Device Attestation: Past, Present, and Future

Orlando Arias*, Fahim Rahman†, Mark Tehranipoor†, and Yier Jin†
*Department of Electrical and Computer Engineering, University of Central Florida
†Department of Electrical and Computer Engineering, University of Florida
oarias@knights.ucf.edu, fahim034@ufl.edu, tehranipoor@ece.ufl.edu, yier.jin@ece.ufl.edu

***Abstract*—In recent years we have seen a rise in popularity of networked devices. From traffic signals in a city's busiest intersection and energy metering appliances, to internet-connected security cameras, these embedded devices have become entrenched in everyday life. As a consequence, a need to ensure secure and reliable operation of these devices has also risen. Device attestation is a promising solution to the operational demands of embedded devices, especially those widely used in Internet of Things and Cyber-Physical System.**

In this paper, we summarize the basics of device attestation. We then present a summary of attestation approaches by classifying them based on their functionality and reliability guarantees they provide to networked devices. Lastly, we discuss the limitations and potential issues current mechanisms exhibit and propose new research directions.

I. Introduction

Under the umbrella terms of Internet of Things (IoT) and Cyber-Physical Systems (CPS), millions of low power devices have become entrenched in our lives. Reports state that currently 15 billion IoT devices are currently deployed [1], and deployment is expected to reach 50 billion by the year 2020 [2]. The number of CPS has also risen, with the advent of the smart grid [3], [4] and autonomous vehicles [5].

This massive deployment of devices has led to significant security concerns. Various attacks have shown weaknesses in IoT and CPS infrastructure, with a swarm of light bulbs potentially leaving a city in darkness [6], rogue devices attacking infrastructure [7], to attacks in critical infrastructure [8], [9].

Device attestation has risen as a promising solution to the security demands of embedded devices. In this paper, we discuss the requirements of an attestation scheme, as well as classify attestation schemes by their functionality and coverage. We then summarize representative device attestation approaches in the different categories. We present a discussion of the approaches as well as possible pitfalls and limitations that we have encountered in our survey of the literature.

The remainder of this paper is structured as follows: we present background concepts in device attestation in Section II. We then introduce different attestation approaches in Section III. A comparative discussion and future directions of research are discussed in Section IV. Lastly, we draw concluding remarks in Section V.

II. Attestation Background

A. Common Terminology

Throughout this paper, we employ the symbols and terminology shown in Table I.

TABLE I
Common symbols and terminology used in this paper.

Symbol	Meaning
$\mathcal{V}$	The *verifier* in an attestation mechanism
$\mathcal{P}$	The *prover* in an attestation mechanism
$\mathcal{M}$	A *measure* computed by a prover
$\mathcal{S}$	A particular *state* in the device being attested

B. Problem Definition

In its most basic form, device attestation is defined as making a claim about the properties of a target [10]. Two mutually exclusive parties are involved in an attestation scheme: a *verifier* $\mathcal{V}$, and a *prover* $\mathcal{P}$. Attestation is performed using a challenge-response mechanism upon $\mathcal{V}$'s requests. During the servicing of an attestation request $\mathcal{P}$ does a *measure* $\mathcal{M}$ of the device. $\mathcal{V}$ receives $\mathcal{M}$ and then determines whether or not $\mathcal{M}$ represents a valid device state.

Formally, let $\mathbb{A}$ represent the set of possible responses from $\mathcal{V}$. Since $\mathcal{V}$ replies with a yes or no answer

$$\mathbb{A} = \{0, 1\}, \tag{1}$$

where 1 represents the device attested successfully, and 0 represents an attestation failure. Let $\mathbb{M}$ be the set of measures returned by $\mathcal{P}$. The operation of $\mathcal{V}$ is then given by the mapping of $\mathbb{M}$ into $\mathbb{A}$, that is

$$V_{k,n} : \mathbb{M} \to \mathbb{A}. \tag{2}$$

$\mathcal{P}$ generates $\mathcal{M}$ by performing a computation over the state $\mathcal{S}$ of the device. $\mathcal{S}$ can be any identifiable quality of the device, such as the code it runs, the contents of its memory, or the an intrinsic characteristic of the system on chip it utilizes. We allow $\mathbb{S}$ to represent the set of possible $\mathcal{S}$ for the device, which may include illegal ones. Then, $\mathcal{P}$ performs the operation

$$P_{k,n} : \mathbb{S} \to \mathbb{M}. \tag{3}$$

We also say that $\mathcal{M} \in \mathbb{M}$ is the result of a measure performed by $\mathcal{P}$. Then, the device is operational if and only if

$$\forall \mathcal{S} \in \mathbb{S}, \mathcal{M} = P_{k,n}(\mathcal{S}) \in \mathbb{M}, V_{k,n}(\mathcal{M}) = 1. \tag{4}$$

Conversely, a device is deemed compromised if

$$\exists \mathcal{S} \in \mathbb{S}, \mathcal{M} = P_{k,n}(\mathcal{S}) \in \mathbb{M}, V_{k,n}(\mathcal{M}) = 0. \tag{5}$$

C. Scheme Requirements

For the results of Equation (4) to be valid a few considerations about the attestation scheme need to be in place.

