FADIA: FAirness-Driven collaboratIve remote Attestation

Mohamad Mansouri
Thales SIX GTS, EURECOM, France
mohamad.mansouri@thalesgroup.com

Wafa Ben Jaballah
Thales SIX GTS, France
wafa.benjaballah@thalesgroup.com

Melek Önen
EURECOM, France
melek.onen@eurecom.fr

Md Masoom Rabbani
KU Leuven, Belgium
mdmasoom.rabbani@esat.kuleuven.be

Mauro Conti
University of Padua, Italy
conti@math.unipd.it

ABSTRACT

Internet of Things (IoT) technology promises to bring new value creation opportunities across all major industrial sectors. This will yield industries to deploy more devices into their networks. A key pillar to ensure the safety and security of the running services on these devices is remote attestation. Unfortunately,existing solutions fail to cope with the recent challenges raised by large IoT networks. In particular, the heterogeneity of the devices used in the network affects the performance of a remote attestation protocol. Another challenge in these networks is their dynamic nature: More IoT devices may be added gradually over time. This poses a problem in terms of key management in remote attestation.

We propose FADIA, the first lightweight collaborative remote attestation protocol that is designed with fairness in mind. FADIA enables fair distribution of load/tasks on the attesting devices to achieve better performance. We also leverage the Eschenauer-Gligor scheme to enable efficient addition of devices to the network. We implement our solution on heterogeneous embedded devices and evaluate it in real scenarios. The evaluation shows that FADIA can (i) increase the lifetime of a network by an order of magnitude and (ii) decrease the remote attestation runtime by a factor of 1.6.

CCS CONCEPTS

• **Security and privacy** → **Security protocols**; *Distributed systems security*; Mobile and wireless security; • **Networks** → *Mobile ad hoc networks.*

KEYWORDS

remote attestation, collaborative attestation, heterogeneous IoT networks, fairness, embedded systems

ACM Reference Format:
Mohamad Mansouri, Wafa Ben Jaballah, Melek Önen, Md Masoom Rabbani, and Mauro Conti. 2021. FADIA: FAirness-Driven collaboratIve remote Attestation. In *Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '21), June 28–July 2, 2021, Abu Dhabi, United Arab Emirates.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3448300.3468284

WiSec '21, June 28–July 2, 2021, Abu Dhabi, United Arab Emirates

ACM ISBN 978-1-4503-8349-3/21/06…$15.00
https://doi.org/10.1145/3448300.3468284

1 INTRODUCTION

The evolution of the Internet of Things (IoT) technology leads to a tremendous increase in the deployment and use of IoT devices in the industries and factories [19]. The number of IoT devices is likely to grow to achieve 125 billion devices in 2030 as suggested by a study in [29]. These devices hold critical roles in production or operational lines and thus their use raises serious concerns with respect to the safety and the security of the system. They are usually designed or deployed so that they can connect to their manufacturers or simply to the Internet. An IoT device could be an entry point or attack vector for malicious hackers to explore [12, 19]. For example, the Mirai attack [12] infected insecure devices on a large scale and resulted on a massive distributed denial of service which hits the Internet access. Due to their low-cost design, IoT devices cannot support sophisticated security measures. This inherent weakness can be exploited by adversaries whose primary goal would be to target the compromise of IoT devices with little effort. One should therefore continuously verify the software integrity of IoT devices. Remote attestation (RA) helps achieve this goal by enabling a device acting as a prover to prove the integrity of its software configuration to a remote verifier. There exist many RA solutions in the literature [2, 25–27]. While existing solutions have improved a lot in terms of scalability, security and robustness, they unfortunately fall short in addressing the heterogeneous aspects of the network. Indeed, IoT devices deployed in the same network may feature different configurations. Furthermore, based on their role and/or position in the network/system they may also differ in terms of battery and computation level. Another problem that is untackled in the existing literature is the increasing size of the network (i.e., devices are added to the network gradually over time). Such problem raises challenges in terms of key management since new keys need to be distributed to existing devices in the network to enable the device-to-device communication. This often turns to be impractical and does not scale well.

To address these challenges, we propose a fairness-driven approach whereby RA tasks/operations are not equally distributed to all devices in the network. Instead, their capabilities in terms of computation and energy are taken into account and devices with more power perform more operations related to the RA compared to more lightweight devices. With this goal, we introduce FADIA, a lightweight collaborative RA protocol whereby all devices participate to the protocol by computing their own attestation and also forwarding others' ones. The protocol uses a tree-based architecture where each node in the tree aggregates its attestation with the ones received by its children nodes and forwards the result to its parent node. The number of children of a given node and its

Table 1: Comparison btween previous work on collaborative remote attestation and FADIA. The big O notation represents the complexity of the cost on a prover with respect to n (number of provers in the network).

	Scalability		*Supported Features*				
	Communication Cost	Storage cost	Mobility	Detects hw attacks	Autonomous	Adding Devices	Fairness
SEDA [5]	$O(1)$	$O(1)$	○	○	○	○	○
SENA [2]	$O(1)$	$O(1)$	○	○	○	○	○
LISA$_\alpha$ [7]	$O(\log n)$	$O(1)$	○	○	○	○	○
LISA$_s$ [7]	$O(1)$	$O(1)$	○	○	○	○	○
DARPA [20]	$O(n)$	$O(n)$	○	●	○	○	○
SALAD [26]	$O(n)$	$O(n)$	●	○	○	○	○
PADS [3]	$O(n)$	$O(n)$	●	○	●	○	○
PASTA [27]	$O(n)$	$O(n)$	●	●	●	○	○
FADIA	$O(1)$	$O(1)$	●	●	○	●	●

role in the tree is determined based on its capabilities (ex. hardware specifications) and current capacity (ex. battery level). With FADIA, we show that by tuning these two parameters, we can achieve a more optimized runtime of the RA protocol and a longer lifetime in sensor networks (i.e., time to the first sensor node failure). FADIA is also lightweight since it uses symmetric message authentication codes and integrates the efficient key management scheme proposed by Eschenauer and Gligor [17]. Thanks to this scheme each node probabilistically shares a symmetric key with other nodes in the network and can therefore authenticate each other without the need for a key redistribution on each device addition.

Our contributions can be summarized as follows:

- We study the heterogeneity problem in RA protocols and we present a solution that integrates fairness by design. We show by experiments that fairness can provide a better performance for a RA protocol;
- We present FADIA a RA protocol that supports efficient addition of devices at runtime, without extra overhead, but only with a limited sacrifice of the connectivity of the devices;
- We implement our solution on heterogeneous embedded systems, namely Rasberry Pi 2 devices and Tmote Sky sensor nodes;
- We evaluate the performance of our solution in heterogeneous and homogeneous networks. We compare the results of our solution to the other best existing protocols [26, 27] in the same network scenarios. The results show that our solution outperforms the previous solutions in all aspects.

The rest of the paper is organized as follows. In § 2 we go over the previous work and compare them to FADIA. We then explain the problem scope in § 3. We give a background about Eschenauer and Gligor which is required for a better understanding of FADIA in § 4. We present the threat model in § 5. We give a high level description of our approach in § 6, and a detailed description of its design in § 7. We analyze the security of FADIA in § 8 and evaluate it in § 9. Finally, we conclude our work in § 10.

2 RELATED WORK

Remote Attestation. Remote Attestation (RA) schemes are protocols that enable a device to prove the integrity of its software to another remote device [22, 24]. It involves two roles: the *prover* (the device that proves its software integrity) and the *verifier* (the device that verifies the integrity of the prover's software). Lately, to adhere to the increasing number of embedded devices in the network, collaborative RA protocols are proposed. Some are designed for static networks such as SANA [2], LISA [7], SEDA [5], and SHeLA [32]. These approaches run on devices deployed in a static tree topology. Another line of research propose RA schemes that share a design of a gossip-based mechanism and run in dynamic networks. Examples of such an approach are DARPA [20], PADS [3], and SALAD [26]. Gossip-based protocols suffer from high bandwidth overhead and long runtime. An alternative approach are remote attestation schemes for IoT services that work on a publish/subscribe paradigm based on asynchronous communication pattern, for example SARA [15]. Nonetheless, this protocol is not applicable for large networks. Recently, Kohnhauser et al. proposed PASTA [27], an autonomous RA protocol in which provers create multiple spanning trees in the network and generate the so-called tokens to attest the provers participating in the tree. Each token embeds the proofs of all the nodes in the tree. Later, these tokens are distributed among all the provers in a gossip-like approach. The parallelized tree generation process in PASTA allows a relatively fast proof aggregation between the provers. Also, the token distribution offers autonomy to the RA protocol. However, PASTA does not scale well in terms of communication due to the flooding process it requires. Also, PASTA requires provers with high storage and computation capabilities. This is because it uses asymmetric cryptography for the token generation. Compared to all the other approaches, FADIA the first solution to introduce the concept of fairness to remote attestation protocols targeting heterogeneous IoT networks. Remote attestation tasks/operations are distributed among nodes in a fairly manner. Furthermore, we also show that FADIA shows better performance when the network is homogeneous with respect to the previous approaches (see § 9). Finally, the cost of the addition of new provers to the network is also optimized thanks to the use of the efficient key management scheme in [17]. We overview and compare existing solutions and FADIA in Table 1 with respect to different features. In terms of communication and computation scalability, we observe that for FADIA these costs do not depend on the number of provers in the network. FADIA, similar to PASTA, supports mobile devices and detects tampering attacks. The only feature that FADIA does not support is autonomy.

