SwapGuard: A Software-Only Solution for Attesting Hot-Swappable Devices in Power Grids

Xinshu Dong
Advanced Digital Sciences Center
Singapore
xinshu.dong@adsc.com.sg

Sumeet Jauhar
Advanced Digital Sciences Center
Singapore
sumeet.j@adsc.com.sg

Binbin Chen
Advanced Digital Sciences Center
Singapore
binbin.chen@adsc.com.sg

***Abstract*—Hot-swappable devices play an important role for achieving high availability and easy maintainability in power grids, especially in substations. However, they also present a potential attack vector for malware to penetrate an otherwise highly closed system. We propose *SwapGuard*, a software-only solution that effectively guards a power grid system against swapped-in devices infected with malware. Built upon existing software-based attestation techniques, SwapGuard's design can securely handle hot-swapping event and accommodate different types of devices and networks. We evaluate SwapGuard on real-world power grid devices. The experimental results show that SwapGuard incurs low overhead to the operation of the system, in terms of the additional time required before a swapped-in device starts to function, the overhead to bootstrap the whole system, and the overhead to defend against unattested devices.**

I. Introduction

As a critical infrastructure that keeps our lights on and business as usual, power grids demand both high availability and easy maintainability. The *hot-swappable* design of many power grid devices on the market provides an important engineering means for achieving both requirements. With such a hot-swapping capability, a faulty device, e.g., an I/O processing board in a substation, can be quickly replaced with another working one that has been prepared in advance. By hot-swapping a working device into the system, one can almost instantly restore the operations of the affected power system. Hence, hot-swapping design reduces the downtime of the systems and saves maintenance staff from the trouble and challenges associated with on-site debugging and diagnosis.

Despite many of their benefits, hot-swappable devices open up a new potential attack vector to power systems. In particular, the control and communications systems of a power grid are often operated in a closed network, isolated from public network. Yet hot swapping of a malware-infected processing board into the system is tantamount to a physical attack assisted by the innocent maintenance staff. The risk of having malware-infected devices cannot be underestimated: many devices in today's power systems are sourced from and maintained by different vendors, and they may be handled by multiple parties during the process. If malware penetrates power systems this way, it can bypass perimeter-based security defenses, such as firewalls, as those control devices inside the perimeter are supposed to be trusted.

There is a lack of effective solution to reduce the risk associated with hot-swappable devices. The prevailing signature- and heuristics-based intrusion detection solutions require substantial computing resources and frequent updates of their databases, making it difficult to deploy them in power grid systems. On the other hand, while secure hardware, e.g., Trusted Platform Modules (TPMs), can render a trusted foothold for security solutions to detect any malicious code residing in untrusted software [15], [1], [2], secure hardware features such as secure storage, access control, etc. are scarce among legacy devices in power grids.

In this work, we propose *SwapGuard*, a software-only solution to obtain high security assurance for hot-swappable devices. Our solution ensures that successfully verified devices are malware-free, while incurring small overhead on the operation of the power grid. The fact that it does not require the addition of any secure hardware makes it easier for adoption. Our solution leverages the existing *software-based attestation techniques* [19], [9], which use timing (instead of any secure hardware) to establish a root-of-trust for software integrity attestation: conceptually, if the running copy of a software has been modified, it will take extra computation time (compared to the original software) to compute a randomly-seeded checksum value over the software's original memory content. We make three contributions in this work.

- First, we identify several key challenges in attesting hot-swappable devices in power grid systems, including the lack of secure hardware support, the need for a lightweight solution, and the need to support different types of devices.
- To address the above challenges, we design *SwapGuard*, a software-only solution that provides a secure and efficient way to attest swapped-in devices. SwapGuard prevents malware-affected devices from being accepted into the system and launching attacks. SwapGuard also provides an efficient simultaneous all-board attestation scheme to bootstrap the trust of the whole system.
- We implement SwapGuard on hot-swappable boards of a remote terminal unit model that is used in real-world power grid systems. Our evaluation shows that SwapGuard's overhead is low, in terms of: 1) the small amount of time (around 2 seconds) to attest a newly swapped-in device; 2) the time to bootstrap the trust of whole system (around 2 seconds as well, thanks to the simultaneous attestation design), and 3) the overhead to secure CAN bus communication against unattested devices (around 1 millisecond of increase in round trip time).

The paper is organized as follows. Section II presents our threat model. Section III provides a brief introduction to the software-based attestation techniques that SwapGuard leverages. Section IV presents the problem setup and the challenges. Section V presents our SwapGuard solution and Section VI evaluates its performance. Section VII discusses the related work, and we conclude in Section VIII.

