HAGAR: Hashgraph-based Aggregated Communication and Remote Attestation

Jo Vliegen
ES&S, COSIC, ESAT, KU Leuven
jo.vliegen@kuleuven.be

Md Masoom Rabbani
ES&S, COSIC, ESAT, KU Leuven
mdmasoom.rabbani@kuleuven.be

Wouter Hellemans
ES&S, COSIC, ESAT, KU Leuven
wouter.hellemans@kuleuven.be

Nele Mentens
ES&S, COSIC, ESAT, KU Leuven
nele.mentens@kuleuven.be
LIACS, Leiden University, The Netherlands
n.mentens@liacs.leidenuniv.nl

ABSTRACT

Over the past decade, an exponential rise in the deployment of Internet-of-Things (IoT) devices engulfed our surroundings. IoT devices have spread into domains like personal smart devices, industrial applications, military applications, and medical applications, to name a few. Brittle security features and large deployment in safety-critical systems make IoT devices an attractive target for cyberattacks. Large collections of data, ranging from personal, and oversensitive to financial data, make it crucial to safeguard IoT applications and data communication from cybercriminals. One key technique to deal with these cyberattacks is Remote Attestation (RA), in which a verifier remotely checks the sanity of an IoT device's firmware. However, implementing RA over large IoT networks can be challenging due to the dynamic nature of the network and the real-time aggregation of attestation results.
We propose 'HAGAR: Hashgraph-based Aggregated Communication and Remote Attestation' to address the aforesaid challenges. HAGAR is a distributed ledger based on a hashgraph architecture that not only provides decentralized security guarantees like traditional blockchain technology, but also makes communication fast and thus offers continuous attestation aggregation in large IoT networks. We use the features of Hashgraphs for data aggregation in remote attestation mechanisms for large dynamic IoT networks.

CCS CONCEPTS

• **Security and privacy** → **Security protocols**; **Embedded systems security**; • **Computer systems organization** → **Embedded systems**;

KEYWORDS

Remote Attestation, Swarm Attestation, Network Security & Privacy, Hashgraph

1 INTRODUCTION

In the past decades, Internet-of-Things (IoT) devices have enriched us with numerous advancements. Recently, the use of IoT devices is no longer restricted to industrial or military applications but IoT applications are also actively used in our daily lives (e.g., smart appliances). These devices are often resource-constrained, and they work together to complete certain tasks. A group of collaborating devices in a specific task is called an 'IoT swarm'[1]. Although IoT networks bring a lot of added value, they also fuel security concerns due to their limitations of employing standard security measures. Therefore, IoT applications are becoming exceedingly targeted by hackers. Attacks like Stuxnet [1], Mirai Bot [7], and smartTVhack [25] have shown us their devastating capabilities. Moreover, the use of IoT in financial transactions makes them even more lucrative targets. Any attack on these devices can result in huge financial losses [3]. Thus, financial transactions executed by IoT devices must be secure and trusted to avoid being exploited by hackers.

Remote Attestation (RA) is an established security measure to detect adversarial presence in a device. The basic principle of RA comprises three main entities, namely a trusted party called the verifier, a potentially 'untrusted' device called the prover, and the network operator. In a typical RA protocol, the verifier sends a challenge to the prover. The prover computes a measurement (usually a hash) of its configuration (e.g., firmware and/or data) and sends its response back to the verifier. The integrity and authenticity of the response is protected by a Message Authentication Code (MAC) or is digitally signed. The received data are then checked by the verifier to see if they are legitimate, and if the received measurement corresponds to an expected 'good configuration' (e.g., one from a database of known 'golden-values') [15, 16]. Furthermore, the attestation process is generally triggered by the verifier, which is referred to as 'on-demand' attestation. While extremely effective in a single verifier-prover setting, RA is difficult to scale in its most basic form. Indeed, the overhead is proportional to the number of provers in the system, making it unsuitable for very large deployments or networks of autonomous devices. Nevertheless, in the past few years, researchers have proposed different schemes [4, 8, 13, 24] to address the scalability issues that come with the remote attestation of such an IoT network. In all these RA protocols, the trust is centralized (i.e., depends on the trustworthiness of the verifier), and researchers are yet to establish distributed trust among untrusted parties in the network [6].

