DARPA: Device Attestation Resilient to Physical Attacks

Ahmad Ibrahim*
TU Darmstadt
Ahmad.Ibrahim@cased.de

Ahmad-Reza Sadeghi
TU Darmstadt
Ahmad.Sadeghi@cased.de

Gene Tsudik
UC Irvine
Gene.Tsudik@uci.edu

Shaza Zeitouni
TU Darmstadt
Shaza.Zeitouni@cased.de

ABSTRACT

As embedded devices (under the guise of "smart-whatever") rapidly proliferate into many domains, they become attractive targets for malware. Protecting them from software and physical attacks becomes both important and challenging. Remote attestation is a basic tool for mitigating such attacks. It allows a trusted party (verifier) to remotely assess software integrity of a remote, untrusted, and possibly compromised, embedded device (prover).

Prior remote attestation methods focus on software (malware) attacks in a one-verifier/one-prover setting. Physical attacks on provers are generally ruled out as being either unrealistic or impossible to mitigate. In this paper, we argue that physical attacks must be considered, particularly, in the context of many provers, e.g., a network, of devices. Assuming that physical attacks require capture and subsequent temporary disablement of the victim device(s), we propose DARPA, a light-weight protocol that takes advantage of absence detection to identify suspected devices. DARPA is resilient against a very strong adversary and imposes minimal additional hardware requirements. We justify and identify DARPA's design goals and evaluate its security and costs.

1. INTRODUCTION

In addition to traditional computing devices that come in various shapes and sizes (e.g., laptops, desktops, smartphones and tablets), so-called *smart* embedded computing devices are increasingly percolating into many spheres of everyday life. Such devices include household appliances, industrial machinery, automotive and avionic components, as well as many kinds of personal gadgets. In general, these smart devices differ from traditional computers in that their mission is not general-purpose computing. Hence, their capabilities and purposes are limited to supporting the goals of the device as a whole, e.g., sensing or actuation. These devices represent attractive attack targets and their proliferation poses a formidable security challenge, for three reasons: **First**, they communicate via wired or wireless interfaces, which means that they can be accessed remotely. **Second**, in order to keep costs low and/or to conserve power, they lack necessary resources to defend themselves against attacks in the manner of general-purpose computers, e.g., via sophisticated OS security features or anti-malware tools. **Third**, because they are used to control (or interface with) physical equipment, a successful attack can cause actual real-world damage.

To address the aforementioned challenge, a lot of effort has been invested into both prevention and mitigation of attacks, especially, remote malware infestations, exemplified by the well-known Stuxnet episode [55]. The most popular approach is to verify the current state of a remote embedded device in order to establish that it behaves as it should, i.e., operates correctly. This translates into verifying device's software integrity, which is typically achieved using *remote attestation*, a distinct security service that provides a proof to a trusted entity (verifier) of software integrity of an untrusted – and possibly compromised – remote embedded device (prover).

Problem Description. Prior remote attestation results consider only remote software attacks, wherein the adversary's power is limited to manipulating prover's software from afar. This is in line with the need to protect the prover against remote malware infestations, under the assumption that the adversary never has physical access to the prover. Furthermore, prior schemes focus on the setting with a single prover. This was a natural first step.

In the single-prover setting, it is reasonable to assume that physical attacks on the prover are either impossible or very unlikely, especially, if the device is physically protected or unreachable, e.g., located on secure premises. Also, in case of multiple stand-alone (not inter-connected) provers, each can be attested individually and no-physical-attack assumption might still hold. However, there are current and emerging scenarios that involve multiple inter-connected devices, e.g., automotive, building, office, and factory automation environments. They differ from the single-prover setting in two important ways:

1. Faced with potentially numerous provers, attesting them individually can become expensive and unscalable, regardless of whether attestation is performed locally or remotely. This motivates a need to perform collective or aggregated attestation of the entire set of provers.
2. Provers might be heterogeneous and distributed over a

*Authors' names are listed in alphabetical order.

WiSec'16 , July 18-22, 2016, Darmstadt, Germany

DOI: http://dx.doi.org/10.1145/2939918.2939938

large physical area, e.g., a factory floor. Consequently, the prior assumption about no physical attacks is no longer valid: some provers could remain physically unreachable while others might be within adversary's grasp. For example, in an office building, devices in public spaces might be easily accessible, in contrast to those inside individual private offices.[1]

The second issue has not been considered at all in the context of remote attestation. In contrast, the first issue has been noticed and progress has been made, to some extent, by Asokan et al. [4] in the design of SEDA– a technique for efficient and scalable attestation of a network of provers. This is attained by distributing the attestation burden across the entire network. However, SEDA focuses on so-called "device swarms". It does not attest topology and merely reports the number of devices that pass attestation. Also, SEDA's threat model only considers remote software-based attacks.

Goals and Contributions. If some devices can be captured and physically attacked, remote attestation techniques must define a stronger adversary model that allows physical attacks, and devise a means of mitigating such attacks. This paper represents the first step towards this goal and makes two main contributions. **First,** it defines the adversary model for collective attestation that allows physical attacks. **Second,** it constructs a collective attestation technique DARPA, that is secure in the presence of the strongest version of the adversary. Specifically, DARPA can detect both software-based and physical attacks. However, our main goal is to detect whether an attack has occurred, rather than identifying malicious devices. DARPA can be extended with a majority voting protocol in order to detect such devices. Finally, due to the dynamic nature of the targeted networks, we acknowledge the possibility of *false positives*, due to device failures, unreachablity, network partitioning or message loss. However, the main focus of this paper is security, i.e., to avoid *false negatives*.

The main premise for our work is that, in order to physically attack a device, the adversary must make it inaccessible for a certain non-negligible amount of time, e.g., to take it apart for the purpose of extracting secrets [5]. (Inaccessibility implies either physical removal of the device or switching it off *in situ*.) Therefore, detecting device's absence can be a sign of capture. In designing DARPA, we take advantage of prior work in Wireless Sensor Networks (WSN) literature [13, 14, 15, 23, 24, 25, 26, 53].

2. BACKGROUND AND OBJECTIVES

Physical Attacks. We consider unattended networks of embedded devices that are infrequently inspected physically. Some devices might be physically protected, while others can be subject to physical attacks, perhaps due to their type and/or location. This paper focuses on physical attacks that require disablement or disconnection of the device(s) from the network for a non-negligible amount of time. We refer to this event as device *capture*.

Physical attacks can be classified into invasive, non-invasive, and semi-invasive.

Invasive attacks [48] aim to extract information from a device by trying to directly access internal components using sophisticated and expensive specialized equipment, e.g., Focused Ion Beam (FIB) and micro-probing stations. Such attacks start with full or partial de-capsulation (i.e., removal of packaging using mechanical or chemical means, or mixture of both), followed by de-processing.

Non-invasive attacks [58], such as *side-channel* (e.g., power, time or electromagnetic radiation) attacks aim to stealthily extract cryptographic keys during normal device operations. They mainly use low-cost electrical engineering tools. Several countermeasures have been proposed to mitigate side-channel attacks at physical, technological and/or algorithmic levels [38, 58]. Examples include: shielding circuits, using analog isolation to hide the correlation between secret key and power consumption, or making execution path independent of cryptographic keys.

Semi-invasive attacks [49] are less expensive and less complicated than invasive attacks, since they involve cheaper equipment (e.g., laser microscopes) and only require decapsulation. Examples include: ultra-violet attacks, laser scanning, thermal imaging and optical fault injections.

Both invasive and semi-invasive attacks require detaching the device for some time (e.g., anywhere from several hours to weeks [47, 48]) using specialized tools. Even micro-probing, which can be performed while the device is operating, requires switching-off the device for at least the duration of de-capsulation and de-processing operations (which is non-negligible) before getting access to internal wiring and signals buried under passivation layer, using a micro-probing station. On the other hand, although side-channel attacks may require as little as few minutes and up to one day (if mounted *in-field* by an insider) they are very likely to be detected since they require bulky physical tools (e.g., an oscilloscope or EM-receiver/demodulator) to be attached to the victim device, or installed in close proximity.

Prevention *vs.* Detection: Attack prevention is clearly more appealing than attack detection. However, we recognize that the former is very difficult to achieve in the envisaged general setting of large distributed networks of heterogeneous devices. We believe that the only means of attack prevention are physical, e.g., placing all devices inside a secure perimeter, protecting them with other types of devices (e.g., alarm sensors that are themselves subject to attack), putting them inside protective containers or encasing them in various hard-to-penetrate materials, such as cement or metal. Besides being expensive and hampering mobility, all these approaches are simply not general. An alternative is to assume that each device is equipped with tamper-resistant components, which can be problematic since tamper-resistant hardware involves extra costs: monetary, power consumption, weight and volume. This motivates our focus on detection, rather than prevention, of physical attacks.