Trustworthiness of $\mathcal{P}$. Since the $\mathcal{P}$ is in charge of collecting measures on the device, it must be devised in such way that it is tamper resistant. That is, $\mathcal{P}$ must be provably trustworthy. Trustworthiness of $\mathcal{P}$ may be established by means of a root of trust [11], [12], [13]; isolation [14], [15] by adding it as an intrinsic part of the platform [16]; or by utilizing properties intrinsic to hardware, such as the effect of process variations [17], [18].

Trust in $\mathcal{P}$ relies not only on its proper operational state, but on its semantic approach at performing the event of measuring the device. $\mathcal{P}$ may be in a proper operational state, but its implementation does not gather sufficient information to construct $\mathcal{S}_i \in \mathbb{S}$, the current state of the device being attested. The trustworthiness of $\mathcal{P}$ has a direct effect in the trustworthiness of $\mathcal{M}$. However, a trustworthy $\mathcal{P}$ does not necessarily reflect a trustworthy $\mathcal{M}$. That is to say, the trustworthiness of $\mathcal{P}$ is a *necessary but not sufficient condition* to determine the trustworthiness of $\mathcal{M}$.

Liveness of $\mathcal{M}$. Any $\mathcal{M}$ computed by $\mathcal{P}$ must reflect the current $\mathcal{S}$ of the system. Any requests by $\mathcal{V}$ must be answered by $\mathcal{P}$ by performing a fresh collection of state $\mathcal{S}_0$ in order to compute $\mathcal{M}$ while still considering all previously measured $\mathcal{S}$ that have not been used in computation to determine a given $\mathcal{M}$. This ensures an attacker is unable to alter the behavior of the device (a state) in a way that $\mathcal{P}$ will not utilize it.

Unforgeability of $\mathcal{M}$. An attacker must not be able to reproduce any $\mathcal{M}$ computation done by $\mathcal{P}$; that is, forge $\mathcal{M}$. Two attributes are necessary for $\mathcal{M}$ to meet this condition: *non-replayability* and *non-reconstructibility*. We discuss the need for these approaches below.

Consider the case where a trustworthy $\mathcal{P}$ has collected states in $\mathbb{S}_\mathbb{C} = \{\mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2, \mathcal{S}s_3\}$. $\mathcal{P}$ has also computed measures in $\mathbb{M}_\mathbb{C} = \{\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3\}$, respectively. Allow for the condition expressed in Equation (4) to be met under these conditions. Without loss of generality, suppose that during the next attestation request $\mathcal{P}$ obtains state $\mathcal{S}_0$. If $\mathcal{P}$ reports $\mathcal{M}_0$ as the computed measure, then an attacker is able to present $\mathcal{M}_0$, or any other previously computed $\mathcal{M} \in \mathbb{M}$ as a response to an attestation request.

Consequently, given a particular $\mathcal{S}_i, \mathcal{S}_j, \mathcal{S}_l ... \in \mathbb{S}$ collected at different times, with $\mathcal{S}_i = \mathcal{S}_j = \mathcal{S}_l = ...$, Equation (3) must return $\mathcal{M}_i, \mathcal{M}_j, \mathcal{M}_l, ... \in \mathbb{M}$ so that $\mathcal{M}_i \neq \mathcal{M}_j \neq \mathcal{M}_l \neq ....$ This is to say, any $\mathcal{M}$ reported to $\mathcal{V}$ by $\mathcal{P}$ must be *non-replayable*. For this particular purpose, an agreed upon nonce n is used by $\mathcal{P}$ and $\mathcal{V}$ during each attestation request to generate $\mathcal{M}$ and determine the functional status of the device, respectively.

Furthermore, an attestation approach must guarantee that any $\mathcal{M}$ is the result of $\mathcal{P}$ collecting state data on the device. That is, under the assumption that an attacker has knowledge of the function and the agreed upon nonce n used by $\mathcal{P}$ to compute $\mathcal{M}$, $\mathcal{M}$ can not be computer by an entity other than $\mathcal{P}$. For this purpose, a secret shared by $\mathcal{P}$ and $\mathcal{V}$ is utilized. The secret can be a preshared key or negotiated session key k. In this instance, standard key management procedures must be followed.