Key Management for IoT devices. In RA protocols, two types of secret keys are defined: the *communication keys* which are used to establish a secure channel between provers and the *attestation keys* which are used for attestation, i.e., to generate a proof of the integrity of a prover's software. A RA protocol may define a unique key shared among all provers for both purposes (e.g., PADS [3]). Such an approach enables the efficient addition of new devices to the network at runtime. Nevertheless, solutions become inefficient if a single node is compromised (as all nodes need to update their keying material). On the other hand, some RA protocols [20, 26, 27] distribute pairwise unique keys for each couple of provers and a unique attestation key for each prover. This approach increases the security of the protocol and enables an efficient revocation mechanism. However, the addition of new devices to the network

becomes costly since one additional key needs to be distributed to each existing prover. FADIA implements the symmetric key management scheme proposed in [17] in order to optimize the cost incurred by the addition and revocation of nodes. In FADIA, each node is pre-loaded a keyring (a set of communication keys) randomly selected from a key pool. Devices which share at least one key in their keyrings can directly establish a secure communication channel. Similar to [20, 26, 27], FADIA defines one attestation key per prover.

3 PROBLEM SCOPE

Large-Scale analysis has been performed on embedded devices [4] and has shown that a large number of them are vulnerable to unknown security bugs. This makes attestation of the software configurations of IoT devices a necessity as a defence-in-depth mechanism against malware infections. The relevance of RA on embedded devices has been studied lately and many solutions are proposed accordingly, addressing scalability, security and robustness features. Unfortunately, these solutions may become inefficient and sometimes even impractical for some applications of IoT networks. This is mainly because the current state of the art solutions disregard two common characteristics of IoT networks: (i) the heterogeneity of IoT network and (ii) the increasing size of the network (i.e., devices are added to the network gradually over time).

Heterogeneity of devices: Large-scale RA protocols involve collaborative tasks across devices. These tasks include generating and forwarding attestations. Existing solutions perform the distribution of this load (i.e., RA tasks) randomly or uniformly. This may result in a significant performance decrease in heterogeneous networks. We define heterogeneity in the IoT network as the diversity in the (hardware and/or software) characteristics of the IoT devices. For instance in Industry 4.0 IoT applications [34], sensor devices in the network (ex. Tmote Sky [11]) have low computational capabilities Microcontroller Units (MCU) compared to a Raspberry Pi operating in robots. The quality of the MCU affects directly the speed of processing the attestation messages from peer devices. Therefore, a RA collaborative protocol that does not consider this gap in the hardware capabilities between devices will end up putting either equal attestation load on different devices or higher load on less capable devices. The number of proofs (i.e., attestations) that a node can receive and forward should thus be depending on its capabilities. The heterogeneity of the network can also threaten the lifetime of the services. Running a RA protocol on a device consumes a significant amount of its battery due to the frequent participation in transmission and reception of attestation messages. Thus, if sensors with lower battery levels engaged in many energy-consuming operations, this can end up with battery depletion of some devices causing potential disruption in the service. To this end, we see heterogeneity as a problem that can have a strong impact on both the performance and lifetime of a collaborative RA protocol.

Dynamic nature of IoT networks: With the continous advancement of IoT applications, IoT networks gradually increase in size (i.e., new devices are added). Existing devices in the network know little about the new device. This leaves a problem for RA since it requires the device's pre-knowledge of shared key materials to secure the communication while running the protocol. A typical solution is to use a centralized server that manages the distribution of keys at run-time. However, this solution lacks scalability and adds a high overhead to the runtime of the RA protocol. Based on that, we identify the need for a dynamic management and distribution of the key materials as a missing requirement for a practical remote attestation protocol.

4 ESCHENAUER AND GLIGOR'S SCHEME [17]

We present a background knowledge of E-G's scheme that is necessary for understanding the rest of the paper. E-G's key distribution scheme follows a probabilistic approach to efficiently distribute the keys over a large number of devices. It facilitates the addition and the revocation of nodes (and the corresponding keys) in the network without substantial computation and communication overhead on the end devices. The scheme defines a main key pool which the participating devices pick a key ring from. Devices having at least one shared key from their key rings can communicate securely. This scheme has been shown to be simple and highly scalable and is therefore, suitable for resource-constrained devices. FADIA utilizes E-G's scheme to distribute the keys to the provers. Only provers who share common keys can establish secure communication channels.

Connectivity Analysis. FADIA's connectivity is defined as the average percentage of provers a prover can connect to (i.e., communicate with). Since FADIA's key-distribution is based on E-G's scheme, the connectivity property translates to the probability of two provers sharing at least one key in their key rings (P_s).

$$P_s = 1 - \frac{((p-r)!)^2}{(p-2r)!p!} \tag{1}$$

For example, with a key ring of size $r = 300$ and a key pool of size $p = 100000$ we obtain a connectivity of $P_s \approx 0.6$.

5 ASSUMPTION AND THREAT MODEL

In this section, we describe the assumptions on the network. Then we define our security model.

5.1 Network Assumptions

We consider a mesh network topology where devices acting as provers, communicate within their communication range. Additionally, all provers are connected to a more powerful device (the controller C) acting as the verifier. For example, this can be the edge router. The network may contain more than one controller such that these controllers share their information and synchronize their data on a different layer. For the sake of simplicity, in this paper, we consider a single controller that connects to all provers in the network. Both, the provers and the verifier are managed by the network operator O. The participating provers can have heterogeneous characteristics. Moreover, they can be static or mobile within the network. New provers may be added to the network at any point in time.

5.2 Security Model

Security assumptions: We assume that provers have the minimal secure hardware features to perform RA [18]. Additionally, provers

are equipped with loosly synchronized real-time clocks. The minimal secure hardware can be implemented using a secure Read-Only-Memory (ROM) to store the keys and a Memory Protection Unit (MPU) that stores FADIA's attestation code. The aforementioned execution space on each prover is referenced as the Trusted Anchor ($\mathcal{T}_{\mathcal{A}}$). Additionally, we assume that the controller is not compromised and fully trusted. Furthermore, similar to previous research, FADIA only considers invasive and semi-invasive physical attacks. Thus, non-invasive attacks such as side-channel attacks are out of the scope of this paper. In this context, we rely on a common assumption that for an attacker to successfully bypass the $\mathcal{T}_{\mathcal{A}}$, it needs to take the devices offline for more than a δ_h time which is predefined and known to be non-negligible [9, 10]. This is because such attacks require expensive and complex laboratory equipment and requires the full possession of the target for a significant amount of time (from hours to weeks)[35, 36]. This becomes even more expensive especially when devices are equipped with tamper-resistant mechanisms [21, 30, 33]. We finally assume that the implementation of FADIA and its cryptographic components does not contain any security bugs.

Adversarial Model: The main objective of an Adversary is to perform malicious activities by corrupting the memory of a prover and also damaging the network communication while being undetected. We consider two types of adversaries, Software Adversary $\mathcal{A}_s$ and Hardware Adversary $\mathcal{A}_h$. $\mathcal{A}_s$ has full control of the execution of a prover apart from the $\mathcal{T}_{\mathcal{A}}$. It also has full access to the prover's memory except the memory protected by the MPU. Thus, $\mathcal{A}_s$ can launch attacks like spoofing attack, Man in the middle attacks, replay attacks. In addition to $\mathcal{A}_s$ capabilities, $\mathcal{A}_h$ has physical access to the devices in the network. This provides him/her with the ability to leak any secret or modify FADIA's code on the targeted prover. However, this is only possible after turning the prover off for more than δ_h time, as stated previously. δ_h is defined by C for all provers participating in the protocol. FADIA is considered secure if an adversary (under the aforementioned assumptions) cannot forge a "healthy" state for a compromised prover. Inline with other RA schemes [2, 3, 5, 27] we keep Distributed Denial of Service (DDoS) attack out of our current context. Nevertheless, in section 8 we mention possible ways to detect DDoS attack in FADIA.