Objectives. We believe that, ideally, a secure *collective* attestation protocol should:

1. Verify collective integrity of the network.
2. Detect (and, ideally, identify) compromised devices.
3. Offer better efficiency than attesting devices individually.
4. Detect absent and thus possibly captured devices.
5. Attest network topology.

Properties (1–3) pertain to scalable attestation of groups (or swarms) of devices. The recently proposed SEDA proto-

[1]Similar examples include factory automation or perimeter monitoring scenarios where devices on the edges of the network are naturally more vulnerable than those in the middle.

col [4] aims at addressing (1–3), in a weaker (remote software-only) adversary model. SEDA does not address either (4) or (5). Property (4) is needed to mitigate a stronger adversary that can physically capture and fully compromise devices – it forms the crux of this paper's contribution. Property (5) is optional; it is important if the topology is a part of overall network's integrity. Our goal is a protocol that satisfies all these properties.

3. DARPA: PRELIMINARIES

3.1 System Model and General Idea

We consider a network of many, possibly heterogeneous, devices with either static or dynamic topology, e.g., automotive networks, industrial control systems, prospecting robots or IoT devices in smart home/office environments. There are s devices in total and each device D_i is uniquely identifiable, i.e., has a distinct id_i. The set of the id_i-s is denoted by $\mathcal{ID}$. We denote the verifier by $\mathcal{A}dm$.

The general idea of DARPA is that the network is mostly left unattended, i.e., in the time between successive attestations. During that time, each D_i periodically monitors all other $s-1$ devices. This is achieved using the heartbeat protocol, executed at regular time intervals t_{hb}. Each D_i broadcasts a secure heartbeat to its immediate neighbors, thereby proving its presence. Each neighbor, in turn, forwards the received heartbeat to its own neighbors, and so on. Each D_i collects all heartbeats, verifies and logs them. At the time of next attestation, $\mathcal{A}dm$ performs a collective attestation protocol (e.g., SEDA) with the entire network and gathers, from each device, a set of logs – one for each heartbeat protocol instance executed since the previous attestation. This is done via a separate collect protocol.

3.2 Adversary Model

Similar to all other attestation techniques, we assume that the verifier is trusted. In the context of the prover, we consider three types of adversaries:

1. **Software** – $\mathcal{A}dv_{v,0}$ can remotely compromise (i.e., via software attacks) up to $v \leq s$ devices. This is the usual adversary model in prior attestation protocols.
2. **Physical** – $\mathcal{A}dv_{0,w}$ can capture (i.e., physically attack) up to $w < s$ devices.
3. **Hybrid** – $\mathcal{A}dv_{v,w}$ can compromise up to v, and capture up to, w devices.

We use v and w to refer to maximum numbers of devices that $\mathcal{A}dv$ can subvert – by software and physical means, respectively – within a certain time interval T_{att}. This interval corresponds to the longest period of time between successive instances of network attestation performed by $\mathcal{A}dm$, i.e., longest inter-attestation time gap. We assume $\mathcal{A}dv$ requires a non-negligible amount of time T_{cap} to physically attack a device. T_{cap} is expected to be appreciably longer than any communication delay within the network. We also limit $\mathcal{A}dv$'s scope of attacks as follows:

1. **No omnipotent adversary:** Given s devices total, we claim that $\mathcal{A}dv_{0,s}$ is impossible to mitigate. However, we consider the next most powerful adversary – $\mathcal{A}dv_{s,s-1}$, which can compromise all devices and capture all-but-one, without making any assumptions about the single uncaptured device.
2. **No noninvasive physical attacks:** $\mathcal{A}dv$ might exploit hardware side-channels (time, power, optical) to extract devices secrets. Such attacks – which do not require switching off a device for a minimum amount of time – represent an orthogonal problem that is beyond the scope of this paper.
3. **No denial-of-service (DoS):** We consider DoS attacks to be out of scope, assuming that the envisaged $\mathcal{A}dv$ wants to remain stealthy and undetected while attempting to compromise and/or physically attack devices.

4. MITIGATING PHYSICAL ADVERSARY

For now, we assume that $\mathcal{A}dv$ can capture all devices – except one D_z – in a single T_{att} interval. However, we also assume that $\mathcal{A}dv$ can not compromise any device that it does not first capture. Hence, "0" in the first subscript of $\mathcal{A}dv_{0,s-1}$.

The intuition behind our approach is as follows: If the lower bound on the time needed to capture a device – T_{cap}– is known, the network can regularly run a collective heartbeat (absence detection) protocol, at intervals t_{hb}, shorter than T_{cap}. Therefore, if the following two conditions hold:

1. Each uncaptured device's heartbeat is unforgeable and uniquely tied to the particular instance (time) of the heartbeat protocol,
   and:
2. Each device periodically (every t_{hb} interval) emits its own time-based heartbeat and collects/records heartbeats of all other present devices

Then, it is safe to make the following assertion:

> **Assertion 1:** [**Alert**] If $0 < k < s$ devices are captured within a given T_{att}, then, by the next attestation instance, the log of at least one uncaptured device (D_z) will lack at least one other device's heartbeat, for at least one heartbeat protocol instance.

Assuming that D_k is absent for longer than T_{cap}, D_z's log would be missing at least one of D_k's heartbeats corresponding to a particular instance of the heartbeat protocol. This is because $T_{cap} > t_{hb}$, where t_{hb} is the interval between successive runs of the heartbeat protocol. Of course, after capturing D_k, $\mathcal{A}dv$ might extract its secrets and later attempt to forge the missing heartbeat(s). However, by that time, it is too late since this would take at least $T_{cap} > t_{hb}$ time and D_z would record D_k's absence. Note that, in practice, the inverse of **Alert** does not hold: a discrepancy between devices' logs at attestation time does not imply that any devices were captured. Indeed, a log discrepancy might occur due to lost heartbeat messages, device failures or (even temporary) unreachability. Thus, we readily acknowledge that *false positives* are possible. However, our main goal is to avoid *false negatives*.

Furthermore, we assert that:

> **Assertion 2:** [**Normalcy:**] At any attestation instance, if logs of all s devices match and contain every device's heartbeat for each heartbeat protocol instance, then no devices were absent for longer than T_{cap} time.

Normalcy implies that no device was captured since none were absent for at least T_{cap}. Therefore, the heartbeat log of D_z contains timely proof of every other device's presence, for

each heartbeat protocol instance since the last attestation. And, because D_z is (at least) the only uncaptured device and its log agrees with all other devices' logs, no device was absent for $\geq T_{cap}$. (We emphasize that D_z's identity, id_z, is unknown to $\mathcal{A}dm$.) Consequently, no physical attacks took place.

4.1 DARPA:heartbeat Protocol

We first introduce some assumptions. The first two correspond to individual devices and the rest apply to the network:

1. ***Reliable Clocks:*** Each D_i has a reliable clock, loosely synchronized with $\mathcal{A}dm$'s clock. δ_t denotes the maximum clock skew between any two devices.
2. ***Secret Keys:*** Each D_i has a unique private signing key SK_i, which is assigned and installed (perhaps by $\mathcal{A}dm$) before deployment.
3. ***Public Keys:*** Each D_i knows the set of all id_i-s – $\mathcal{ID}$, and, for each D_j – the public key PK_j, as well as $\mathcal{A}dm$'s public key – $PK_{\mathcal{A}dm}$. For better efficiency, signatures can be replaced by message authentication codes (MACs). This variant is discussed in Section 6.
4. ***Connectivity:*** there always exists a path between any two devices, i.e., the network is always connected.
5. ***Mobility:*** devices might be mobile, i.e., network topology is subject to change. However, during DARPA's execution, topology is assumed to be static.

heartbeat is initiated in a distributed manner: D_i either receives a heartbeat message from one of its neighbors, or wakes up based on a timer, indicating that t_{hb} has passed since the last run of the protocol. In either case, D_i generates its own heartbeat, timestamped and signed with SK_i, and broadcasts it to neighbors. Whenever D_i receives a heartbeat from D_j, it verifies the timestamp and the signature. If both are valid, D_i logs the heartbeat as part of the current protocol and re-broadcasts it to the neighbors. Once D_i receives heartbeats from all peers, it terminates the protocol. Alternatively, D_i terminates the protocol based on a time-out – t_{acc} (see Table 1). Non-receipt of a valid heartbeat from some D_j within t_{acc} from the protocol start indicates that, from D_i's perspective, D_j is absent. The detailed description of the heartbeat protocol is shown in Figure 1. Table 1 summarizes our notation. In it, t_{tx} denotes the global upper bound for a network delay. However, since the time T_{cap} needed to physically attack a device is expected to be considerably greater than any other delay, a tight upper bound on network delay is not necessary. As mentioned earlier, the heartbeat protocol for D_i can start based on a timeout or a received heartbeat message. However, for the sake of clarity, we illustrate the protocol as starting only upon a timeout, assuming that heartbeat messages received prior to step 0 are queued in a buffer.

