P_r, the bad and good neighbor SN populations are adjusted only when code attestation and recovery are performed. That is,

$$n_g^*(t) = n_g(t) + q \times n_b(t) \times (1 - P_{fn}^{DCA}(t)) \times P_r \quad (6)$$

$$n_b^*(t) = n_b(t) - q \times n_b(t) \times (1 - P_{fn}^{DCA}(t)) \times P_r \quad (7)$$

where $n_g^*(t)$ and $n_b^*(t)$ are good and bad node populations after code attestation. Next we calculate the probability that a SN is diagnosed as compromised at time t. There are two possible ways by which a SN is diagnosed as compromised. The first case is that the SN is compromised and it is correctly identified as a bad SN with probability $1 - P_{fn}^{DCA}$. The second case is that the SN is not compromised and it is incorrectly misidentified as a bad SN with probability P_{fp}^{DCA}. Hence, the probability that a SN is diagnosed as compromised at time t is given by:

$$\theta(t) = P_b(t) \times (1 - P_{fn}^{DCA}(t)) + P_g(t) \times P_{fp}^{DCA}(t) \quad (8)$$

Here $P_b(t)$ is the probability of the SN being a bad node and $P_g(t)$ is the probability of the SN being a good node at time t prior to code attestation being performed. Because of node homogeneity, we can calculate $P_b(t) = n_b(t)/n$ and $P_g(t) = n_g(t)/n$, with $n_g(t)$ and $n_b(t)$ defined by Equations 4 and 5, respectively. A SN will consume energy as a target node with probability q in every sensing period T. If this happens, $n_v \times (E_c + E_v)$ energy would be consumed because there are n_v verifiers with which it must communicate to execute the challenge-response procedure. Thus, on average $q \times n_v \times (E_c + E_v)$ energy will be consumed in a sensing interval. Further, it may be diagnosed as compromised with probability $\theta(t)$ in which case with probability P_r additional E_r energy is necessary for code recovery. This will consume $q \times \theta(t) \times P_r \times E_r$ energy on average in a sensing interval. In a sensing period there will be on average $n \times q$ target SNs in a SN's neighborhood needing it to serve as a verifier with probability n_v/n. This consumes $n \times q \times (n_v/n) \times E_v$ more energy. Lastly, A SN will consume E_s energy for sensing and reporting in every sensing period T and will also consume about $(n/4) \times E_R$ energy for forwarding packets from about $n/4$ neighbors who are located in the bottom left quadrant relative to the sink node, assuming geographic routing. Summarizing above, the amount of energy consumed by a SN in an interval $[t, t+T]$, denoted by $E_u(t)$, is given by:

$$E_u(t) = E_s + \frac{nE_R}{4} + q[n_v(E_c + E_v) + \theta(t)P_r E_r + n_v E_v] \quad (9)$$

Consequently, a SN will exhaust its energy after N_q sensing and reporting periods, with N_q given by:

$$E = \sum_{i=1}^{N_q} E_u(t = iT) \quad (10)$$

Let $R_q(t)$ denote the probability that a SN returns valid sensing readings in the sensing and reporting interval $[t, t+T]$, which is exactly the same as the probability that the node is a good node at time t when it returns sensor readings. Because of node homogeneity, $R_q(t)$ can be computed by:

$$R_q(t) = n_g^*(t)/n \quad (11)$$

In Equation 11 we use $n_g^*(t)$ as given by Equation 6 to account for our delay reporting strategy, i.e., sensing/reporting is performed after code attestation (if it is invoked) in a sensing interval. We follow the Byzantine system failure definition that if more than 1/3 of the SNs fail then the WSN fails. Let $R_s(t)$ denote the probability that the WSN is still healthy (i.e., with at least 2/3 of the SNs bring healthy) in the sensing and reporting interval $[t, t+T]$. We make use of $R_q(t)$ in Equation 11 to compute $R_s(t)$ as:

$$R_s(t) = \sum_{i=2/3N}^{N} \frac{N!}{i!(N-i)!} (R_q(t))^i (1 - R_q(t))^{N-i} \quad (12)$$

The Mean Time to Failure (MTTF) of the WSN, denoted by L_s, hence can be calculated by:

$$L_s = \int_0^{N_q T} R_s(t)dt \approx \sum_{i=1}^{N_q} T \times R_s(t = iT) \quad (13)$$

In the above formulation, we have used $N_q \times T$ as the upper bound to account for the maximum lifetime of the WSN due to energy exhaustion.