6 OUR APPROACH

We present our approach solving the problems mentioned in § 3 (namely, heterogeneity and device addition). We design a lightweight collaborative RA protocol (FADIA). To solve the heterogeneity problem, FADIA is designed with fairness in mind. In a collaborative RA protocol, we define fairness as the ability to distribute the load of the protocol according to the capabilities of the provers. The goal is to increase the performance of the protocol and to reach a better lifetime for the network. In a fair RA protocol, provers in FADIA will be assigned a score depending on their capabilities and behave accordingly. This score will be frequently computed and the protocol should adapt to any change. In FADIA, similar to PASTA [27], a group of provers collaborate and create a spanning tree in which parent nodes collect attestations from children nodes. However, in contrast to PASTA, the choice of the position and the number of children of a prover in the tree are adaptively regulated.

Table 2: Notations

Entities	
$\mathcal{P}_i, C, O$	A prover of index i, Collector, and Network Operator
$\mathcal{A}_h, \mathcal{A}_s$	Hardware adversary and software adversary
Parameters	
uid	Unique id of a prover
$cntr$	Counter for the number of attestation of a prover
α_g	Max size of the set of uids in an attestation message.
δ_h	Minimum time required by $\mathcal{A}_h$ to compromise a prover
δ_c	Time a prover waits to receive *invite* before it calls *generateTree*()
c_{max}	Maximum number of children for a node in a tree
c_{limit}	Maximum number of children a prover can accept in a tree
K_{ic}	Secret key shared between $\mathcal{P}_i$ and the controller
K_s	Secret key shared between C and O
K_{ij}^{ID} , K_{ij}	Secret key (and it corresponding key id) shared between $\mathcal{P}_i$ and $\mathcal{P}_j$
sch	The hash of the software configuration on a prover

These are determined by the scoring function which is computed based on the hardware capabilities (e.g., CPU) of the prover and its current residual battery. The *score* function outputs a score value between 0 and 1. When the score is closer to 1, the prover can assign more tasks with respect to the RA protocol (for example, the node can have a higher number of children). To cope with the dynamic nature of the network, the protocol should support the addition of new devices at runtime. As mentioned previously, involving a central key server for key distribution at runtime results in a significant overhead. Instead, we propose to rely on Eschenauer-Gligor's (E-G) scheme [17] for the distribution of *communication keys*. Thanks to this scheme, FADIA easily addresses the trade-off between the connectivity of the provers and the security of the communication. Furthermore, thanks to [17], the addition of a new prover to the protocol does not require any modification at the other provers. Additionally, each prover is assigned a unique *attestation key*. The proofs are aggregated by the provers on the way towards the initiator.

7 DESIGN OF FADIA

In this section, we describe the design of FADIA. The protocol is composed of four different phases: the *initialization*, *joining*, *attestation* and *revocation* phases. The first *initialization* phase consists of an offline setup phase where the keying material are installed at all involved provers. During the *joining* phase, a prover identifies itself to the controller. The prover further starts the attestation phase. During the *attestation* phase (which is the core phase of FADIA), the active prover periodically participates in virtual attestation trees to send its attestation report and forward others'. The active prover keeps running this phase until it is dropped from the network (either intentionally because it is detected as malicious or incidentally because it left the network). On such an event, the *revocation* phase starts whereby the dropped prover becomes offline and its keys are revoked. In the following, we describe each phase in details.

7.1 Initialization Phase

Provers, before participating in FADIA, are considered offline. In order for a prover $\mathcal{P}_i$ to enter the protocol, it has first to be set by the network operator O. O defines a key pool ($O.Pool$) according to [17], (see § 4). This key pool is stored in a safe location (offline).

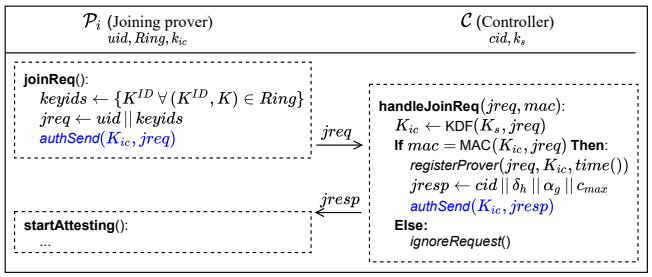


Figure 1: FADIA's joining phase

The key pool contains p symmetric keys together with their key ids $\{(K^{ID}, K)\}$. $\mathcal{P}_i$ randomly receives a key ring of size r from the key pool:

$$\mathcal{P}_i.Ring \leftarrow \{(K^{ID}, K)\} \subset \mathcal{O}.Pool$$

$\mathcal{O}$ also assigns a unique id (uid) for $\mathcal{P}_i$ and a cryptographic hash of the current software configuration (sch). $\mathcal{P}_i$ then obtains a unique key K_{ic} (the *attestation key*) derived from the chosen keyring ids and the prover's unique id. This key is computed by $\mathcal{O}$ using a key derivation function (KDF) and a secret key (K_s) known only by $\mathcal{O}$ and the controller. More formally,

$$K_{ic} \leftarrow KDF(K_s, \{K^{ID} \forall (K^{ID}, K) \in \mathcal{P}_i.Ring\} \cup \{uid\}) \quad (2)$$

K_{ic} is used in the later phases of the protocol to provide secure communication between a $\mathcal{P}_i$ and the controllers. Moreover, $\mathcal{O}$ also defines a function $score()$ which evaluates at the runtime the required load $\mathcal{P}_i$ should take based on its hardware capabilities and its current capacity. This function outputs a value between 0 and 1 that indicates the amount of load $\mathcal{P}_i$ can take (0 indicating that this device should take the least load possible, and 1 indicating that it should take the maximum load that can be given to one device). The implementation of $score()$ depends on the underlying environment and application. For example, in the case of a network of wireless devices with batteries, the function $score()$ will evaluate the battery percentage level of a prover; For a network of devices with microcontrollers of different computational speed, $score()$ will categorize different types of microcontrollers into different classes, and output a higher value for more powerful classes.

7.2 Joining Phase

Once $\mathcal{P}_i$ is initialized, it can join the network. $\mathcal{P}_i$ sends a join request message ($jreq$) to the controller (C) in the network. The join message contains its unique ID (uid) and the set of key ids in its key ring. This message is authenticated using a message authentication code (MAC) computed with K_{ic}. Based on the received uid and the key ids, C computes K_{ic} (see equation 2) and authenticates $\mathcal{P}_i$. Upon validation, C registers $\mathcal{P}_i$ in the table of provers currently participating in the protocol. The table of registered provers records the current state of each prover (being either *healthy* or *unhealthy*) along with the last time the prover attested. $\mathcal{P}_i$ is first registered as "healthy". C sends back a response message $jresp$ (authenticated using K_{ic}) to confirm the joining process. $jresp$ includes (i) cid: the controller's id (ii) δ_h which corresponds to the maximum time a prover can stay active without attesting in the network, (iii) α_g which defines the maximum number of attestations that can be aggregated, and (iv) c_{max} that is the maximum number of children a prover can have. After $\mathcal{P}_i$ is registered, C regularly checks its status and if $\mathcal{P}_i$ does not attest in δ_h time, its status becomes "unhealthy". Figure 1 provides the specification of the joining phase.

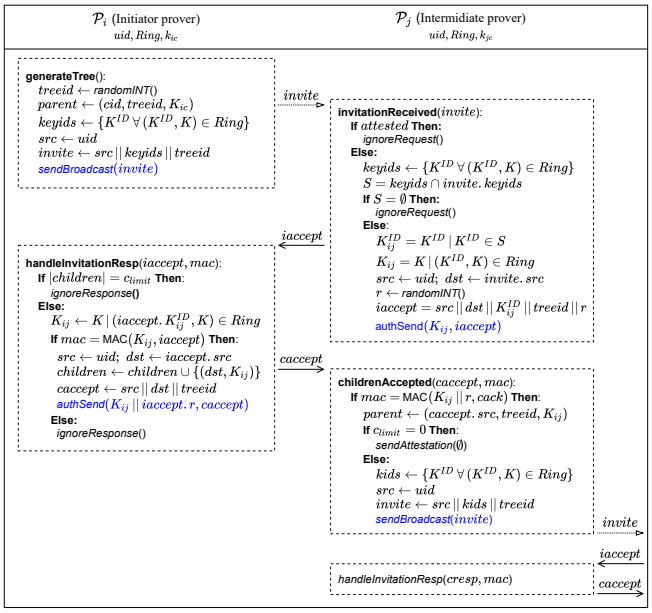


Figure 2: Tree construction in FADIA's attestation phase.