4.2 DARPA:collect Protocol

Periodically (at intervals upper-bounded by T_{att}), $\mathcal{A}dm$ initiates the collect protocol. We make no assumptions about $\mathcal{A}dm$'s location at that time: it could be local or remote. $\mathcal{A}dm$ generates a request: $Ch = N, t_{\mathcal{A}dm}, SIG_{\mathcal{A}dm}$ where: N is a random challenge, $t_{\mathcal{A}dm}$ is a timestamp, and $SIG_{\mathcal{A}dm}$ is $\mathcal{A}dm$'s signature over N and $t_{\mathcal{A}dm}$.

Upon receipt of Ch, D_i verifies the timestamp and the signature. If both are valid, D_i replies with the set of all logs collected since the last attestation instance – $LOG_i = \{log_{i,p}, \ ; 0 < p \leq \#hb\}$ – timestamped and signed with SK_i. Once $\mathcal{A}dm$ receives LOG_i, it verifies the timestamp and D_i's signature. If both are valid, it stores the corresponding log set. Having received LOG_i-s from all devices,

s	total number of devices
i, j, z, k	device indexes $\in [1, s]$
$v,\ w$	max #-s of devices $\mathcal{A}dv$ can compromise or capture
id_j	identifier of D_j
$\mathcal{ID}$	set of all device id-s: $\{id_1 ... id_s\}$
SK_i, PK_i	dev_i's public and private keys, respectively
k_{ij}	symmetric key shared between D_i and D_j
K_i	symmetric key shared between D_i and $\mathcal{A}dm$
T_{att}	maximum interval between two consecutive attestations
T_{cap}	minimum time needed by $\mathcal{A}dv$ to capture a device; $T_{cap} < T_{att}$
δ_t	maximum clock skew between any two devices
t_{tx}	maximum time to transmit a message between any two devices. (includes processing time at intermediate devices)
t_{acc}	acceptance interval; $t_{acc} = \delta_t + t_{tx}$
t_{hb}	interval between successive heartbeat instance; $2 \cdot t_{acc} < t_{hb} < T_{cap} - t_{acc} - \delta_t$
$\#hb$	number of heartbeat protocol runs between successive attestations; $1 \leq \#hb = \lfloor \frac{T_{att}}{t_{hb}} \rfloor$
p	heartbeat protocol index; $0 < p \leq \#hb$
$hb_{i,p}$	D_i heartbeat message for p-th heartbeat protocol instance; $hb_{i,p} = \{p, t_{i,p}, id_i\}$
$SIG_{i,p}$	signature, computed by D_i, using SK_i over $hb_{i,p}$
$time()$	returns current time
$t_{i,p}$	D_i clock when $hb_{i,p}$ is created
$log_{i,p}$	D_i's log of all valid received heartbeats for p-th protocol instance
$present[s]$	bitmap indicating which devices were ever absent, $present[i] = 0$ implies that D_i was absent (at least once)
OK	flag indicating if any device was ever absent, $OK = 0$ implies that (at least) one device was absent at least once

$accept(t_{start}, t_{j,p}) = 0\|1$	timestamp verification; returns "true" if $t_{j,p} = t_{start}$
sign(SK, MSG)	signature generation: using SK and MSG, yields a signature on MSG
verify(PK, MSG, SIG)	signature verification: using PK, checks SIG on MSG; yields $0\|1$
mac(k, MSG)	computes MAC of MSG using k
vermac(k, MSG, MAC)	MAC verification: using k, checks MAC on MSG; yields $0\|1$

Table 1: Notation

Figure 1: DARPA: heartbeat Protocol (as viewed by D_i)

$Timeout(timer_{hb})$

Start-timer($timer_{acc} = t_{acc}$)

$t_{i,p} = t_{start} = time()$, $hb_{i,p} = \{p, t_{i,p}, id_i\}$

$SIG_{i,p} = sign(SK_i, hb_{i,p})$

Set $log_{i,p} = hb_{i,p}$, Start-timer($timer_{hb} = t_{hb}$)

$D_i \xrightarrow{hb_{i,p},\ SIG_{i,p}}$ all neighbors

while ($\neg$ $Timeout(timer_{acc})$ $\wedge$ $sizeof(log_{i,p}) < s$) **do**

  $D_i \xleftarrow{hb_{j,p},\ SIG_{j,p}}$ neighbor (D_j)

  if $(accept(t_{start}, t_{j,p}) \wedge hb_{j,p} \notin log_{i,p})$ **then**

    if $Verify(PK_j, hb_{j,p}, SIG_{j,p})$ **then**

      $Append(\{hb_{j,p}, SIG_{j,p}\},\ log_{i,p})$

      $D_i \xrightarrow{hb_{j,p},\ SIG_{j,p}}$ all neighbors

    else

      Discard $hb_{j,p}$

    end

  else

    Discard $hb_{j,p}$

  end

end

$p = p + 1$

Figure 2: DARPA: collect Protocol (as viewed by $\mathcal{A}dm$)

for $0 < i \leq s$ **do**

  $t = time()$, $N \in \{0,1\}^{\ell_N}$,

  $SIG_{\mathcal{A}dm} = \mathsf{sign}(SK_{\mathcal{A}dm}, t|N)\}$

  $\mathcal{A}dm \xrightarrow{Ch=\{t,N,SIG_{\mathcal{A}dm}\}} D_i$

  $\mathcal{A}dm \xleftarrow{resp_i=\{LOG_i,SIG_i\}} D_i$

  if $\left(verify(PK_i, LOG_i|Ch, SIG_i)\right)$ **then**

    Store LOG_i

  else

    Discard $resp_i$

  end

end

Figure 3: DARPA: collect Protocol (as viewed by D_i)

$D_i \xleftarrow{Ch=\{t,N,SIG_{\mathcal{A}dm}\}} \mathcal{A}dm$

if *If* $\left(verify(PK_{\mathcal{A}dm}, t|N, SIG_{\mathcal{A}dm})\right)$ **then**

  $SIG_i = \mathsf{sign}(SK_i, LOG_i|Ch)$

  $D_i \xrightarrow{resp_i=\{LOG_i,SIG_i\}} \mathcal{A}dm$

else

  Discard Ch

end

$\mathcal{A}dm$ can easily determine whether any devices were absent for one or more heartbeat protocol instances since last attestation. Furthermore, $\mathcal{A}dm$ can identify the devices that were absent at each heartbeat protocol instance. According to the **Normalcy** assertion, if all logs match and contain all devices' heartbeats, then no device has been absent. The collect protocol is shown, in more detail, in Figure 2 and 3.

4.3 Efficiency Improvements

Several aspects of the above protocols involve some potentially impractical assumptions and costly operations. This is done mainly in order to simplify presentation; substantial cost reductions and simplifications can be made, as follows:

Communication Model: The collect protocol in Figures 2 and 3 is shown in an idealized communication setting, where each $\mathcal{A}dm$ "talks" *directly* to every D_j. While plausible, this is not a realistic assumption. However, recall that $\mathcal{A}dm$'s attestation request (Ch) is signed and so is D_i's reply – $resp_i$. These signatures essentially result in an authentic channel between $\mathcal{A}dm$ and D_i. Consequently, protocol's security is the same, regardless of whether $\mathcal{A}dm$ and D_i communicate directly, or through a sequence of intermediate hops, i.e., one or more D_j-s.