[1] https://www.techrepublic.com/article/iot-device-swarm-intelligence-think-about-security-before-its-too-late/

One important technique to establish trust in a decentralized, untrusted network is to employ blockchain technology. The first use of blockchain technology was as part of the cryptocurrency Bitcoin [22]. Entities can establish trust in an untrusted network. Indeed, blockchain has a lot of potential, but there are some important things to consider before implementing blockchain technology in the context of swarm attestation. For example, the time to establish trust is relatively long (e.g., 10 minutes to authorize a block in Bitcoin). In some cases this can even take hours, which implies that we cannot use blockchain applications in our daily lives for small transactions and quick purchases [2]. Another concern for blockchain applications is that they have limited to no scalability. To deal with the scalability issue without compromising any security, researchers recently proposed schemes like IoTA [23] and Hashgraph [10]. These new schemes try to overcome the limitations of blockchain technology and to open the road for the adoption of distributed ledgers in IoT domains by adopting Directed Acyclic Graphs (DAGs) as a fundamental technique.

Idea and Contribution. In this paper, we propose a novel protocol called 'HAGAR: Hashgraph-based Aggregated Communication and Remote Attestation'. The aim of this paper is to provide an efficient and effective method for aggregating RA proofs from IoT devices in large networks. Overall, HAGAR provides the following main contributions:

(1) **Efficient continuous attestation evidence collection in large IoT networks.** Unlike state-of-the-art on-demand swarm attestation techniques, HAGAR continuously performs attestation (i.e., self-triggered attestation) and report collection. Therefore the network operator always has a 'view' of the network. Moreover, HAGAR exploits the secure and fast built-in message propagation and collection techniques of Hashgraph. HAGAR does not require an overlay in the form of a spanning tree to collect attestation results at the root verifier.
(2) **Efficient attestation of dynamic or unstructured IoT networks.** The assumption of an overlay spanning tree provides large IoT networks with the ability to perform and aggregate attestation results efficiently at the root verifier. However, the static network structure prevents device mobility during attestation and does not support devices joining or leaving the network during attestation. HAGAR does not only allow device mobility but also allows devices to join and/or leave the network during attestation. In other words, HAGAR does not depend on a static network constellation for the aggregation of the attestation results.
(3) **Decentralized Verification.** Different from classical collective attestation protocols, HAGAR does not depend on a centralized verification method. Individual IoT devices in a large network attest each other and reach a network-wide consensus. Nevertheless, depending on the application, a verifier can communicate with any of the devices in the network and upon authentication can get the aggregated (complete or partial) attestation result from the device, without performing any network-wide attestation operation.
(4) **Security against Distributed Denial-of-Service attacks & physical attacks.** HAGAR, by design principle, can 'partially' address Distributed Denial-of-Service (DDoS) attacks. A network can reach consensus as long as more than $\frac{2}{3}$ of the devices in the network are operational. Moreover, by the same principle, a physical adversary cannot prevent HAGAR to reach consensus as long as the network has more than $\frac{2}{3}$ devices in the correct operational state. In addition, it is very unlikely for a physical adversary to 'hijack' or destroy a large number of devices in a network without the network operator being alerted.
(5) **No connectivity assumption.** HAGAR, unlike most of the collective attestation techniques, does not require IoT devices to always be online and/or reachable during attestation.

2 RELATED WORK

Remote Attestation is a security technique that enables a trusted third party to verify whether or not a potentially untrusted device is compromised. In its most basic form, RA, while effective, is difficult to scale. Because the timing overhead is proportional to the number of devices in the system, traditional RA techniques are unsuitable for very large deployments or networks of autonomous and/or collaborating small devices. As a result, in recent years, researchers have started to develop novel protocols, called as 'swarm attestation', that allow RA to be scaled while maintaining its important security features [4, 5, 8, 14, 21, 24].