7.3 Attestation Phase

Provers start running this phase immediately after completing the joining process. $\mathcal{P}_i$ enters a new attestation period every $\delta_h/2$ time. ($\delta_h/2$ is chosen to guarantee that two consequent attestations are always received within no longer than δ_h time.) At each attestation period a $\mathcal{P}_i$ starts by performing an integrity check on its software configuration. The check is performed against the software configuration hash (sch) stored in the prover. The method of this check and the format of sch is out of the scope of this paper. This can be a simple technique based on computing the hash of the firmware or more complex techniques such as the ones proposed in [1, 8, 14, 38]. If the integrity check fails, then $\mathcal{P}_i$ quits the attestation phase and will be eventually dropped from the protocol. After succeeding the integrity check, $\mathcal{P}_i$ should particiapte in the construction of a tree in which it will attest in. First, $\mathcal{P}_i$ generates a unique proof of its attestation using the function $generateProof()$. It further evaluates $score()$ and uses the result to decide on its role in an attestation tree. More specifically, $\mathcal{P}_i$ updates two parameters namely, c_{limit} and δ_c:

- c_{limit} is the maximum number of children $\mathcal{P}_i$ can accept in the upcoming attestation tree.

$$c_{limit} \leftarrow score() \times c_{max}$$

- δ_c (which is less than $\delta_h/2$) is the amount of time $\mathcal{P}_i$ waits to receive a participation invitation to an attestation tree. When δ_c is reached and $\mathcal{P}_i$ did not receive any invitation, it starts the tree construction protocol.

$$\delta_c \leftarrow score() \times \delta_h/2$$

Neighboring provers that share common keys and are ready to provide their attestations construct a tree. Note that if a prover does not share any key with its neighbors, it directly sends the attestation to the controller. However, such cases appear with a low probability in realistic networks. For example, if the average number of neighbors for a prover is 5, and the connectivity $P_s = 0.6$, then the probability of a prover being isolated is $(1 - P_s)^5 \approx 0.01$. The tree construction and the attestations collection processes are described next.

7.3.1 Tree Construction. A tree construction starts by an initiator prover ($\mathcal{P}_i$) after waiting for δ_c time. The latter broadcasts an invitation message (*invite*) to its neighbors. The invitation message includes the unique id (*uid*) of $\mathcal{P}_i$ as well as a tree id (*treeid*) (generated from a timestamp to guarantee freshness) and the set of key ids which $\mathcal{P}_i$ holds in its keyring. When $\mathcal{P}_j$ receives *invite*, it first checks if it did not attest in the last $\delta_h/2$ time. Then it checks if it shares at least one key with $\mathcal{P}_i$. Let K_{ij} denote the shared key between $\mathcal{P}_i$ and $\mathcal{P}_j$, and K_{ij}^{ID} be its corresponding key id. $\mathcal{P}_j$ responds with an invitation acceptance message (*iaccept*) containing the key id K_{ij}^{ID} . The message also includes the unique ids of the source and the destination, the tree id sent by $\mathcal{P}_i$, and a random value (r) and a Message Authentication Code (MAC) computed using the shared key K_{ij} . $\mathcal{P}_i$ accepts $\mathcal{P}_j$ as a child only if it has not acquired c_{limit} children yet and responds with a child acceptance message *caccept* authenticated with the shared key concatenated with the random value (r). To this end, both provers established a secure channel with the shared key K_{ij} . Next, $\mathcal{P}_j$ either extends the tree and thus acts as an intermediate prover, or it finishes the tree construction process thus acts as a leaf prover. A prover acts as a leaf prover in two cases: either it does not accept any children (i.e., $c_{limit} = 0$) or its *invite* message is timed out without receiving any response from its neighbors. The details of the tree construction messages are shown in Figure 2. The secure channels established between parent nodes and children nodes in the tree are used next to transmit the attestation messages.

Algorithm 1: Aggregation of two attestations.

$a_x = \{proof_a : uids_a, proof_b : uids_b, ...\}$;
$uids_x \subset \{uid_1, uid_2, ..., uid_n\}$;
Algorithm aggregateAttestation(a_1, a_2)
  $res \leftarrow a_1$
  for $proof, uids$ ***in*** a_2 **do**
    if $|uids| = \alpha_g$ **then**
      $a_2 \leftarrow a_2 \setminus \{proof : uids\}$
      $res \leftarrow res \cup \{proof : uids\}$
    else
      for $proof', uids'$ ***in*** res **do**
        if $|uids| + |uids'| < \alpha_g$ **then**
          $a_2 \leftarrow a_2 \setminus \{proof : uids\}$
          $res \leftarrow res \setminus \{proof' : uids'\}$
          $res \leftarrow res \cup \{proof \oplus proof' : uids \cup uids'\}$
          break
  for $proof, uids$ ***in*** a_2 **do**
    $res \leftarrow res \cup \{proof : uids\}$

7.3.2 Attestations collection. The leaf nodes send attestation messages (*attst*) to their parents. The attestation message of $\mathcal{P}_i$ contains the generated proof which is computed as follows:

$$generateProof() : proof \leftarrow MAC(K_{ic}, uid \,||\, cntr \,||\, treeid) \quad (3)$$

where *cntr* is a counter that is incremented each time $\mathcal{P}_i$ attests. Parent nodes aggregate the attestation messages from their children using the function *aggregateAttestation*(). Similarly, the new attestation message is sent to the parent and processed. This is repeated until the initiator prover receives all the aggregated attestation messages. It then sends the final attestation message to the controller. An attestation message is composed of multiple sets of prover ids. The number of provers in a set is controlled by the parameter α_g. Each set is linked with a single proof which is the XOR of all proofs provided by the provers in that set. The algorithm that describes *aggregateAttestation*() is depicted in Algorithm 1.

The granularity of the aggregation of the attestation is parameterized by α_g. If α_g is 0, then none of the proofs are XORed, and thus, all individual proofs are transmitted to the controller. If α_g larger than the number of provers participating in the tree, all proofs are XORed forming a single proof for all provers.

When C receives the aggregated attestation it validates the proofs. For each set, it computes the MAC according to equation 3 using the unique keys (K_{jc}) of each prover $\mathcal{P}_j$. For each verified aggregated proof, the status of all provers in the set is updated. More specifically, C updates the time of the last proof received from these provers by the current time. We show the attestations collection details in Figure 3.

7.4 Key Revocation Phase

When prover $\mathcal{P}_i$ becomes "unhealthy", it cannot be trusted anymore. So it is required by C to revoke all the shared keys between $\mathcal{P}_i$ and the other provers. Since C knows the ids of all the keys in the keyrings of all provers, it can find out the affected devices (i.e., the ones which share at least a key with the "unhealthy" device). C sends a revocation message (*revk*) to each of them. The message contains the *uid* of the receiving prover, and the set of key ids that should be revoked. *revk* is authenticated with a MAC using K_{ic} of the corresponding prover.

$$revk \leftarrow cid \,||\, uid \,||\, \text{affectedKeys}() \quad (4)$$

When a prover receives *revk*, it removes the affected keys from its keyring. When the size of its keyring goes under a certain threshold θ, the prover goes to the offline state and requires reinitializing to go back online.

7.5 Role of the *score* function

The fairness by design approach can be achieved thanks to the tuning of mainly two parameters set with the help of the *score* function: c_{limit} and δ_c. These help configure the behavior and the load of each prover during the attestation phase and their correct setting hence ensure a fair distribution of the load caused by FADIA on the active devices.

The first parameter c_{limit} represents the number of children a prover can hold during one attestation round in the virtual attestation tree. The number of children has a direct impact on the amount of load put on the prover. The load involves the use of cryptographic tools (MACs) to establish a secure channel with each child and forwarding the attestation proofs.

The second parameter δ_c is the time which a prover waits for to receive the invitation message (*invite*) before it decides to start

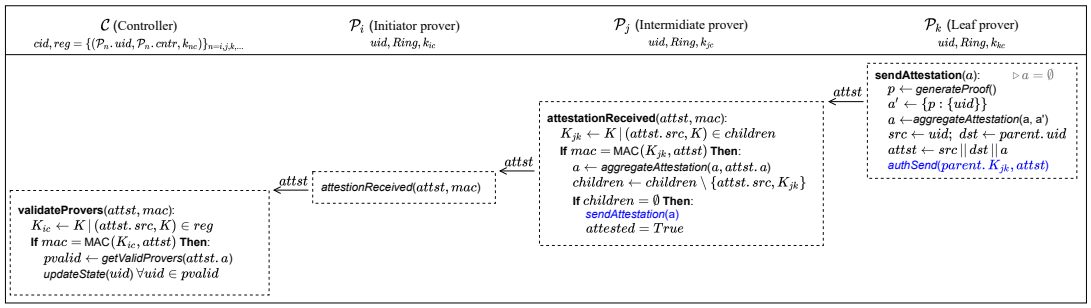


Figure 3: Attestation collection in FADIA's attestation phase.

its own tree and sends *invite* itself. It is important to mention here that when a prover finishes an attestation round, it can switch to a sleeping state since it has completed its attestation for this round. Reasearches on wireless ad-hoc networks have studied the scheduling of the nodes sleeping state to optimize the network liftime [28]. Inlighted by these studies we control the sleeping time of a prover based on a fair policy. In FADIA for each attestation round, a prover is first being actively waiting for invitation messages, then performs the requested attestation operation in the tree and finally switches to a sleep state until the next round. Consequently, δ_c is a critical parameter that influences directly the average amount of time a prover spends in the sleep state and hence optimizes its resource consumption. Research has shown that being in an active state (listening or sending) can be intensively resource-consuming compared to being in a sleep state [23]. Therefore, δ_c parameter is also used to control the amount of load on a prover in FADIA.