II. Threats Posted by Hot-Swappable Devices

Albeit being valuable for efficient failure recovery and system upgrade, hot-swappable devices can introduce a serious attack vector into the otherwise highly closed system of power grids. The replacement device hot-plugged into the power systems can potentially carry malware that has been implanted into the device via various channels, e.g., in laboratory, along the supply chain, during vendor customization, etc. Although such threats cannot be ruled out even for the original manufacturing and assembly processes of the devices, the more ad-hoc nature of post-sales maintenance and the involvement of different vendors/contractors pose a greater risk.

A. Threat Model

To study such cyber threats to hot-swappable power grid devices, we consider a concrete threat model as follows. We assume that in an operational power grid, any control device in substations can be hot-swapped with a replacement unit at any time. For identification purposes, each device has a read-only or hardware-configurable identifier, accessible to software. The control center's computer is trusted, as it resides in a well protected environment, where its security can be readily verified locally by the operators. As a common practice, we consider the control and communications network in the grid is closed, and only authorized devices and personnel can obtain access to it. By trusting the usual practice of stringent physical access control protocols guarding the control rooms and substations of power grids, we consider that the operators and engineers that program and install the replacement devices are not active attackers.

However, due to the malware in the vendor's programming and testing environment, the programmed control devices can be unwittingly infected by and further spread malware. Therefore, we assume the replacement units hot-swapped into the power systems during maintenance can contain malicious software, but without any altered or added hardware components. Nevertheless, we assume the authorized personnel maintaining the devices still has an out-of-band communication channel with the control center, which cannot be tampered with by malware, e.g., an LED indicator, authenticated web applications, or even SMS. Our last assumption is that the malware that enters the power systems via the hot-swapping channel can bypass traditional perimeter security solutions, such as firewall, as well as prevailing fault detectors [13], [7].

B. Problem Worsened with CAN Bus

One popular implementation of such hot swapping devices is connecting them via a Controller Area Network (CAN) bus in a remote terminal unit (RTU) [22]. With the built-in support from CAN bus, when a new device is plugged onto the CAN bus, it can instantly start communicating with the other devices. When a device functions in an unexpected manner (e.g., becomes unresponsive), or is unplugged from the CAN bus, it can be isolated from the operations and communications of other devices on the bus.

Nevertheless, the CAN protocol is known for its inherent lack of security. For example, devices on the CAN bus do not authenticate each other, and the messages sent on the bus are visible to all nodes connected to it. This makes the CAN protocol a convenient attack vector for perpetrators who have obtained a foothold in a system to compromise other parts of the network, e.g., manipulating or shutting down the engine, braking systems, and doors of a car from a compromised car control network [16], [6].

III. Leveraging a Software-Only Approach

We leverage the technique of software-based attestation for defending against malicious hot-swappable devices.

A. Software-based Attestation in a Nutshell

Software-based attestation [19], [9] starts with a trusted verifier sending a request with a pseudorandom number (*nonce*) to an untrusted prover device. Based on the nonce, the prover starts a random walk over its memory and computes a checksum based on the memory content it accesses in the walk.

To attest to the verifier that it has only genuine software in the memory, the prover needs to satisfy two conditions: 1) to produce the correct checksum, and 2) to produce it within an expected amount of time. The latter is an important constraint, since otherwise a malware-infected prover may launch memory copy attacks or memory compression attacks [19], [10] that can manipulate the memory content to restore the expected values and hide the malicious code. Researchers have designed the checksum computation logic in a careful way such that any such manipulation inevitably incurs more time in computing the expected checksum.

After the checksum step succeeds, a root of trust has already been established on the prover device. Then the verifier can trustfully request the root of trust to perform further checks, such as computing a keyed hash of the content of all storage units on the device, including secondary storage.

B. Benefits of Software-based Attestation

Several features of software-based attestation make it a desirable candidate for detecting malware that penetrates via hot-swappable devices.

Low deployment overhead. Hardware-based attestation relies on secure hardware, e.g., secure storage and access control, which is scarce among legacy devices in power grids. Instead, software-based attestation can be deployed without incurring any overhead to upgrade the hardware of legacy devices, as a software patch to the devices.

High assurance. Unlike traditional security mechanisms based on signatures and heuristics that are primarily capable of detecting known malicious code and attacks, properly developed software-based attestation solutions can produce high assurance that only expected legitimate code resides on the device

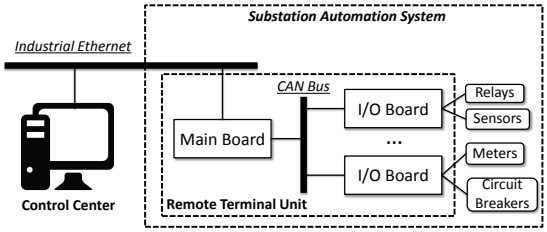

Figure 1. Example Scenario for Attesting Hot-Swappable Power Grid Control Devices. All boards in the RTU are hot swappable.

memory. This is especially important for critical infrastructures such as power grids.