D. Types of Attestation

Device attestation may take many forms. We first define two methods $\mathcal{P}$ may utilize to collect $\mathcal{S}$. We say an attestation approach is *static* if $\mathcal{P}$ halts normal device operation to collect state information about the device. Likewise, we say an attestation approach is *dynamic* if $\mathcal{P}$ collects state information about the device without interrupting normal operation.

Attestation may also be *local* or *remote*. In a *local attestation* setting, both $\mathcal{V}$ and $\mathcal{P}$ reside within the device. In the event of an attestation failure $\mathcal{V}$ must take action a reporting action to record the failure and possibly set the device into a known stable failsafe mode. Under a *remote attestation* setting, $\mathcal{V}$ is a far away trusted entity which sends attestation requests to $\mathcal{P}$. In this case, requests and responses must be authenticated and properly secured from tampering.

We further categorize attestation approaches based on the portion of the device which is measured by $\mathcal{P}$.

Software Attestation. In a software attestation model, $\mathcal{P}$ checks the functionality of the code in the device. Here, $\mathcal{S}$ may consist of code segments on the device, control-flow metadata, and the contents of RAM. $\mathcal{P}$ computes $\mathcal{M}$ based on this type of state data, for example by using a cryptographic hash function, then sends $\mathcal{M}$ to $\mathcal{V}$.

Swarm Attestation. A swarm attestation model concentrates on checking the trustworthiness and proper functionality of multiple interconnected devices. In a swarm, devices may not have the same hardware and software configuration. Devices that are part of a swarm are only able to communicate with their direct neighbors. The protocol used for swarm attestation must be designed so that a compromised device in the swarm does not tamper with any measure performed in the system. Each device in the swarm may contain its own $\mathcal{V}$ and $\mathcal{P}$. $\mathcal{P}$ collects $\mathcal{S}$, computes $\mathcal{M}$, which then sends to $\mathcal{V}$ in a neighboring node to complete the attestation request.

Hardware Attestation. Hardware attestation techniques center on validating the authenticity and trustworthiness of the underlying hardware. This may be printed circuit boards, system on chips, on-board sensors, or communication channels. Hardware attestation schemes require a robust design in $\mathcal{P}$, often utilizing intrinsic hardware properties such as physical unclonable functions [17], [18], to compute $\mathcal{M}$.

III. Notable Approaches

A. Attacker Capabilities

Before presenting a summary of previously proposed approaches, we summarize the objectives and capabilities of an attacker. Attestation assumes an attacker wishes to compromise a device to alter its functionality. For example, an attacker may wish to change the values reported by a current meter to underpay for energy utilization; or an attacker may wish to tamper with a network of connected traffic lights to cause

TABLE II
COMPARISON OF DIFFERENT APPROACHES. ATTESTATION MECHANISMS FLAGGED AS STATIC WILL HALT NORMAL DEVICE OPERATION DURING EXECUTION OF $\mathcal{P}$. ATTESTATION MECHANISMS FLAGGED AS DYNAMIC ALLOW DEVICE OPERATION TO REMAIN UNINTERRUPTED WHILE $\mathcal{P}$ PERFORMS OPERATIONS TO COMPUTE $\mathcal{M}$.

Approach	Attestation Parameters						
	Type	Local	Remote	Software	Swarm	Board	SoC
Samsung Group, KNOX [19]	Static	●	●	●			
Eldefrawy et al, SMART [14]	Static		●	●			
Abera et al, Control-Flow Attestation (C-FLAT) [15]	Static		●	●			
Zeitouni et al, ATRIUM [16]	Dynamic		●	●			
Asokan et al, Scalable Embedded Attestation (SEDA) [20]	Static		●	●	●		
Ibrahim et al, DARPA [21]	Dynamic		●	●	●		
Suh et al, PUF for Device Authentication and Secret Key Gen. [22]	Static	●	●				●
Wei et al, BoardPUF [23]	Static	●	●			●	

infrastructure damage. An attacker may choose to follow one or many of the following avenues:

Change the Software. The attacker may change the code being run by the device with malicious code. This can be performed by injecting new code into the device [24], [25], overwriting existing code through open programming interfaces [26] or a vulnerable firmware update process [27], or glitching the memory bus to change fetched data [16].

Alter Software Behavior. The attacker may alter the behavior of the software run by means such as code-reuse attacks [28], [29], [30], [31]. This only requires the corruption control-flow information through memory errors to execute arbitrary sequences of code. This attack methodology is meant to bypass attestation approaches that only check code memory.