In FADIA, these two parameters are calculated at runtime by each prover. Their corresponding values are updated at each round of the attestation phase using the *score* function. We present examples of how to build this function depending on the scenario of deployment in section § 9.2.1.

8 SECURITY ANALYSIS

The main goal of an adversary is to perform malicious activities and evade detection. However, a remote attestation scheme should identify the presence of malicious actors in the network to safeguard the network operation. We consider the system secure if an attacker cannot forge a "healthy" state for a "non-healthy" prover. Remainder of this section informally discuss the security of FADIA w.r.t. adversarial assumptions mentioned in 5.

Attacks Performed by $\mathcal{A}_s$:

- Spoofing attacks: FADIA is immune to attackers trying to spoof a prover's identity. Since all message exchanges are protected by keys stored in an inaccessible location for software attacker and can only be accessed through $\mathcal{T}_{\mathcal{A}}$, an attacker will not be able to produce authenticated messages without the keys from the key ring. Please note that the *invite* message is an exception, as this message is not authenticated. However, this does not affect the security of FADIA since attackers spoofing this message will not be able to complete the 3-way handshake, (i.e., respond with a valid *caccept*).
- MITM attacks: FADIA will identify this attack as it suffers the same limitations of spoofing attacks. $\mathcal{A}_s$ using this attack technique will not be able to manipulate messages without being detected since the integrity of these messages is ensured using the keys from the key rings that is protected by $\mathcal{T}_{\mathcal{A}}$.
- Replay attacks: FADIA prevents replay attacks since all messages are unique and cannot be used twice without being detected. Specifically, freshness is guaranteed in *invite* messages thanks to using a timestamp in the *treeid*. Similarly, *iaccept* and *caccept* includes randomness for each prover-to-prover communication. Further, *attst* messages are also resilient to replay attacks since they contain a counter which is incremented at each attestation round.
- DoS Attacks: Although FADIA does not include these types of attacks in its threat model, it is worth noting that FADIA can detect the effect of such attacks. This is because $\mathcal{A}_s$ performing DoS attacks on a prover (or prover group) will prevent the attestation of these provers. The controller will therefore, inevitably discover a missing attestation from the provers under attack.

Attacks Performed by $\mathcal{A}_h$. In addition to the software adversaries, hardware adversaries have the ability to tamper with a device to extract the keys from its keyring or alter the attestation code. Under the assumption that a hardware adversary needs to take the system offline for at least a δ_h duration, FADIA provides resilience against these attackers. This is because FADIA requires that within every δ_h period, each prover provides evidence of its "healthiness". This helps the network owner to ensure that the provers are not taken offline and thus not corrupted. On the other hand, a hardware adversary may use leaked keys to attack other provers sharing the same key. However, the key revocation process ensures that if a prover no longer participates in the protocol, these keys are revoked. Additionally, even if an attacker leaked all the key materials from a prover, the attacker will not be able to forge a legitimate proof as this forgery involves the targeted prover's K_{ic}. The state of a "unhealthy" device will thus not be forged by $\mathcal{A}_h$.

9 IMPLEMENTATION AND EVALUATION OF FADIA

In this section, we evaluate the performance of FADIA in a heterogeneous network and demonstrate the advantages originating from

Table 3: Benchmarks and energy consumption measurements of cryptographic functionalities of FADIA when implemented on Tmote Sky and Raspberry PI 2 devices.

		HMAC-SHA256		SHA256	
		Time (ms)	Energy (mJ)	Time (ms)	Energy (mJ)
PI 2	32 B	0.068	-	0.025	-
	4 KB	1.075	-	1.049	-
	8 KB	2.083	-	2.032	-
	32 KB	8.131	-	8.079	-
SKY	32 B	63.28	0.3384	15.54	0.0862
	4 KB	1035	5.5908	988	5.3388
	8 KB	1998	10.7892	1960	10.6128

its fairness-driven approach by evaluating the energy consumption, the computational cost and the bandwidth. We further study its performance in a homogeneous network and show that even in this case, FADIA outperforms the relevant state-of-the-art solutions, namely PASTA [27] and SALAD [26].

9.1 Implementation of FADIA on Tmote Sky and Raspberry PI 2

To illustrate heterogeneity, we implement FADIA on two types of devices: Tmote Sky [11] and Raspberry PI 2. The Tmote Sky which represents a resource-constrained device, is equipped with an i16-bits 8 MHz msp430 MCU, 10 KB of RAM, and 48 KB of non-volatile memory. On the other hand, the Raspberry PI 2 is more powerful as it is equipped with a 900MHz quad-core ARM Cortex-A7 CPU, 1 GB of RAM, and 32 GB of non-volatile memory. Both types of devices are equipped with CC2420 RF transceivers. The CC2420 chip operates on 2.4 GHz and is compliant with IEEE 802.15.4 standards. We do not follow a certain security architecture for implementing FADIA on the devices. However, FADIA can be implemented based on any security architecture. Previous research has shown that achieving a security architecture on sensor devices is indeed possible [6, 13, 16, 31]. In our implementation, and without loss of generality, an ARM TEE can be used for the Raspberry PI, and TYTAN [6] can be used for the Tmote Sky. We use HMAC-SHA256 for authenticating the messages and creating the proofs. We use the SHA256 of the device's firmware as the software configuration hash (*sch*).

In order to conduct our study, we first evaluate the cost of one attestation round for a prover in terms of the execution time and the energy consumption of FADIA. Since one round mainly involves MAC and hash computation operations, we have obtained some benchmarks on HMAC-SHA256 and SHA-256 respectively. Results are shown in Table 3. As expected, Tmote Sky takes more time to generate attestation proofs. We also benchmark the throughput and the round-trip time (RTT) of the CC2420 transceiver in a real environment. The throughput on the application layer is 25.2 Kbps and the RTT is 61.4 ms.

9.2 Evaluation

We first evaluate FADIA on a heterogeneous network to measure the influence of fairness on the performance in terms of energy consumption and computational cost. To evaluate the benefits of fairness, we consider two variants of FADIA. Variant (1) with fairness activated and variant (2) without fairness. Our results show

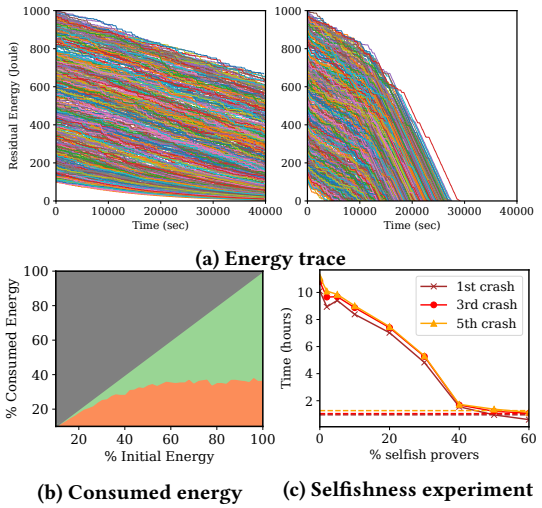


Figure 4: Evaluation of the energy consumption. (a) the residual energy over time. w/ (left side) and w/o (right side) activating fairness. (b) the average consumed energy w.r.t. initial energy after 16.6 **hours of running FADIA with fairness activated. The green-colored area represents the residual energy and the orange-colored area represents the consumed energy at the end of the experiment. (c) the time taken until 1st, 3rd, and 5th crash (energy depletion at a prover) with different percentage of selfish provers in the network. Dotted lines represent the results when fairness is not activated.**

that fairness improves the lifetime of the network and the runtime of the RA protocol. We then evaluate the performance of FADIA on a homogeneous network in terms of storage, computation and communication cost. Our results show that FADIA outperforms the state-of-the-art solutions. Additionally, we evaluate the robustness of FADIA against selfish provers. We also evaluate the efficiency of the revocation phase. Due to the lack of space, we present it in appendix C. We perform our simulations using the Omnet++ simulator [37]. We implement FADIA on the application layer of the devices. For the lower layers, we use a simplified medium access control version which makes sure that no device within the communication range transmits at the same time. Provers communicate in a half-duplex fashion and store messages in queues when the medium is busy.

