IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 15, 2021, accepted March 16, 2021, date of publication March 22, 2021, date of current version March 30, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3067697

CRAFT: A Continuous Remote Attestation Framework for IoT

L. MOREAU[1,2], E. CONCHON[1], AND D. SAUVERON[1]
[1]XLIM Laboratory, UMR CNRS 7252, Université de Limoges, 87000 Limoges, France
[2]SAS ICOHUP, 87000 Limoges, France

Corresponding author: L. Moreau (louis.moreau@etu.unilim.fr)

This work was supported in part by an ANRT CIFRE thesis convention (2017/1623), through the Région Nouvelle-Aquitaine under the grant for project "SVP-IoT," and in part by the MIRES research federation under grants for projects "CANIoT" and "SECIoT".

ABSTRACT As Internet of Things (IoT) networks keep growing with regards to the number of devices they contain, they become more attractive targets for attackers. Protecting these networks and the IoT devices they encompass is a major security challenge, and remote attestation enables checking of the integrity of devices (and thus networks). There are three categories of existing remote attestation protocols: software, hardware and hybrid attestation protocols. However, they all tackle specific issues only, such as small networks, IoT swarms, static networks, device-to-device attestation and network attestation. To provide as generic a solution as possible, which enhances security, we propose CRAFT, the first agnostic continuous remote attestation framework for IoT. CRAFT can be used in any real-world IoT network topology and can use any preexisting remote attestation protocol while remaining open to upgrades and extensions. A rigorous performance evaluation shows that CRAFT is very flexible and improves network security with little or no overhead, depending on the chosen parameters.

INDEX TERMS Computer security, continuous attestation, Internet of Things, remote attestation, smart devices.

I. INTRODUCTION

Connected objects are taking a growing part in everyday life. Devices range from simple connected sensors like thermometers to drones or even cars. These extremely diverse objects form interconnected networks such as drone swarms [1], smart homes [2], smart factories [3] or smart cities [4], and are called the Internet of Things (IoT) on a global scale. The IoT enables huge amounts of data to be exchanged, and new services to be provided. However, to fully benefit from IoT-provided data, the adoption of such devices needs to be done on a large scale, and to do so buyers need to have sufficient trust in connected objects. The increasing numbers of connected objects also mean a larger attack surface. All such devices are remotely accessible and benefit from limited resources, usually to comply with economic (hardware price), functional (integration to existing infrastructures) or ecological (energy consumption) reasons. That makes them prime targets for attackers, and insecure devices often end up in large botnets [5] resulting in increasingly large attacks. Remote attestation has recently been introduced as a solution to detect attacks on IoT devices. Devices either attest each other or transmit their attestation to a centralized verifier, the goal being to maintain trust throughout the network's life. Most existing attestation protocols aim at protecting devices from software attacks, and only a few also consider physical attacks.

The main issue with existing schemes is that they focus on specific remote attestation mechanisms, which only work efficiently in specific contexts, such as small networks, networks with no or little mobility, or networks with a short lifespan. To address these issues we propose **C**ontinuous **R**emote **A**ttestation **F**ramework for Io**T** (CRAFT), the first continuous remote attestation framework that can be used in any given network as depicted in Fig. 1. We represent any network using different classes of devices according to how secure the devices are: K-devices belong to the *core network* and L-devices belong to the *outer network*. Devices can enter the network, communicate with their neighbours, move and thus have different neighbours. They can also be banned from the network if they cannot be trusted anymore because their attestation failed or because they were unresponsive for too long. These mechanisms make CRAFT a useful security framework for IoT devices, being very flexible and adaptive.

The associate editor coordinating the review of this manuscript and approving it for publication was Adnan M. Abu-Mahfouz.

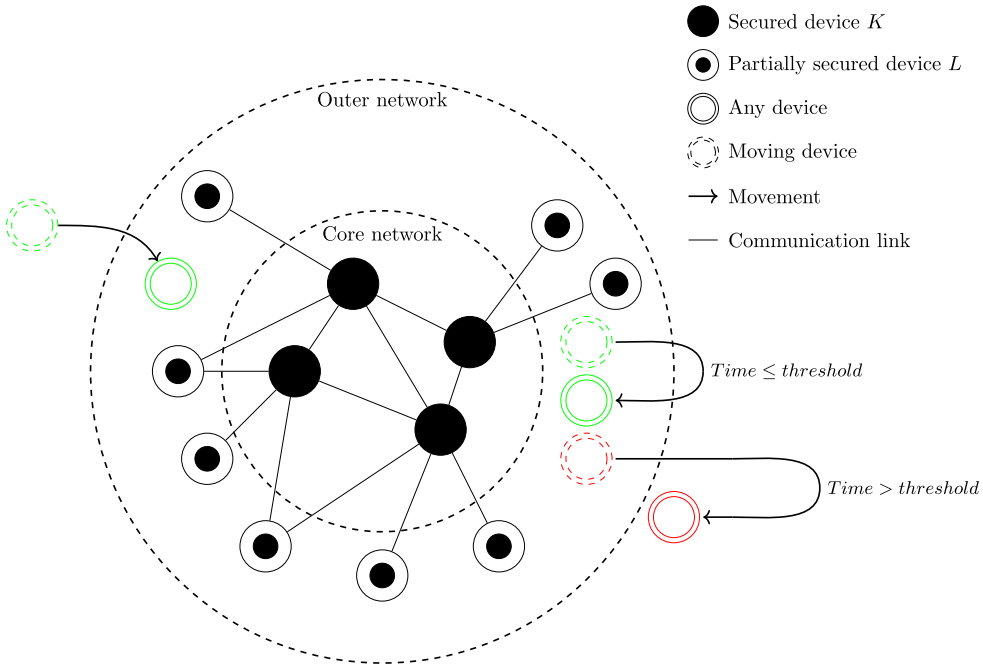


FIGURE 1. General network.

A. CONTRIBUTION

The main contribution in this paper is CRAFT, a general continuous remote attestation framework that can be used on any real-world IoT network. It can also leverage any existing remote attestation protocols, and even several in the same network. CRAFT is designed to be more flexible and efficient than standalone protocols, and it can be adjusted to fit an actual context.

The additional contributions of this paper are:

- A general network structure definition that can be used to describe any IoT network and that serves as a basis for CRAFT explanations. Based on this definition, the requirements needed to enable CRAFT to be generic and thus fit any scenario are listed in this paper.
- A comparative evaluation of CRAFT against two existing attestation protocols, SEDA [6] and US-AID [7] shows that the framework provides great advantages in terms of flexibility, security and performances over standalone attestation protocols.

B. STRUCTURE

Section II provides a general overview of existing attestation protocols. Section III then describes the threat model. Then, Section IV details CRAFT, a general continuous remote attestation framework, and provides its security analysis. Finally, Section V shows the benefits of CRAFT by comparing it to two existing attestation protocols.

II. RELATED WORK

CRAFT is the first proposed framework for attestation, to the best of our knowledge. As it is the first framework and since CRAFT is agnostic of the underlying attestation mechanisms, the following analysis of related work is dedicated to studying the main categories of attestation that CRAFT can encompass.

Several attestation techniques emerged in previous work aimed at keeping a network trusted, but they only focus on specific aspects of attestation. Many remote attestation protocols use an initial prover to broadcast an attestation request through the network using a broadcast tree and then fetch the result back [6]. Others enable the final attestation report to be fetched by any prover [8]. Some work has also exclusively focused on the internal attestation mechanism of a single device so it can be attested in the most trusted way [9]. Therefore, existing remote attestation protocols can be classified in the following categories: Software attestation, Hardware attestation and Software-Hardware, also called Hybrid attestation.

A. SOFTWARE ATTESTATION

Software-based attestation does not require any additional hardware components. That makes this solution the cheapest, but also offers a larger attack surface as such systems lack hardware security. It is commonly used for devices with very low-capabilities, such as 8-bit microcontrollers. As not all devices can comply with the hardware requirements, several solutions have been proposed. They are usually based on a challenge-response mechanism.

SWATT (in 2009) [10] uses a challenge-response protocol initiated by an external verifier, as the device is not trusted to verify itself. SWATT randomly checks parts of a device's memory, and shows that without changing the hardware,

an attacker will add noticeable overhead to the process, and thus be detected.

LRMA (in 2015) [11] is based on SWATT, but also takes into account network delays to achieve better performance than SWATT. It also counts the number of attestation failures to dynamically adjust the attestation frequency, and thus increase the probability of detecting an attacker.

Steiner *et al.* (in 2019) [12] aim at solving one of the issues of software-based attestation; i.e. evaluating the maximum Round-Trip Time (RTT) accurately at runtime. Their protocol relies on several smaller challenges instead of a single one, which improves RTT observation and thus the precision in estimating whether a device is compromised or not, while also reducing global overhead.

B. HARDWARE ATTESTATION

Hardware-based attestation requires additional hardware security features, such as a TPM (Trusted Platform Module) [13] or a TEE (Trusted Execution Environment) [14]. Hardware attestation solutions provide security against a wider range of attacks than software ones as hardware devices offer better protection by integrating some physical security mechanisms. However, hardware-secure devices are a complete remote attestation solution on their own.

ReDAS (in 2009) [15] uses a TPM to provide secure runtime attestation on dynamic systems, using two dynamic properties (structural integrity and global data integrity) to ensure attestation of dynamic applications at the device level.

SARA (in 2017) [9] uses TPM-enforced devices as cluster heads to isolate and secure parts of the network. Devices in the cluster and cluster head validate each other, and cluster heads are also validated by the main verifier in order to provide attestation at the network level.

SAFES (in 2018) [16] uses a TEE (ARM TrustZone), which provides isolation between the normal world and the secure world. TrustZone is used to monitor code in the normal world from the secure world. Code is controlled at each I/O event to ensure the correct execution of instructions.

C. HYBRID ATTESTATION

Hybrid-based attestation requires some additional hardware security features. It usually consists of regular hardware like a ROM (Read-Only Memory) and an MPU (Memory Protection Unit), with the MPU ensuring that critical code and data (such as cryptographic keys) are only accessed as they should be. It is less costly than a TPM or TEE, but still more secure than software-only attestation. Papers on hybrid attestation either focus on how to efficiently use minimal hardware [17]–[20], or on protocols using this hardware [6], [8], [21], [22].