Swarm Attestation. Swarm attestation techniques like SEDA [8], SANA [4], and LISA [11] assume that the network is connected and static throughout the attestation period. As a result, these methods enable the network to be built as a balanced binary tree with parent-child relationships using a spanning tree overlay. Devices interact synchronously during attestation in these schemes: each device attests its child devices and reports back the aggregated attestation report to its parent. Other schemes such as SHeLA [24] assume the presence of a (relatively) powerful edge device between the verifier and the low-end devices in the network. The attestation process is performed by these edge devices, which give the network the flexibility to allow mobile devices. However, the underlying structure of the network is based on a spanning tree principle and the attestation aggregation takes time. Furthermore, techniques such as DARPA [17] and SeED [18] improve SEDA by allowing network devices to collaboratively identify hardware-compromised devices. However, they depend on the SEDA principle (i.e., spanning tree based network architecture rooted at the verifier for initiation of the attestation process along with the message aggregation) to aggregate attestation results and thus inherit SEDA's drawbacks.

In contrast to the aforementioned swarm attestation techniques, HAGAR does not depend on spanning tree based static network models and thus does not depend on continuous connectivity among devices. Moreover, HAGAR can withstand loss of connectivity in the network as long as $\frac{2}{3}$ of the network devices are online (i.e., in order to reach consensus).

Consensus-based Swarm Attestation. To support dynamic or unstructured large networks, researchers proposed dynamic swarm attestation techniques such as PADS [5] and SALAD [19]. Unlike classical swarm attestation schemes, PADS and SALAD do not assume complete network connectivity or the presence of an overlay of a spanning tree. These dynamic swarm attestation schemes exploit the concept of consensus. Any device in the network can communicate with any other device in the network. Upon mutual authentication, they share their respective views on the network. After receiving a message, every device performs 'consensus' and reaches the same 'view' about the network's health.

Although these approaches are effective in performing attestation in large dynamic networks, they require a significant amount of time to achieve consensus. Through the use of the Hashgraph mechanism, HAGAR is much faster in reaching network-wide consensus.

3 BACKGROUND

We now provide some background knowledge about the Hashgraph architecture and network view.

3.1 Hashgraph

Hashgraph is a distributed ledger technology [10] that offers an alternative to traditional blockchain. Unlike traditional blockchain, Hashgraph uses a Directed Acyclic Graph (DAG) data structure and a consensus algorithm known as 'gossip about gossip' to reach consensus on the state of the ledger. The data structure of Hashgraph is a combination of a hash and a graph. The hash is a simple algorithm that serves the security mechanism. The graph contains the entire history of with whom and in what specific order the nodes have gossiped. Hence the data structure is expandable because it can grow over time.

3.2 GOSSIP protocol based on Hashgraph

Hashgraph offers an advantage over traditional blockchain due to its efficient transaction processing. In Hashgraph, a device shares any event or message with two other devices at the same time, leading to high transaction throughput and low latency [9]. On the other hand, in traditional blockchains, each transaction must be validated and added to a block of transactions in the chain. This leads to slower transaction times and increased computational complexity. However, Hashgraph uses a distributed consensus protocol that allows for parallel validation of transactions known as 'gossip about gossip'. Consequently, this approach results in much faster transaction processing times compared to traditional blockchain technology. Gossip about gossip enables every node in Hashgraph to attain complete or partial network information. For example, in a network, any node α can synchronize with any two chosen nodes β and γ at any time. Through this process of gossiping, nodes in the network are able to build a complete view of the entire event history, and can then use this information to reach consensus on the state of the ledger. The algorithm considers the order and timing of the events to ensure that consensus is reached fairly and efficiently.

3.3 Network View through consensus

In HAGAR, we adapt the Hashgraph mechanism to aggregate the device's 'health' information (represented by Hash digests on the underlying firmware). While communicating, devices share respective aggregated knowledge of other devices' health with each other. For example, when devices α and β communicate, they acquire the network view and self-attestation report of each other. Upon verification (see Sect. 5.2), a device expands its 'view' by recording the attestation result of the encounter. This concept is taken further so communicating devices learn each other's 'view' on the network. As a result, devices in HAGAR can quickly get a 'snapshot' of the network status (i.e., completely or partially).

4 SYSTEM DETAILS

Before elaborating on HAGAR, this section discusses the assumptions on the system model and the adversary.