9.2.1 *Evaluation in a heterogeneous network.*

Test Case 1: Optimizing the energy consumption. To measure the optimization of energy consumption, we simulate FADIA on 500 Tmote Sky sensors acting as provers. *sch* is set as the SHA256 hash of the firmware of size 30 KB. The network and cryptographic delays are set according to the values measured in Table 3. Additionally, we consider a simple energy consumption model: The model updates the current energy consumption based on the status of the transceiver and the microcontroller of the prover. The transceiver can either be transmitting, listening, or OFF. Similarly, the microcontroller can either be ON or idle. For each of the following statuses, the energy is computed according to the energy measurements shown in Table 4. The provers move in a random

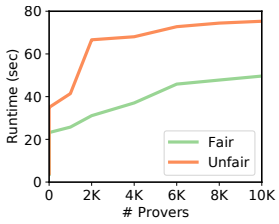


Figure 5: Runtime of FADIA with (in green) and without (in orange) fairness integrated in dynamic network.

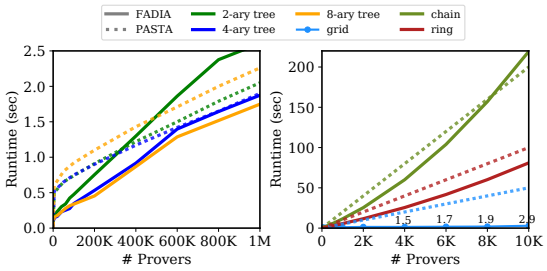


Figure 6: Runtime of FADIA and PASTA with different number of provers in tree, grid, chain, and ring topologies.

waypoint model at a linear speed uniform between $1mps$ and $2mps$ in a $300m \times 300m$ area. Each of the provers is equipped with a battery of $1000J$ max capacity. The initial energy level a prover starts with is chosen randomly (uniformly $[100J, 1000J]$). Evaluation of different random distribution functions for the initial energy are shown in appendix A. Note that the maximum capacity of an alkaline AA battery is around $13,000J$. But we use $1000J$ as the maximum capacity for the seek of the feasibility of the experiment. We run FADIA for 60,000 seconds. For variant (1) of FADIA we implement the $score()$ to return the current battery level of a device. Alternatively, for variant (2) $score()$ always retrurns 0.5.

We measure the consumption of energy of each prover with respect to time. Figure 4a shows the energy traces of the provers in both variants of FADIA. As expected, variant (2) of FADIA (i.e. fairness is not activated), shows a fast depletion of the energy of all the provers while for variant (1), most of the provers remain active after 40,000 seconds. The reason for the fast depletion of energy in the "unfair" protocol (i.e. variant 2) is that provers with low energy are treated indifferently from high energy provers. This leads to putting a significant load on these devices due to the high (i.e. unfair) number of children they need to collect attestations from. Moreover, since δ_c is not adapted to the energy level of the device, provers may spend more time waiting to be invited to a tree construction process. This keeps the transceivers of these devices in the listening state for a longer time instead of switching sooner to the OFF/idle state. This brings a serious problem since the part that mostly consumes energy in sensor devices is the transceiver. This is the case for Tmote Sky as shown in Table 4. Notice that, when provers with low battery crash, this decreases the number of provers in the network and causes fewer tree constructions to appear, causing a domino effect and faster depletion for other devices. Differently, in the case of FADIA variant (1), the low battery provers preserve their energy which prevents the early loss of provers.

We also look into the consumed energy of the provers at the end of the experiment (i.e., after 16.6 hrs). We group provers that had similar initial battery levels at initialization and we measure their average consumed energy level after the experiment. Figure 4b shows the relation between the consumed energy with respect to the initial energy. The graph shows that provers with high initial battery level (more than 50%) consumed more energy than provers with fewer initial battery level thus providing a longer lifetime to the network. Additionally, we measure the time taken until we detect 1st, 3rd and 5th crash of a prover (i.e., its energy is completely depleted). We observe that the lifetime of the provers in a fair protocol is an order of magnitude longer.

Test Case 2: Optimizing the runtime. In order to measure the optimization of the runtime achieved thanks to fairness, we evaluate the runtime of FADIA on a heterogeneous static tree topology. In this scenario, we use two types of devices such that 50% of provers are Tmote Sky (MSP430) devices and the other 50% are Raspberry PI 2 devices. Both types of devices use the same transceiver (CC2420). We use the throughput, network delay, and the cryptographic delays for each type of device according to the benchmarks measured in table 3. We measure the time taken from the start of the tree construction, until the final attestation is sent from the root node to the collector. The function $score()$ is defined such that it returns 0.05 for the MSP430 provers and 1 for Raspberry PI provers. Accordingly, the number of accepted children (c_{limit}) will be 1 for the less powerful provers and 20 for the powerful ones. On the contrary, we simulate FADIA variant (2) which assigns 10 children for each prover regardless of its type. Figure 5 shows the runtime results of both approaches with respect to a varying number of provers in the network. The results show that FADIA with fairness option can run 1.6 faster in static topologies.

9.2.2 *Evaluation in a homogeneous network.*

Memory consumption of FADIA. Each prover in FADIA stores one key ring. The storage consumption derived from the key ring is $(4B + 32B) \times R$, where 4 is the size of the key ring in number of keys. The prover also stores uid ($4B$), the controller key k_{ic} ($32B$), the attestation counter $cntr$ ($4B$), and other FADIA parameters ($20B$). The total memory consumption is $56 + 36 \times R$ Bytes. For example, with a key ring of size 300, this results in $10.6KB$ of memory. It is worth to notice that the memory consumption depends only on the key ring size and is independent from the number of provers on the network. This provides very high scalability compared to most of the state-of-the-art solutions that incurs a cost linear to the number of provers. Our experimental study shows that FADIA can run on 10,000 provers while each of them consuming only 10.6 KB which is significantly low compared to PASTA with $780KB$ of memory usage and SALAD with $365KB$.

Runtime of FADIA. To measure the runtime for FADIA, we have implemented the protocol in a static network defined under four common topologies: the tree, chain, ring and grid topologies. We consider the construction of single attestation tree where all provers participate to it. The running time is evaluated by measuring the time it takes until the report is collected by the controller. For a fair comparison with the state-of-the-art, the scenario, the types of the

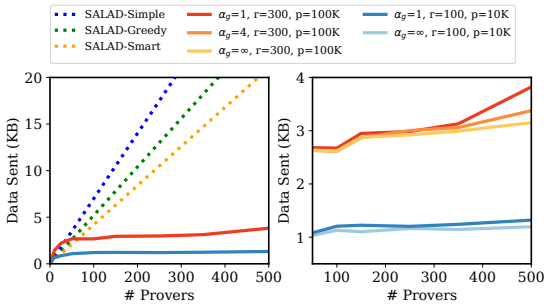


Figure 7: Average amount of data sent by a prover during a one round of attestation in FADIA and SALAD protocols.

devices, the network delays and the cryptographic benchmarks are all set as the ones used in the evaluation of PASTA protocol in [27]. We use ESP32-PICO-D4 devices in the simulation as provers and set the size of their firmware to $50KB$. The throughput of the provers on the application layer is 12.51 MB/s and the round trip time is 4.63 ms. The time a device takes to generate the proof is set according to Table 5 in the appendix. The provers perform 10 attestation rounds and the runtime of the attestation is averaged. The position of the nodes are randomized between rounds to force reinitialization of the tree costruction. We set the keyring size $r = 300$, and $\alpha_g = inf$. Figure 6 shows the runtime results of FADIA and PASTA.

We observe that FADIA shows a low runtime in tree topologies. It can attest $1,000,000$ provers in less than 2 seconds in a 4-ary tree. The runtime of FADIA and PASTA are close to each other in a tree topology since most of the attestation time corresponds to the network delays. Moreover, FADIA shows better performance at higher degree trees because messages are broadcasted during the construction of the attestation tree, whereas for PASTA, one-to-one tree commitment requests are sent from the prover to its neighbors. Additionally, FADIA is faster than PASTA by approximately 17 times for grid topologies and 1.3 times for ring topologies. In grid topologies, the tree construction results in an unbalanced tree. This creates a problem for PASTA since the tree construction happens in two steps. In the first step, all provers first commit to the tree. Then in the second step, all provers receive the aggregated commitment. This requires all provers to wait for the tree construction to finish before they start attesting. FADIA does not have this problem since the leaf nodes can immediatly send their attestations without waiting for the tree construction to finish. On the other hand, PASTA outperforms FADIA in chain topologies with a large number of provers (i.e., > 8000). This is explained by the fact that in FADIA, every prover sends all the key ids in the keyring to its neighbors which turns to be not effective in large chain topologies. Fortunately, such topologies do not often exist in real applications.

