SMATT: Smart Meter ATTestation Using Multiple Target Selection and Copy-Proof Memory

Haemin Park, Dongwon Seo, Heejo Lee and Adrian Perrig

Abstract A smart grid is verging on a promising technology for reforming global electrical grids. Currently, attackers compromise security and privacy by maliciously modifying the memory of smart grid devices. To thwart such attacks, software-based attestation protocols ensure the absence of malicious changes. A verifier and a target device locally generate their own checksums by memory traversal, and the verifier attests the target device by comparing the checksums. For smart grids, however, two challenges are arise in practically deploying the attestation protocol: verification overhead for large-scale networks and evasion of attestation by memory replication. To address these challenges, we propose a novel software-based attestation technique, termed SMATT (Smart Meter ATTestation), to address the aforementioned two challenges by leveraging multiple target selection and copy-proof memory. A verifier randomly selects multiple smart meters, and receives checksums. The verifier only compares the checksums instead of performing memory traversal, thereby remarkably reducing the computational overhead. To prevent memory replication, we design a customized copy-proof memory mechanism. The smart meter outputs garbage values when copy-proof memory sections are being accessed, and thus, attackers cannot replicate the memory. Furthermore, we define an SI epidemic model considering two

H. Park · D. Seo · H. Lee (✉)
Division of Computer and Communication Engineering, Korea University, Seoul, Korea
e-mail: heejo@korea.ac.kr

H. Park
e-mail: gaiger@korea.ac.kr

D. Seo
e-mail: aerosmiz@korea.ac.kr

A. Perrig
CyLab/ECE, Carnegie Mellon University, Pittsburgh, PA, USA
e-mail: adrian@ece.cmu.edu

S.-S. Yeo et al. (eds.), *Computer Science and its Applications*,
Lecture Notes in Electrical Engineering 203, DOI: 10.1007/978-94-007-5699-1_90,

attestation parameters, the number of infectious smart meters and the number of selected smart meters by a verifier, to enhance the malware detection accuracy of SMATT. In our experimental environments, SMATT takes only 20 s for a verifier to attest millions of smart meters. In addition, SMATT detects malware with over 90 % probability, when the malware tampers with 5 % of the memory.

1 Introduction

A smart grid, an ongoing modernization of a power grid, is a system that integrates existing electricity infrastructure with IT networks. This provides further benefits to customers and power utilities: reducing electricity costs for customer and controlling power generation for power utilities [1, 2]. Especially, a smart meter plays a key role to facilitate such beneficial services, and thus it becomes an attractive target for an adversary to gain an advantage and cause social chaos [1, 3–5]. For example, malware infiltration on easily accessible smart meters is a perplexing problem on a faraway utility, because it can manipulate electricity costs or fabricate metering information for a targeted home. This vulnerability can be monetized by attackers.

A technology for malware detection in an embedded device is to check the integrity of the device's firmware. In particular, software-based attestation techniques remotely verify whether the memory of the embedded device is maliciously changed [6, 7]. These techniques are based on a challenge-response protocol. A verifier sends a challenge to a target device, and then it reads entire memory in a pseudo-random fashion. The checksum as a result of pseudo-random memory traversal is sent from the target device to the verifier. Finally, the verifier computes the checksum in the same fashion, and compares it with the checksum from the target device.

However, the existing attestation techniques cannot be easily applied to smart meters. The complexity and scale of power grids cause new challenges: management of millions of devices and routine maintenance [2]. Especially, from attestation perspective, it is difficult for the verifier to attest millions of smart meters due to the heavy processing burden caused by checksum computation. Furthermore, recent research indicates that malware can subvert memory [8], and contribute to evasion of attestation [9].