Tamper With Communications. The attacker may also attempt to tamper with the communications between devices in a swarm, or between $\mathcal{P}$ and a remote $\mathcal{V}$. Tampering may consist of inserting messages, removing messages, or changing the contents of messages sent between devices or $\mathcal{P}$ and $\mathcal{V}$. The attacker's objective is to imitate or leak secrets from a swarm to conceal a compromised device.

Compromising the Hardware Root of Trust. Hardware attacks use semi-invasive and invasive techniques, such as probing the device's circuit board to expose backdoors. To facilitate the process, an attacker may be an insider in a fabrication house where a counterfeit printed circuit board is used to manufacture the device allowing for the extraction of cryptographic secrets. Attackers with physical access to the device may extract secrets held in non-volatile memories to facilitate the intrusion into larger systems or to steal the device's unique identity in order to launch different types of attacks against the system, e.g., relay and replay attacks [32].

B. Software Attestation

Samsung introduced KNOX in 2013 for some of their Android-based devices as an enterprise grade security solution [19]. KNOX builds upon an ARM TrustZone [33] environment while adding new features to an ARM-based SoC. KNOX-enabled devices provide a root of trust as part of the secure world software, as well as operating system safeguards and verification mechanisms to ensure a safe execution environment. Prior to launching security critical tasks, such as enterprise applications, the KNOX $\mathcal{P}$ reads software configuration as $\mathcal{S}$ and computes a signature as $\mathcal{M}$. $\mathcal{V}$ deems some variations from the original configuration as a compromise. Security sensitive data, held in *KNOX Containers*, is only accessible if the security of the device is not deemed as compromised. For example, in the event the boot chain of the device is found to have been tampered with, an e-fuse internal to the SoC is blown branding the unit as untrusted. From this point KNOX Containers can no longer be created and any secured data becomes inaccessible.

Eldefrawy et al propose a small root of trust for embedded devices in SMART [14], providing a static remote attestation solution. It incorporates $\mathcal{P}$ into a small on-chip ROM. $\mathcal{P}$ utilizes a range of the device's application code as $\mathcal{S}$ and computes a HMAC [34] over it as $\mathcal{M}$. The hash is sent to a remote $\mathcal{V}$ to affirm its correctness. A preshared attestation key is stored in the device and gated so that only ROM code is capable of accessing it. To ensure that no leakage occurs, a safe memory erasure is performed every time the device reboots and whenever the ROM code finishes executing. Further precautions are taken to avoid indirect leakage by controlling code execution within the ROM. ROM code is only allowed to be executed from a fixed entry point to a fixed return point as to ensure that no code-reuse attacks attempt to leak the secret key.

In SMART, when $\mathcal{P}$ receives an attestation request from the remote $\mathcal{V}$, $\mathcal{P}$ suspends the currently running task and hashes the requested portion of code memory. SMART was implemented and tested in an OpenMSP430 [35] core, with a reported 9.2% area overhead. HMAC computation was reported to range from $48\,\mathrm{ms}$ to $287\,\mathrm{ms}$ on a block size ranging from $32\,\mathrm{B}$ to $1\,\mathrm{kB}$ when run at a clock frequency of $8\,\mathrm{MHz}$.

Abera et al demonstrated in [15] that it is insufficient to use only device code as $\mathcal{S}$ in order to compute $\mathcal{M}$. Code-reuse attacks are able to bypass attestation mechanisms that use this type of system. As a result, they proposed Control-Flow Attestation (C-FLAT) [15]. C-FLAT is a remote attestation mechanism that statically aggregates the execution path of a running program, including branches and function returns. In this approach, $\mathcal{P}$ collects control-flow information as $\mathcal{S}$ and hashes it to compute $\mathcal{M}$. On an attestation request, $\mathcal{V}$ receives the hashed control-flow information and computes the

expected $\mathcal{M}$ on its end. The device is said to attest correctly if both $\mathcal{M}$ match.

C-FLAT was implemented and tested in a Raspberry Pi 3 single board computer. Code is instrumented so that control-flow instructions target a trampoline section which belongs to a *runtime tracer*. The trampolines allow software to transition to a BLAKE2 [36] Measurement Engine which resides in a TrustZone environment. The Measurement Engine is part of $\mathcal{P}$ and is responsible for hashing control-flow. When $\mathcal{V}$ sends an attestation request to the device $\mathcal{P}$ replies with the collected information. The authors report that as the number of control-flow events increase, overhead increases linearly. When testing against dedicated functions in Open Syringe Pump a relatively large overhead 72% to 80% was reported. The authors further caution that a limiting factor in their target application is that of external sensors and actuators. Thus, the perceived overhead may be reduced to a mere 0.03% to 0.13%.