Bandwidth consumption of FADIA. We evaluate the bandwidth consumption of FADIA in a dynamic network. We consider devices moving in a random way point model (i.e., provers choose random destinations and move toward them) at a linear speed uniform between $1mps$ and $2mps$. The provers move in an area of $500m \times 500m$. We measure the average amount of data sent and received by a prover in an attestation round (i.e., for all provers to attest). Notice that the scenario, the device type, the network delays, and the cryptographic benchmarks are all set the same as the ones used in the evaluation of SALAD protocol in [26]. We use Stellaris LM4F120H5QR devices in the simulation as provers and set the size of their firmware to $30KB$. The throughput of the provers on the application layer is set to 35.0 kbps and the round trip time is set to $15ms$. The cryptographic delays are set according to Table 5 in the appendix. We choose the value of α_g as 1, 10, and infinity. We also use different values for the keyring and pool sizes: more specifically we set $r = 100, p = 10,000$ and $r = 300, p = 100,000$. Both cases give the same connectivity of the graph being $P_s = 0.6$ (see 4). Figure 7 shows the results. In particular, the results show that FADIA has highly scalable bandwidth consumption since the data consumption is nearly constant with respect to the number of provers. It also shows that the bandwidth consumption depends mostly on the size of the keyring. However, this cost always remains less than the average consumption of SALAD which increases linearly with the number of provers in the network.

9.2.3 Evaluation with selfish provers. We evaluate the impact of selfish provers on the lifetime of the network (time till 1st, 3rd and 5th crash of a prover). A selfish prover is a prover that does not will to participate in the tree generation process. It thus greedily attest indvidualy to the controller and goes to sleep state as early as possible. Note that such extreme selfish behavior can be easily detected. We consider this extreme case to evaluate the worst case scenario. A more careful selfish prover will still collaborate however less than it is supposed to. We consider the same energy consumption scenario in *Test Case 1* (see 9.2.1). However, selfish provers are chosen with initial battery level greater than $250J$. Figure 4c shows the results with different percentage of selfish provers. We observe that FADIA is robust against selfishness. This is because the lifetime drops significantly only when more than 40% of the provers are selfish. With high number of selfish provers the performance degrades since there is a sort of race toward the collector to provide attestation. This creates too many contention between provers accessing the wireless medium leading for the attestations to be delayed.

10 CONCLUSION AND FUTURE WORK

We propose FADIA, a lightweight collaborative attestation protocol that can be deployed on heterogeneous networks of IoT devices. FADIA is the first RA protocol that integrates fairness in its design. We show that fairness is an important feature for remote attestation protocols. Fairness can increase the performance of the protocol by a factor of 1.6 in a network where Tmote Sky sensors and Raspbery PIs coexsists. The lifetime of the network can increase by an order of magnitude, thus achieving less failures. We also show that FADIA outperforms the state-of-art-solutions in terms of scalability. The only drawback of FADIA is that it is not suited for autonomous networks.

In future work, we aim to improve FADIA to apply it on autonomous networks without affecting its scalability. We are also interested in investigating other possible parameters for the *score* function (e.g. the importance of a prover in the network, the bit-rate of a prover's transceiver, etc.). Also, we look further into devloping automatic methods to optimize the $score()$ function.

REFERENCES

[1] T. Abera, N. Asokan, L. Davi, J. E. Ekberg, T. Nyman, A. Paverd, A. R. Sadeghi, and G. Tsudik. 2016. C-FLAT: Control-Flow Attestation for Embedded Systems Software. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16)*. Association for Computing Machinery.

[2] M. Ambrosin, M. Conti, A. Ibrahim, G. Neven, A. R. Sadeghi, and M. Schunter. 2016. SANA: Secure and Scalable Aggregate Network Attestation. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16)*. Association for Computing Machinery.

[3] M. Ambrosin, M. Conti, R. Lazzeretti, M. M. Rabbani, and S. Ranise. 2018. PADS: Practical Attestation for Highly Dynamic Swarm Topologies. In *2018 International Workshop on Secure Internet of Things (SIoT)*.

[4] C. Andrei, Z. Jonas, F. Aurélien, and B. Davide. 2014. A Large-Scale Analysis of the Security of Embedded Firmwares. In *23rd USENIX Security Symposium (USENIX Security 14)*. USENIX Association.

[5] N. Asokan, F. Brasser, A. Ibrahim, A. R. Sadeghi, M. Schunter, G. Tsudik, and C. Wachsmann. 2015. SEDA: Scalable Embedded Device Attestation. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS '15)*. Association for Computing Machinery.

[6] F. Brasser, B. El Mahjoub, A. Sadeghi, C. Wachsmann, and P. Koeberl. 2015. TyTAN: Tiny trust anchor for tiny devices. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*.

[7] X. Carpent, K. ElDefrawy, N. Rattanavipanon, and G. Tsudik. 2017. Lightweight Swarm Attestation: A Tale of Two LISA-s. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security (ASIA CCS '17)*. Association for Computing Machinery.

[8] X. Carpent, G. Tsudik, and N. Rattanavipanon. 2018. ERASMUS: Efficient remote attestation via self-measurement for unattended settings. In *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*.

[9] M. Conti, R. Di Pietro, A. Gabrielli, L. V. Mancini, and A. Mei. 2010. The Smallville Effect: Social Ties Make Mobile Networks More Secure against Node Capture Attack. In *Proceedings of the 8th ACM International Workshop on Mobility Management and Wireless Access (MobiWac '10)*. Association for Computing Machinery.

[10] M. Conti, R. Di Pietro, L. Vincenzo Mancini, and A. Mei. 2008. Emergent Properties: Detection of the Node-Capture Attack in Mobile Wireless Sensor Networks. In *Proceedings of the First ACM Conference on Wireless Network Security (WiSec '08)*. Association for Computing Machinery.

[11] Moteiv Corporation. 2016. Tmote Sky Details. "http://www.snm.ethz.ch/snmwiki/pub/uploads/Projects/tmote_sky_datasheet.pdf".

[12] The New Jersey Cybersecurity and Communications Integration Cell (NJCCIC). December 28, 2016. "Mirai Botnet". Retrieved 28 December 2016..

[13] K. M. El Defrawy, N. Rattanavipanon, and G. Tsudik. 2017. HYDRA: hybrid design for remote attestation (using a formally verified microkernel). *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks* (2017).

[14] G. Dessouky, S. Zeitouni, T. Nyman, A. Paverd, L. Davi, P. Koeberl, N. Asokan, and A. R. Sadeghi. 2017. LO-FAT: Low-Overhead Control Flow ATtestation in Hardware. In *Proceedings of the 54th Annual Design Automation Conference 2017 (DAC '17)*. Association for Computing Machinery.

[15] E. Dushku, M. M. Rabbani, M. Conti, L. V. Mancini, and S. Ranise. 2020. SARA: Secure Asynchronous Remote Attestation for IoT Systems. *IEEE Transactions on Information Forensics and Security* (2020).

[16] K. Eldefrawy, A. Francillon, D. Perito, and G. Tsudik. 2012. SMART: Secure and Minimal Architecture for (Establishing a Dynamic) Root of Trust. In *NDSS 2012, 19th Annual Network and Distributed System Security Symposium, February 5-8, San Diego, USA*.

[17] L. Eschenauer and V. D. Gligor. 2002. A Key-Management Scheme for Distributed Sensor Networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS '02)*. Association for Computing Machinery.

[18] A. Francillon, Q. Nguyen, K. B. Rasmussen, and G. Tsudik. 2014. A minimalist approach to Remote Attestation. In *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*.

[19] W. He, M. Golla, R. Padhi, J. Ofek, M. Dürmuth, E. Fernandes, and B. Ur. 2018. Rethinking Access Control and Authentication for the Home Internet of Things (IoT). In *Proceedings of the 27th USENIX Conference on Security Symposium (SEC'18)*. USENIX Association.

[20] A. Ibrahim, A. R. Sadeghi, G. Tsudik, and S. Zeitouni. 2016. DARPA: Device Attestation Resilient to Physical Attacks. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks (WiSec '16)*. Association for Computing Machinery.

[21] V. Immler, J. Obermaier, K. Kuan Ng, F. Xiang Ke, J. Lee, Y. Peng Lim, W. Koon Oh, K. Hoong Wee, and G. Sigl. 2018. Secure Physical Enclosures from Covers with Tamper-Resistance. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2018).

[22] C. Kil, E. C. Sezer, A. M. Azab, P. Ning, and X. Zhang. 2009. Remote attestation to dynamic system properties: Towards providing complete system integrity evidence. In *2009 IEEE/IFIP International Conference on Dependable Systems Networks*.

[23] W. Kim and I. Jung. 2019. Smart Sensing Period for Efficient Energy Consumption in IoT Network. *Sensors* (2019).

[24] P. Koeberl, S. Patrick, S. Schulz, A. R. Sadeghi, and V. Varadharajan. 2014. TrustLite: A Security Architecture for Tiny Embedded Devices. In *Proceedings of the Ninth European Conference on Computer Systems (EuroSys '14)*. Association for Computing Machinery.