The first published solution for hybrid attestation was SMART (2012) [17]. SMART stores critical code and keys in a ROM, and an MPU ensures that keys are only read by in-ROM code, and that this code can only be executed the intended manner.

Trustlite (in 2014) [18] extends SMART, with larger access control rules for the MPU, taking into account different kinds of memories (e.g. DRAM and Flash) and peripherals (such as timers) and introduces the use of OS and trusted states called Trustlets.

SEDA (in 2015) [6] was one of the first solutions to define a hybrid-based swarm attestation protocol. It relies on SMART and Trustlite in its test implementation. Attestation is requested by a trusted verifier from a device, which transmits it to its neighbours and so on, forming a tree. In the end, the first device will receive the sum of all responses and transmit information to the verifier about the number of compromised devices. In later papers [7], [23], SEDA has been criticized by its own authors for not being adequate for real-world application as it only focuses on remote software adversaries and a specific network model (swarms). However, it is used as a basis for comparison in many other studies.

DARPA (in 2016) [23] uses heartbeats to ensure devices are not disconnected from the network, and thus tries to detect physical attacks. It assumes that to physically attack a device, the attacker must disconnect it from the network (leaving aside side-channel attacks).

US-AID (in 2018) [7] uses PONAs (Proofs Of Non-Absence), similar to heartbeats, to show devices are continuously connected to the network, and have not been under physical attack. Using PONAs, devices can move, under the condition that their new neighbour is next to their previous neighbour.

SIMPLE (in 2020) [24] can be considered as lying between Hybrid and Software attestation solutions as it does not require additional hardware security features and thus encompasses more IoT devices. Its security comes from Security MicroVisor, a formally verified software-based memory isolation technique.

What is lacking in the literature is a more global proposal, able to leverage all these solutions by being adaptable to the network deployment context. That is what CRAFT is aiming at by defining general requirements, so that multiple remote attestation solutions can exist and even coexist in a single network to best suit security requirements, while taking into account the diversity of networks, which may include swarms of drones, smart-homes, smart-cities, and many other variations.

III. THREAT MODEL

As in most previous work, the operator O, which oversees the network, is assumed trusted. All other devices are subject to attacks from an adversary Adv, with different attack capabilities:

- Following the Dolev-Yao model [25], Adv has full control over communications: he can listen, modify, delete or generate messages between devices.
- Adv can also compromise the software of any device and can thus control the device execution and memory content.

TABLE 1. Notations.

General definitions	
O	Network operator
N	Total number of devices
D	Set of all devices with $D = \{D_i; 1 \leq i \leq N\}$
Device parameters	
id_i	Identity of a device D_i
s_i	Security strength of a device D_i
h_i	Maximum device mobility. A device D_i at h_i network hops from D_j can physically move next to $D_{j \neq i}$ without further communication with other devices between D_i and D_j. If $h_i = 0 \; \forall i$, then the devices relative positions are fixed.
T_{a_i}	Time with no heartbeat after which D_i is considered compromised
T_{b_i}	Maximum elapsed time between two heartbeats sent by D_i after which a broadcasted message defined in section IV-B is sent up to the h_i th neighbour, with $T_{b_i} < T_{a_i}$
SK_i, PK_i	D_i private and public keys
k_{ij}	Common key shared by two devices (i.e. D_i and D_j)
$OCert_i$	D_i certificate delivered by O
H_i^t	Heartbeat of device D_i received at time t by $D_{j \neq i}$
H_i^{list}	List of heartbeats from $D_{j \neq i}$ known by D_i
P_i	D_i parameters, $P_i = \{id_i, s_i, h_i, T_{a_i}, T_{b_i}\}$
$Sign_O(P_i, PK_i)$	Signature by O of D_i parameters and D_i public key
Network parameters	
st_L	Minimal security strength for a device to be included in the network
st_K	Minimal security strength above which a device is considered secured
K	Set of secured devices such as $K = \{D_i \mid st_K \leq s_i; 1 \leq i \leq N\}$
L	Set of partially secured devices such as $L = \{D_i \mid st_L \leq s_i < st_K; 1 \leq i \leq N\}$
Functions	
$Auth(input, key)$	Authenticates a given $input$ using a given key
$Verify(auth, key)$	Verifies an authentication $auth$ using a given key
$Exist(value, list)$	Verifies that a given $value$ exists in a given $list$
$Hash(input)$	Computes the hash of a given $input$ using hash functions (e.g. SHA1, SHA-256,...)
$Enc(input, key)$	Encrypts a given $input$ using a given key
$Dec(input, key)$	Decrypts a given $input$ using a given key

- *Adv* can perform physical attacks, and bypass any hardware protection, provided he gets enough time to do so: we assume that executing a physical attack requires disconnecting the device from the network for a significant amount of time. *Adv* then has access to hardware-protected code and memory.

Side-channel attacks are excluded from the scope of this work, as they could be performed on running devices, and require another set of detection and mitigation solutions. Denial of Service (DoS) is not considered in this paper, as in most related work.

IV. FRAMEWORK

CRAFT is a continuous remote attestation framework for IoT networks. Such a framework needs to consider factors such as IoT devices heterogeneity, the variety among IoT networks, and the diversity of already existing attestation protocols. CRAFT achieves this by defining a set of parameters to represent networks, devices, and how devices interact in the network. It is also open to future extensions. CRAFT is described in Section IV-A, along with specific notations and requirements. Its operations, and a deployment example, are given in Section IV-B. Finally, an informal security analysis of CRAFT is detailed in Section IV-C.

A. FRAMEWORK GENERAL PRESENTATION

1) NOTATIONS

The notations presented in Table 1 will be used in the remainder of the paper to precisely describe CRAFT. This table is composed of four parts: General definitions, Device parameters, Network parameters and Functions.

General definitions establish preliminary bases for other notations. Device parameters are specific to each device and are either configuration parameters or internal states. Network parameters are used to set up the network and classify devices. Finally, functions enable better description of the algorithms used.

2) DESCRIPTION

The proposed framework aims at improving IoT network security by enabling continuous remote attestation of network devices. Networks are controlled by an operator O and are composed of N devices D_i, each being identified by id_i.

Each device is defined by several parameters to make the framework suitable for any network setup. Devices are first split into subsets according to their security strength parameters s_i. These parameters are based on the deployment context and device characteristics and are given by O at setup time. Devices are assigned to one of three categories using two threshold values st_L and st_K, which are also fixed according to the deployment context. Secure devices with a $st_K \leq s_i$ are called K-devices and are part of the *core network*, and partially secured devices with a $st_L \leq s_i < st_K$ are called L-devices and are part of the *outer network*. Other devices are excluded from the network. The less secure a device is, the stricter other security parameters must be; for example, heartbeat timers T_{a_i} and T_{b_i} or maximum mobility h_i will have

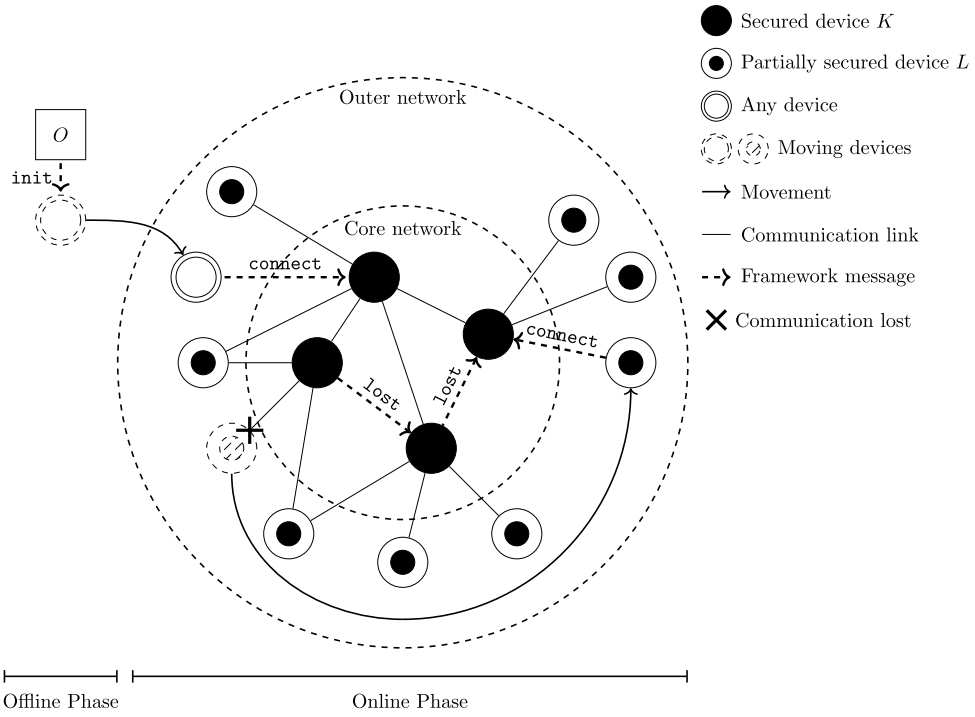


FIGURE 2. Main phases of CRAFT lifecycle.

restrictive values (i.e the smaller those values are, the less freedom a device has). Given these parameters, any device will be instantiated by O with $P_i = \{id_i, s_i, h_i, T_{a_i}, T_{b_i}\}$. Every device D_i also possesses a (SK_i, PK_i) key pair to initiate a secure connection with another device D_j by creating a session key k_{ij}. During the setup step, O also signs each device parameters with $Sign_O(P_i, PK_i)$. This enables devices to prove they are genuine on their first connection with other devices. All notation used is provided in Table 1.

The framework has two main phases, depicted in Fig. 2: an *Offline Phase* which prepares the device for introduction into the network, and an *Online Phase* during which the device is communicating with its neighbours to perform the tasks required to attest each device in the network, and therefore attest the whole network (both *core* and *outer networks*). The *Offline Phase* is performed only before introducing a new device and consists of two steps: setup and initialization. Any device is first categorized according to the deployment context and its hardware features, and then initialized with its own security parameters, functional parameters, keys and certificate in order to be able to join the network. The *Online Phase* happens continuously once the device is introduced into the network. A device always starts with a first connection with its direct neighbours, before proceeding with regular heartbeats and attestation. During the *Online Phase*, except in the case of a static network, devices are moving. Additional messages are exchanged to enable continuous attestation despite the movement.