Nevertheless, the development and deployment of software-based attestation for hot-swappable devices in power grid need to tackle several challenges in reality, as we will discuss next.

IV. Problem Setup & Challenges

As shown in Figure 1, we consider a trusted control center verifying the authenticity of software running on hot-swappable devices in a remote substation automation system (SAS). As there is inevitable downtime during attestation, we consider the control center attests one substation at a time in isolation, e.g. with direct connection between the control center and each substation. The RTU at each SAS comprises a main board that is connected to the control center via an industrial Ethernet, and connected to a few I/O boards via the CAN bus. The I/O boards are connected to sensors and meters for measurement, as well as relays and circuit breakers for digital control. All boards of the RTU can be hot swapped. The verifier needs to detect any malware when it enters the closed network via hot-swapping during substation maintenance.

To accomplish such attestation via software-only means, several challenges need to be addressed as we elaborate below.

Hot-swapping may happen at any board, in any time. It is important to ensure that the attestation can be securely performed regardless of whether the newly plugged-in board is the main board (which can perform man-in-the-middle attack between the verifier and the other IO boards) or the IO board (which does not directly connect to the verifier). Furthermore, we need to ensure that the new board cannot cause any physical harm before its trustworthiness is attested. For power grids, even a single piece of malicious command or false data from a malware-infected plugged-in board can cause serious damage. As discussed in Section II-B, the broadcasting nature and insecure design of CAN bus makes it rather easy for malware to send out crafted messages and impersonate others.

The requirement of low overhead and high availability. When one introduces security countermeasures to industrial control systems like power grids, it is compulsory that the new measure should reduce its impact to the system to the minimum. There should be as small as possible (if any) overhead for the system's regular operations, especially the computing and communication capability of many legacy devices in the system are rather limited. Also, the system's downtime (if any) should be kept as low as possible.

Proxy attacks among interconnected devices. While one can assume that the procedure requires the swapped-in devices to be inserted and attested one by one, there are cases (e.g., the initial trust bootstrapping, or major system upgrades) where all boards in the network need to be attested. As timing is a key factor in software-based attestation, a secure scheme needs to ensure that when one device is being attested, another device cannot impersonate it while retaining its malware afterwards, i.e., the proxy attack [19], [11]. The presence of devices with the same or different specifications, together with the presence of different types of networks, make the problem non-trivial to solve.

V. SwapGuard: Attesting Hot-Swappable Devices

To address the challenges discussed earlier, our solution SwapGuard employs several techniques. We first consider the handling of a single plugged-in device in Section V-A, then we present in Section V-B a simple yet efficient scheme for performing an all-board attestation over an entire substation automation system, so as to bootstrap the needed trust.

A. Secure and Efficient Handling of a Swapping Event

SwapGuard uses *lightweight encryption* and *admission requests* to securely handle a newly plugged-in device.

Defending against pre-attestation attacks. To prevent a plugged-in but yet-to-be-attested device from launching attacks, some form of authentication is required over communication channels. We achieve so by letting the verifier dispatches a symmetric key to a new device only after its successful attestation. The key is refreshed periodically and used to encrypt all CAN bus communications between devices. In principle, a keyed hash can be used to authenticate the messages. However, since each CAN bus frame has only 8 bytes of data field, we use a fast stream cipher HC128 [3] in this work to avoid incurring additional CAN bus frames to transmit the hash values. Beyond our proof-of-concept prototype of such a CAN encryption scheme, we can potentially adopt more comprehensive CAN bus security enhancements in future [26], [5].

When a new device d_i' is hot-swapped into the system, it does not possess the correct key, without which it is unable to send proper messages to other devices. Thus, it cannot leak anything out of the closed system, send malicious commands to other existing devices, or propagate to infect other devices[1].

Nonetheless, the malware on d_i', if any, can still launch two types of attacks. 1) It can ignore its read-only or hardware-configured identifier, and use the CAN bus address with the highest priority (i.e., with the smallest ID value) to send CAN bus messages as fast as possible, so as to jam the bus communications. 2) It can issue malicious digital signals to intelligent electronic devices (IEDs) connected to it, such as relays, which do not go through the CAN bus.

For 1), an I/O board attempting to constantly occupy the CAN bus can be detected and reported by a *bridging device* that is connected to both the CAN bus network and the network with the verifier. For example, the main board in

[1]There is, however, a theoretical possibility that network drivers that are configured to ignore d_i''s messages can be potentially compromised with a mal-formed message. However, we do not consider such a constrained attack vector in this paper.