III. Numerical Results and Analysis

A. Environment Setup

We consider a WSN rapidly deployed in a hostile military environment for motion sensing. Mobile soldiers equipped with communication devices acting as mobile sink nodes are able to receive sensing reports regarding motions for combat advantages. The average number of neighbor SNs (n) is 15. The initial energy $E = 2.5$ joules of each SN is high enough to sustain at least 2-3 days of operations. The bandwidth is 250 Kbps. The transmission energy consumption rate of Crossbow Mica2 motes is 82.33 mJ/s [4] and the reception energy consumption rate is 50 nJ/bit. Therefore, the energy consumption for sensing and reporting, E_s, assuming 256 bits of sensing data, is about $256 \times 82.33/(250 \times 10^3) \approx 0.084$ mJ. The energy consumption for routing a packet, E_R, involves both reception and retransmission and is about $0.084 \text{ mJ} + 256 \times 50 \times 10^{-6} \approx 0.096$mJ. The computational power consumption rate is 50 mJ/s, the CPU power of a sensor node is 10 MIPs, and the code length is 2000 instructions [4]. Therefore, the energy consumption of a target SN for performing code attestation, E_c, is about $2000 \times 50/(10 \times 10^6) = 0.01$mJ. Code verification involves a verifier and a target SN exchanging a challenge/response pair. Assume that a challenge/response packet length is 8 bytes. Hence, the energy consumed for sending/receiving a challenge/response pair, E_v, is about $8 \times 8 \times 82.33/(250 \times 10^3) + 8 \times 8 \times 50 \times 10^{-6} \approx 0.024$ mJ. A recovery process involves reloading and checking the integrity of the code. The code size is 2KB. The energy consumed for code checking is the same as E_c above. So the energy consumption E_r for recovery is $2\text{K} \times 8 \times 50 \times 10^{-6} + 0.01 \approx 0.82$ mJ. The node false positive and false negative probabilities are small for code attestation after formal verification of the protocol design and testing before deployment. A range of 1-2% is reasonable and we set it to 1.5% in the paper. The sensing and reporting interval T is 1 min as dictated by the combat mission. The code recovery probability P_r is assumed to be 0.75. We consider $F_c(t)$ being an exponential distribution function with capture rate λ_c in the range of once per 10 minutes to once per 30 minutes, such that $F_T = 1 - e^{\lambda_c T}$ following Equation 3. The per-hop packet delay is computed by $(T_{RTS} + T_{CTS})/p + T_D$ where $T_{RTS} = 38 \times 8/(250 \times 10^3) \approx 1.22$ msec is the time to transmit a RTS packet based on DSSS, $T_{CTS} = 44 \times 8/(250 \times 10^3) \approx 1.41$ msec is the time to transmit a CTS packet, $T_D = 256/(250 \times 10^3) \approx 1.03$ msec is the time for a SN to transmit a sensor report packet, and $p = e^{-(n-1) \times \mu \times (T_{RTS} + T_{CTS})}$ is the probability other $n-1$ SNs within radio range not transmitting during $T_{RTS} + T_{CTS}$ and thus $1/p$ is the number of trials before a SN clears the channel for transmission based on RTS/CTS. Here μ is the packet rate per SN computed as $(1 + n/4 + q \times 2 \times n_v + q \times \theta(t) \times P_r) \times 1/T$ accounting for reporting, routing, code attestation, and code recovery activities during $[t, t+T]$ time interval.

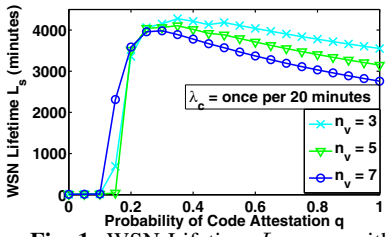

Fig. 1: WSN Lifetime L_s vs. q with varying n_v.

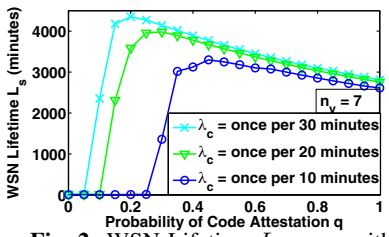

Fig. 2: WSN Lifetime L_s vs. q with varying λ_c.

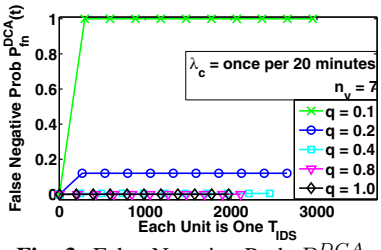

Fig. 3: False Negative Prob P_{fn}^{DCA} vs. Time with varying q.