The requirements that CRAFT fulfills are described in Section IV-A3.

3) REQUIREMENTS

The proposed continuous remote attestation framework must follow the requirements listed below. Those requirements are split into two categories: functional requirements and security requirements. Functional requirements define the basic functionalities any continuous remote attestation framework needs to fulfill. Security requirements define security features that must be supported to ensure a network remains trusted.

a: FUNCTIONAL REQUIREMENTS

- **(FR1)** A continuous remote attestation framework must be able to support any attestation protocol (given the protocol can be used in the deployment context).
- **(FR2)** A continuous remote attestation framework must be able to adapt its parameters according to the deployment context including network mobility, devices' security, devices' hardware performances and threat model.
- **(FR3)** A continuous remote attestation framework must add as little data exchange as possible, to reduce network usage and energy consumption, while still adding enough security.
- **(FR4)** A continuous remote attestation framework must make as few calls to cryptography primitives as possible in order to reduce computation time and energy consumption, while still adding enough security.

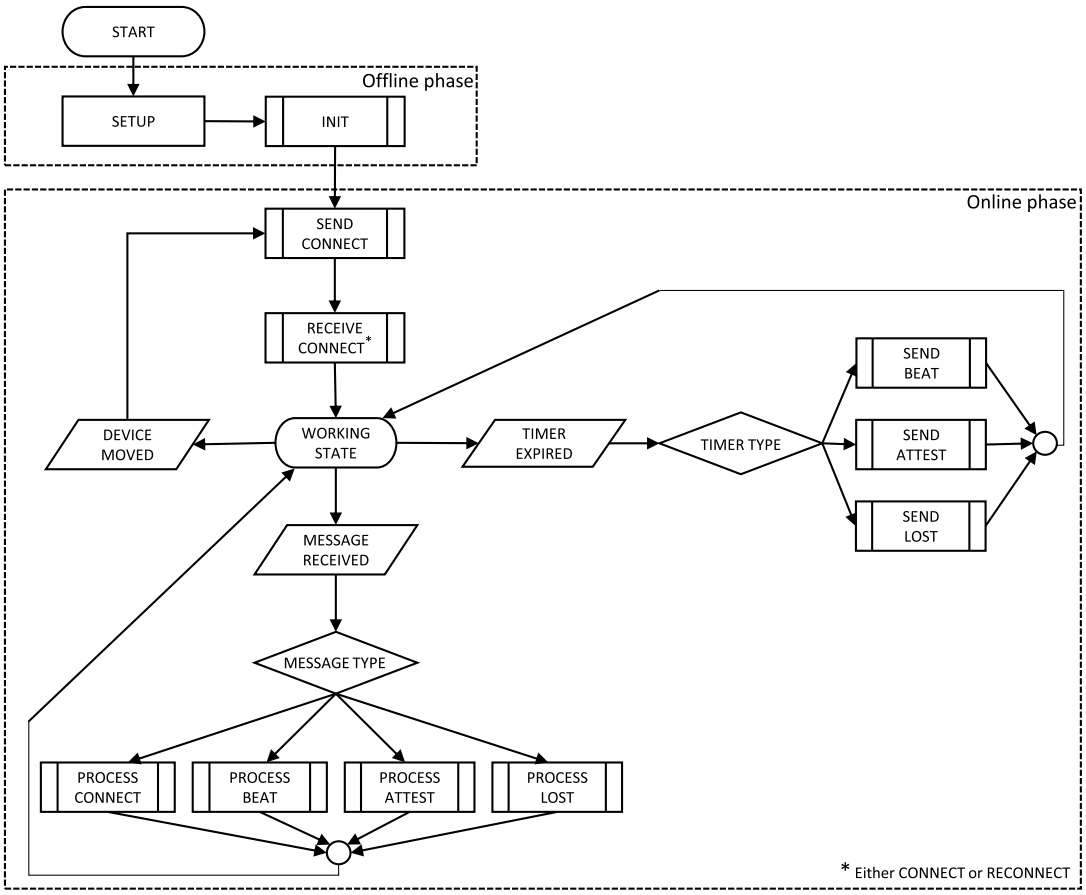


FIGURE 3. Different states of a device in CRAFT.

- **(FR5)** A continuous remote attestation framework must be open to future extensions.

b: SECURITY REQUIREMENTS

- **(SR1)** A continuous remote attestation framework must reject any malicious device trying to be part of the network.
- **(SR2)** A continuous remote attestation framework must be able to exclude a device from the network when continuous attestation is broken; i.e., when the device attestation fails or when it cannot be reached for a certain amount of time.

B. FRAMEWORK PHASES

During the lifetime of the network as illustrated in Fig. 2, the continuous remote attestation framework goes through two main phases, and each phase consists of different steps. These steps are also depicted in Fig. 3 from the point of view of a device inside of CRAFT.

The first phase is the *Offline Phase* and takes place before the network is actually running. The *Offline Phase* begins with a setup step, during which devices are categorized and parameters are set. Then, an init step takes place, during which previously set parameters are provided by O to all devices.

The second phase is the *Online Phase*, which encompasses the lifecycle of the network as explained in Section IV-B2.b. At the beginning of that phase, connections are established between neighbours, and then regular messages are exchanged to maintain continuous attestation even when devices move, by broadcasting information required to enable reconnection of devices.

1) OFFLINE PHASE

a: SETUP

The setup step consists of categorizing a device before introducing it to the network. We consider two classes of devices, named K and L. The limit between these two classes is chosen by O and depends on the deployment context.

As depicted in Fig. 4, devices D_i can belong either in K or L according to how secure the device is. Some devices might not qualify to be part of D: they are not included in the network. The security strength parameter s_i defines two other characteristics of devices: networking role and device parameters.

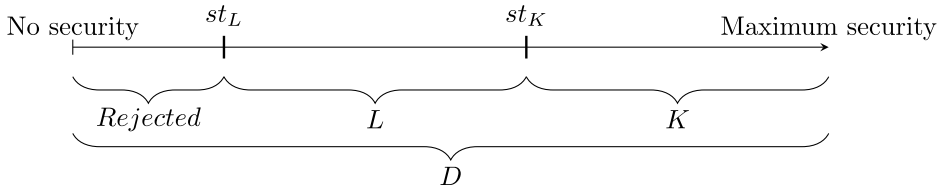


FIGURE 4. Devices classification by security strength (arbitrary scale).

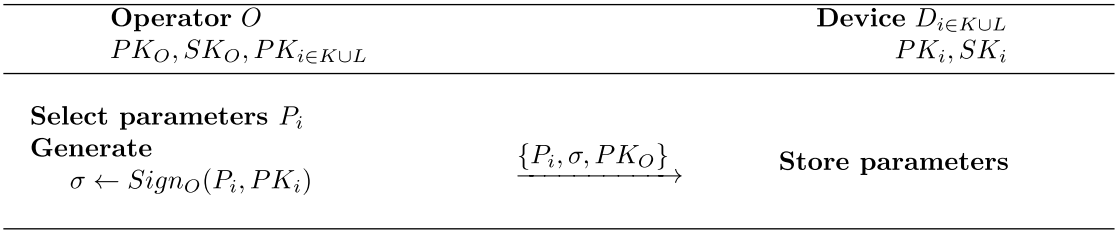


FIGURE 5. Init step.

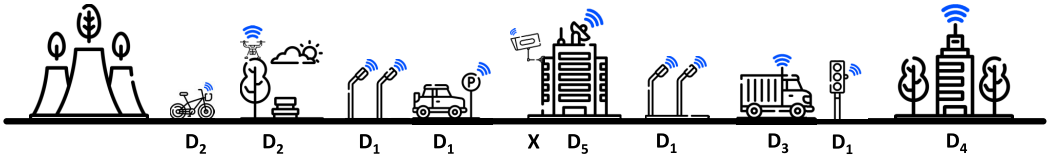


FIGURE 6. Illustration of the real-world smart-city example.

The networking role is either *node* or *endpoint*. Any K-devices have the ability to act as either a node or as an endpoint, whereas L-devices can only act as endpoints (i.e. the *outer network* as illustrated in Fig. 2; hence devices in K build the *core network*. An endpoint communicates exclusively with nodes to transmit application data or attestation data. A node communicates with other nodes and endpoints and participates in attesting the network and routing all data. We provide details as a network deployment example below.

A device D_i can be hardware-secured or software-secured, and devices can each have a different solution with several degrees of efficiency. All solutions existing in the network must be listed and ordered by security strength, and any given device must observe a minimum security level determined by the deployment context. Two identical devices in a network can have different s_i depending on the deployment context; e.g., an indoor device might be less prone to attack than an identical device deployed outdoors.

Once devices have been classified, O can give them parameters according to s_i. Two devices with the same s_i should have the same parameters, as it makes no sense to have tighter parameters for the same level of security. In networks and to ease deployment, devices with different s_i values can share the same parameters.

b: INIT

During the init step, O gives the device its security and functional parameters. O also provides P_i, PK_O and $Sign_O(P_i, PK_i)$ following the procedure depicted in Fig. 5 so the device can join the network. This can be done using a distinct type of message or during a factory configuration. Depending on the underlying attestation protocol and additional options, other parameters may also be provided.

c: REAL-WORLD DEPLOYMENT EXAMPLE

The network considered in this example is a smart-city with many devices in it. Those devices are split into groups, according to how secure they are. At the time of network deployment, the Operator O assigns a security level to these devices following a scale O chooses. Every single device from these groups could have a different security level, but for the sake of the example, they are grouped by categories. Fig. 6 illustrates these different devices in a smart-city context.

- Last generation base stations, defined as $(D_5, s = 28)$
- Older base stations, defined as $(D_4, s_4 = 25)$
- Official city vehicles, defined as $(D_3, s_3 = 21)$
- Air pollution sensors attached to mobile devices (bicycles, drones) defined as $(D_2, s_2 = 14)$
- Smart parking, smart lighting and smart traffic lights, defined as $(D_1, s_1 = 12)$
- Low-cost security cameras, defined as $(X, s_X = 5)$

The Operator O also selects context-adapted security thresholds for the K and L categories giving $st_L = 10$ and $st_K = 20$. As shown in Fig. 7, this gives $K = \{D_3, D_4, D_5\}$ and $L = \{D_1, D_2\}$. X is excluded from the network due to its low security strength parameter.