Zeitouni et al showed how a series of Time of Check Time of Use (TOCTOU) attacks in previous static attestation schemes that can compromise the way $\mathcal{S}$ is collected, and consequently manipulate the results of $\mathcal{P}$ computing $\mathcal{M}$. The authors then proposed ATRIUM as a solution in [16]. ATRIUM is a dynamic remote attestation mechanism that borrows concepts from C-FLAT [15] and SMART [14], in that it utilizes control-flow and code that is executed as $\mathcal{S}$. However, unlike C-FLAT and SMART, ATRIUM adds an instruction filter as an extension to the execution stage of a pipelined processor. All executed instructions are collected in an instruction buffer. The instruction filter checks if the currently executing instruction is a control-flow instruction and sends a signal to a loop encoder. Loop information as well as function call information is kept as states in the ATRIUM core. When a full basic block is executed, the collected instructions are hashed using a hardware BLAKE2b implementation and the results are stored in a dedicated memory. If the instruction buffer fills while instruction hashing is taking place, a signal is sent to the CPU to halt execution until hashing has completed. The ATRIUM core is what allows $\mathcal{P}$ to dynamically collect $\mathcal{S}$ to compute $\mathcal{M}$. When an attestation request arrives the stored hashes are sent as a response to the remote $\mathcal{V}$ for checking. Attestation concludes in a fashion similar to C-FLAT.

A RISC-V PULPino core [37] was extended to include ATRIUM. The modified core was synthesized and tested targeting a Virtex-7 XC7Z020 FPGA with minimal hardware overhead. Under the configured conditions, the authors report a total resource utilization of 15% of the total slice registers, 20% of slice LUTs, and $18\,\mathrm{kbit}$ of BRAM. Performance overhead ranged from 1.7% to 22.69%, depending on the amount and frequency of control-flow instructions in the tested algorithm.

C. Swarm Attestation

Asokan et al propose SEDA to attest a swarm of devices in [20]. The SEDA protocol has two phases, an off-line device initialization and registration phase, and an online device attestation and swarm attestation phase. During the off-line phase each device in the swarm is initialized by a trusted operator and given a signing key pair alongside an identity certificate. The device is then registered into the swarm, executing the `join` action of the SEDA protocol. This produces a handshake between the device and its neighbors, where it learns and verifies the public parameters of the neighbors. Furthermore, an attestation key is established during the join operation between the device and its neighbors in the swarm. During the online phase, devices can request attestation from its neighbors using an `attdev` request. The key generated during the off-line phase is used to attest the neighboring nodes. Swarm attestation starts with $\mathcal{V}$ sending an attestation request `attest` to a random device in the swarm. This device becomes the *initiator* of the swarm attestation. By recursively sending `attdev` messages to neighboring devices, a spanning tree is built, with devices that have not received the message being added. Eventually, $\mathcal{V}$ receives a reply containing the results of the swarm attestation.

SEDA was tested with a SMART-based [14] and TrustLite-based [13] implementations. The protocol is disjoint from the details regarding $\mathcal{P}$ and the state information it uses to compute $\mathcal{M}$. The protocol was evaluated using *computational cost*, *communication cost*, *memory cost*, *runtime cost*, and *energy cost* as metrics under a simulation using up to 10^6 devices using different spanning tree configurations. Performance was shown to fluctuate depending on the number of devices in the network and the number of child nodes in the spanning tree. Energy consumption was shown to increase linearly with respect to the number of neighboring devices, and to be evenly distributed by all members of the swarm. Only the *initiator* exhibits higher energy consumption due to the extra number of computations it must perform, however the authors state that this energy cost can be amortized throughout the swarm by choosing a different *initiator* on every swarm attestation request.

DARPA [21] is a lightweight collective attestation protocol aimed at defending unattended networks of embedded systems from physical attacks. The types of physical attacks considered are focused upon those that require disabling or disconnecting the device from the network. In considering such an adversary, DARPA leverages the non-negligible time required during disconnection to identify a devices absence using a *heartbeat*. The *heartbeat* identifies a device in the network uniquely, and is intermittently probed at regular intervals by its immediate neighbors to detect liveness. If at any time a device heartbeat fails to be detected, then the offending device can be considered compromised and appropriately quarantined.