In this paper, we propose a Smart Meter ATTestation mechanism, termed SMATT, that reduces computational overhead of a verifier and prevents memory replication to evade attestation. It leverages multiple target selection and copy-proof memory sections. Multiple meters manufactured with the same configuration, namely *identical smart meters,* are randomly selected by the verifier. Then, the verifier only compares the checksums without pseudo-random memory traversal. As a result, SMATT reduces the verification overhead. To prevent memory replication attacks, the smart meter sets the copy-proof memory sections

and monitors memory-related APIs. If the copy-proof memory sections are being accessed through the APIs, then the smart meter outputs garbage values as if they are real ones. Attackers obtain memory values ruined by the garbage, and the replicated smart meter fails to attest. Moreover, we define attestation parameters to enhance the malware detection accuracy of SMATT by using the popular SI epidemic model [10]. Our experiments show that SMATT reduces verification overhead, and detects memory falsification by malware with a high probability.

The main contribution of our work is twofold. First, SMATT reduces the verification overhead on the verifier; it is efficient for large-scale smart grids that consist of a large number of smart meters with a small number of verifiers. Second, SMATT enhances the robustness of attestation by preventing memory replication attacks.

The remainder of this paper is organized as follows. In Sect. 2, we introduce several problems, and describe the detail attestation procedure in the next section. Security analysis and evaluation are followed in Sects. 4 and 5. Finally, Sect. 6 presents our conclusion.

2 Problem Definition

In this section, we describe the weaknesses in software-based attestation protocols and state the problem that we aim to solve throughout this paper.

2.1 Weaknesses in Software-Based Attestation Protocols

Software-based attestation protocols have been proposed for embedded devices. Due to the constrained resources of embedded devices, the existing protocols pursue a lightweight design, and it can be suitable for the smart meter [2]. Nevertheless, there are several remaining issues yet to be solved in terms of verification overhead and malware infiltration [2].

Immense verification overhead by large-scale networks. Considering the complexity and scale of smart grids, a large number of smart meters will create a great computational overhead for a verifier. That is, a verifier linked to many smart meters should compute the checksums of each smart meter. For example, the verifier can become exhausted when many attackers send an overwhelming amount of miscalculated checksums. Therefore, it is a challenge to design an efficient software-based attestation protocol for the verifier that manages a large number of smart meters.

Evasion of attestation by memory replication attacks. Malware utilizes a linear scan of memory to create trial data from all possible memory positions [9]. An attacker can obtain whole memory values including critical information, and

can produce replicated smart meters. The attacker modifies the firmware of one's smart meter in order to make illegal profits such as modifying electricity usage. Meanwhile, the replicated smart meter transmits correct responses to verifiers and success to attest. For example, Song et al. [11] designed a one-way attestation protocol that leverages built-in values as challenges to compute a checksum. However, the challenges can be predicted by a memory replication attack, and thus, the attacker can compute the correct checksum.

2.2 Problem Statement

A viable attestation mechanism needs to address the following requirements:

- To reduce verification overhead on the verifier that handles a large number of smart meters.
- To prevent evasion of attestation by the memory replication attack.

Our goal is to design the attestation protocol that satisfies the two requirements.

3 SMATT: Smart Meter ATTestation

We introduce three assumptions that comply with the NIST guidelines for smart grids [3]. Then, we describe SMATT in detail. Finally, we determine two attestation parameters to enhance the malware detection accuracy of SMATT.

3.1 Assumptions

SMATT should incorporate the following three technical assumptions.

1. Time synchronization: Accurate and reliable time synchronization is essential to ensure that equipments operate correctly in the smart grid. Especially, a metering service based on time-of-use pricing is the reason that time synchronization is necessary.
2. Encrypted communication: Existing key management solutions such as Public Key Infrastructure (PKI) are considered as the basis of further innovation for secure communication between smart meters and verifiers [3].
3. Identical smart meters: A manufacturer produces identical smart meters installed an exact same firmware and configuration. The identical smart meters generate same checksums as the result of memory traversal. A verifier has information that indicates the groups of identical smart meters.