Public Key Signatures: Both heartbeat and collect protocols involve heavy use of signatures. This is done mainly for illustration purposes, as it makes the protocol more concise, though also more expensive. In the heartbeat protocol, it is easy to replace signatures with hop-by-hop MACs, assuming that a key shared is by every pair of neighboring devices. It is equally easy to replace signatures with MACs in collect, as long as each device shares a unique key with $\mathcal{A}dm$. In our context, MACs are no less secure than signatures, since our $\mathcal{A}dv_{0,s-1}$ is purely physical and can not capture *all* devices. Thus, as long as at least one D_z remains always present (uncaptured), it would accurately log at least one absent device (see Section 6 for details). One obvious downside of using MACs is the need for a separate key establishment protocol between adjacent devices, or a key pre-distribution scheme, to let each device share a symmetric key with every neighbor.

Heartbeat Logs: Figures 2 and 3 show each D_i collecting, and later sending to $\mathcal{A}dm$, accumulated logs (of the form $log_{i,p}$) bundled into LOG_i at collect time. Once again, this is done for ease of illustration. Instead, it is sufficient for each $log_{i,p}$ to contain a list of either absent or present devices, whichever is smaller. At the very minimum, we simply use a binary flag indicating whether at least one device was absent during any heartbeat instance. An optimized MAC-based implementation of DARPA is described in Section 6.

4.4 heartbeat: Correctness and Security

Correctness of the heartbeat protocol means that, if no device is absent, then all log sets (LOG_i-s) of all devices must match and contain every other device's heartbeat.

We assume that D_i is present and functional during the entire inter-attestation time gap T_{att}. Thus, D_i periodically emits an authenticated heartbeat $hb_{i,p} = \{p, t_{i,p}, id_i, \}$, and $SIG_{i,p}$ to all its neighbors. Since all devices have reliable clocks loosely synchronized with $\mathcal{A}dm$, the clock of every neighbor D_j deviates from $t_{i,p}$ by at most δ_t. Consequently, D_j receives $hb_{i,p}$ at $t \in [t_{i,p} - \delta_t, t_{i,p} + \delta_t + t_{tx}]$. Recall that t_{tx} is the maximum time to transmit a message between any two devices. Since $hb_{i,p}$ is a genuine heartbeat of D_i for the current heartbeat interval, i.e., $accept(p, t_{i,p}) = 1$, and verify$(PK_i, SIG_{i,p}) = 1$; and is received within the acceptance interval $t_{acc} = t_{tx} + \delta_t$, D_j appends $hb_{i,p}$ to its $log_{j,p}$ and forwards $hb_{i,p}$ it to all its neighbors. Similarly, each neighbor of D_j appends $hb_{i,p}$ to its $log_{j,p}$ and forwards it onwards. Since the network is always connected and its topology is static only for the duration of heartbeat instance, all heartbeat messages are received by all devices, within t_{tx} time. Thus, every D_k appends $hb_{i,p}$ to its $log_{k,p}$. It follows that LOG_j of every D_j matches and contains $hb_{i,p}$ of every other D_i for all p heartbeat instances.

Security of heartbeat means that, if $0 < w < s$ devices are absent, then logs of present devices lack at least one heartbeat of each absent one.

We again assume D_i is present, i.e., not captured. According to the definition of $accept()$ in Table 1, every uncaptured D_k only logs the heartbeat $hb_{i,p}$ (of D_i at p-th heartbeat interval) in its log if it was received in the same interval. We derive the upper and lower-bound on the time when $hb_{i,p}$ can be logged in $log_{k,p}$ (log time). Let t_1 be the expiration time of the p-th $timer_{hb}$ on any present D_k, i.e., at t_1, D_k generates its heartbeat $hb_{k,p}$, and logs in $log_{k,p}$ all heartbeats received either before t_1 or within $t_{acc} = t_{tx} + \delta_t$. However, since present devices (such as D_i) only generate correct heartbeats (i.e., based on the current timestamp), the acceptable $hb_{i,p}$ can be generated no earlier than $t_1 - \delta_t$. Consequently, log time is lower-bounded by $t_1 - \delta_t$, and upper bounded by $t_1 + t_{tx} + \delta_t$. Similarly, if $t_2 = t_1 + t_{hb}$ is the expiration time of $(p+1)$-th $timer_{hb}$, log time of $hb_{i,p+1}$ is in $[t_2 - \delta_t, t_2 + t_{tx} + \delta_t]$. Therefore, D_i can be absent (and undetected) for at most: the time between sending $hb_{k,p}$ at the earliest time possible and emission of $hb_{k,p+1}$ at the latest time possible:

$$t_2 + t_{tx} + \delta_t - (t_1 - \delta_t) = t_1 + t_{hb} + t_{tx} + \delta_t - t_1 + \delta_t = t_{hb} + 2 \cdot \delta_t + t_{tx}$$

Since $T_{cap} > t_{hb} + 2 \cdot \delta_t + t_{tx}$, the log of every uncaptured D_k will lack at least one heartbeat of every captured D_i (either $hb_{i,p}$ or $hb_{i,p+1}$).

5. MITIGATING SOFTWARE ADVERSARY

In contrast to a physical adversary $\mathcal{A}dv_{0,s-1}$, a purely software $\mathcal{A}dv_{s,0}$ can not capture devices. However, it can remotely compromise any device's software, which is subject to attestation. In this section, we assume that the adversary can software-compromise *all* devices.

To guarantee security under $\mathcal{A}dv_{s,0}$, the attestation protocol must remotely verify overall software integrity of the network and, thus, of each D_i. To achieve this, we need to make some minimal assumptions about hardware security features of the underlying devices. In particular, the integrity measurement mechanism (attestation code) residing on D_i must be immune to software attacks. This condition can be satisfied by assuming availability of minimal hardware protection on each D_i, as done in the attestation literature for low-end embedded devices [4, 8, 19, 29]. In brief, minimal requirements/features are: Read-only Memory (ROM) for storing attestation code and associated cryptographic key(s) and a simple memory protection unit (MPU). The MPU has several tasks: (i) grants access to cryptographic key(s) **only** to attestation code, (ii) ensures non-interruptibility of execution of attestation code,and (iii) cleans up (flushes all registers and other temporary storage) at the end of attestation code execution.

Based on the above, compromised devices can be detected. Indeed, this is exactly what is achieved by a collective attestation protocol as recently proposed in SEDA [4]. In SEDA, $\mathcal{A}dm$ chooses an arbitrary device (initiator), and sends it an attestation request. The initiator forwards the request to its immediate neighbors who repeat the process. The request propagates over the entire network forming a spanning **tree** rooted at the inititator. Next, each leaf sends its parent an integrity report. The parent verifies each child's report and aggregates them (along with its own) into a single report, which includes the number of correctly attested devices. Reports are then propagated and aggregated, in reverse, along the spanning tree towards the initiator, and on to the verifier.

6. MITIGATING HYBRID ADVERSARY

The strongest adversary $\mathcal{A}dv_{s,s-1}$ can compromise all devices by software means, and physically attack all-but-one devices – D_z, within a single inter-attestation time gap T_{att}. To mitigate this adversary we integrate a secure collective attestation protocol (such as SEDA) which detects compromised devices, with DARPA:collect and combine it with DARPA:heartbeat to detect absent devices.

This requires the same hardware assumptions as in Section 5 to secure both the attestation and DARPA protocols. Most importantly, we assume that ROM-resident attestation code now includes heartbeat and collect protocols. Also, cryptographic keys used by heartbeat and collect are stored in ROM and accessible only to that attestation code. Private intermediate data, on the other hand is stored in access-protected RAM.

Figure 4 shows our implementation of the integrated protocol over a generalization of the SMART architecture [19] introduced in [8, 29]. In it, a Memory Protection Unit (MPU) controls access to the part of memory (ROM and RAM) storing secret keys and private protocol data. Access is controlled according to hardware-based access control rules in the table of Figure 4. For example, rule # 1 states that: *only code residing at address a_0 to a_3 (i.e., attestation,* heartbeat*, and* collect*) has read access to keys residing at addresses a_4 to a_5*. Similarly, heartbeat and collect have protected read and/or write access to their private intermediate data.

6.1 Protocol

The integrated protocol is shown in Figure 5; it is based entirely on symmetric cryptography. Every D_i is assumed to share a key K_i with $\mathcal{A}dm$, and k_{ij} – with each direct neighbor D_j.