4.1 System Model Assumptions

We consider a network that consists of N IoT devices D_i, with $i = 1,2,\ldots,N$, a network operator and a number of verifier devices. We use the following terminology and assumptions for these entities:

- IoT device (**Dev**): Each **Dev** in the network is homogeneous. The devices can have different functionalities in terms of which parts of the firmware is executed, but the firmware itself is considered to be identical.
- Network Operator (**Op**): The **Op** ensures the secure bootstrap of the firmware on each deployed **Dev** D_i. Afterwards, it securely introduces the **Dev** in the network. This is a one-time process and generally happens during the initialization of the network setup procedure.
- Verifier (**Vrf**): A number of **Vrf** devices are also present in the network. These devices can be more computationally powerful compared to the **Dev** in the network. When a **Vrf** communicates with a **Dev** in the network, all the log entries that are stored in the **Dev** are transferred to the **Vrf**. The **Vrf** thus aggregates all attestation outcomes. It is the responsibility of the **Op** to decide how all attestation outcomes should be handled. This decision should be tailored to the intended network and should depend on the targeted application.

We assume the existence of three trusted components on a **Dev**, based on similar assumptions stated in the literature [5, 8, 13, 24]:

- Read-Only Memory (ROM): The read-only memory stores the firmware. Next to the **Dev**'s default program, it also lets the **Dev** perform the attestation-related operations and lets it create the messages for communication with other devices.
- Secure Key Storage: The memory region that stores cryptographic details for performing secure MAC operations. This storage is read-accessed only by HAGAR. Generally, during attestation, this memory area is not modified.
- Secure writable memory: This memory is used to store the modified aggregated message securely. Only HAGAR has read/write access to this memory area.

The aforesaid memory regions are protected and can be accessed only by authorized entities based on security guarantees.

4.2 Adversarial Assumptions

We consider the following adversaries for our networks. The behaviour of these adversaries is consistent with the adversary model described in [5, 20].

- Communication Adversary (Adv_{comm}): We consider that the Adv_{comm} has all the capabilities described in [12] and can obtain any message as a legitimate user of the network. Further, this adversary is also capable of a variety of attacks, including: (i) drop messages between devices; (ii) replay old messages from devices such as a correct attestation report even after a **Dev** has been compromised; and (iii) forge attestation reports and alter messages.
- Remote Software Adversary (Adv_{sw}): This type of adversary tries to corrupt the firmware of a **Dev** by injecting malware, altering (parts of) the firmware, or manipulating unprotected memory regions in a **Dev**. This adversary operates remotely.
- Mobile Adversary (Adv_{mob}): This type of adversary has the capability to corrupt a number of devices in the network by constantly changing its location from one device to another in the network.

In line with other RA literature [5, 8] we assume that an adversary cannot tamper with protected memory regions and we keep non-invasive physical attacks such as side-channel attacks out of our current scope. In addition, please note that HAGAR does not prevent Physical Adversary (Adv_{hw}) and DDoS attacks. However, as described in Section 1 HAGAR can partially withstand the presence of Adv_{hw} and DDoS attacks. Moreover, in Section 6 we describe a potential way to limit the DDoS attack.

4.3 Security Requirements.

In line with state-of-the-art RA protocols like [8, 13] HAGAR should satisfy the following security properties:

- **Integrity of the Dev.** HAGAR should provide reliable evidence that devices in the network are in a legitimate state and should guarantee the integrity of the underlying firmware of the **Dev**.
- **Authenticity of the Dev.** The *Adv* should not be able to forge the data and impersonate the **Dev**.
- **Integrity of communication data.** HAGAR should ensure that the communication among the devices is authentic and that the integrity of the data is preserved.
- **Freshness.** HAGAR should be able to recognize a corrupted device that publishes obsolete legitimate data in order to avoid detection of an ongoing attack.

5 HAGAR: HASHGRAPH-BASED AGGREGATED COMMUNICATION AND REMOTE ATTESTATION

Before presenting HAGAR's details, it is stressed that HAGAR will not take any action when a (possibly) malicious **Dev** is detected. The goal is to notify the **Op** who should take appropriate actions as the custom policy dictates.

HAGAR consists of three main phases: (1) deployment (2) message aggregation and attestation, and (3) verification. We present the notation of HAGAR in Table 1 and in the following, we provide comprehensive details for each of the phases of the protocol.