D. Hardware Attestation

Suh et al propose utilizing a physical unclonable function (PUF) to authenticate integrated circuits and generate cryptographic keys in [22]. A PUF is a hardware function which utilizes process variations[1] as a source of entropy

[1]Process variations are uncontrollable and unavoidable variations in the manufacturing process of integrated circuits which add irregularities to the dimensions of transistors.

to generate random numbers based on a challenge-response setup [17], [18]. Because the operation of a PUF is subject to environmental factors, the authors add an error correction mechanism to ensure reliability on response (key) generation. Chip attestation works as follows: $\mathcal{V}$ applies a challenge (attestation request) to the chip. $\mathcal{P}$ utilizes the PUF circuit as $\mathcal{S}$ to compute $\mathcal{M}$. $\mathcal{V}$ uses a previously populated challenge-response database to finish the attestation request. Authors discuss that the proposed method has a 2.1×10^{-21} at misidentifying identifying integrated circuits, and a probability of less than 5×10^{-11} at failing to identify the integrated circuit. Although the authors claim that the method can be used in very low power environments, such as RFID, no metrics are provided regarding energy consumption. Also, no formal protocol description is provided.

Wei et al propose BoardPUF in [23]. BoardPUF provides an attestation mechanism for circuit boards using capacitance variations during the manufacturing process. The approach has two components, a source of variations fabricated on the board, and a chip to perform measurement and authentication. The variation units are designed as capacitor structures in the internal layers of the circuit board. Protecting them from noise sources are uninterrupted ground planes in the outer layers of the circuit board. Burr edges resulting from the chemical etching process of the boards, as well as the misalignment of layers while laminating, and changes in the thickness of boards are used as variations that alter the capacitance of the structures on the board. The chip component of the mechanism serves as $\mathcal{P}$, capturing $\mathcal{S}$ as a series of frequencies generated by an oscillator circuit using the on-board capacitances. $\mathcal{M}$ is computed by utilizing the captured frequencies as clock sources for sequential logic. The final logic state is error corrected and used by $\mathcal{V}$ to determine the authenticity of the board.

The proposed mechanism was manufactured and tested under different environmental conditions. Authors report a variation between $130\,\mathrm{kHz}$ to $158\,\mathrm{kHz}$ in the measured frequencies. Results claim that boards are incorrectly rejected with a ratio of 5.89×10^{-6}, and incorrectly accepted wit a ratio of 3.01×10^{-11}. However, no discussions on board area overhead and power are given.

IV. Discussion and Future Directions

Throughout the literature, we have seen a race with constantly increasing stakes between attacks and defenses. As attackers use more sophisticated methods to tamper with devices, attestation defenses have become more complex. We argue that a successful device attestation scheme must not only meet the requirements stated in Section II, they must also meet a deployability criteria. Attestation defenses should be readily implementable by vendors wishing to add security to their devices. Furthermore, attestation defenses should concern themselves with the power, memory, and computational constrains of embedded devices.

For example, SMART [14] and ATRIUM [16] require the processor to be modified in invasive ways. This is impossible for device manufacturers under the current Intellectual Property licensing model used in industry, or limited to a select few such as with the case of Samsung KNOX [19]. This limits the deployment of any proposed security mechanism. Although the implementation in C-FLAT [15] requires no hardware modifications it was shown to be vulnerable attack [16], which makes it undesirable as a security solution.

Of the discussed approaches, we note that *static* attestation approaches (those where $\mathcal{P}$ interrupts device operation) may be detrimental to timing sensitive systems, where a small delay can result in the stability of the system being lost. Furthermore, Zeitouni et al demonstrated in [16] that the switch to a $\mathcal{P}$ results in a timing channel that can be exploited by an attacker to give $\mathcal{P}$ a false $\mathcal{S}$. Our survey points to *dynamic* approaches result in more accurate collection of $\mathcal{S}$ by $\mathcal{P}$ to compute $\mathcal{M}$. Dynamic data collection also results in reduced performance overhead during execution. As such, we argue that future attestation mechanisms should be dynamic in nature.

We also note the difference in the way performance overhead is reported. SMART [14] reports the latency in execution caused by the HMAC function, whereas C-FLAT [15] reports overhead in partially instrumented code in a particular application (Open Syringe Pump [38]). We find that computational overhead is a function of any instrumentation required by $\mathcal{P}$, any delays introduced by $\mathcal{P}$ including its cryptographic requirements, and any delays caused by hardware modifications, such as in the case with ATRIUM [16]. Furthermore, of the discussed approaches, only SEDA [20] reports energy measurements. This is an important aspect for power-constrained devices, as it directly affects the field life of the units. Much like computing performance can be tested with a set of industry standard benchmarks such as SPEC CPUINT2006 [39], a common embedded test suite should be used to test performance and energy requirements of proposed device attestation solutions. The test suite should consist of representative applications in IoT and CPS devices. For example, Open Syringe Pump [38] and OpenPLC [40] can serve as baselines for testing new attestation methods for CPS, while demonstration programs targeting sensor boards can be used as a baseline for IoT devices. We urge researchers in this area to take this as a consideration when presenting new attestation solutions.