Figure 1 serves as the bridging device for the whole RTU. The bridging device can promptly report to the verifier via the network that the swapped-in board has no control. An out-of-band channel to maintenance staff can be used to trigger the necessary response. If it is the bridging device itself that contains the bus-jamming malware, the jamming will be gone if the device attests itself successfully (since it needs to load the genuine image in order to pass the attestation). Otherwise, through a periodic polling to the bridging device, the verifier will be able to detect the absence of an attested bridging device. For 2), we also require the verifier to provide an out-of-band channel to the staff who hot swap the devices, informing them whether the newly plugged-in device has passed the attestation. They should only connect the plugged-in device with its peripherals after it has been successfully attested.

Admission requests and preventing impersonation attacks. For the newly plugged-in device to acquire the symmetric key and start communicating with the rest of the system, it needs to explicitly issue a plaintext request to the verifier for attestation. Such an explicit *admission request* design allows the verifier to react to swapping event at any time on demand, without the need to periodically polling all hot-swappable devices. In other words, it addresses the shortcoming of the present CAN bus implementation that no reliable notification is provided when a new device joins the group. Once the verifier receives such an admission request, it issues an attestation request to the requesting device, and if successful, sends the symmetric key to it[2].

It requires an additional design to guard the admission request design against malware. Note that such a newly swapped-in device d_{new} can be malicious, and thus can impersonate another clean device d_c in its admission request. Upon receiving the admission request, the verifier would proceed to issue an attestation request to d_c, which being malware-free, in turn produces a good checksum within the time constraint. Then the verifier sends the symmetric key to d_c. The problem occurs if both d_{new} and d_c share a same CAN bus, where all messages are broadcast. d_{new} would receive the symmetric key as well, even without going through the attestation process. The security mechanism would be bypassed this way.

To prevent such impersonation attacks for admission requests, whenever the bridging device receives an admission request apparently from a device d_{req}, it will send a plaintext status-inquiry message to d_{req}. If d_{req} is indeed a newly plugged-in device, it will respond that it has not acquired the key also in plaintext. However, if d_{req} is impersonated by another device d_{mal}, d_{req} will respond to the bridging device with a message stating that it possesses the key. This message will be encrypted by the key. If d_{mal} attempts to tamper with or block such a response message (e.g., by occupying the CAN bus), it will be detected by the bridging device. The bridging device will only forward the admission request to the verifier, if it receives only one plaintext message that d_{req} has no key and during which the CAN bus is not jammed.

[2]This per-device attestation as well as the all-network attestation described later are conducted via plaintext messages.

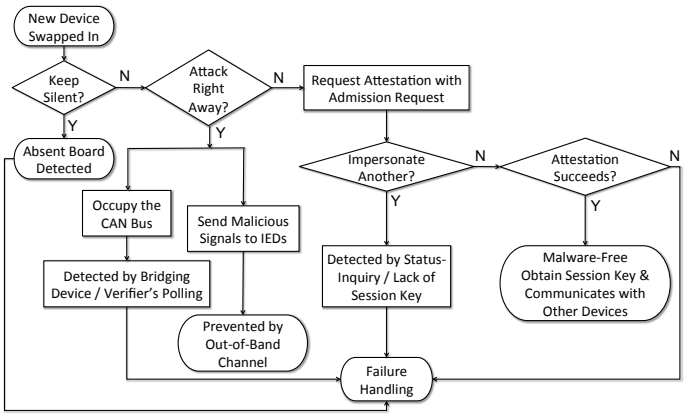

Figure 2. Possible Behaviors of a Swapped-in Device under SwapGuard.

This prevents impersonation of newly swapped-in devices other than the bridging device itself. A malicious bridging device can still skip the status-inquiry message, and directly send an admission request to the verifier, impersonating another device. To address this threat, we require the admission request to the verifier to be encrypted with the current key, unless the request claims to be from the bridging device itself. A newly swapped-in bridging device does not have the key, and can only send a plaintext admission request to the verifier, honestly claiming itself as the initiator.

Figure 2 illustrates the possible behaviors of a newly swapped-in device under SwapGuard. As can be seen, our design ensures that only a device with authenticate software can be attested successfully, and no device can cause harmful physical impacts prior to successful attestation.

B. A Simultaneous All-board Attestation Scheme.

We now briefly introduce an all-board attestation scheme, which bootstraps a trustworthy state of the system to guard against cyber threats from subsequent hot swapping of devices. This can be performed during initial solution deployment, and on demand during major system upgrade operations. For such all-board attestation, a key consideration is to keep disruption to system availability as little as possible.