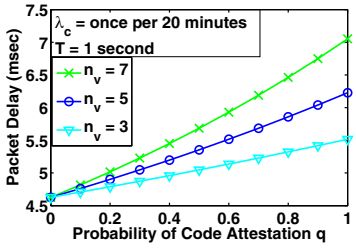

Fig. 4: Hop Delay vs. q with varying n_v.

B. Results

We summarize the major findings of the paper through 4 figures. Figure 1 shows the WSN lifetime L_s vs. q (probability of code attestation invocation in a cycle) with varying n_v (the number of verifiers for a target SN being attested) in the range of 3 to 7. The capture rate λ_c is set to once per 20 minutes to isolate its effect. We first observe that there exists an optimal q at which the WSN lifetime is maximized due to the tradeoff of energy consumption vs. reliability gain. Furthermore, the optimal q increases as n_v decreases. This is due to the fact that if we use fewer verifiers (i.e., a smaller n_v), we have to compensate it with a higher code attestation rate (i.e., a higher q) to be able to detect compromised SNs effectively. We also observe that using fewer verifiers ($n_v = 3$) results in the best WSN reliability because in this setting, energy conservation by using fewer verifiers outweighs reliability gain by using more verifiers to prolong the system lifetime.

Figure 2 shows the WSN lifetime L_s vs. q with varying λ_c (node compromise rate) in the range of once per 10 minutes to once per 30 minutes. The number of verifiers n_v is set to 7 to isolate out its effect. We again observe that there exists an optimal q value at which the WSN lifetime is maximized. Furthermore, the optimal q value increases as λ_c increases. The reason is that as λ_c increases, SNs are more likely to be compromised, so the system will have to increase the code attestation frequency (i.e., a higher q) to avoid security failure.

Figure 3 vividly displays how the system-level false negative probability P_{fn}^{DCA} evolves over time under a given q value. It confirms that a minimum q level exists (e.g., 0.1) below which P_{fn}^{DCA} is 1 (i.e., not being able to detect any bad node) because when q is too small (code attestation is not done frequently), there are simply too many bad nodes in the system, and because of bad node collusion (through good mouthing attacks during voting), the system-level false negative probability is 1. On the other hand, when q is greater than this minimum threshold, P_{fn}^{DCA} converges to a constant value approaching zero as q increases. The tradeoff between energy consumption vs. reliability gain is manifested by the fact that there is a diminishing return in P_{fn}^{DCA} as q increases further. For example when $q = 0.4$, P_{fn}^{DCA} is 4% but when $q = 1.0$, P_{fn}^{DCA} only improves to 1%. The effect of q on the system-level false positive probability P_{fp}^{DCA} is similar and not repeated here.

Finally, Figure 4 summarizes the impact of code attestation to performance measured by packet delay. As q or n_v increases, the packet collision probability increases due to increased traffic with code attestation and recovery. Consequently, the per-hop packet delay also increases. The most striking conclusion is that excessive code attestation (e.g., $q = 1$ or $n_v = 7$) hurts not only the system lifetime but also the system performance and there exists an optimal setting (e.g., $q = 0.4$ and $n_v = 3$) under which the system lifetime is maximized without sacrificing performance.

IV. Conclusion

By means of a novel probability model, we discovered the optimal operational settings for running distributed code attestation, including how often code attestation should be invoked (the q parameter) and how many neighbor verifiers should be used per code attestation event (the n_v parameter), so that the embedded WSN lifetime is maximized without sacrificing performance. In the future, we plan to extend the research to consider adversaries which can perform strategic capture, random or opportunistic attacks to elude detection.

References

[1] A. Seshadri, *et al.*, "SCUBA: secure code update by attestation in sensor networks," in *Proc. 2006 ACM Workshop on Wireless Security*, pp. 85–94.

[2] Y. Yang, *et al.*, "Distributed software-based attestation for node compromise detection in sensor networks," in *Proc. 2007 IEEE Symposium on Reliable Distributed Systems*, pp. 219–230.

[3] I. R. Chen, Y. Wang, and D. Wang, "Reliability of wireless sensors with code attestation for intrusion detection," *Inf. Process. Lett.*, vol. 110, no. 17, pp. 778–786, 2010.

[4] M. Calle and J. Kabara, "Measuring energy consumption in wireless sensor networks," in *2006 IEEE Symp. Personal, Indoor and Mobile Radio Communications*.

[5] L. F. Akyildiz, *et al.*, "Wireless sensor networks: a survey," *Computer Networks*, vol 38, pp. 393–422, 2002.

[6] C. Castelluccia, *et al.*, "On the difficulty of software-based attestation of embedded devices," in *2009 ACM Conference on Computer and Communications Security*.