Table 1: Notation Summary

Term	Description
Vrf	Verifier
Dev	IoT Device
ID	Device's Unique Id
Att_S	Self-attestation of the firmware on device S
log	stored information
N	the network size
n	a subset of devices, with $n <$ **N**
Procedure	**Description**
H_i	hash corresponding to the network view of **Dev** i
σ	computation of MAC at sender's end
σ'	computation of MAC at receiver's end
$verify_MAC$	MAC verification operation to compare σ and σ'
$selfAtt$	computation of the individual device attestation

5.1 Phase 1: Deployment

A deployment phase is carried out before any other action for the secure setup of the network. It is an offline phase in which the **Op** bootstraps the devices. In addition, **Op** is also responsible for the key management of the devices. Further, **Vrf** receives the necessary information for the attestation operation in the deployment phase.

Key management. For the sake of simplicity, we have assumed symmetric-key cryptography based on a pre-shared key for the computation of the MAC. Please note that, with

different application scenarios and device capabilities, HAGAR could optionally adopt alternative key management schemes, including any public-key-based scheme.

5.2 Phase 2: Message aggregation and attestation

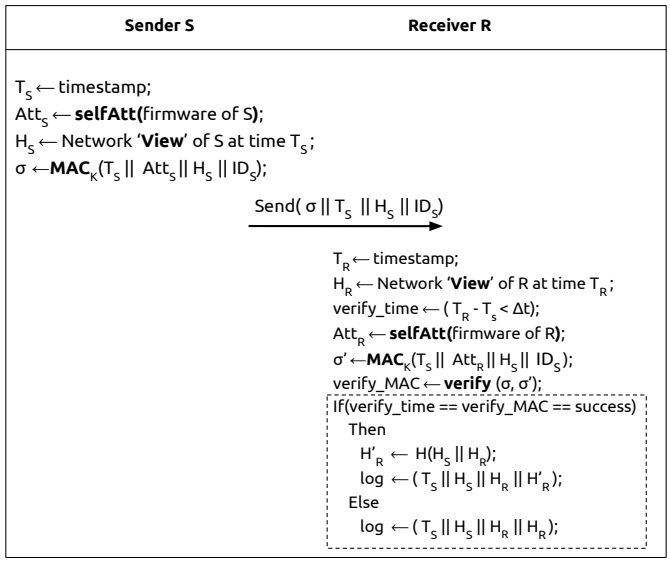

Figure 1: HAGAR - phase 2: Message aggregation and attestation

In HAGAR, every time the **Dev** sends or receives a message from its neighbours, it performs a self-attestation. Figure 1 shows the steps involved in HAGAR's message aggregation and attestation between a sender **Dev** S and receiver **Dev** R. Before sending a message **Dev** S performs the following: ① get current timestamp T_s, ② perform the self-attestation Att_s over the firmware of S, ③ get the most recent network 'view' of S at time T_S i.e., H_S , and ④ compute a MAC: $\sigma \leftarrow MAC_K(T_s||Att_s||H_S||ID_S)$. ⑤ Finally, the sender **Dev** S sends $(\sigma||T_s||H_S||ID_S)$ to a receiver **Dev** R. Although the message σ needs to be sent to 2 **Dev**'s, for the sake of simplicity, we have only shown one single receiving **Dev**.

The receiving **Dev** R ① first checks its own timestamp T_R, ② R computes its most recent network 'view' H_R at time T_R, ③ R verifies the received timestamp T_S. This is achieved by verifying that the difference with the incoming timestamp is within a predefined threshold: $T_R - T_S < \Delta t$. This step is required to thwart replay attacks. ④ After verifying the timestamp, R performs its own self-attestation Att_R. The computation of Att_R is essential because utilising Att_R the **Dev** R can validate the received MAC, as the firmware digest of every device is the same (i.e., homogeneous), ⑤ Upon completion of the self-attestation, **Dev** R also computes a MAC i.e., σ' over $(T_S||Att_R||H_S||ID_S)$, ⑥ R verifies (i.e., *verify_MAC*) (σ, σ'), ⑦ if the verification is successful then R updates its network 'view' to $H'_R \leftarrow$ H($H_R||H_S$). ⑧ Upon completion of the view update, R updates the **log**. The **log** consists of $(T_S||H_S||H_R||H'_R)$. Further, in case the *verify_MAC* operation fails then R does not update its current view but, nonetheless, updates **log** to reflect the unchanged view of R.