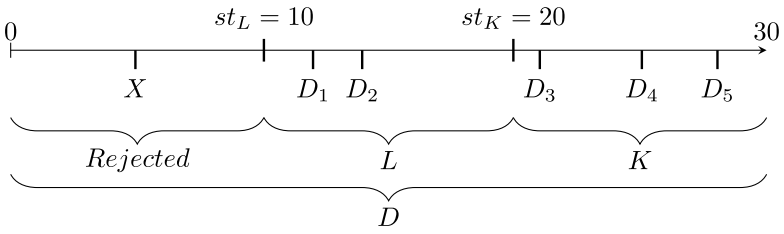


FIGURE 7. Device classification by security strength for our real-world smart-city example (arbitrary scale from 0 to 30).

Once the devices have been classified, O can define their parameters. L-devices are set with $T_{a_i} = 3600s$ and K-devices with $T_{a_i} = 7200s$ as they are considered more secure. T_{b_i} is set to $60s$ for both L- and K-devices: all of them must signal their presence to the network with the same regularity. Finally, L-devices are set with a h_i value of $h_1 = 0$ and $h_2 = 1$ allowing small movements, and K-devices are further distinguished by setting $h_3 = 3$ and $h_4 = h_5 = 0$ since city vehicles can have greater mobility than D_2 devices and base stations are fixed.

Thus, devices can be instantiated using the definition $P_i = \{id_i, s_i, h_i, T_{a_i}, T_{b_i}\}$:

$$P_1 = \{1, 12, 0, 3600, 60\}$$
$$P_2 = \{2, 14, 1, 3600, 60\}$$
$$P_3 = \{3, 21, 3, 7200, 60\}$$
$$P_4 = \{4, 25, 0, 7200, 60\}$$
$$P_5 = \{5, 28, 0, 7200, 60\}$$

Devices can then be initialized by O, join the network and take part in regular message exchanges.

2) ONLINE PHASE

During the *Online Phase* as depicted in Fig. 8, several messages are exchanged, named `connect`, `beat`, `attest`, `lost` and `reconnect`. `connect` is sent during the beginning of the *Online Phase* while other messages are exchanged regularly during the Standard Lifecycle of the network. These messages are the minimal set of messages required to maintain all the desired features of our framework. Depending on the attestation protocol in use and its specificities (consensus, aggregation, ...), additional messages might be needed. Packet length is not defined here, only content, as most packets can change depending on the cryptographic operations used or the inclusion of optional fields as requested by requirements **FR3** to **FR5**.

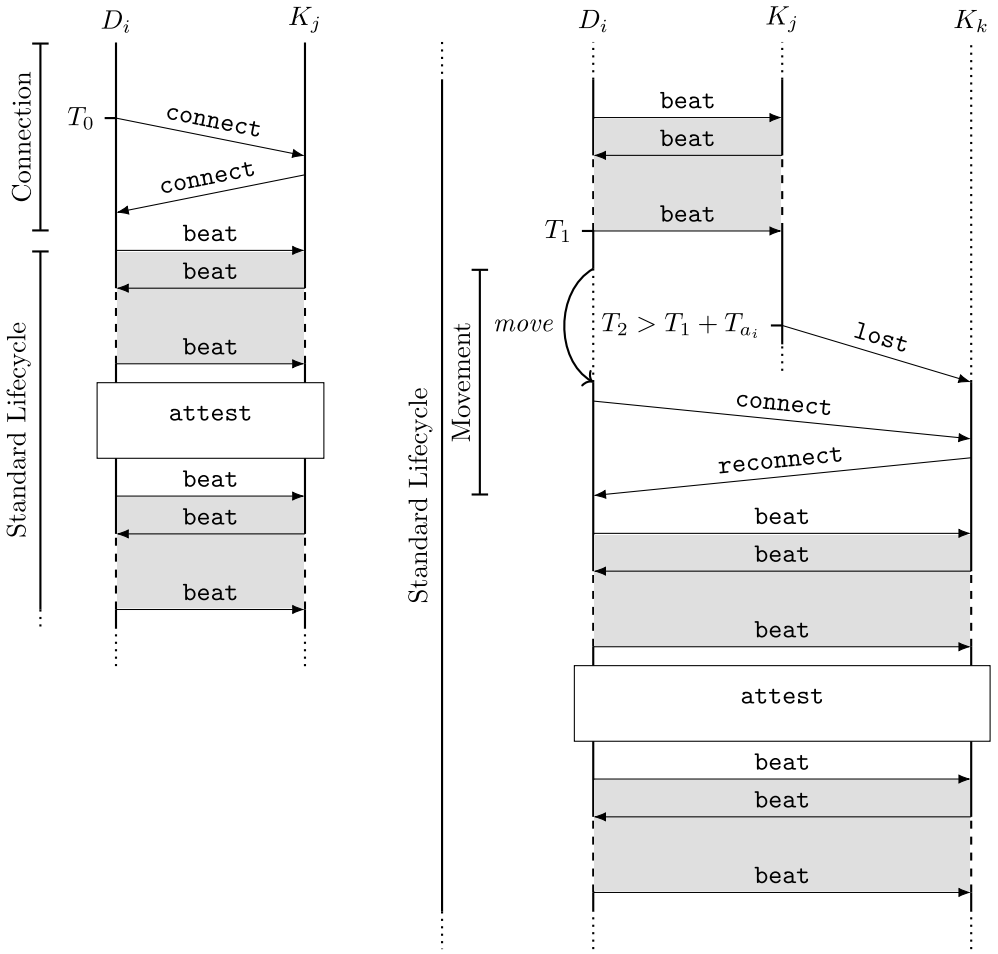
As depicted in Fig. 9 every message starts with a field called `Head` to distinguish different messages. It is followed by a `Size` field which is the full size of the message in groups of 32 bits. The next packet part is a field for `Parameters` so devices can know how to receive a given message: for instance, a part of the bitfield set to $(01)_2$ would mean to use SHA-1 as the hash algorithm, while $(10)_2$ would mean SHA-256. Thus, the `Parameters` field enables **FR2**. The `Device Id` field identifies the message sender. Finally, the `Timestamp` field is used to avoid message replay, and to ensure that any old message is dropped. There is no need for a checksum as the proposed protocol is operating at the OSI application layer and it relies on lower layers of protocols (e.g. TCP) to enforce packet integrity. Messages such as `beat` are authenticated, using either a signature or a Message Authentication Code (MAC). A signature takes more computation time but is more secure and bound to a single device. A MAC is faster to compute, more flexible in size, and relies on symmetric keys established from an asymmetric key pair. Choosing between the two will depend on the network devices and on their respective security requirements. This flexibility also contributes to fulfilling requirements **FR3** by adjusting how much data volume overhead is added by cryptography protocols, and **FR4** by adjusting cryptography depending on the deployment context.

a: INTRODUCING DEVICES IN THE NETWORK

`connect` as illustrated in Fig. 8a is a message (depicted in Fig. 10) sent by a new device in the network in order to be accepted by its neighbours, and sent back by the neighbour to the connecting device. `connect` contains the sending device parameters h_i, T_{a_i} and T_{b_i}. The `Other Options` field enables CRAFT to be extended with additional features the network might require but are too network-specific to be detailed here as required by **FR5**, without the need for defining other packets. The `Options Size` and `Other Options` can be omitted, setting a bit in the `Parameters` bitfield to signal these option s. A public key PK_i is shared to enable encryption through the use of a shared secret created with an algorithm used upon `connect` reception and validation, such as ECDH for instance. The genuineness of the public key and the device parameters are attested by O signature (i.e. the `Public Key and Device Parameters Signature by` O field). The signature expires according to the validity date placed in the `Signature Expiration` field, which ensures only a new device is accepted as genuine by other devices.

b: STANDARD LIFECYCLE

During the lifetime of the network in Fig. 8, `beat` and `attest` messages are regularly exchanged to ensure the continuous attestation of the network.



(a) Connection and Standard Lifecycle (b) Device movement and reconnection

FIGURE 8. Typical message exchange between devices during the *Online Phase*.

Head	Size	Parameters	Device Id
Timestamp	...		

FIGURE 9. Common packet header definition.

Head	Size	Parameters	Device Id
Timestamp	h_i	T_{a_i}	T_{b_i}
Options Size	Other Options		
PK_i			Signature Expiration
Public Key and Device Parameters Signature by O			

FIGURE 10. `connect` packet definition.

`beat` presented in Fig. 11 is a message sent by D_i at regular intervals to its direct neighbours from the *core network* (K-devices) as shown in Fig. 12. It proves that D_i has not moved or been disconnected from the network. To reduce the number of messages and thus the data and energy overheads, `Optional Data` can be sent using `beat`, fulfilling both

Head	Size	Parameters	Device Id
Timestamp	Optional Data		
Authentication			

FIGURE 11. `beat` **packet definition.**

Device $D_{i\in K\cup L}$		**Device** $K_{j\neq i}$
id_i, k_{ij}		H_j^{list}, k_{ij}
when T_{b_i} **elapsed:** $H_i^t = \{id_i, t = now\}$ $a = Auth(H_i^t, k_{ij})$	$\xrightarrow{\{H_i^t, a\}}$	**if** $Exist(H_i^{t-1}, H_j^{list})$ **and** $Verify(a, k_{ij})$: replace H_i^{t-1} by H_i^t in H_j^{list} **else:** do nothing

FIGURE 12. `beat` **exchange between any device and a *K*-device.**

Head	Size	Parameters	Device Id
Timestamp	Lost Device Id	Initial Sender Id	TTL
Expiry Time	Lost Device Configuration Hash		
Encrypted Authentication of Initial Sender			
Authentication			

FIGURE 13. `lost` **packet definition.**

FR3 and **FR5**. For instance, instead of using a separate message, attestation can be triggered using the same message, thus reducing overhead. The whole message must be authenticated with k_{ij} to ensure communications only take place between trusted devices.

In Fig. 12 we can see that device D_i, when T_{b_i} has passed, sends a `beat` message containing the device id id_i and the current time. Every receiving device K_j must verify the message authentication using the shared key k_{ij} and must check that H_i^{t-1} exists in H_j^{list}. If both conditions are fulfilled, these devices can update their information about D_i; if not, they simply ignore it.