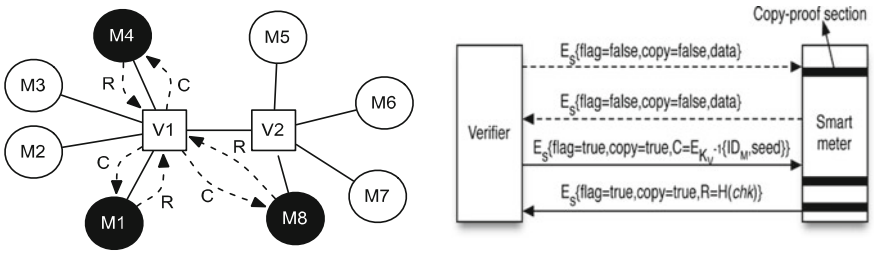

Fig. 1 Randomly selected smart meters (M1, M4, M8) exchange challenges (C) and responses (R) with a verifier (V1) (*left*). During the attestation period (*solid arrows*), V1 inserts a challenge (C), and receives a response (R) from the smart meter (*right*)

3.2 SMATT Operations

SMATT operations are based on a challenge-response protocol, which consists of four steps: multiple target selection, copy-proof memory generation, checksum generation and verification.

Multiple target selection. In existing attestation mechanisms, a verifier should perform checksum computing operations to attest a single target (smart meter), and it causes very high computational overhead for the verifier in large-scale networks like smart grids. Multiple target selection is designed to reduce the overhead problem by comparing checksums instead of computing them.

A verifier selects multiple identical smart meters to be attested, and receive checksums. Then, the verifier compares the checksums and is able to identify which smart meter sends a different checksum. Hence, the verifier can detect compromised meters without checksum computing operations. A detailed description is as follows:

1. A verifier has a private key (K_V^{-1}). Then, it shares a public key (K_V) and a symmetric key (S) with smart meters. All packets are encrypted by the symmetric key ($E_S\{\cdot\}$).
2. During usual period, the verifier occasionally transmits the encrypted packet, $E_S\{flag = false, copy = false, data\}$, to its managed meters. The $flag = false$ means that the verifier is in usual period.
3. During attestation period, the verifier (V1) determines a group to be attested. In Fig. 1 (left), there are two identical groups, G1 = {M1, M4, M8} and G2 = {M2, M3, M5, M6, M7, M8}, and V1 selects G1. Among the G1, V1 randomly selects multiple smart meters $M_N$to be attested.
4. The verifier sets the $flag = true$ and inserts a challenge, $C = E_{K_V^{-1}}\{ID_M, seed\}$. After decryption using K_V, ID_M is utilized to identify each selected meter. The seed becomes the input for the memory traversal algorithm.
5. The selected meter returns a response ($R = H(chk)$) to the verifier, where $H(\cdot)$ denotes a cryptographic hash function and chk denotes a checksum.

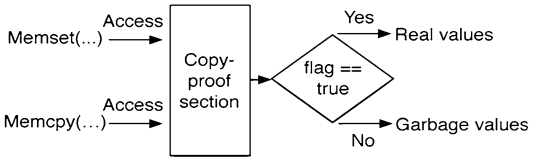

Fig. 2 Memory-related APIs are monitored by SMATT. When copy-proof selections are being accessed through those APIs, SMATT determines whether the outputs become real values according to the flag

Copy-proof memory section. An attacker can evade attestation using the memory replication attack (viz., Sect. 2). The effective scheme to prevent the attack is to utilize memory randomization. The values of special memory sections are periodically changed, so that the attacker cannot obtain the real values of whole memory. However, since the smart meter uses a flash memory as storage, the frequent changes of specific memory section can cause the "wear-out" problem that shortens flash memory life.

Therefore, we define copy-proof memory sections in the smart meter that addresses the wear-out problem. The copy-proof sections are scattered throughout the memory, and they are filled with garbage values. The smart meter monitors memory-related APIs, such as *memset* and *memcpy*. During attestation period, the memory-related APIs output real values of the copy-proof sections. During usual period, however, the APIs output random garbage values instead of real ones. Consequently, the attacker cannot replicate the copy-proof sections, and thus, the attacker's smart meter containing garbage copy-proof sections fails to attest. Figure 2 shows conceptual operations that output different memory values depending on the flag.