5.3 Phase 3: Verification

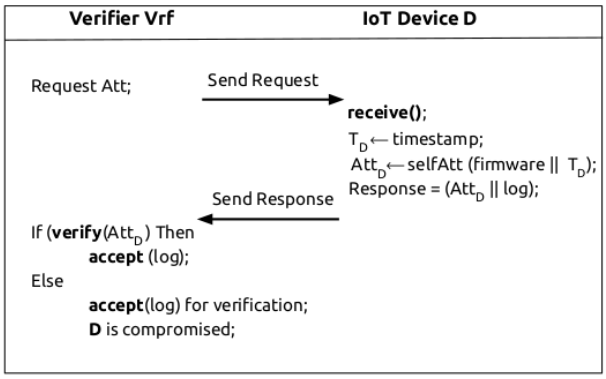

Figure 2: HAGAR - phase 3: Verification

In Figure 2, we explain the steps involved in the verification process. ① The **Vrf** initiates communication with a **Dev** D in the network by sending an attestation request, ② upon receiving such a request, D will perform self-attestation over its firmware, gets the current timestamp T_D and computes Att_D, ③ D replies the **Vrf** with the self-attestation result Att_D and 'current' *log*, ④ **Vrf** verifies the received attestation result and if it is legitimate then the **Vrf** accepts the *log* otherwise it creates an entry that states **Dev** D is compromised. Furthermore, **Vrf** collects the **log** from the compromised **Dev** to validate at which 'point in time' the **Dev** got compromised. Note that, in order to create a comprehensive 'network-wide' attestation view, the **Vrf** has to communicate with many devices. The number of **Vrf** devices greatly impacts the time that is required to aggregate the network-wide attestation result at the **Op**'s end.

6 SECURITY ANALYSIS

The goal of a remote adversary in a swarm attestation protocol is to alter the configuration (e.g., firmware) of one or more **Dev** and evade detection by the **Vrf**. HAGAR is a secure scheme if the MAC that is used in HAGAR is immune to selective forgery, and the message aggregation and distributed consensus is secure. In what follows, we sketch a security proof of HAGAR with respect to the adversarial assumptions in Sect. 4.2 and security requirements in Sect. 4.3.

- **Security against** Adv_{comm}. An Adv_{comm} can delay, drop or forge a message. In HAGAR, every message is cryptographically secured by the MAC on the message guaranteeing the authenticity and integrity of the message along with the **Dev**'s legitimacy. Indeed an Adv_{comm} can alter the plain text of the message and

generate a valid MAC. Nonetheless, this will be noticed as the comparison of σ and σ' will not succeed. As HAGAR follows the Hashgraph-based distributed message communication, this guarantees network-wide consensus among devices even if few devices fail to communicate.

- **Security against Replay Attack**. In order to detect a replay attack, the HAGAR protocol should guarantee the freshness of the attestation result. HAGAR can prevent replay attacks in line with Hashgraph. HAGAR forms a 'directed acyclic graph architecture (DAG)' which is a finite oriented graph where the timestamp-based 'transactions' are represented as points on this graph. These points can be connected with each other but can never form a loop. The formation of a loop will indicate the presence of a replay attack to the **Vrf**, or of a Hash collision.
- **Security against** Adv_{sw}. An Adv_{sw} can inject malware into one or more devices in the network. However, HAGAR is immune to this type of attack as during every event a **Dev** D_i will initiate attestation of the underlying software. The self-attestation process is secure as it is stored within the trusted-memory region. Thus, the presence of a Adv_{sw} will be notified and the integrity of the **Dev** is guaranteed.
- **Security against** Adv_{mob}. An Adv_{mob} tries to corrupt a number of devices in the network by changing its location continuously. The location change is also an evasive manoeuvre to evade detection by the **Vrf**. HAGAR conforms to the Hashgraph data structure. Every event in HAGAR will be registered within the **log**. This happens 'almost immediately' which means the copying happens as soon as one **Dev** gossips about that event and all other devices get to know about it. The **Vrf** will therefore know the exact position of the event in the history of the hash graph.
  Thus even if Adv_{mob} manages to corrupt a number of devices, HAGAR can still reach to consensus as long as more than $\frac{2}{3}$ of devices are legitimate.
- **Security against DDoS attacks**. The security of the HAGAR scheme strongly depends on the security of the Hashgraph protocol. A requirement for a properly functioning network is to be resistant to DDoS attacks. HAGAR is distributed in a manner that is DDoS resistant. A **Dev** can be overrun by an attacker with packets, resulting in a temporarily disconnection of the network, while the rest of the entire network can continue to function normally. An attack on the whole system would require attacking a large portion of the members which is much more difficult to achieve.
  Additionally, in the traditional blockchain, the longest chain rule is used to determine the ledger's true state, making it vulnerable to 51%-attacks. In contrast, Hashgraph uses a distributed consensus algorithm called "gossip about gossip" that does not rely on the longest chain rule. The consensus algorithm in Hashgraph takes into account the order and timing of events, which makes it more resistant to 51%-attacks.
- **Security against Man-in-the-middle attack.** A MAC is used instead of the more traditional digital signature. The reason for doing so is to lighten the required processing on the **Dev**. From a security point-of-view, this allows receiving devices to alter a message. In contrast with using digital signatures, this malicious behaviour would go undetected. However, to overcome this, the malicious **Dev**'s self-attestation will not be approved by any other **Dev** as the firmware differs which will be reflected in H_S upon sending a message. Upon aggregating attestation reports, the network operator will be able to detect and identify the malicious **Dev**.