To reduce impact on system availability, SwapGuard adopts a simple strategy that runs software-based attestation on all boards in an RTU almost "simultaneously", i.e., starting almost at the same time, and finishing at times with very small intervals between one another. The intuition is that since all devices being attested in the network are kept busy for almost the same period of time, they cannot help one another by launching proxy attacks. The reason that they cannot have exactly the attestation start and finish time is due to practical concerns, where the bridging device (the main board) needs to be up and running to receive attestation requests returned in CAN bus messages by the I/O boards. Moreover, since the processing of CAN bus messages at the bridging device can be a performance bottleneck, the other devices need to have slightly different attestation finish time so that their response messages will not all arrive at the bridging device simultaneously.

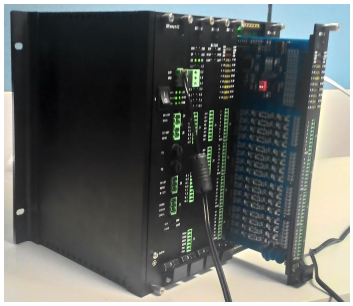

Figure 3. Photo of the RTU in Our Experiment

Unified checksum response time ($UTime$). To accommodate the differences between devices in terms of capabilities of finishing the computation and communication for the checksum result, we propose a *unified checksum response time*, $UTime$ to characterize the overall time needed for a device to respond with a correct checksum for a given checksum function, including the time for checksum computation, e.g., device architectures [20], as well as that for delivering the result back to the verifier, e.g., network latency [11].

Adjusting device $UTime$ with memory walk iterations. Based on $UTime$, SwapGuard designs checksum functions for different devices in the substation automation systems such that they each incur *almost* the same expected $UTime$ to make proxy attacks impossible to evade detection. In particular, SwapGuard first computes the minimal number of iterations required for a specified probability (e.g., 99.99999999% in this paper) that all memory locations are accessed in the random memory walk of the checksum function. Then it selectively increases the number of iterations for certain devices to adjust their $UTime$ for a desirable schedule, where the bridging device is expected to finish attestation first, while others finish marginally later in succession.

The proposed simultaneous attestation *largely* achieves a close-to-shortest full-network attestation time, based on a given probability required for memory coverage. As we show in our experiments in Section VI, such attestation can take around 2.1 seconds for an entire substation automation system.

VI. Evaluation

We evaluate SwapGuard on an RTU model that is used in real-world power grid substation automation systems.

A. Evaluation Setup and Checksum Logic Implementation

As shown in Figure 3, the RTU evaluated is equipped with 1 main board and 4 I/O boards, which are connected by a CAN bus. All boards are hot swappable. The specification of the computation capability of the boards can be found in Table I. The main board can allow up to 60 I/O modules to be connected to it via the CAN bus. The I/O boards sample analog and digital data from the connected sensors and meters, send digital signals to IEDs, and write the data to the CAN bus for transmission to the main board. The main board has an Ethernet communication interface (with a WIZnet W5100 Ethernet chip [23], which implements a fully hardwired TCP-IP stack operating at 100Mbps) for communication with the upstream control center via the IEC 60870-5-104 protocol.

Board	Processor	CPU Clock	RAM	ROM
Main	NXP LPC2292, ARM7TDMI-S	60MHz	16KB	256KB
I/O	NXP LPC1756, Cortex-M3	100MHz	16/32KB	256KB

Table I
SPECIFICATIONS OF THE HOT-SWAPPABLE BOARDS IN THE RTU

Prob.[Full coverage]	Computation time
99.99%	0.546 sec
99.9999%	0.668 sec
99.999999%	0.787 sec
99.99999999%	0.951 sec

Table II
CPU TIME NEEDED FOR CHECKSUM COMPUTATION AT THE 16KB-RAM I/O BOARD, GIVEN THE DIFFERENT REQUIREMENTS ON THE FULL COVERAGE PROBABILITY

All the boards have on-chip CAN controllers for CAN bus communications. The CAN protocol requires that all the devices on the shared CAN bus operate at the same bit/baud rate, which is set to 125kbps in our experiments.

We have implemented a checksum function based on that in [9], where the detailed design of the checksum logic can be found. We claim no contribution for the checksum logic in this work, and due to space limit we have to refer interested readers to the original paper for the checksum logic design. Our checksum function contains 26 ARM instructions for each memory access in the random memory walk. The random access speed of ROM is lower than RAM, however, with an on-chip hardware accelerator, their sequential access speeds (for loading code) are almost identical. Hence, when applying the checksum scheme in [9], we check both ROM and RAM to prevent an attacker from launching a memory copy attack by executing malicious code from the ROM while keeping the RAM unchanged.

B. Performance Evaluation

We set up our experiment scenario as in Figure 1, and use a desktop computer to mimic the verifier at the control center. The attestation request is sent from the verifier to the main board of the RTU via a switched Ethernet network.