Moreover, the smart meter sets $copy = true$ if the memory-related APIs access to the copy-proof sections during attestation period. Thus, the verifier can recognize the illegal access to the copy-proof sections if the smart meter sends packets including $copy = true$ during usual period.

Checksum generation. SMATT adopts a block-based pseudo-random memory traversal technique that generates a checksum over the memory cells of the smart meter [7]. A seed (S) substitutes for the input value of $RC4(\cdot)$, which is a pseudo-random number generator. The memory traversal addresses (A) meets *size_of_memory* by modulo operation. *memory_traversal*(A, *size_of_block*) reads memory from address A to A+*size_of_block*, and outputs its result to MEM_i, where MEM_i denotes the memory values of the ith traversal. The *chk* is derived from the concatenation of MEM_i and CP_MEM_i, where CP_MEM_i denotes the memory values of copy-proof sections of the ith traversal. After generating the *chk*, A becomes the new seed for the next step. These four sequences are repeated until the number of iterations reaches a predefined maximum iteration number (T). Finally, the *chk* is hashed by a cryptographic hash function such as SHA-1 to avoid exposing the original *chk*. Equation (1) represents the checksum generation.

$$H(chk) = H(\sum_{i=1}^{T} (MEM_i || CP\_MEM_i) \quad (1)$$

Checksum verification. Selected meters send a $H(chk)$ to a verifier. In the verifier, the $H(chk)$ is compared with that of each selected meter. If these $H(chk)$'s are identical, the verifier determines that the smart meters are valid. Thus, SMATT reduces verification overhead because the verifier does not need to perform memory traversal.

3.3 Determining Attestation Parameters

SMATT leverages randomization and multiple target selection to reduce verification overhead and prevent evasion of attestation. However, if there are a sufficient number of compromised meters, only compromised meters can be selected by a verifier.

In other words, the verifier cannot distinguish between legitimate meters and compromised meters, since the compromised meters can provide the same incorrect checksum. To avoid the situation, a verifier should select one legitimate meter at least; Eq. (2) shows the condition

$$k > I(t), \quad (2)$$

where $I(t)$ denote the malware spreading ratio at a time t, and k denote the number of meters to be selected by a verifier. Here, we define two attestation parameters, $I(t)$ and k, in order to enhance the malware detection accuracy of SMATT.

Determining $I(t)$. Supposing a smart meter becomes either susceptible (S) or infected (I). We utilize a deterministic epidemic model, the SI model to analyze malware propagation in the smart grid. The SI epidemic model [10] for smart meters is described in Eq. (3).

$$\frac{dI}{dt} = \alpha SI, \quad (3)$$

where α is the infection rate of smart meters. It depends on the number of susceptibles (S) and infectives (I). Also, Eq. (4) explains α:

$$\alpha = \frac{\beta x}{N}, \quad (4)$$

where x means contacts per unit of time. β is the probability of infection when an infective contacts a susceptible. N means the total number of smart meters. Since we assume that a smart meter can be either S or I, S becomes $N - I$. Now, we can derive Eq. (5):