Hardware attestation approaches are limited by area overhead and energy needs of the deviced solution. PUF based approaches like the ones presented suffer from accuracy issues and thus need extra circuitry to correct errors, adding to area and power overhead. Also, PUF-based security mechanism present scalability issues. As the number of devices increases, the number of challenge-response pairs that must be evaluated during factory time increase as well, elevating manufacturing the cost. Further research in this area should address these limitations. Furthermore, we argue that these approaches should report area and energy metrics as to facilitate comparison and view applicability.

V. Conclusion

In this paper, we provided a formal framework for device attestation as well as discussed the requirements of an attestation mechanism from the view of the prover. We presented a classification of attestation mechanisms and provided an overview of previous work in the area. We also discussed limitations of previous work and proposed potential directions for future research.

Acknowledgements

This work is partially supported by National Science Foundation (DGE-1802701), Semiconductor Research Corporation (2016-TS-2724), Florida Center for Cybersecurity (FC2), and Cisco. Mr. Orlando Arias is also supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. 1144246.

References

[1] D. Evans, "The internet of things - how the next evolution of the internet is chaging everything," *White Paper. Cisco Internet Business Solutions Group (IBSG)*, 2011.

[2] A. Nordrum, "Popular internet of things forecast of 50 billion devices by 2020 is outdated," *IEEE Spectrum*, 2016.

[3] S. M. Amin and B. F. Wollenberg, "Toward a smart grid: power delivery for the 21st century," *IEEE power and energy magazine*, vol. 3, no. 5, pp. 34–41, 2005.

[4] H. Farhangi, "The path of the smart grid," *IEEE power and energy magazine*, vol. 8, no. 1, 2010.

[5] L. D. Burns, "Sustainable mobility: a vision of our transport future," *Nature*, vol. 497, no. 7448, pp. 181–182, 2013.

[6] E. Ronen, A. Shamir, A.-O. Weingarten, and C. OFlynn, "Iot goes nuclear: Creating a zigbee chain reaction," in *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 2017, pp. 195–212.

[7] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the mirai botnet," in *26th USENIX Security Symposium (USENIX Security 17)*, 2017, pp. 1093–1110.

[8] M. Abrams and J. Weiss, "Malicious control system cyber security attack case study–maroochy water services, australia," *McLean, VA: The MITRE Corporation*, 2008.

[9] R. Langner, "Stuxnet: Dissecting a cyberwarfare weapon," *IEEE Security & Privacy*, vol. 9, no. 3, pp. 49–51, 2011.

[10] G. Coker, J. Guttman, P. Loscocco, A. Herzog, J. Millen, B. OHanlon, J. Ramsdell, A. Segall, J. Sheehy, and B. Sniffen, "Principles of remote attestation," *International Journal of Information Security*, vol. 10, no. 2, pp. 63–81, 2011.

[11] W. A. Arbaugh, D. J. Farber, and J. M. Smith, "A secure and reliable bootstrap architecture," in *Security and Privacy, 1997. Proceedings., 1997 IEEE Symposium on*. IEEE, 1997, pp. 65–71.

[12] F. Brasser, B. El Mahjoub, A.-R. Sadeghi, C. Wachsmann, and P. Koeberl, "Tytan: tiny trust anchor for tiny devices," in *Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE*. IEEE, 2015, pp. 1–6.

[13] P. Koeberl, S. Schulz, A.-R. Sadeghi, and V. Varadharajan, "Trustlite: A security architecture for tiny embedded devices," in *Proceedings of the Ninth European Conference on Computer Systems*. ACM, 2014, p. 10.

[14] K. Eldefrawy, G. Tsudik, A. Francillon, and D. Perito, "Smart: Secure and minimal architecture for (establishing dynamic) root of trust." in *NDSS*, vol. 12, 2012, pp. 1–15.

[15] T. Abera, N. Asokan, L. Davi, J.-E. Ekberg, T. Nyman, A. Paverd, A.-R. Sadeghi, and G. Tsudik, "C-flat: control-flow attestation for embedded systems software," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 743–754.

[16] S. Zeitouni, G. Dessouky, O. Arias, D. Sullivan, A. Ibrahim, Y. Jin, and A.-R. Sadeghi, "Atrium: Runtime attestation resilient under memory attacks," in *International Conference On Computer Aided Design (ICCAD)*, 2017.