Time needed for attesting individual devices. Figure 4, Figure 5 and Figure 6 plot the cumulative distribution functions for the end-to-end latency of attesting 2 types of I/O boards and the main board of RTU over 100 runs, respectively. They demonstrate that the time required for attesting an individual swapped-in device takes from around 1 to slightly more than 2 seconds, with most of the time spent on checksum computation. As required by software-based attestation, we compute the checksum at the highest microprocessor frequency, i.e., 60MHz for the main board, and 100MHz for the I/O board. Compared to the manual process of hot swapping itself, which usually takes more than 10 seconds, such an extra overhead of $1-2$ seconds is likely acceptable. The attestation on the I/O boards is almost twice faster as compared to the main board (0.95 vs 2.03 seconds) for computing the checksum. This is due to that the I/O board processors operate at a higher frequency than that of the main board.

The variance in the response time is mainly due to the communication delay variation. For both types of boards, the variation is less than 0.5ms. In comparison, even under a rather pessimistic assumption[3], the malware needs to incur at least

[3]Our implementation uses 30 CPU cycles for each checksum update, and an attacker needs to incur at least one additional CPU cycle for each update.

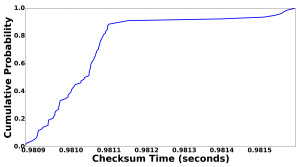

Figure 4. Cumulative Distribution Function for Time Taken for Attesting 32KB-RAM IO Board (100 runs)

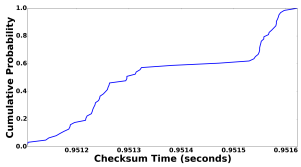

Figure 5. Cumulative Distribution Function for Time Taken for Attesting 16KB-RAM IO Board (100 runs)

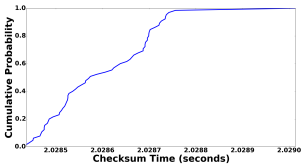

Figure 6. Cumulative Distribution Function for Time Taken for Attesting Main Board of RTU (100 runs)

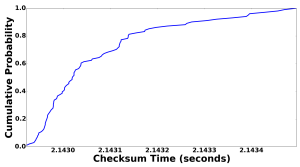

Figure 7. Cumulative Distribution Function for Time Taken for Attesting the Entire RTU (100 runs)

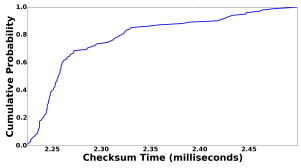

Figure 8. Cumulative Distribution Function on Overall RTT between verifier and I/O Board: *Original, plaintext CAN messages* (100 runs)

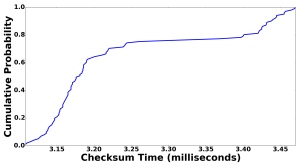

Figure 9. Cumulative Distribution Function on Overall RTT between verifier and I/O Board: *with Encrypted CAN Messages* (100 runs)

Board	#Base Iterations	#Adjusted Iterations	Expected Duration
Main Board	260000	260000	2.03 sec
I/O Board 1 (16KB)	260000	569000	2.05 sec
I/O Board 2 (16KB)	260000	572000	2.06 sec
I/O Board 3 (32KB)	275000	575000	2.07 sec
I/O Board 4 (32KB)	275000	578000	2.08 sec

Table III
BASE & ADJUSTED NUMBER OF ITERATIONS OF MEMORY WALK, AND THE EXPECTED ATTESTATION DURATION (*excluding* NETWORK LATENCY)

3% of additional time (or around 28ms for the I/O board and around 60ms for the main board) in order to produce the correct checksum. Both are significantly larger than the 0.5ms response time variation. As a result, we observe no false-positive that a genuine image fails to attest itself due to the varying response time.

Recall that the checksum algorithm conducts random accesses to on-chip memory, which are repeated for a number of times to achieve a high probability of covering every word in the memory. The probability can be estimated by solving a coupon collector problem. Table II estimates how the overall attestation time increases with a higher target probability for achieving full coverage. Our default setting achieves ten of 9, making the expected duration before one encounters a memory missing event more than 100 years, even if one repeatedly runs the checksum algorithm. As shown in the Table, one can reduce the attestation time by around 43%, if one allows a lower coverage guarantee of 99.99%.