$$\frac{dI}{dt} = \alpha \cdot (N - I)I = \alpha NI - \alpha I. \quad (5)$$

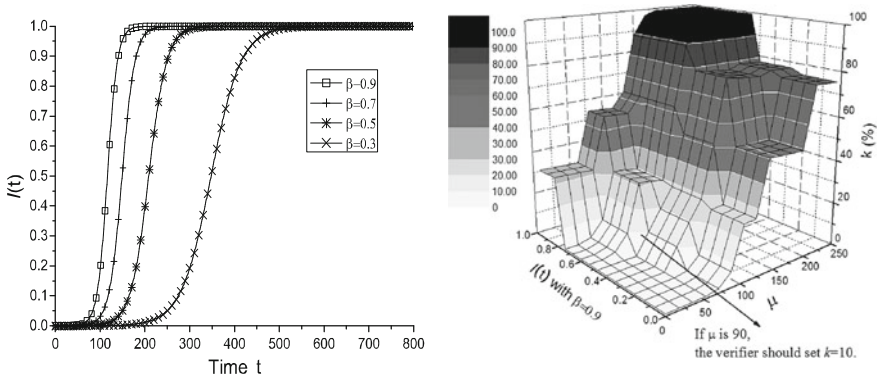

Fig. 3 According to research on malware propagation [17], malware has the characteristics of slow start and exponential propagation (*left*). The malware spreading ratio at a time t ($I(t)$) determines the number of meters to be selected (k). That is, k should be greater than the number of infected smart meters at a time μ, where μ means the attestation cycle (*right*)

Then, Eq. (6) is also derived by the general solution.

$$I(t) = \frac{N}{1 + cNe^{-\alpha Nt}}, \tag{6}$$

where c is a constant that depends on the initial conditions.

We use the same environment considered in the reports [12, 13]. The number of smart meters per house is 3 (gas, electricity, and water meters), and there are 105,000 meter/km^2 in an urban area such as London. The maximum contacts per unit of time is set to 0.1/s. Hence, we set $I_0 = 3$, $N = 105,000$ and $x = 0.1$, and obtain Eq. (7). The result of Eq. (7) is shown in Fig. 3 (left).

$$I(t) = \frac{N}{1 + \frac{1}{3}(N-3)e^{-\alpha Nt}}. \tag{7}$$

Determining *k*. Next, we determine the k. As we mentioned, a verifier should select at least one legitimate meter. That is, k is greater than the number of infected smart meters during attestation. By using the parameter, $I(t)$, the verifier sets $k > I(\mu)$, where μ means the attestation cycle. The k related to $I(\mu)$ is shown in Fig. 3 (right). Depending on the availability of network resources, an administrator can consider two possible cases to determine the k:

- In the case of a low-bandwidth smart grid network, a verifier should set k high. For example, if $\beta = 0.9$ and $\mu = 90$, then $k = N \times 0.1$.
- In the case of a high-bandwidth smart grid network, a verifier should set k low. For example, if β= 0.9 and $\mu = 60$, then $k = N \times 0.01$.

4 Security Analysis

In this section, we analyze the efficiency of SMATT in terms of time complexity, and we describe how SMATT prevents evasion of attestation by malware.

4.1 Verification Overhead

In practice, estimated numbers of meters per sector range from approximately 12,000 to 35,000 in London [12]. It can be seen that there is a large variation in the number of meters per sector. However, SMATT does not need memory traversal of the verifier. The time complexity of checksum verification in SMATT takes $O(N)$, while that of the existing attestation protocols takes $O\left(\frac{Nm \ln m}{b}\right)$, where N is the number of smart meters, b means the size of a block, and m denotes the memory size [7, 14].

4.2 Malware Infiltration

We define feasible attack scenarios, and then we discuss how SMATT thwarts them.

Attack scenarios and countermeasures. An attacker with the objective of monetization aims to inject malware into easily accessible smart meters. The malware contributes to evasion of attestation by using a proxy attack and a memory attack.

A proxy attack scenario. The attacker installs a proxy meter that generates a correct checksum instead of a compromised meter, and evades attestation. The attacker abuses 2 m: compromised and proxy meters. The compromised meter simply forwards the attestation request from the verifier to the proxy meter. Then, the proxy meter returns the correct checksum to the compromised meter. This enables the compromised one to evade attestation by forwarding the checksum to the verifier.

Unfortunately, the *ID* of each meter prevents against the proxy meter. The compromised meter should encrypt the challenge that includes the proxy *ID*, and sends it to the proxy meter. Because malware cannot infer the private key for the encryption, it has no chance to generate the new challenge, and evasion of attestation by the proxy meter is defeated.

A memory replication attack scenario. The attacker abuses the compromised meter that sends the correct checksum to the verifier, and evades attestation. By the

memory replication attack, the attacker obtains knowledge of the entire program memory and the correct checksum. The compromised meter returns the checksum to the verifier when attestation begins.