`attest` is a message or group of messages sent according to the underlying attestation protocol. As it relies on the attestation protocol under use, we provide no further description, other than it should respect the same packet format. Attestation can also be broadcast using `beat` messages, resulting in fewer messages, which may lead to smaller overhead. Indeed, the `Optional Data` field of the `beat` can be used to trigger and/or broadcast the attestation without adding overhead due to repeating message headers. This flexibility in CRAFT features addresses requirement **FR1**.

To enable devices' **movements**, the following packets `lost` and `reconnect` are also defined. How these packets are integrated in the Standard Lifecycle is described in Fig. 8b.

`lost`, as depicted in Fig. 13, is a message sent by K_j to its neighbours to announce that D_i is no longer responding, and thus might have moved. In addition to the common fields presented in Fig. 9, this message contains id_i as the `Lost Device Id` to identify the missing device. It also contains id_j as the `Inital Sender Id` to identify the reporting device. The `TTL` field indicates how many network hops the message can still do, and it is decreased before `lost` is forwarded to the next device. For instance, a `TTL` value of *one* means that the message will not be transmitted after the current hop. The four remaining fields enable the lost device to safely reconnect to a distant device having received a `lost` message:

- The `Expiry Time` field prevents the `lost` message from being intercepted by an adversary and kept for a long time. `lost` messages received after T_a are discarded.
- The `Lost Device Configuration Hash` field is a hash of h_i, T_{a_i}, T_{b_i} and PK_i. Thus, when D_i connects back to the network, its new neighbour K_k can check that the parameters sent upon reconnection match the hash received in the `lost` message, proving that D_i

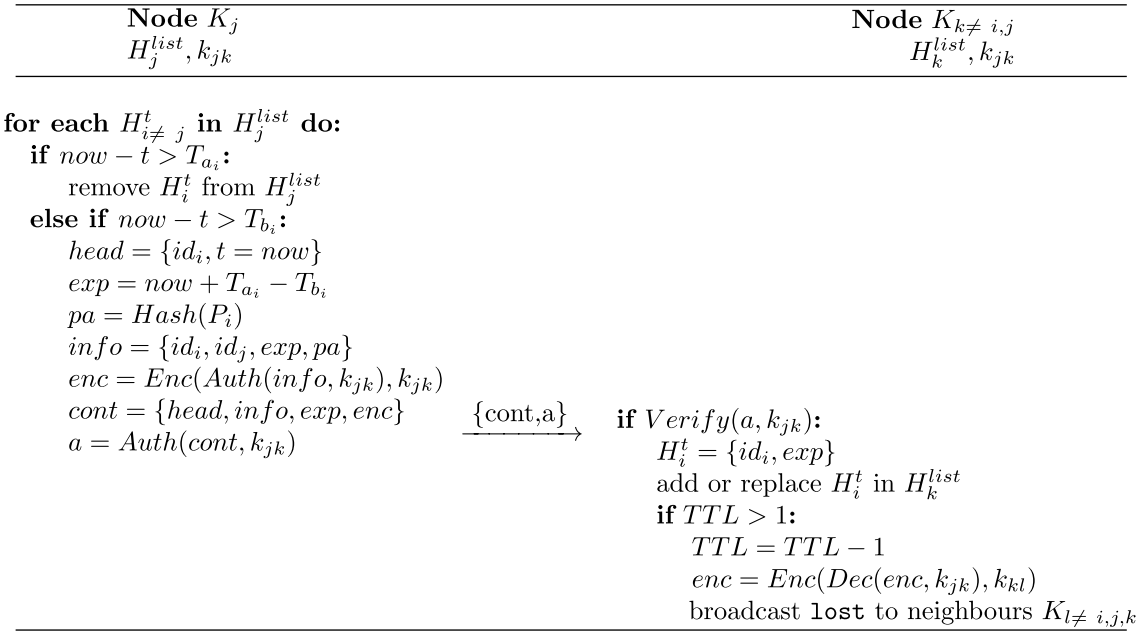


FIGURE 14. `lost` broadcasting between two nodes.

is genuine. Any device can send the right parameters during a `connect`, but only D_i is able to authenticate further communication. Indeed, D_j knows PK_i, but only D_i knows SK_i, and thus only D_i can create the correct session key k_{ij}.

- The `Encrypted Authentication by Initial Sender` field enables the lost device to know that it has reconnected to a genuine device that previously received the `lost` message through a chain of other genuine devices: the initial sender K_j authenticates (e.g. with the HMAC algorithm and using k_{ij}) the `Lost Device Id`, `Initial Sender Id` and `Lost Device Configuration Hash` fields, which are immutable during the `lost` message transfer from one device to another. D_i can thus check that the `lost` message was indeed emitted by K_j using HMAC and k_{ij}. Before sending the `lost` message to its neighbour K_k, K_j encrypts this authentication using k_{jk} in the `Encrypted Authentication of Initial Sender` field. This `Authentication of Initial Sender` is decrypted at each hop by trusted neighbours receiving the message, and then re-encrypted before forwarding the `lost` message with the updated `Encrypted Authentication of Initial Sender` field to the next device. Thus, the `lost` message can only follow a trusted path, and D_i can trust the device it connects back to since it is a device from the chain of trust.
- The final `Authentication` field is the same as in the `beat` message, simply enabling a device to quickly check the authenticity of the message it just received.

Fig. 14 illustrates how the `lost` message is sent and received:

- K_j checks regularly whether any of its neighbours $D_{i\neq j}$ has not sent any `beat` for longer than T_{a_i}
- If it has not sent any, it simply removes $D_{i\neq j}$ from its memory and thus considers it as not trustworthy anymore
- If it has sent some, it then checks weather $D_{i\neq j}$ has been silent for longer than T_{b_i}
- Then, if it has been silent, it broadcasts a `lost` message to all its neighbours $K_{k\neq i,j}$
- Upon reception, neighbours K_k will add $D_{i\neq j}$ to their H_k^{list}, waiting for it to reconnect
- If the `TTL` is greater than one, K_k decreases it and then forwards the `lost` message to its own neighbours
- If D_i does not connect before T_{a_i} elapsed, it is removed from H_k^{list}

When D_i reconnects to a node K_k after movement, it will send K_k a `connect` message minus the `Signature Expiration` and `Parameters Signature by` O fields. These fields are no longer needed as the trust now relies on the fact that K_k received a `lost` message. In response to that, K_k will send a `reconnect` message depicted in Fig. 15. This packet acts both as a `connect` message and a `lost` message. This enables D_i to establish a connection with K_k and to prove that K_k received a trusted `lost` message that followed a trusted path.

C. SECURITY ANALYSIS

In this section, we first discuss the security bases of CRAFT. Then we present how an adversary *Adv* can try to

Head	Size	Parameters	Device Id
Timestamp	h_i	T_{a_i}	T_{b_i}
Options Size	Other Options		
PK_i			Lost Device Id
Initial Sender Id	Lost Device Configuration Hash		
Encrypted Authentication of Initial Sender			
Authentication			

FIGURE 15. `reconnect` **packet definition.**

overcome the security requirements **SR1** and **SR2** provided in Section IV-A3.

1) SECURITY BASIS

CRAFT security relies on three elements:

- The attestation protocols it relies on
- The new messages defined in CRAFT as presented in Section IV
- The security parameters under use such as h_i, T_{a_i} and T_{b_i} defined in Section IV-A1

The security of the underlying attestation protocol used is out of the scope of this paper as it is considered an off-the-shelf secure component that has been proved to be secure beforehand. The more secure the protocol is, the more security CRAFT can provide. The security of CRAFT messages is detailed in Section IV-C2, and relies on cryptographic schemes. Finally, the security based on CRAFT parameters is based on how well these parameters are suited to the deployment context. This implies that the Operator has a good knowledge of the deployment scenario as well as of the devices that will be part of the network. The advantage of these parameters is the flexibility that they bring, but the limitation is the human factor during the context assessment.

2) ATTACKS ON CRAFT MESSAGES

a: IMPERSONATION

Adv could try to look like a genuine device in order to be included in the network. However, as parameters including the public key PK_i are signed by the operator O, such a device is not able to join the network. Therefore, **SR1** is complied with in relation to impersonation.

b: MESSAGE REPLAY

By storing a genuine message and sending it at a later time, *Adv* could try to interfere with devices. However, a timestamp is always included in protocol messages and hashed from neighbour to neighbour using a key shared only between the two neighbours. As devices are loosely synchronised, any message that was too old would then be rejected. Therefore, **SR1** is complied with in regard to message replays.

c: MESSAGE FORGERY

Adv could try to craft a message. For that message to be accepted, the authentication field must be correct and thus *Adv* either breaks the authentication primitive (e.g. HMAC) or fetches the key from the device memory. Fetching the key requires the device to be disconnected from the network for a significant amount of time, and so the device would be excluded and the key invalidated. Therefore, **SR1** is complied with in regard to message forgery.

d: DEVICE CLONING

Adv could clone an entire device and replace it in the network. To do so, the target device would need to be disconnected from the network for a significant amount of time, and would therefore be considered as compromised and consequently excluded. Thus, a cloned device with the right keys could not pursue communication with the network, as **SR2** is complied with.

e: DEVICE CONTROL

By taking control of a device, *Adv* could try to be part of the network. However, that would require bypassing the attestation protocol used and/or physically attacking the device, which would take a significant amount of time and be detected. Therefore, complying with **SR2** prevents this.

f: WORMHOLE ATTACK

Adv could establish a direct link between distant devices by just relaying their messages to each other. This will not work, as `lost` messages need to be transmitted following a trusted path. Since continuous attestation is not broken, **SR2** is complied with.

V. FRAMEWORK EVALUATION

In order to prove that CRAFT satisfies the requirements while being competitive with bare protocols, it was evaluated in the Omnet++ simulation framework [26]. During simulations, CRAFT was compared to two attestation protocols: SEDA [6] and US-AID [7]. SEDA is a relevant point of comparison as it was one of the first hybrid remote attestation protocols to emerge and it also serves as a comparison point in several

TABLE 2. Protocols features comparison.