Time needed for all-board attestation. Thanks to our simultaneous design, the overall time needed for attesting all the boards in the RTU is only slightly longer than that for the main board. To bootstrap the attestation, the main board sends out a broadcast CAN bus message once it receives an attestation request from the verifier. Any board on the CAN bus that receives the message, including the main board itself, immediately starts running the checksum on its software image. One implementation challenge we have to deal with here is that, we need to switch the main board's frequency between 60MHz (for its own checksum computation) and 48MHz (for CAN bus communication), since 60Mhz is the highest frequency of the microprocessor but 48Mhz is the highest microprocessor frequency allowed if it needs to use its CAN bus controller. In addition, we find that on our experimented RTU, the main board becomes a performance bottleneck when processing CAN bus messages that are continuously generated by IO boards. To make sure all CAN bus messages carrying the checksum results are properly processed by the main board, we have to adjust the iterations of memory walk in the checksum functions for different boards, as shown in Table III. Besides, we also introduce a 5-millisecond delay between sending each adjacent CAN bus messages. The variance in the expected unified checksum response time $UTime$ between different boards is around 10 millisecond, i.e., 0.5% over the $UTime$ for one board in the all-board attestation. This is smaller than the minimal 3% additional time needed by malware to bypass the detection of the attestation. The total duration for attesting the 5 boards is only slightly longer than that for attesting the main board (2.14 vs 2.03 seconds), as shown in Figure 7. This illustrates that our attestation scheme has a highly reduced duration compared to schemes that attest one device after another, e.g., in [11].

Performance overhead for message encryption. We implement message encryption for CAN bus communications using the HC128 stream cipher available via the wolfSSL library [25], which is a crypto library specifically designed for embedded devices. The HC128 stream cipher is one of the finalist ciphers in the eSTREAM project, and the fastest stream cipher available in wolfSSL.

Figure 8 illustrates the round trip time (RTT) without encryption from the verifier to an I/O board (via the Ethernet, main board, and the CAN bus), which is less than 2.45 milliseconds. Figure 9 illustrates the RTT when the communications over the CAN bus are encrypted with HC128. The time has been increased by around 1-millisecond to around 3.45 milliseconds. The overhead comprises of 4 cryptographic operations, i.e., a message encryption by the main board, a message decryption by the I/O board, followed by a message encryption by the I/O board, and a message decryption by the main board. We expect such overhead can be further reduced with more optimized implementation of the cryptographic primitives.

VII. RELATED WORK

Hot-swapping is a popular technique for various electrical and electronic systems [12], [14]. In a general context, malware infection through hot-swappable devices has been well studied. For example, [21] analyzes sophisticated malware embedded in USB firmwares. However, in industry control

systems like power grids, there has been scarce research literature that systematically studies the technical mechanisms to defend against such an attack vector. In practice, many organizations rely on policy/procedure countermeasures to mitigate the risk. SwapGuard provides an effective technical means to support the existing policy/procedure measures.

A rich set of literature have attempted to address challenges with software-based attestation (e.g., [11], [19], [8], [20], [17], [18]), mostly focusing on attesting an individual device that is assumed to be isolated from others. The closest work to us is perhaps the VIPER scheme [11], which aims to securely attest multiple computer peripherals on a bus. Viper considers the case where a malware-infected peripheral may acquire assistance from other peripherals or external devices to produce the expected evidence while still meeting the timing constraint. Our system model and threat model differ from Viper in several aspects. Specifically, Viper does not consider the attack prior to the attestation of a device, and they assume it is possible to sort all peripherals according to their computing speed. Also, availability is less a concern in Viper's setting. In comparison, SwapGuard addresses these unique challenges that arise in a power grid setup.

The lack of basic security mechanisms with the CAN bus receives increasing attention over the past few years, especially with some major security breaches to cars [16]. Several research efforts have proposed mechanisms to strengthen its security [24], [4]. SwapGuard uses a lightweight secure enhancement of CAN protocol as a building block to guard against new plugged-in device that contains malware. The focus of SwapGuard is how to use such a component to help ensure the security of the overall design. Hence, SwapGuard can potentially leverage other security enhancement for the CAN bus as a building block.

VIII. Conclusion

In this paper, we present SwapGuard, a software-only solution for attesting hot-swappable power grid devices. SwapGuard can effectively prevent a malicious device swapped into the power system from being accepted or launching various attacks. It also provides an efficient all-board attestation scheme that can bootstrap a trustworthy state of the entire system. We evaluate SwapGuard with power grid devices presently deployed in power grids. The time required for attesting a hot-swappable device is around 2 seconds, so is the time to bootstrap the trust of a multi-board system. While this is likely an acceptable delay given that the physical operation of swapping devices usually takes more than 10 seconds, one of our ongoing efforts looks at how to further reduce the attestation time by optimizing the checksum design.

Acknowledgment

The work was funded in part under the Energy Innovation Research Programme (EIRP, Award No. NRF2014EWT-EIRP002-040), administrated by the Energy Market Authority (EMA) and in part by the research grant for the Human-Centered Cyber-physical Systems Programme at the Advanced Digital Sciences Center from Singapore's Agency for Science, Technology and Research(A∗STAR). The EIRP is a competitive grant call initiative driven by the Energy Innovation Programme Office, and funded by the National Research Foundation (NRF).