In SMATT, however, copy-proof sections prevent the memory replication attack. Since SMATT differentiates the output values of the copy-proof sections when the attacker attempts to access, the attacker cannot replicate the smart meter memory and fails to attest. A sophisticated attacker attempts to access the copy-proof sections only during attestation period, because it is the only time that SMATT gives real values. Nevertheless, the attacker cannot anticipate when attestation period begins ($false = true$) because of the packet encryption; therefore, the attacker cannot help guessing the timing of attestation period, and it leads $flag = false$ and $copy = true$, which means illegal access to the copy-proof sections during usual period.

5 Experiment and Results

We first establish an experimental environment. Then, we examine the computational overhead of the verifier in the environment, and measure the malware detection rate.

5.1 Experimental Environment

SMATT is implemented on two Android devices (as smart meters) and a laptop (a verifier). Verifier's hardware specification is Intel(R) Atom 1.60 GHz CPU and1 GB RAM, while the two Android devices are the Google development smartphone (Nexus-dev1), Scorpion 1 GHz CPU and 512 MB RAM. The Android smartphone provides similar hardware specifications to the smart meter [15]; moreover, Google announced that the Android OS can be applied at home in the smart grid [16].

5.2 Verification Overhead

We examine verification overhead by measuring the time overhead to attest lots of smart meters. The Android device simulates lots of smart meters by sending k checksums at once. And, we assume several variables related to smart grid requirements and the attestation parameters that we mentioned in Sect. 3. Based on

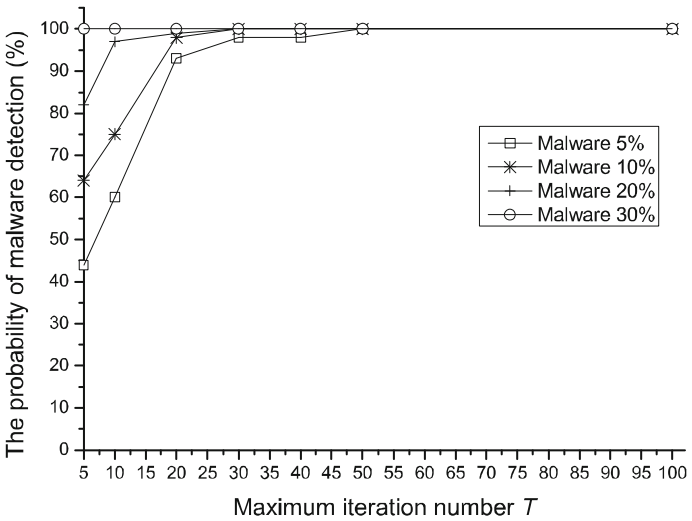

Fig. 4 The probability of malware detection: SMATT thwarts over 90 % of the attacks, if the malware tampers with 5 % of the memory of the smart meter

the requirements [12, 13], the network latency between the verifier and a smart meter is 5,000 ms and the data size is 35 bytes. In addition, we set $k = N \times 0.2$ and $N = 105{,}000$. In our environment, SMATT takes 20,047 ms, which is more efficient than the time of the typical software-based attestation as the verifier respectively sends a challenge to each smart meter. The time is approximately 6 days in the worst case. Therefore, the experimental result shows that SMATT has low verification overhead.

5.3 Malware Detection

We vary the fraction of malware in smart meter memory to measure the probability of malware detection. Figure 4 shows the probability of malware detection. SMATT on an Android device has a different fraction of malware ranging from 5 to 30 %, since the attacker can exploit the available space, approximately 11.6 %, on average [8] to inject malware. If malware takes 10 % of program memory, a smart meter should traverse memory contents approximately 30 times. Although malware changes a small amount of memory contents, such as 5 % of program memory, it can be detected by traversing memory 50 times. The experimental result shows that SMATT detects malware with a high probability.