[25] F. Kohnhäuser, N. Büscher, S. Gabmeyer, and S. Katzenbeisser. 2017. SCAPI: A Scalable Attestation Protocol to Detect Software and Physical Attacks. In *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '17)*. Association for Computing Machinery.

[26] F. Kohnhäuser, N. Büscher, and S. Katzenbeisser. 2018. SALAD: Secure and Lightweight Attestation of Highly Dynamic and Disruptive Networks. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security (ASIACCS '18)*. Association for Computing Machinery.

[27] F. Kohnhäuser, N. Büscher, and S. Katzenbeisser. 2019. A Practical Attestation Protocol for Autonomous Embedded Systems. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*.

[28] S. Mahfoudh and P. Minet. 2008. Survey of Energy Efficient Strategies in Wireless Ad Hoc and Sensor Networks. In *Seventh International Conference on Networking (icn 2008)*.

[29] IHS Markit. 2017. Number of Connected IoT Devices Will Surge to 125 Billion by 2030, IHS Markit Says. https://news.ihsmarkit.com/prviewer/release_only/slug/number-connected-iot-devices-will-surge-125-billion-2030-ihs-markit-says.

[30] S. Moein, T. Aaron Gulliver, F. Gebali, and A. Alkandari. 2017. Hardware Attack Mitigation Techniques Analysis. *International Journal on Cryptography and Information Security* (2017).

[31] J. Noorman, J. Van Bulck, J. Tobias Mühlberg, F. Piessens, P. Maene, B. Preneel, I. Verbauwhede, J. Götzfried, T. Müller, and F. Freiling. 2017. Sancus 2.0: A Low-Cost Security Architecture for IoT Devices. *ACM Trans. Priv. Secur.* (2017).

[32] M. M. Rabbani, J. Vliegen, J. Winderickx, M. Conti, and N. Mentens. 2019. SHeLA: Scalable Heterogeneous Layered Attestation. *IEEE Internet of Things Journal* (2019).

[33] S. Ravi, A. Raghunathan, and S. Chakradhar. 2004. Tamper resistance mechanisms for secure embedded systems. In *17th International Conference on VLSI Design. Proceedings.*

[34] V. Roblek, M. MeÅ¡ko, and A. KrapeÅ¾. 2016. A Complex View of Industry 4.0. *SAGE Open* 2 (2016).

[35] S. Skorobogatov. 2011. Physical Attacks on Tamper Resistance: Progress and Lessons. 2nd ARO Special Workshop on HW Assurance, Washington DC.

[36] S. Skorobogatov. 2012. *Physical Attacks and Tamper Resistance.* Springer New York.

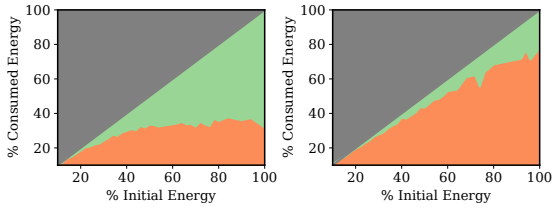
[37] A. Varga and R. Hornig. 2008. An Overview of the OMNeT++ Simulation Environment. In *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops (Simutools '08)*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[38] S. Zeitouni, G. Dessouky, O. Arias, D. Sullivan, A. Ibrahim, Y. Jin, and A. Sadeghi. 2017. ATRIUM: Runtime attestation resilient under memory attacks. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*.

A ENERGY CONSUMPTION EVALUATION

STATE	Energy Consumption
MCU ON	0.0054 J
TX	0.0585 J
RX	0.0654 J
IDLE	0.00016 J

Table 4: Energy consumption of Tmote Sky devices while in different states. MCU ON represents a state where the microcontroller performing computations. TX and RX represents the CC2420 state while transmitting or receiving respectively. IDLE represents and idle state where the transceiver is off and the MCU is in low power mode. Data taken from Tmote Sky datasheet [11].



(a) Normal distribution (b) Pareto distribution

Figure 8: Evaluation of the fairness in energy consumption for 500 provers. Figures show the average of the percentage of consumed energy with respect to the percentage of initial energy. Details of the experiment are shown in section 9.2.1 (a) the initial energy of the provers is chosen following a normal distribution ($\mu = 500, \sigma = 200$). (b) the initial energy of the provers is chosen following a shifted Pareto distribution (80% of the provers has an energy between 100J and 300J. The rest has an energy between 300J and 1000J).

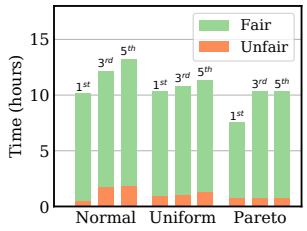


Figure 9: Time taken until 1st, 3rd, and 5th crash is seen in a network of provers running FADIA with (in green) and without (in orange) fairness integrated.

B BENCHMARKS OF ESP32-PICO-D4 DEVICES AND STELLARIS LM4F120H5QR MICROCONTROLLERS

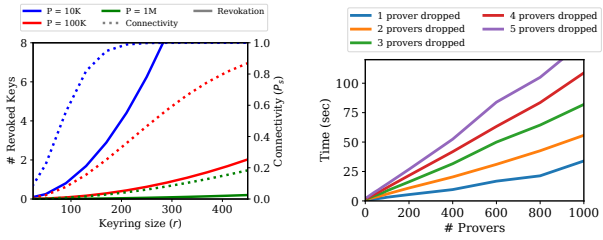
	ESP32-PICO-D4		LM4F120H5QR	
Function	Size	Time (ms)	Size	Time (ms)
SHA256	5 KB	13.171	32 KB	40.02
HMAC-SHA256	16 B	0.042	32 B	0.23
	1024 B	0.301	32 kB	39.86

Table 5: Benchmarks of SHA256 and HMAC-SHA256 with different input sizes on ESP32-PICO-D4 devices and Stellaris LM4F120H5QR MCUs (data from [27] and [26] respectively).

C EVALUATION OF THE REVOCATION PHASE IN FADIA

The revocation phase starts when a prover is dropped from the network. The controller detects the event during the next δ_h time and sends a revocation message *revk* to all the affected provers. To measure its efficiency, we evaluate the following:

- ($\mathbb{E}_1$) *The number of provers affected when a device is dropped.*
- ($\mathbb{E}_2$) *The number of connections affected.*
- ($\mathbb{E}_3$) *The number of keys revoked for each prover.*
- ($\mathbb{E}_4$) *Time taken until all affected provers receive the revocation.*



(a) Number of revoked keys (b) Revocation time

Figure 10: Revocation efficiency. (a) shows the the number of revoked keys on a prover with respect to different key ring sizes. (b) shows the average amount of time taken till the revocation process is completed with respect to variable number of provers in the network.

$\mathbb{E}_1$ is estimated as the number of provers in the network multiplied by the probability of two provers being connected ($n \times P_s$). This means that the average number of affected devices is proportionally tied to the connectivity of the provers in the network. An example of a keyring of size 300 and a key pool of size 100,000 gives $P_s = 0.6$, thus in this example 60% of the provers are affected with the revocation process. However, this is not a problem since most of the keys revoked at the prover are not actually used. Therefore, a more important metric to look into is the average number of connections affected (i.e., connections established using a key which is revoked). This is equal to $c \times r/p$ which estimates $\mathbb{E}_2$ (c is the total number of connections established at the time of the revocation). Following our example, the averaged affected connections will be only $0.003 \times c$ thus only 0.3% of the current connections need to be re-established. Third, $\mathbb{E}_3$ estimates the expected amount of decrease in the size of a keyring on each revocation process and it is equal to the following:

$$\sum_{i=0}^{K} \frac{\prod_{j=0}^{i-1}(K-j) \times \prod_{j=0}^{K-i-1}(P-K-j)}{\prod_{j=0}^{K-1}(P-j)} \times i \times \binom{i}{K} \tag{5}$$

We demonstrate this equation in Figure 10a. The figure shows the relation between the averages number of keys revoked per prover and the connectivity of the graph with respect to different keyring size and key pool size. We can see that for $r = 300$ and $p = 100,000$, 0.8 keys are revoked per prover on average. It thus requires 190 provers to drop for an active prover to revoke half of its keys. Finally, to evaluate $\mathbb{E}_4$, we run a simulation in which we deploy FADIA in a network of provers and we set $r = 300$ and $p = 100,000$. During the run of the protocol, we start the revocation process at a random point in time. We measure the time it takes till all provers receive the revocation message. We also consider cases where multiple revocations start simultaneously. Results are shown in Figure 10b. The revocation process increases linearly with the number of provers in the network and is performed within 34 seconds for 1,000 provers. When multiple provers are dropped simultaneously, the revocation time scales linearly with the number of dropped provers.