References

[1] N. Asokan, F. Brasser, A. Ibrahim, A.-R. Sadeghi, M. Schunter, G. Tsudik, and C. Wachsmann. SEDA: Scalable embedded device attestation. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS, 2015.

[2] K. E. Defrawy, A. Francillon, D. Perito, and G. Tsudik. SMART: Secure and minimal architecture for (establishing dynamic) root of trust. In *Proceedings of 19th Annual Network & Distributed System Security Symposium*, NDSS, 2012.

[3] ECRYPT. estream: the ecrypt stream cipher project. http://www.ecrypt.eu.org/stream/.

[4] B. Groza, S. Murvay, A. Herrewege, and I. Verbauwhede. Libracan: A lightweight broadcast authentication protocol for controller area networks. In *Proceedings of the 11th International Conference on Cryptology and Network Security*, CANS, 2012.

[5] A. Happel. Secure communication for CAN FD. http://vector.com/portal/medien/cmc/press/RDI/Security_CAN_Newsletter_201411_PressArticle_EN.pdf.

[6] A. G. Illera and J. V. Vidal. Dude, wtf in my can! https://www.blackhat.com/docs/asia-14/materials/Garcia-Illera/Asia-14-Garcia-Illera-Dude-WTF-In-My-Can.pdf, 2014.

[7] J. Kim and L. Tong. On topology attack of a smart grid: Undetectable attacks and countermeasures. *IEEE Journal on Selected Areas in Communications*, 31(7):1294–1305, July 2013.

[8] X. Kovah, C. Kallenberg, C. Weathers, A. Herzog, M. Albin, and J. Butterworth. New results for timing-based attestation. In *Proceedings of the IEEE Symposium on Security and Privacy*, SP, 2012.

[9] Y. Li, Y. Cheng, V. Gligor, and A. Perrig. Establishing software-only root of trust on commodity systems: Facts and fiction. In *Proceedings of the 23rd Security Protocols Workshop*, SPW, 2015.

[10] Y. Li, J. M. McCune, and A. Perrig. SBAP: Software-based attestation for peripherals. In *Proceedings of the 3rd International Conference on Trust and Trustworthy Computing*, TRUST, 2010.

[11] Y. Li, J. M. McCune, and A. Perrig. VIPER: Verifying the Integrity of PERipherals' Firmware. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS, 2011.

[12] Linear Technology. Hot swap controllers. http://www.linear.com/products/hot_swap_controllers.

[13] Y. Liu, P. Ning, and M. K. Reiter. False data injection attacks against state estimation in electric power grids. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS, 2009.

[14] Maxim Integrated. Hot swap controllers. https://para.maximintegrated.com/results.mvp?fam=hot_swap.

[15] J. M. McCune, B. J. Parno, A. Perrig, M. K. Reiter, and H. Isozaki. Flicker: An execution infrastructure for tcb minimization. In *Proceedings of the 3rd ACM SIGOPS/EuroSys European Conference on Computer Systems*, Eurosys, 2008.

[16] P. Paganini. Car hacking: You cannot have safety without security. http://resources.infosecinstitute.com/car-hacking-safety-without-security/, 2014.

[17] A. Seshadri, M. Luk, A. Perrig, L. V. Doorn, and P. Khosla. Using fire and ice for detecting and recovering compromised nodes in sensor networks. Technical report, Carnegie Mellon University, December 2004.

[18] A. Seshadri, M. Luk, A. Perrig, L. van Doorn, and P. Khosla. SCUBA: Secure code update by attestation in sensor networks. In *Proceedings of the 5th ACM Workshop on Wireless Security*, WiSe, 2006.

[19] A. Seshadri, M. Luk, E. Shi, A. Perrig, L. van Doorn, and P. Khosla. Pioneer: Verifying code integrity and enforcing untampered code execution on legacy systems. In *Proceedings of the Twentieth ACM Symposium on Operating Systems Principles*, SOSP, 2005.

[20] 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*, SP, 2004.

[21] SRLabs. Turning USB peripherals into BadUSB. https://srlabs.de/badusb/.

[22] Texas Instruments. Introduction to the controller area network (CAN). http://www.ti.com/lit/an/sloa101a/sloa101a.pdf, 2008.

[23] WIZnet. W5100. http://www.wiznet.co.kr/product-item/w5100/.

[24] M. Wolf, A. Weimerskirch, and C. Paar. Security in automotive bus systems. In *Workshop on Embedded Security in Cars(ESCAR)*, 2004.

[25] wolfSSL. wolfSSL - Embedded SSL Library. https://www.wolfssl.com/.

[26] J. Yoshida. CAN Bus Can Be Encrypted, Says Trillium. http://www.eetimes.com/document.asp?doc_id=1328081.