6 Conclusion

In this paper, we propose a novel software-based attestation protocol for smart meters, SMATT, which provides a technique to reduces the processing burden of a verifier and prevents evasion of attestation by memory replication attacks. Multiple target selection contributes to simplify the verification operation of a verifier, and copy-proof memory protects against memory replication attacks. We analyze the attestation parameters to enhance the malware detection accuracy of SMATT. In our experiments, SMATT is remarkably faster than typical attestation protocols due to its reduction of verification overhead, and it has a high detection probability with low memory falsification rates.

Acknowledgments This research was supported by the National IT Industry Promotion Agency (NIPA) under the program of Software Engineering Technologies Development and Experts Education. Additionally, this research was supported by Seoul R & BD Program (WR080951).

References

1. McDaniel, P.D., McLaughlin, S.E.: Security and privacy challenges in the smart grid. IEEE Secur. Priv. **7**(3), 75–77 (2009)
2. Khurana, H., Hadley, M., Lu, N., Frincke, D.A.: Smart-grid security issues. IEEE Secur. Priv. **8**(1), 81–85 (2010)
3. National Institute of Standards and Technology (NIST): NIST IR 7628: Guidelines for Smart Grid Cyber Security (2010) http://csrc.nist.gov/publications/PubsNISTIRs.html
4. McLaughlin, S.E., Podkuiko, D., McDaniel, P.: Energy theft in the advanced metering infrastructure. In: CRITIS, pp. 176–187 (2009)
5. Mo, Y., Kim, T.H., Brancik, K., Dickinson, D., Lee, H., Perrig, A., Sinopoli, B.: Cyber-physical security of a smart grid infrastructure. In Proceedings of the IEEE (2011)
6. Seshadri, A., Perrig, A., van Doorn, L., Khosla, P.K.: SWATT: Software-based attestation for embedded devices. In: IEEE Symposium on Security and Privacy. (2004)
7. AbuHmed, T., Nyamaa, N., Nyang, D.: Software-based remote code attestation in wireless sensor network. In: GLOBECOM, pp. 1–8 (2009)
8. Castelluccia, C., Francillon, A., Perito, D., Soriente, C.: On the difficulty of software-based attestation of embedded devices. In: ACM Conference on CCS, pp. 400–409 (2009)
9. Hargreaves, C., Chivers, H.: Recovery of encryption keys from memory using a linear scan. In: ARES (2008)
10. Khelil, A., Becker, C., Tian, J., Rothermel, K.: An epidemic model for information diffusion in MANETs. In: Proceedings of the 5th ACM international workshop, MSWiM '02, pp. 54–60 (2002)
11. Song, K., Seo, D., Park, H., Lee, H., Perrig, A.: OMAP: One-way memory attestation protocol for smart meters. In: EEE ISPA Workshop SGSC, pp. 111–118 (2011)
12. Himayat, N., Johnsson, K., Talwar, S., Wang, X.: Functional requirements for network entry and random access by large number of devices. Technical Report IEEE802.16ppc-10/0049r1, IEEE 802.16 Broadband Wireless Access Working Group (2010)
13. Himayat, N., Talwar, S., Johnsson, K.: Smart grid requirements for IEEE 802.16 M2MNetwork. Technical Report IEEE C802.16ppc-10/0042r2, IEEE (2010)
14. Mitzenmacher, M., Upfal, E.: Probability and Computing—Randomized Algorithms and Probabilistic Analysis. Cambridge University Press, Cambridge (2005)

15. SmartSynch: Functions and Features i-210+c SmartMeter. http://smartsynch.com/pdf/i-210+c_smartmeter_e.pdf (2012)
16. Berst, J.: Will Google destroy Zigbee? http://www.smartgridnews.com/artman/publish/Business_Strategy/Will-Google-destroy-ZigBee-3681.html. (2011)
17. Zhu, Z., Cao, G., Zhu, S., Ranjan, S., Nucci, A.: A social network based patching scheme for worm containment in cellular networks. In: IEEE INFOCOM (2009)