7 SUMMARY AND DISCUSSION

In this section, we highlight the advantages and limitations of HAGAR.

Advantages. First, to the best of our knowledge, HAGAR is the first Hashgraph-based swarm attestation technique that continuously aggregates network-wide attestation reports. In addition, HAGAR can address the DDoS attack and Physical adversary as long as more than $\frac{2}{3}$ of the IoT devices in the network are not compromised.

The second important advantage of HAGAR is that it does not require the network to be static or organized as an overlay of spanning tree. Thus, it also eliminates the need of a **Vrf** to initiate attestation in contrast to the state-of-the-art which works on demand. The **Dev** in HAGAR can self-initiate the attestation process and corroborate in the process of attestation aggregation for the entire network through the 'gossip about gossip' mechanism.

Third, HAGAR is lightweight and can be employed by a low-end embedded devices.

Limitations. Indeed HAGAR introduces improvements with respect to state-of-the-art, on-demand swarm attestation techniques. Despite our efforts, HAGAR is not perfect and comes with some limitations.

First, HAGAR aims to aggregate the attestation result in the **log**. However, aggregation of large network-wide attestation is memory-consuming and thus unsuitable for very low-end embedded devices. To avoid this situation, HAGAR nodes normally monitor message logs and keep from filling up. Nodes can, for instance, prioritise messages or regularly erase their message logs. Furthermore, nodes can restrict the amount of RAM or storage they dedicate to message logs.

Second, although HAGAR can withstand the presence of a physical adversary, we did not provide any solution to counter a physical adversary.

Third, the presumption of a homogeneous network, where all devices possess the same firmware, may limit the applicability of HAGAR in accommodating various application scenarios. Nevertheless, the adoption of HAGAR can be extended to heterogeneous networks, wherein devices engage

in mutual authentication using symmetric or public keys before aggregating information. In this context, the verification process can be delegated to the verifier, relieving the burden on the participating devices. This adaptation enables the broader utilization of HAGAR across diverse network environments, accommodating the distinct requirements and configurations associated with heterogeneous deployments.

Finally, the **Op** has a continuous view on the health of the network. This is guaranteed by the fact that the entire hash-tree of the swarm can be reconstructed and should never fork. The chances of a fork occurring in the entire tree is kept minimal because of the redundancy that is inherently present.

8 CONCLUSION AND FUTURE WORK

This paper presented HAGAR, an effective, lightweight, and secure protocol for continuous attesting large networks of migrating, resource-constrained IoT devices. HAGAR overcomes the limitations of previous on-demand state-of-the-art swarm attestation schemes. It leverages the recently proposed concept of Hashgraph to securely aggregate attestation reports of segments of the entire network. HAGAR is strengthened by its built-in redundancy.

In future work, we will implement our protocol and evaluate the applicability and performance on real IoT swarms. Additionally, we will explore the usage of more powerful devices that can be used to offload the aggregated logs from the devices that are filled.