Feature	CRAFT+SEDA	SEDA	CRAFT+US-AID	US-AID
Static network	✓	✓	✓	✓
Mobile network	✗	✗	✓	✓
Unlimited network lifespan	✓	✓	✓	✗
Heartbeats	✓	✗	✓	✓
Attestation spread	Global	Global	Device-to-Device	Device-to-Device

previous studies. It is also interesting to compare CRAFT and US-AID as it is one of the most recent protocols available, and it shares common features with CRAFT, such as heartbeats. SEDA and US-AID also have different kinds of attestation: SEDA reports all attestations to an initiator while US-AID does neighbour-to-neighbour attestation. However, SEDA and US-AID are context-specific whereas CRAFT is a general continuous remote attestation framework able to fit any context. Table 2 shows the main functional differences between the protocols.

Section V-A describes the metrics as well as the scenarios used and the methodology applied to obtain the results. Section V-B describes how the simulations in Omnet++ were done. Finally, the results of the simulation are detailed in Section V-C.

A. SCENARIOS DESCRIPTION, METRICS AND METHODOLOGY

a: METRICS

To compare the performances of CRAFT to SEDA [6] and US-AID [7], two metrics were taken into account. First, the total amount of data exchanged by the protocols was observed, as it can impact normal operations of the network and global energy consumption. Second, the number of HMAC operations were also compared, as these also impact execution time and energy consumption. Finally, we compared how these metrics evolved in different mobility scenarios.

b: SCENARIOS DESCRIPTION

Protocols were tested on two scenarios and with different numbers of devices to compare their performances in different conditions. The global parameters of the simulation are listed in Table 3. The first scenario consisted of randomly placed, static devices in a square layout. The duration was fixed, and the device density was also fixed to 250 devices per square kilometer (i.e. when the number of devices increased, the area increased accordingly). Communication range was also fixed to 100 m, so that the devices were usually in range for at least one other device. The second scenario had the same parameters, except that all devices moved according to the Gauss-Markov mobility model included in Omnet++.

c: METHODOLOGY

To have significant measurements, and to be able to provide confidence intervals around average values, each scenario simulation was repeated 100 times with a different seed (e.g., in a scenario with 50 static devices, changing the seed would mean changing the random position of the devices). This ensured that the chosen scenario gave no advantage to our framework. First, the assumption was made that data volume and number of HMAC operations followed a normal distribution. To show that this assumption was valid, we repeated the simulation of a scenario using the CRAFT +SEDA implementation 100 times, and we then proceeded to check our assumption using graphical methods: for both metrics, we first created a histogram, as shown in Fig. 16a and Fig. 16c, to show whether the shape of our values distribution was similar to a normal distribution. Both graphs show a clear bell shape, lightly skewed to the right.

We then created a Q-Q plot diagram in Fig. 16b and Fig. 16d. The fact that the dots mostly followed a straight line shows that the distribution was similar to a normal one, except at the right end of the graph, which also shows skewness.

This shows that our data did not strictly follow a normal distribution, but using the Central Limit Theorem, and because simulations were repeated 100 times for each scenario, the results were well approximated by a normal distribution. Therefore, we calculated confidence intervals using Student's t-distribution.

TABLE 3. Omnet++ simulation parameters.

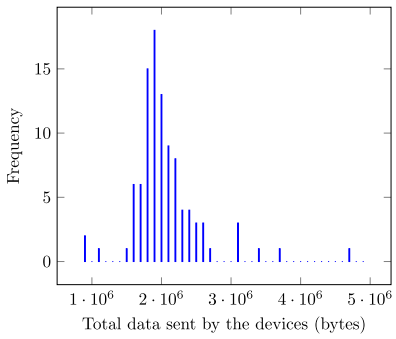
Parameter	Value
Device density (device per km^2)	250
Communication range (m)	100
Node positioning	Random (seeded)
Simulation duration (s)	86400
Number of runs per scenario	100
Radio model	UnitDiskRadio
Wireless Interface model	WirelessInterface
Mobility model	GaussMarkovMobility

B. IMPLEMENTATION DETAILS

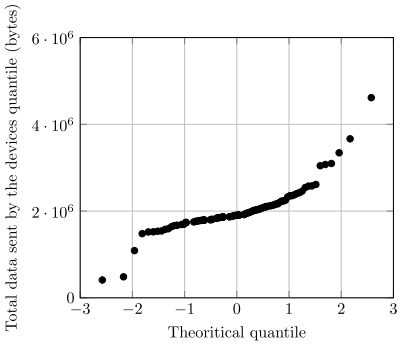
All implemented devices in CRAFT simulations were K-devices, which represented a worst-case scenario for CRAFT performance as K-devices exchange more data than L-devices.

We also re-implemented two existing protocols, SEDA [6] and US-AID [7], in order to compare them with CRAFT with as much accuracy as possible.

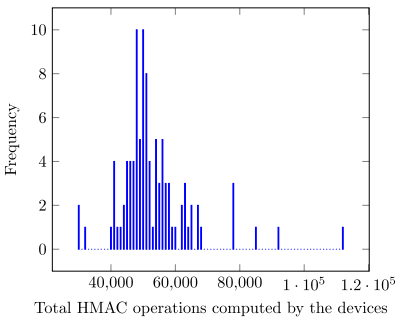
As CRAFT is a framework with no attestation capability of its own, we added SEDA attestation to CRAFT (resulting in CRAFT +SEDA) when comparing it to SEDA, and US-AID



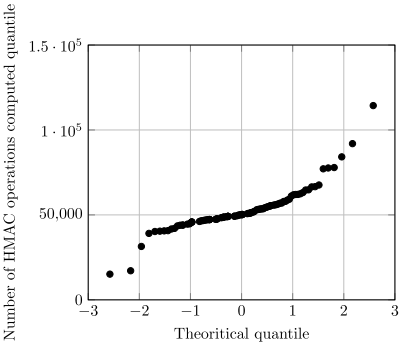
(a) Frequency of the total data bytes sent in the network over 100 iterations



(b) Q-Q plot of the total data bytes sent in the network over 100 iterations



(c) Frequency of the total HMAC calculated in the network over 100 iterations



(d) Q-Q plot of the total HMAC calculated in the network over 100 iterations

FIGURE 16. Graphs showing data is moderately skewed.

TABLE 4. Delays used to simulate cryptography in Omnet++.

Function	HMAC (SHA-256)	Encryption (AES)	ECDSA	PRNG
Delay (ms)	0.4	0.4	347.2	3.8

attestation (resulting in CRAFT +US-AID) when comparing it to US-AID.

Thus, CRAFT +SEDA has the same attestation mechanism as SEDA but provides heartbeats, which either add security to the network or reduce the data overhead caused by attestation. For CRAFT +US-AID, it has the same attestation mechanism as US-AID but brings more mobility and uses the CRAFT heartbeat, comparable to US-AID's Proof Of Non-Absence but with a smaller data overhead as details in Section V-C.

All cryptographic operations of these scenarios are simulated with delays detailed in Table 4 using values taken from SEDA [6] and wolfCrypt benchmarks [27]. With regard to parameters, all simulations lasted 86400 seconds (or 24 hours).

SEDA simulations were run with attestations every 3600 seconds (23 attestations over the simulation lifetime). CRAFT was compared to SEDA with two sets of parameters called CRAFT +SEDA A and CRAFT +SEDA B, both depicted in Fig. 17. These parameters are also presented in Table 5. CRAFT +SEDA A ran with attestations every 3600 seconds and heartbeats every 3600 seconds, making exchanges every 1800 seconds and thus doubling the number of times the devices' presence was checked and lowering the chances an attacker could perform physical attacks on a disconnected device. CRAFT +SEDA B ran with attestations every 7200 seconds and heartbeats every 2400 seconds, which also resulted in an exchange every 1800 seconds while reducing overhead significantly, but relatively decreasing the provided security level.

TABLE 5. Scenario parameters in SEDA and CRAFT+SEDA.

Parameter	SEDA	CRAFT+SEDA A	CRAFT+SEDA B
Attestation frequency (s)	3600	3600	7200
Heartbeat frequency (s)	-	3600	2400

US-AID simulations were also run with attestations every 3600 seconds (23 attestations over the simulation lifetime) and heartbeats every 1100 seconds. CRAFT was compared to US-AID with two sets of parameters called CRAFT +US-AID A and CRAFT +US-AID B, as depicted in Table 6. Both CRAFT +US-AID A and CRAFT +US-AID B used

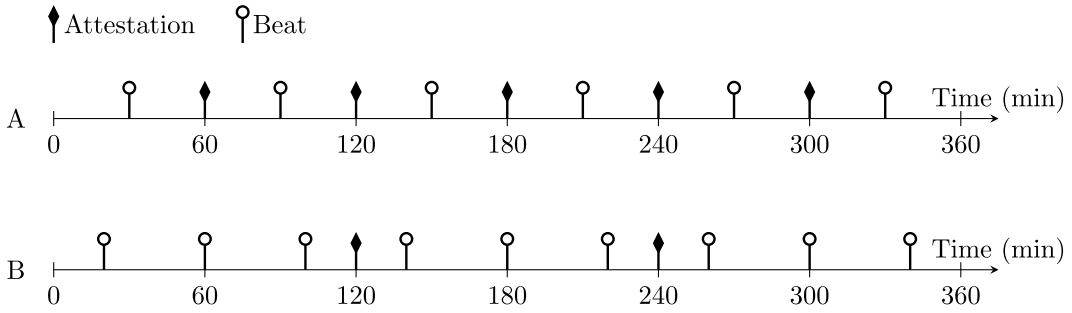
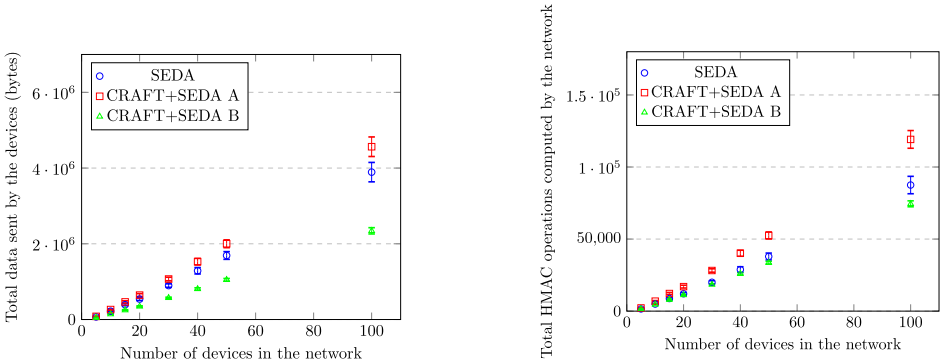


FIGURE 17. Attestation and heartbeat timings in CRAFT +SEDA A and B simulations.