[17] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas, "Silicon physical random functions," in *Proceedings of the 9th ACM conference on Computer and communications security*. ACM, 2002, pp. 148–160.

[18] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld, "Physical one-way functions," *Science*, vol. 297, no. 5589, pp. 2026–2030, 2002.

[19] S. Knox, "White paper: An overview of samsung knox," *xhttps://www.samsung. com/global/business/businessimages/resource/white-paper/2013/06/Samsung_KNOX_whitepaper _June-0. pdf*, 2013.

[20] 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*. ACM, 2015, pp. 964–975.

[21] A. Ibrahim, A.-R. Sadeghi, G. Tsudik, and S. Zeitouni, "Darpa: Device attestation resilient to physical attacks," in *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. ACM, 2016, pp. 171–182.

[22] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proceedings of the 44th annual design automation conference*. ACM, 2007, pp. 9–14.

[23] L. Wei, C. Song, Y. Liu, J. Zhang, F. Yuan, and Q. Xu, "Boardpuf: Physical unclonable functions for printed circuit board authentication," in *Computer-Aided Design (ICCAD), 2015 IEEE/ACM International Conference on*. IEEE, 2015, pp. 152–158.

[24] A. Francillon and C. Castelluccia, "Code injection attacks on harvard-architecture devices," in *Proceedings of the 15th ACM conference on Computer and communications security*. ACM, 2008, pp. 15–26.

[25] A. One, "Smashing the stack for fun and profit (1996)," *Phrack Magazine*, 2007.

[26] L. Garcia, F. Brasser, M. H. Cintuglu, A.-R. Sadeghi, O. Mohammed, and S. A. Zonouz, "Hey, my malware knows physics attacking plcs with physical model aware rootkit," in *Proceedings of the Network & Distributed System Security Symposium, San Diego, CA, USA*, 2017, pp. 26–28.

[27] A. Cui, M. Costello, and S. J. Stolfo, "When firmware modifications attack: A case study of embedded exploitation." in *NDSS*, 2013.

[28] F. Schuster, T. Tendyck, C. Liebchen, L. Davi, A.-R. Sadeghi, and T. Holz, "Counterfeit object-oriented programming: On the difficulty of preventing code reuse attacks in c++ applications," in *Security and Privacy (SP), 2015 IEEE Symposium on*. IEEE, 2015, pp. 745–762.

[29] T. Bletsch, X. Jiang, V. W. Freeh, and Z. Liang, "Jump-oriented programming: a new class of code-reuse attack," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*. ACM, 2011, pp. 30–40.

[30] M. Prandini and M. Ramilli, "Return-oriented programming," *IEEE Security & Privacy*, vol. 10, no. 6, pp. 84–87, 2012.

[31] S. Checkoway, L. Davi, A. Dmitrienko, A.-R. Sadeghi, H. Shacham, and M. Winandy, "Return-oriented programming without returns," in *Proceedings of the 17th ACM conference on Computer and communications security*. ACM, 2010, pp. 559–572.

[32] B. Krebs, "Cyber incident blamed for nuclear power plant shutdown," *Washington Post, June*, vol. 5, p. 2008, 2008.

[33] A. ARM, "Security technology building a secure system using trustzone technology (white paper)," *ARM Limited*, 2009.

[34] M. Bellare, R. Canetti, and H. Krawczyk, "Message authentication using hash functions: The hmac construction," *RSA Laboratories CryptoBytes*, vol. 2, no. 1, pp. 12–15, 1996.

[35] OpenCores, "Openmsp430 project," http://opencores.org.

[36] J.-P. Aumasson, S. Neves, Z. Wilcox-OHearn, and C. Winnerlein, "Blake2: simpler, smaller, fast as md5," in *International Conference on Applied Cryptography and Network Security*. Springer, 2013, pp. 119–135.

[37] E. Zurich and U. of Bologna, "PULP Platform," http://www.pulp-platform.org/.

[38] B. Wijnen, E. J. Hunt, G. C. Anzalone, and J. M. Pearce, "Open-source syringe pump library," *PloS one*, vol. 9, no. 9, p. e107216, 2014.

[39] J. L. Henning, "Spec cpu2006 benchmark descriptions," *ACM SIGARCH Computer Architecture News*, vol. 34, no. 4, pp. 1–17, 2006.

[40] T. R. Alves, M. Buratto, F. M. de Souza, and T. V. Rodrigues, "Openplc: An open source alternative to automation," in *Global Humanitarian Technology Conference (GHTC), 2014 IEEE*. IEEE, 2014, pp. 585–589.