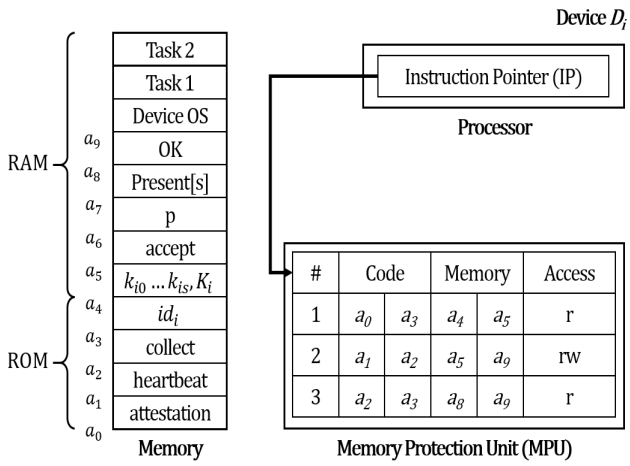

Figure 4: Implementation of DARPA

The protocol operates as follows: every t_{hb} interval, each D_i (D_w and D_k in the figure) generates a timestamped authenticated heartbeat $hb_{i,p}$ and σ_{ij} and broadcasts it to all neighbors D_j. Upon receiving an authenticated heartbeat from a neighbor D_j, D_i checks that:

1. $hb_{j,p}$ is not a duplicate.
2. $hb_{j,p}$ is received within acceptable time, i.e., $accept = true$.
3. $hb_{j,p}$ corresponds to current expected heartbeat instance, i.e., p.
4. the MAC σ_{ij} on $hb_{j,p}$ (re-computed with k_{ij}) verifies correctly.

If all checks succeed, D_i marks D_j as present ($present[j] = 1$), generates a new MACs on $hb_{j,p}$, and forwards it to its neighbors. After a fixed period of time ($t = 2 \cdot \delta_t + t_{tx}$), D_i checks whether all devices were present during the current heartbeat instance. If at least one D_j was absent, D_i sets $OK = 0$.

At attestation time, $\mathcal{A}dm$ randomly picks an initiator device – D_w – which serves as the interface to the rest of the network, see Figure 5. $\mathcal{A}dm$ creates a random challenge N, authenticates it along with the current time $t_{\mathcal{A}dm}$ with a MAC based on a shared key K_w. $\mathcal{A}dm$ then sends the authenticated challenge CH_w to D_w. This triggers both attestation and collect. Once D_w verifies CH_w, it creates a separate CH_j (authenticated with k_{wj}-based MAC) and forwards it to each neihboring D_j. Upon receiving CH_j, each neighbor authenticates the request and forwards an authenticated copy to each of its neighbors, and so on. This way, the challenge securely propagates throughout the network along a spanning tree rooted at D_w. Each leaf D_i creates a reply which includes:

1. $resp_i$, indicating whether a single device was absent at any heartbeat interval ($resp_i = \mathsf{mac}(K_i, OK|N)$). $resp_i$ is authenticated in an end-to-end manner.
2. its attestation response $attest_i$ to its parent D_j. $attest_i$ depends on the integrated collective attestation protocol. In SEDA, it represents the number of successfully attested devices.

The parent accumulates $attest_i$ and XORs $resp_i$ (along with its own) and propagates the result upstream, towards D_w. Finally, D_w forwards the accumulated reply to $\mathcal{A}dm$. $\mathcal{A}dm$ re-generates all MACs, XORs them, and compares the result to $resp_w$. If they match, $\mathcal{A}dm$ decides that no device has been captured and can now trust the attestation results in determining compromised devices, if any. The protocol is shown in Figure 5.

A distinct session identifier is needed to avoid double-counting benign devices. This identifier is sent along with the challenge and each device includes it in the attestation response. However, to allow devices to attest their neighbors and account for software updates, every device is initialized with its own software certificate, i.e., its software configuration signed by $\mathcal{A}dm$. Upon joining the network, each device sends its software certificate to its neighbors, each of whom verifies and stores it for future attestation. Further information about session management and distribution of software configurations for collective attestation can be found in [4].

6.2 Correctness and Security

Somewhat surprisingly, the integrated protocol is actually **not secure** against the strongest hybrid adversary. The main (and only) reason is due to the fact that the *Reliable Clocks* assumption is insufficient to mitigate $\mathcal{A}dv_{s,s-1}$. Consider the following attack scenario:

> *First $\mathcal{A}dv_{s,s-1}$ captures D_i during $p-th$ heartbeat instance and extracts its keys. Concurrently, $\mathcal{A}dv_{s,s-1}$ compromises (via software) all devices: $D_1, ..., D_s$ and infects them with malware, which tampers with their clocks, and extends the time a received heartbeat is accepted to $\geq T_{cap}$. After capture, D_i runs the p-th heartbeat instance. On each compromised device, D_i's heartbeat is accepted by the ROM-resident code running the* heartbeat *protocol, since it appears to be correct.*

Other similar attack scenarios are possible, where the common feature is the ability of malware (which is resident on compromised devices) to manipulate their clocks.

This insecurity leads us towards an additional requirement that we claim to be both necessary and sufficient to mitigate a hybrid adversary: the need for a **Reliable Read-Only Clock (RROC)** on each device. We use this terminology in order to stress the fact that each device's clock must be non-malleable, i.e., not modifiable by software from either ROM or RAM. However, we do not require the clock to be *secure* in terms of any kind of physical protection, i.e., no tamper-resistance or tamper-evidence is needed. An RROC can be realized with an inexpensive commercially available Real-Time Clock (RTC), such as [1].[2]

One alternative to the RROC requirement is **secure writable memory**. Such memory must be writeable only from ROM. It is not hard to see that, if each D_i has just a tiny amount of such memory, it can securely record the timestamp of the last heartbeat protocol. This way, once malware resets the D_z's clock to a prior heartbeat protocol instance, it would attempt to introduce backdated heartbeat messages from previously absent devices, ROM-resident attestation code would recognize stale timestamps, detect the inconsistency and take appropriate action. At the very least, D_z can ignore backdated heartbeats.

[2]Note that integrity of any intermediate software that reads the RROC is assumed to be assured, similar to attestation code in Figure 4.