(a) Data volume comparison

(b) HMAC operations comparison

FIGURE 18. SEDA comparison with CRAFT +SEDA A and B with regard to data volume and HMAC operations.

TABLE 6. Scenario parameters in US-AID and CRAFT+US-AID.

Parameter	US-AID	CRAFT+US-AID A	CRAFT+US-AID B
Attestation frequency (s)	3600	3600	3600
Heartbeat frequency (s)	1100	1100	1100
h_i (# of hops)	-	3	1

TABLE 7. Overhead comparison of CRAFT+SEDA over SEDA.

Metric	% of attestations	CRAFT overhead
Data volume	100%	+18.3%
	50%	−38.0%
# of HMAC	100%	+38.6%
	50%	−11.3%

the same attestations and heartbeat intervals, but the first had its h_i parameter (or TTL) set to $h_i = 3$, and the second had it set to $h_i = 1$. Setting it to 3 showed CRAFT flexibility with regard to mobility, while setting it to 1 enabled the comparison of almost equal functionalities for US-AID and CRAFT +US-AID and showed performance differences more accurately.

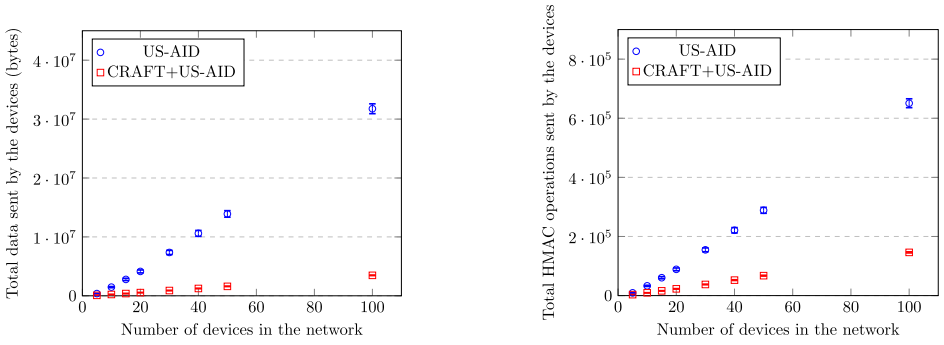
C. PERFORMANCE EVALUATION

In this section, the metrics detailed in Section V-A — data volume and cryptography — are compared between the framework plus the attestation protocol and the attestation protocol alone. The basis of comparison is always the existing attestation protocol (e.g., when comparing SEDA to CRAFT +SEDA, percentages are relatives to SEDA values). Values mentioned below relate to the 50 devices scenario. First, SEDA is compared to CRAFT +SEDA. That comparison will show that the proposed framework is far more flexible than SEDA: it will either provide a higher level of trust with minimal performance overhead; or a performance improvement while still providing an equal or better level of trust. US-AID is then compared to CRAFT +US-AID, showing that while both implementations provide the same base functionality and an equivalent level of trust and security, the proposed framework has a reduced data overhead.

1) SEDA VS CRAFT +SEDA

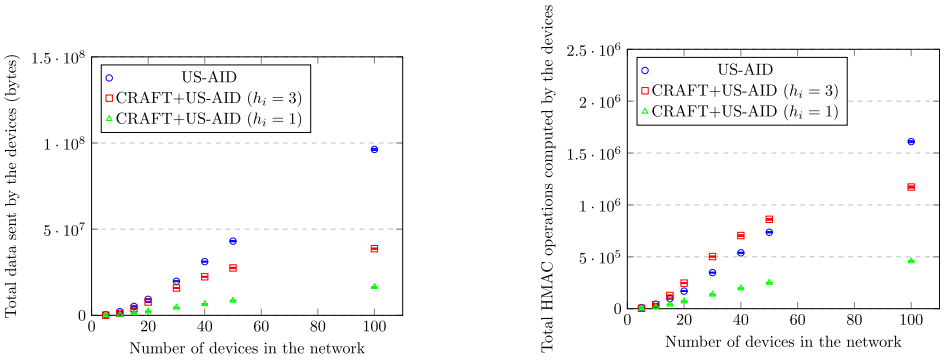
The two implementations share the same attestation mechanism, but SEDA does not have a heartbeat protocol: that means CRAFT has improved security by trading off performance. However, for an equivalent trust and security level, we can reduce the attestation frequency, which is balanced by the `beat` messages, and have a lower performance overhead.

As summarized in Table 7, Fig. 18a shows that, given the same attestation frequency, CRAFT had a data volume overhead of 18.3%, but had 38.0% less data volume by reducing the number of attestations by half while maintaining the average message frequency using `beat` messages. Similarly, Fig. 18b shows that, given the same attestation frequency,



(a) Data volume comparison (b) HMAC operations comparison

FIGURE 19. US-AID comparison with CRAFT +US-AID in a static network with regard to data volume and HMAC operations.



(a) Data volume comparison (b) HMAC operations comparison

FIGURE 20. US-AID comparison with CRAFT +US-AID in a mobile network with regard to data volume and HMAC operations.

CRAFT had an HMAC overhead of 38.6%, but had 11.3% less HMAC by reducing the number of attestations by half.

Thus, depending on the deployment context, CRAFT can be balanced between security and performance. Mobility is not compared here, as the SEDA attestation model has been shown not to work at all with moving devices.

These results show that CRAFT is far more flexible than SEDA, and can do better in security, performance or both, depending on the context and the chosen parameters.

2) US-AID VS CRAFT +US-AID

The two implementations share the same attestation mechanism but have different heartbeat protocols. US-AID offers PONAs, which is similar to the `beat` message. Even though US-AID and CRAFT send the same number of messages, each message is a lot smaller in CRAFT. However, CRAFT also has the `lost` message, to enable devices to move farther away, which increases the number of exchanged messages.

As summarized in Table 8, these points are verified in Fig. 19a-20b. Fig. 19a shows that CRAFT used 88.5% less data than US-AID in a static configuration. Fig. 20a shows that CRAFT used 36.1% less data than US-AID in a mobile configuration with $h_i = 3$ and 80.1% with $h_i = 1$. With regard to HMAC, CRAFT outperformed US-AID in a static mobility scenario where it did 76.6% of HMAC operations. This was not the case in mobile networks due to `lost` messages, as the `lost` message is propagated further away: CRAFT had an overhead of 16.9%. However, when h_i was fixed to 1, both protocols had the same degree of mobility and CRAFT was in fact more efficient, using 66%fewer HMAC operations.

Finally, if devices are moving, we can compare how many of them fail to stay in the network, as shown in Fig. 21. It shows that devices in US-AID were more frequently excluded from the network than in CRAFT: in the 50 devices scenario and over 100 simulations, US-AID excluded 2.07 ± 0.42 devices on average, while CRAFT only excluded 0.27 ± 0.11 devices on average with both h_i (average values of lost devices were the same for $h_i = 1$ and $h_i = 3$ in the tested scenarios). In the mobile network scenario, our data followed what appears to be a logarithmic curve as the number of devices increased, whereas US-AID values increased linearly as depicted in Fig. 20b with the CRAFT +US-AID ($h_i = 3$). The explanation is that US-AID keeps track of all its previous neighbours through the whole network life while we chose to only keep track of current active neighbours to limit the overall memory footprint. This meant that in our 50 devices scenario, each US-AID device ended up sending

TABLE 8. Overhead comparison of CRAFT+US-AID over US-AID.

Metric	Scenario	CRAFT overhead
Data volume	Static	−88.5%
	Mobile ($h_i = 3$)	−36.1%
	Mobile ($h_i = 1$)	−80.1%
Number of HMAC	Static	−76.6%
	Mobile ($h_i = 3$)	+16.9%
	Mobile ($h_i = 1$)	−66.0%

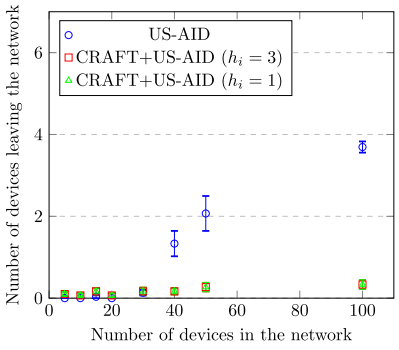


FIGURE 21. US-AID comparison with CRAFT +US-AID in a mobile network with regard to devices unintentionally leaving the network.

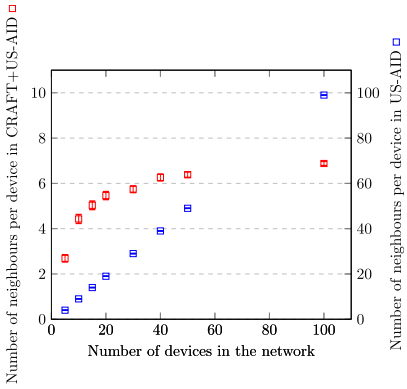


FIGURE 22. Logarithmic evolution of the number of neighbour per device in CRAFT +US-AID in a mobile network.

PONAs messages containing information about the 49 other nodes, whereas each CRAFT +US-AID device only sent `beat` and `lost` messages to 6.4 neighbours on average. For the 100 devices scenario, the number of neighbours evolved linearly for US-AID (from 49 to 99) while it evolved in a logarithmic fashion for CRAFT +US-AID (from 6.4 to 6.9) as depicted in Fig. 22. According to these results, CRAFT is better than US-AID in both performance and mobility, for an equivalent level of security.

In summary, both comparisons showed that CRAFT has multiple advantages over existing protocols such as flexibility, security and performance. Experiments comparing CRAFT and SEDA highlighted the relevance of the `beat` messages for increasing security with a limited messages overhead. CRAFT can also be used to increase the time period between two attestations while keeping the same level of security as in SEDA. Experiments with US-AID showed that CRAFT is suitable in mobile networks and can handle high levels of mobility without losing as many nodes as US-AID, thanks to the trusted paths in the network. Indeed, trusted paths help to join the network back anytime and anywhere while US-AID allows mobility only from neighbours to neighbours.