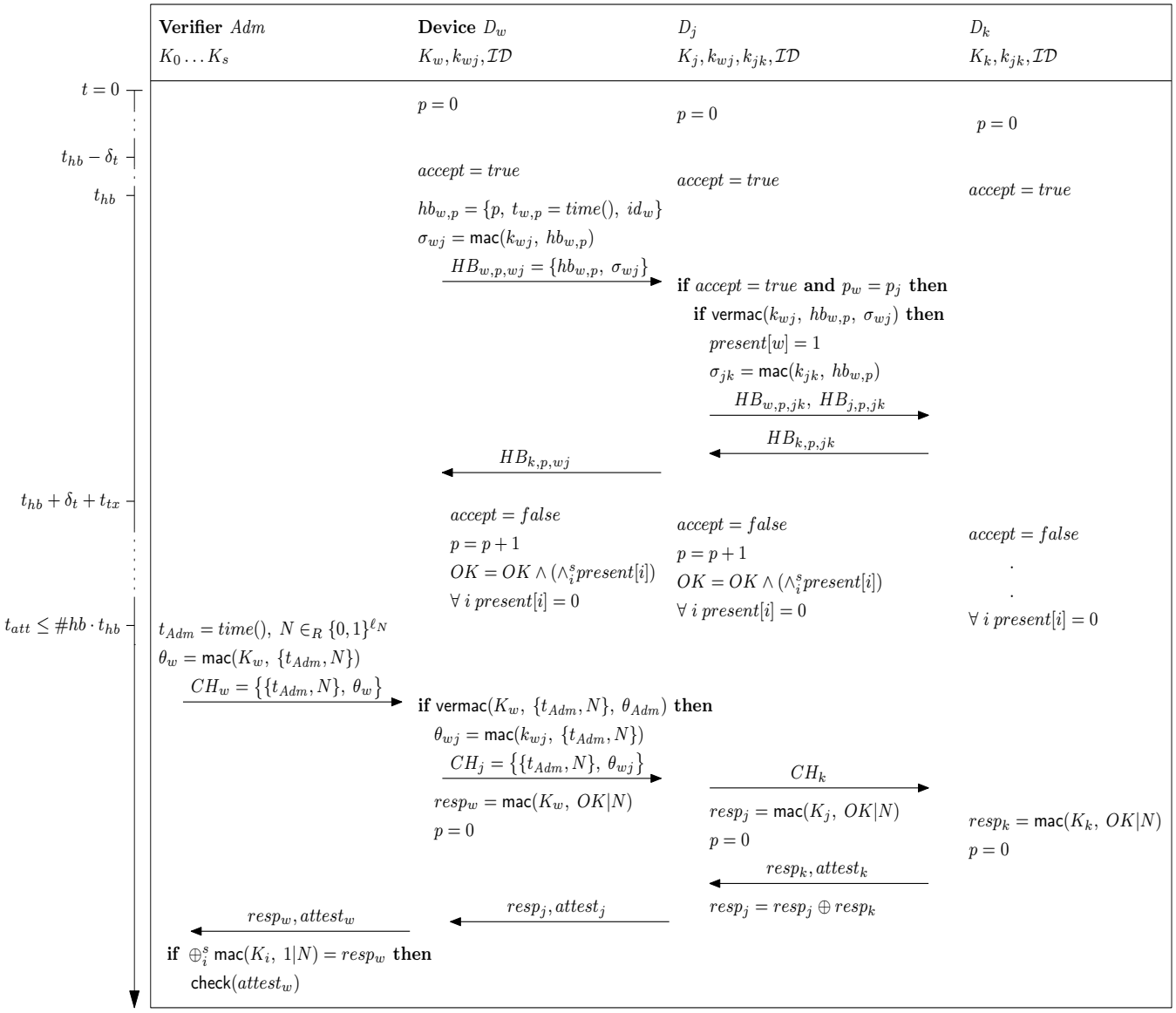

Figure 5: The complete DARPA protocol

Correctness of the integrated protocol means that, if no device was absent or compromised, then: (1) $attest_w$ indicates that no device is compromised, and (2) $resp_w$ matches $\oplus_i^s\ \mathsf{mac}(K_i,\ 1|N)$.

(1) follows directly from correctness of the collective attestation protocol. According to correctness of heartbeat, all devices receive valid heartbeats for all heartbeat instances, within T_{att}. Consequently, at each heartbeat instance and for every D_i, $present[i]$ is set to 1. Thus, at attestation time OK at D_i is also 1 ($OK = OK \wedge (\wedge_i^s present[1]) = 1 \wedge \dots \wedge 1 = 1$), and $resp_i$ is initially set to $\mathsf{mac}(K_i,\ 1|N)$. Therefore, since all devices receive and accept CH, $resp_w = \oplus_i^s\ \mathsf{mac}(K_w,\ 1|N)$.

Security of the integrated protocol means that: (1) if no device was captured, then $attest_w$ correctly indicates the number of compromised devices; and (2) if at least one device is captured, $resp_w$ reflects that.

(1) follows directly from security of the collective attestation protocol. Furthermore, if no device is captured, hop-by-hop MACs and digital signatures are equally secure, since (due to SMART-like device architecture) malware has no access to keys. Thus, based on security of heartbeat, capture of (at least) one D_j is detected by all other devices. Of course, D_j can later forge heartbeats of absent devices, thus helping them evade detection. However, it can not reset an OK flag of an uncaptured device to 1 once it is set to 0 ($OK = \mathbf{OK} \wedge (\wedge_i^s present[i])$). Meanwhile, since keys shared with $\mathcal{A}dm$ and digital signatures are equally secure, $resp_w$ correctly indicates that at least one device (say, D_j) was captured. Consequently, $\mathcal{A}dv$ can try to evade detection of D_j's capture in several ways: (1) modifying heartbeat or collect protocol code, (2) extracting K_j before capturing D_j, (3) extracting K_z or modifying OK for every uncaptured D_z within the same T_{att}, or (4) fooling every uncaptured D_z into extending the time a received heartbeat is accepted, as illustrated in the attack example above. However, (1)

is impossible since heartbeat and collect are part of ROM-resident attestation code. (2) and (3) are prevented since cryptographic keys and private data are accessible only to the attestation code. Finally, (4) is ruled out since D_z has a Reliable Read-Only Clock (RROC) which can not be modified by any software. Therefore, $\mathcal{A}dv_{s,s-1}$ can not evade detection of D_j by software compromise. Consequently, $resp_w$ will correctly indicate that at least one device (i.e., D_j) was captured.

7. PERFORMANCE ANALYSIS

We evaluated computation, communication, and energy costs of both variant of DARPA We also compared, via simulations, performance of MAC- and signature-based implementations, for varying numbers of devices.

Computation Overhead: In the signature-based implementation, every D_i generates one and verifies s signatures in heartbeat. In collect, D_i generates one and verifies $(c_i + 1)$ signatures. Let c_i be the number of children of D_i in the spanning tree rooted in D_w. In the MAC-based implementation, D_i verifies s and generates $(N \cdot s)$ MACs in heartbeat. In collect, D_i verifies $(c_i + 1)$, and generates $(N + 1)$ MACs, where N is the number of D_i's neighbors.

Communication Overhead: In the digital signature-based implementation, D_i receives and sends at most N heartbeats in heartbeat. D_i, also receives N challenges and c_i responses; meanwhile, it sends 1 challenge and 1 response. The number of log sets in received and sent responses depends on D_i's position in the spanning tree. It is upper bounded by $s - 1$ and s respectively. In the MAC-based implementation, D_i receives s and sends s heartbeats as well as $N \cdot s$ MACs in heartbeat. D_i also receives N authenticated challenges and c_i MACs, while it sends 1 challenge and N MACs.

Energy Costs: Our estimate of DARPA's energy consumption is shown in Figure 6 and 7. We base it on energy costs of cryptographic operations and communication reported for TelosB sensor node [16]. We also set the number of heartbeat protocol instances to 20. As the results show, energy consumption is quadratic in the network size in the signature-based implementation, and linear in the MAC-based version. This significant improvement is mainly due to the significant reduction in communication. Moreover, on low-end embedded devices MACs are obviously much cheaper energy-wise than digital signatures. Figure 11 also shows an increase in energy consumption as a function of number of neighbors for the MAC-based version. This is due to the hop-by-hop MAC verification and re-generation.

Simulation Results: DARPA was evaluated using omnet++ simulator [40] with several topologies: chain, star, tree (with fan-out degrees: 2, 4, 8 and 12), and networks with fixed number of neighbors (4, 8 and 12). We emulated cryptographic operations as delays based on measurements from TyTAN [8]. Simulation uses 20-kbps as the communication rate for links between devices. It corresponds to the minimum bandwidth provided by ZigBee – a common protocol for IoT devices. We simulated collect and heartbeat based on digital signatures, described in Section 4, and the optimized MAC-based implementation described in Section 6. Results are shown in Figure 8, 9, 10, and 11. Results for networks with fixed number of neighbors are very similar to those for trees and are hence omitted, due to space limitations.

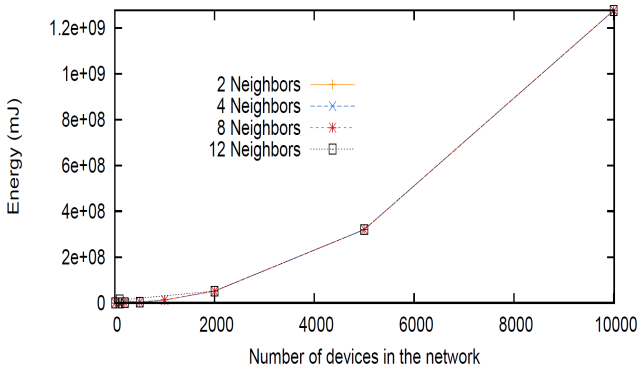

Figure 6: Energy consumption of signature-based DARPA

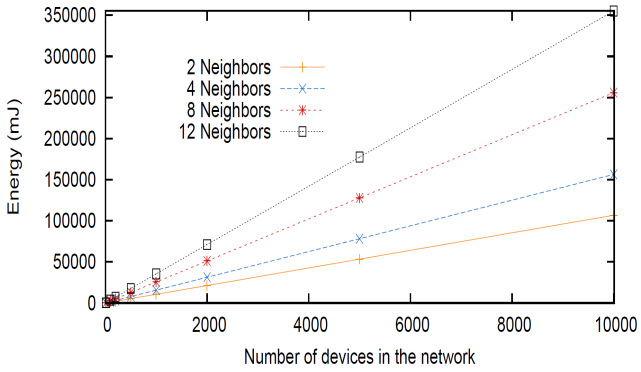

Figure 7: Energy consumption of MAC-based DARPA

As shown in Figure 8 and 9, for both implementations, and aforementioned topologies, run-time of heartbeat is linear in network size. However, the MAC-based implementation performs better, particularly in a chain topology. Computational overheard of heartbeat is linear (in network size) in chain and star, while logarithmic in tree topologies. Its communication overhead is always linear in network size. For this reason, the effect of replacing digital signatures with MACs is reduced. On the other hand, Figure 9 shows that run-time of heartbeat in $1,000 - node$ networks is about 13 seconds. This number is less than T_{cap}, and is therefore realistic.

Figure 10 and 11 show run-times of collect with digital signatures and MACs, respectively. The quadratic run-time of collect in the signature-based implementation is due to the quadratic communication overhead, since each of the s devices has to communicate its s log lists to $\mathcal{A}dm$. Meanwhile, in the MAC-based implementation, only a constant size XOR-ed MAC is communicated. Consequently, this significant difference in run-times between the two implementations is due to the reduction in communication overhead from sending concatenated logs to sending XOR-ed constant size MACs. Note that the run-time of collect in the MAC-based implementation consequently converges to the computational overhead, which is linear in chain and star (and logarithmic in tree) topology.

Finally, as shown in the figures, chain is the worst-performing topology for collect, since responses are always sent through s hops, compared to $log(s)$ hops for trees and 1 hop for

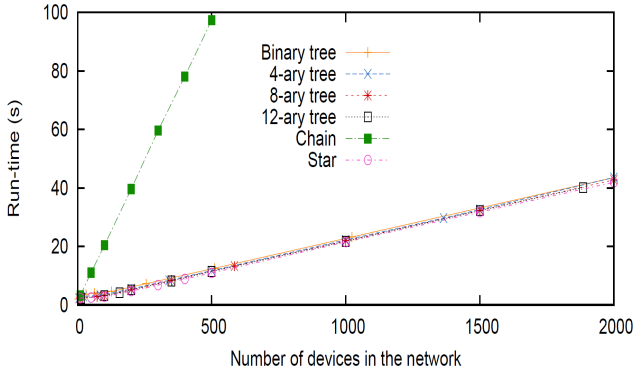

Figure 8: DARPA:heartbeat signature-based performance

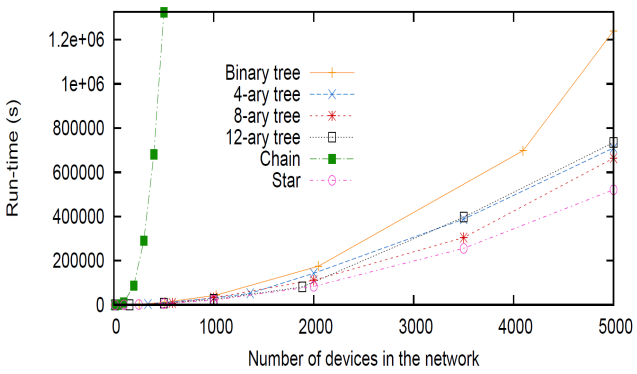

Figure 10: DARPA:collect signature-based performance

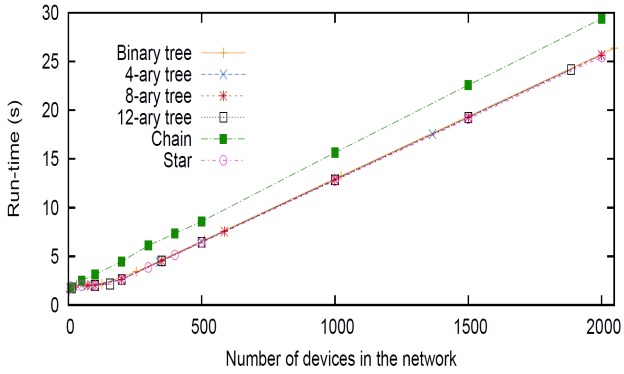

Figure 9: DARPA:heartbeat MAC-based performance

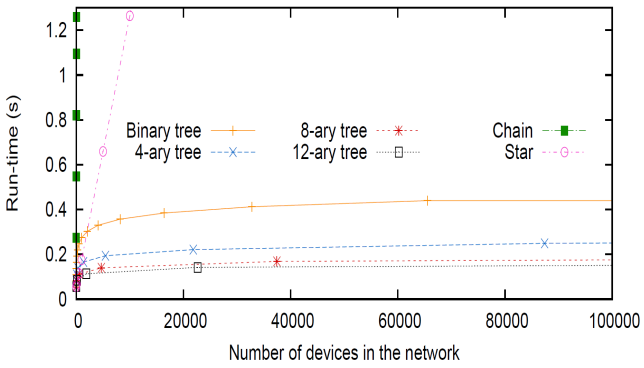

Figure 11: DARPA:collect MAC-based performance

star. On the other hand, star topology performs best in protocols where communication overhead dominates, i.e., in signature-based implementation of collect, and worst when computation overhead is dominant, i.e., in MAC-based. In the latter case, the central node in the star has to verify s MACs.

8. RELATED WORK

Remote Attestation is a popular research topic and many schemes have been proposed in the literature. They all share a scenario where the prover sends to the verifier a status report of its current software configuration. Authenticity of the report is typically assured by some form of secure hardware [19, 30, 31, 43, 44, 52] and/or trusted software [2, 22, 28, 31, 34, 45, 46, 54]. Attestation based on secure hardware is often too complex and expensive for low-end embedded systems. Software-based attestation [22, 28, 34, 45, 46] does not require any hardware and involves no cryptographic keys. However, its security relies on strong assumptions that are hard to achieve in practice [3], e.g.: adversarial silence while the attestation protocol runs, optimality of the attestation algorithm and its implementation, and fixed round-trip prover-verifier delay. Hence, common wisdom implies that secure and practical remote attestation requires at least a few security features in hardware [19, 21, 29].

Current attestation schemes consider only a single prover and do not accommodate groups thereof. Moreover, they consider only (remote) software attackers, with no physical access to provers. Only the recent SEDA [4] scheme performs collective attestation of interconnected devices by distributing attestation burden across the entire network. However, SEDA considers only remote software attacks, while DARPA identifies both compromised and captured devices.

Absence Detection is used (particularly, in sensor networks) to detect node failures and captures. Detecting failures by absence detection has been studied for static [50] and dynamic topologies [13, 23]. However, since these schemes are not designed with security in mind, they are ineffective in our adversarial setting.

A physical attack on a device requires expert knowledge, costly equipment, and most importantly, in most cases, removal of the device(s) from the network for a non-negligible amount of time [5]. Therefore, absence detection has been used to detect node capture attacks, by letting each device keep track of the time it needs to re-encounter a predetermined subset of peer devices [14, 15]. This time is then compared to a predetermined threshold. In static networks, each device simply measures absence time for each neighbor, and compares it to the minimal time needed to capture a device [24]. Techniques for dynamic networks allow a certain number of false negatives, depending on the threshold. However, static network techniques can not be extended to dynamic networks. Moreover, all proposed schemes are vulnerable to remote software attacks. In contrast, DARPA, besides being applicable to static and dynamic networks, can detect both remote software and physical attacks.

Secure data aggregation is a fundamental communication primitive in wireless sensor networks. It reduces communication overhead considerably, through combining data received from distinct sensor nodes, while preserving its security (secrecy and/or integrity). Several approaches have been proposed for integrity-preserving data aggregation. Some are based on cryptographic techniques [10, 11, 27, 32, 37, 42, 57], while others rely on trust relations [41] or witness-based solutions [17]. Integrity-preserving data aggregation can be combined with DARPA to provide security against physical adversary. However, these approaches involve computationally expensive asymmetric cryptography [32], or require globally shared keys [37]. Moreover, they have high computation and communication complexity [12, 17, 39]. They also mainly focus on detecting integrity breaches, rather than the identification of misbehaving nodes. Schemes that are able to detect captured nodes require a number of rounds which is at least logarithmic in the size of the network [51]. Moreover, such schemes are unable to detect captured nodes, unless they misbehave (i.e., alter with the aggregation result).

Key management protocols aim to achieve a trade-off between storage requirements, key connectivity, and resilience to node compromise. Most of proposed schemes aim at decreasing the number of keys each node needs to store. Storage reduction is achieved on the expense of connectivity, i.e., by leveraging knowledge of the network topology [56], or the probability of every two nodes communicating [20]. In such schemes, two nodes share a key if they consider each other reachable [35], or can not rely on other node(s) to provide a secure path [33]. It can also be achieved at the expense of resilience to node compromise. This is done using various mathematically flavored key distribution techniques [6, 7, 9, 18, 36]. While such schemes do not aim to detect captured nodes, they try to minimize the effect of node capture in terms of compromise of communication links. Moreover, some enable revocation of leaked credentials. DARPA requires either one private (signing) key per device or several symmetric keys shared between neighbors, and between every device and the verifier. Thus, its storage requirement is minimal. However, end-to-end authenticity is maintined due to attestation code's exclusive access to keys.