VI. CONCLUSION AND FUTURE WORKS

In this paper, we proposed CRAFT, the first generic and agnostic continuous remote attestation framework for the IoT. To embrace all the requirements of such a framework, we first gave a general definition of what an IoT network is. Then we created CRAFT, which provides all basic functionalities required to leverage continuous remote attestation in any real-world IoT network and works in both static and mobile networks. Moreover, CRAFT can use any preexisting remote attestation protocols while remaining open to upgrades and extensions.

Using simulations, we also showed that CRAFT is much more flexible than standalone attestation protocols such as SEDA and US-AID. As a result, it improves real-world IoT network security thanks to more frequent interactions with a small message overhead. In a mobility scenario, CRAFT also proved to be more reliable than US-AID as devices were more prone to remain in the network.

In future work, as CRAFT is able to deal with several attestation protocols at the same time, we plan to evaluate this feature in regard to both security and performances in different network topologies.

REFERENCES

[1] D. Wu, D. I. Arkhipov, M. Kim, C. L. Talcott, A. C. Regan, J. A. McCann, and N. Venkatasubramanian, "ADDSEN: Adaptive data processing and dissemination for drone swarms in urban sensing," *IEEE Trans. Comput.*, vol. 66, no. 2, pp. 183–198, Feb. 2017.

[2] J. M. Batalla, A. Vasilakos, and M. Gajewski, "Secure smart homes: Opportunities and challenges," *ACM Comput. Surveys*, vol. 50, no. 5, pp. 1–32, 2017.

[3] B. Chen, J. Wan, L. Shu, P. Li, M. Mukherjee, and B. Yin, "Smart factory of industry 4.0: Key technologies, application case, and challenges," *IEEE Access*, vol. 6, pp. 6505–6519, 2018.

[4] K. Zhang, J. Ni, K. Yang, X. Liang, J. Ren, and X. S. Shen, "Security and privacy in smart city applications: Challenges and solutions," *IEEE Commun. Mag.*, vol. 55, no. 1, pp. 122–129, Jan. 2017.

[5] M. Antonakakis, T. April, M. Bailey, and M. Bernhard, "Understanding the mirai botnet," in *Proc. 26th USENIX Secur. Symp. (USENIX Secur.)*, 2017, pp. 1093–1110.

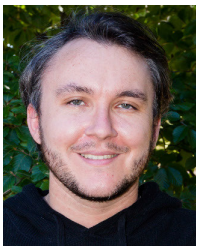
[6] N. Asokan, F. Brasser, A. Ibrahim, A.-R. Sadeghi, M. Schunter, G. Tsudik, and C. Wachsmann, "SEDA: Scalable embedded device attestation," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*. New York, NY, USA: ACM, 2015, pp. 964–975, doi: 10.1145/2810103.2813670.

[7] A. Ibrahim, A.-R. Sadeghi, and G. Tsudik, "US-AID: Unattended scalable attestation of IoT devices," in *Proc. IEEE 37th Symp. Reliable Distrib. Syst. (SRDS)*, Oct. 2018, pp. 21–30.

[8] F. Kohnhäuser, N. Büscher, and S. Katzenbeisser, "SALAD: Secure and lightweight attestation of highly dynamic and disruptive networks," in *Proc. Asia Conf. Comput. Commun. Secur. (ASIACCS)*. New York, NY, USA: ACM, 2018, pp. 329–342, doi: 10.1145/3196494.3196544.

[9] P.-H. Yang and S.-M. Yen, "SARA: Sandwiched attestation through remote agents for cluster-based wireless sensor networks," *Int. J. Distrib. Sensor Netw.*, vol. 13, no. 7, Jul. 2017, Art. no. 155014771719, doi: 10.1177/1550147717719192.

[10] A. Seshadri, A. Perrig, L. van Doorn, and P. Khosla, "SWATT: Software-based attestation for embedded devices," in *Proc. IEEE Symp. Secur. Privacy*, May 2004, pp. 272–282.
[11] X. Yang, X. He, W. Yu, J. Lin, R. Li, Q. Yang, and H. Song, "Towards a low-cost remote memory attestation for the smart grid," *Sensors*, vol. 15, no. 8, pp. 20799–20824, Aug. 2015.
[12] R. V. Steiner and E. Lupu, "Towards more practical software-based attestation," *Comput. Netw.*, vol. 149, pp. 43–55, Feb. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128618307631
[13] T. C. Group. (2008). *Trusted Platform Module (TPM) Summary*. [Online]. Available: https://trustedcomputinggroup.org/resource/trusted-platform-module-tpm-summary/
[14] G. Platform. (2015). *The Trusted Execution Environment: Delivering Enhanced Security at a Lower Cost to the Mobile Market*. [Online]. Available: https://globalplatform.org/wp-content/uploads/2018/04/GlobalPlatform_TEE_Whitepaper_2015.pdf
[15] C. Kil, E. C. Sezer, A. M. Azab, P. Ning, and X. Zhang, "Remote attestation to dynamic system properties: Towards providing complete system integrity evidence," in *Proc. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, Jun. 2009, pp. 115–124.
[16] T. Kobayashi, T. Sasaki, A. Jada, D. E. Asoni, and A. Perrig, "SAFES: Sand-boxed architecture for frequent environment self-measurement," in *Proc. 3rd Workshop Syst. Softw. Trusted Execution*. New York, NY, USA: ACM, Jan. 2018, pp. 37–41, doi: 10.1145/3268935.3268939.
[17] K. E. Defrawy, A. Francillon, D. Perito, and G. Tsudik, "SMART : Secure and minimal architecture for (establishing a dynamic) root of trust," in *Proc. Netw. Distrib. SystemSecurity Symp. (NDSS)*. Reston, VA, USA, Internet Society, 2012, pp. 1–15.
[18] P. Koeberl, S. Schulz, A.-R. Sadeghi, and V. Varadharajan, "TrustLite: A security architecture for tiny embedded devices," in *Proc. 9th Eur. Conf. Comput. Syst. (EuroSys)*. New York, NY, USA: ACM, 2014, pp. 10:1–10:14, doi: 10.1145/2592798.2592824.
[19] X. Carpent, N. Rattanavipanon, and G. Tsudik, "Remote attestation of IoT devices via SMARM: Shuffled measurements against roving malware," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, Apr. 2018, pp. 9–16.
[20] G. Dessouky, T. Abera, A. Ibrahim, and A.-R. Sadeghi, "LiteHAX: Lightweight hardware-assisted attestation of program execution," in *Proc. Int. Conf. Comput.-Aided Design*, Nov. 2018, pp. 1–8.
[21] M. Ambrosin, M. Conti, A. Ibrahim, G. Neven, A.-R. Sadeghi, and M. Schunter, "SANA: Secure and scalable aggregate network attestation," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*. New York, NY, USA: ACM, Oct. 2016, pp. 731–742, doi: 10.1145/2976749.2978335.
[22] M. Ambrosin, M. Conti, R. Lazzeretti, M. M. Rabbani, and S. Ranise, "PADS: Practical attestation for highly dynamic swarm topologies," *CoRR*, vol. abs/1806.05766, pp. 18–27, Sep. 2018. [Online]. Available: http://arxiv.org/abs/1806.05766
[23] A. Ibrahim, A.-R. Sadeghi, G. Tsudik, and S. Zeitouni, "DARPA: Device attestation resilient to physical attacks," in *Proc. 9th ACM Conf. Secur. Privacy Wireless Mobile Netw. (WiSec)*. New York, NY, USA: ACM, 2016, pp. 171–182, doi: 10.1145/2939918.2939938.
[24] M. Ammar, B. Crispo, and G. Tsudik, "SIMPLE: A remote attestation approach for resource-constrained IoT devices," in *Proc. ACM/IEEE 11th Int. Conf. Cyber-Phys. Syst. (ICCPS)*, Apr. 2020, pp. 247–258.
[25] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Trans. Inf. Theory*, vol. IT-29, no. 2, pp. 198–208, Mar. 1983.
[26] O. Ltd. (2020). *Omnet++ Discrete Event Simulator*. [Online]. Available: http://omnetpp.org/
[27] wolfSSL Inc. (2020). *WolfCrypt Embedded Crypto Engine*. [Online]. Available: https://www.wolfssl.com/docs/benchmarks/



L. MOREAU received the dual M.Sc. degree in information security from the University of Limoges, Limoges, France, and computer science from 3IL Engineering School, Limoges, in 2017. He is currently pursuing the Ph.D. degree in computer science with the University of Limoges, under a CIFRE thesis signed with ICOHUP Company, Limoges. His main research interests revolve around IoT security including both software protocols and hardware features.

E. CONCHON received the M.Sc. and Ph.D. degrees in wireless communication from the Institut National Polytechnique de Toulouse (INPT), Toulouse, France, in 2002 and 2006, respectively. In September 2008, he joined Champollion University as an Associate Professor and the Institut de Recherche en Informatique de Toulouse (IRIT) as a Researcher. Since September 2015, he has been an Associate Professor with the University of Limoges and a Researcher with XLIM. His research interests include security solutions for wireless networks, context-aware systems, and secure middleware solutions for health applications.

D. SAUVERON received the M.Sc. and Ph.D. degrees in computer science from the University of Bordeaux, France. He has been an Associate Professor (Habilitation) with the XLIM Laboratory (UMR CNRS 7252, University of Limoges, France), since 2006. He is the Dean of the Faculty of Science and Technology, University of Limoges. His research interests include smart card applications and security (at hardware and software level), RFID/NFC applications and security, mobile network applications and security (particularly UAV), sensor network applications and security, the Internet of Things (IoT) security, cyber-physical systems security, and security certification processes. Since 2011, he has been a member of the CNU 27, the National Council of Universities (for France). In December 2013, the General Assembly of the International Federation for Information Processing (IFIP) awarded him the IFIP Silver Core award for his work. He has been involved in more than 100 research events in a range of capacities (including the PC Chair, the General Chair, the Publicity Chair, an Editor/Guest Editor, a Steering Committee Member, and a Program Committee Member). Since 2014, he has been the Chair of IFIP WG 11.2 Pervasive Systems Security, having previously been appointed as a Vice-Chair of the working group. He has served as an External Reviewer for several Ph.D. thesis in foreign countries and France. More on `http://damien.sauveron.fr/`