9. SUMMARY AND FUTURE WORK

This paper proposes DARPA – a scheme which mitigates a very powerful adversary capable of physical attacks, under reasonable assumptions. However, despite its benefits, DARPA has certain limitations that need to be addressed, including:

- False positives due to device failures and temporary unreachability, e.g., network partitioning.
- Lack of identification of potentially compromised devices.
- Relatively high communication overhead of the heartbeat protocol.

There are several concrete issues that we plan to tackle in the near-term:

- Use majority voting to identify compromised devices.
- Apply witness-based approach to mitigate network partitioning, i.e., if $\mathcal{A}dv$ physically attacks $\leq k$ devices within the inter-attestation gap, then each device must have at least $k+1$ witnesses for each heartbeat period.
- Lower heartbeat protocol overhead by sending heartbeats only to neighbors.
- Extend DARPA to support device mobility during heartbeat.

Acknowledgements

We thank anonymous reviewers for their useful comments, as well as Levente Buttyán for his insightful and constructive feedback. At TU Damstadt, this research was co-funded by the German Science Foundation, as part of project S2 within CRC 1119 CROSSING, EC-SPRIDE, the European Union's 7^{th} Framework Programme, under grant agreement No. 609611, PRACTICE project, and Intel Collaborative Research Institute for Secure Computing (ICRI-SC). At UC Irvine, this research was supported by funding from the National Security Agency (H98230-15-1-0276) and the Department of Homeland Security (under subcontract from HRL Laboratories).

References

[1] Real-time clocks (rtc) ics. https://www.maximintegrated.com/en/products/digital/real-time-clocks.html.
[2] W. Arbaugh et al. A secure and reliable bootstrap architecture. In *IEEE S&P*, 1997.
[3] F. Armknecht et al. A security framework for the analysis and design of software attestation. In *ACM CCS*, 2013.
[4] N. Asokan et al. Seda: Scalable embedded device attestation. In *ACM CCS*, 2015.
[5] A. Becher et al. *Tampering with motes: Real-world physical attacks on wireless sensor networks*. Springer, 2006.
[6] R. Blom. An optimal class of symmetric key generation systems. In *EUROCRYPT Workshop*, 1984.
[7] C. Blundo et al. Perfectly secure key distribution for dynamic conferences. *Information and Computation*, 1998.
[8] F. Brasser et al. Tytan: Tiny trust anchor for tiny devices. In *DAC*, 2015.
[9] S. Çamtepe et al. Combinatorial design of key distribution mechanisms for wireless sensor networks. *IEEE/ACM Transactions on Networking*, 2007.
[10] H. Chan et al. Secure hierarchical in-network aggregation in sensor networks. In *ACM CCS*, 2006.
[11] H. Chan et al. SIA: Secure information aggregation in sensor networks. *JCS*, 2007.
[12] C.-M. Chen et al. RCDA: Recoverable concealed data aggregation for data integrity in wireless sensor networks. *TPDS*, 2012.
[13] W. Chen et al. On the quality of service of failure detectors. *IEEE TC*, 2002.
[14] M. Conti et al. Emergent properties: Detection of the node-capture attack in mobile wireless sensor networks. In *WiSec*, 2008.
[15] M. Conti et al. Mobility and cooperation to thwart node capture attacks in manets. *EURASIP WCN*, 2009.
[16] G. de Meulenaer et al. On the energy cost of communication and cryptography in wireless sensor networks. In *WiMob*, 2008.
[17] W. Du et al. A witness-based approach for data fusion assurance in wsn. In *GLOBECOM*, 2003.
[18] W. Du et al. A pairwise key predistribution scheme for wireless sensor networks. *ACM TISSEC*, 2005.
[19] K. Eldefrawy et al. SMART: Secure and minimal

architecture for (establishing a dynamic) root of trust. In *NDSS*, 2012.
[20] L. Eschenauer et al. A key-management scheme for distributed sensor networks. In *ACM CCS*, 2002.
[21] A. Francillon et al. A minimalist approach to remote attestation. In *DATE*, 2014.
[22] R. Gardner et al. Detecting code alteration by creating a temporary memory bottleneck. *IEEE TIFS*, 2009.
[23] N. Hayashibara et al. Failure detectors for large-scale distributed systems. In *SRDS*, 2002.
[24] J.-w. Ho et al. *Distributed Detection of Node Capture Attacks in Wireless Sensor Networks*. InTech, 2010.
[25] C. Hsin et al. Self-monitoring of wireless sensor networks. *Computer Communications*, 2006.
[26] C.-f. Hsin et al. A distributed monitoring mechanism for wireless sensor networks. In *ACM workshop on Wireless security*, 2002.
[27] L. Hu et al. Secure aggregation for wireless networks. In *SAINT Workshops*, 2003.
[28] R. Kennell et al. Establishing the genuinity of remote computer systems. In *USENIX*, 2003.
[29] P. Koeberl et al. TrustLite: A security architecture for tiny embedded devices. In *EuroSys*, 2014.
[30] J. Kong et al. PUFatt: Embedded platform attestation based on novel processor-based PUFs. In *DAC*, 2014.
[31] X. Kovah et al. New results for timing-based attestation. In *IEEE S&P*, 2012.
[32] V. Kumar et al. Secure hierarchical data aggregation in wireless sensor networks: Performance evaluation and analysis. In *IEEE MDM*, 2012.
[33] J. Lee et al. A combinatorial approach to key predistribution for distributed sensor networks. In *WCNC*, 2005.
[34] Y. Li et al. VIPER: Verifying the integrity of peripherals' firmware. In *ACM CCS*, 2011.
[35] D. Liu et al. Location-based pairwise key establishments for static sensor networks. In *SASN*, 2003.
[36] D. Liu et al. Establishing pairwise keys in distributed sensor networks. *ACM TISSEC*, 2005.
[37] A. Mahimkar et al. Securedav: A secure data aggregation and verification protocol for sensor networks. In *GLOBECOM*.
[38] S. Mansouri et al. An architectural countermeasure against power analysis attacks for fsr-based stream ciphers. In *COSADE*. 2012.
[39] S. Nath et al. Secure outsourced aggregation via one-way chains. In *SIGMOD*, 2009.
[40] OpenSim Ltd. OMNeT++ discrete event simulator. http://omnetpp.org/.
[41] S. Ozdemir. Secure and reliable data aggregation for wireless sensor networks. In *Ubiquitous Computing Systems*. 2007.
[42] S. Papadopoulos et al. Exact in-network aggregation with integrity and confidentiality. *TKDE*, 2012.
[43] B. Parno et al. Bootstrapping trust in commodity computers. In *IEEE S&P*, 2010.
[44] S. Schulz et al. Short paper: Lightweight remote attestation using physical functions. In *WiSec*, 2011.
[45] A. Seshadri et al. SWATT: Software-based attestation for embedded devices. In *IEEE S&P*, 2004.
[46] A. Seshadri et al. SAKE: Software attestation for key establishment in sensor networks. In *DCOSS*. 2008.
[47] S. Skorobogatov. Physical attacks on tamper resistance: Progress and lessons. In *Workshop on Hardware Assurance*, 2011.
[48] S. Skorobogatov. Physical attacks and tamper resistance. In *Introduction to Hardware Security and Trust*. 2012.
[49] S. P. Skorobogatov. *Semi-invasive attacks: a new approach to hardware security analysis*. PhD thesis, 2005.
[50] P. Stelling et al. A fault detection service for wide area distributed computations. *Cluster Computing*, 1999.
[51] G. Taban et al. Efficient handling of adversary attacks in aggregation applications. In *ESORICS*, 2008.
[52] Trusted Computing Group (TCG). Website. http://www.trustedcomputinggroup.org, 2015.
[53] R. van Renesse et al. A gossip-style failure detection service. In *Middleware*, 1998.
[54] A. Vasudevan et al. CARMA: A hardware tamper-resistant isolated execution environment on commodity x86 platforms. In *ASIACCS*, 2012.
[55] J. Vijayan. Stuxnet renews power grid security concerns, 2010.
[56] Z. Yu et al. A key management scheme using deployment knowledge for wireless sensor networks. *TPDS*, 2008.
[57] W. Zhang et al. Secure data aggregation in wireless sensor networks: A watermark based authentication supportive approach. *Pervasive and Mobile Computing*, 2008.
[58] Y. Zhou et al. Side-channel attacks: Ten years after its publication and the impacts on cryptographic module security testing. *IACR Cryptology ePrint Archive*, 2005.