ACKNOWLEDGMENT

This work is partially supported by the COllective Research NETworking (CORNET) project "TrustedIOT: Trusted Computing Architectures for IoT Devices". The Belgian partners are funded by VLAIO under grant number HBC.2021.0895. This work is also partially supported by Cybersecurity Initiative Flanders (VR20192203). Additionally, Wouter Hellemans is an SB Ph.D. fellow at FWO (Research Foundation Flanders) under grant agreement 1SH3824N.

REFERENCES

[1] 2014. Countdown to Zero Day: Stuxnet and the Launch of the World's First Digital Weapon. https://www.wired.com/2014/11/countdown-to-zero-day-stuxnet.

[2] 2020. The 5 Big Problems With Blockchain Everyone Should Be Aware Of. https://www.bernardmarr.com/default.asp?contentID=1354.

[3] 2021. The Daily Swig. https://portswigger.net/daily-swig/ransomware.

[4] Moreno Ambrosin, Mauro Conti, Ahmad Ibrahim, Gregory Neven, Ahmad-Reza Sadeghi, and Matthias Schunter. 2016. SANA: Secure and Scalable Aggregate Network Attestation. In *CCS '16*.

[5] Moreno Ambrosin, Mauro Conti, Riccardo Lazzeretti, Md Masoom Rabbani, and Silvio Ranise. 2018. PADS: Practical Attestation for Highly Dynamic Swarm Topologies. In *2018 International Workshop on Secure Internet of Things (SIoT)*. 18–27. https://doi.org/10.1109/SIoT.2018.00009

[6] Moreno Ambrosin, Mauro Conti, Riccardo Lazzeretti, Md Masoom Rabbani, and Silvio Ranise. 2020. Collective Remote Attestation at the Internet of Things Scale: State-of-the-Art and Future Challenges. *IEEE Communications Surveys Tutorials* 22, 4 (2020), 2447–2461. https://doi.org/10.1109/COMST.2020.3008879

[7] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. 2017. Understanding the mirai botnet. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*. 1093–1110.

[8] N Asokan, Ferdinand Brasser, Ahmad Ibrahim, Ahmad-Reza Sadeghi, Matthias Schunter, Gene Tsudik, and Christian Wachsmann. 2015. SEDA: Scalable Embedded Device Attestation. In *CCS '15*. 964–975.

[9] Leemon Baird. 2016. The swirlds hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance. *Swirlds Tech Reports SWIRLDS-TR-2016-01, Tech. Rep* 34 (2016), 9–11.

[10] Leemon Baird, Mance Harmon, and Paul Madsen. 2017. Hedera: A Public Hashgraph Network & Governing Council. https://hedera.com/hh_whitepaper_v2.1-20200815.pdf. 2017 (2017).

[11] Xavier Carpent, Karim ElDefrawy, Norrathep Rattanavipanon, and Gene Tsudik. 2017. LIghtweight Swarm Attestation: a Tale of Two LISA-s. In *Proceedings of the 2017 ACM Asia Conference on Computer and Communications Security (ASIACCS '17)*. in press.

[12] Danny Dolev and Andrew Yao. 1983. On the Security of Public Key Protocols. 29, 2 (1983), 198–208.

[13] 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* 15 (2020), 3123–3136.

[14] Ambrosin et al. 2017. Toward Secure and Efficient Attestation for Highly Dynamic Swarms: Poster. In *WiSec '17*. 281–282.

[15] Aurelien Francillon, Quan Nguyen, Kasper Bonne Rasmussen, and Gene Tsudik. 2012. Systematic Treatment of Remote Attestation. *IACR Cryptology ePrint Archive* 2012 (2012), 713.

[16] Aurélien Francillon, Quan Nguyen, Kasper B Rasmussen, and Gene Tsudik. 2014. A Minimalist Approach to Remote Attestation. In *Proceedings of the conference on Design, Automation & Test in Europe (DATE '14)*. 244.

[17] Ahmad Ibrahim, Ahmad-Reza Sadeghi, Gene Tsudik, and Shaza 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)*. 171–182.

[18] Ahmad Ibrahim, Ahmad-Reza Sadeghi, and Shaza Zeitouni. 2017. SeED: Secure Non-Interactive Attestation for Embedded Devices. In *WiSec '17*.

[19] Florian Kohnhäuser, Niklas Büscher, and Stefan 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*.

[20] Florian Kohnhäuser, Niklas Büscher, and Stefan 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*. 329–342.

[21] Mohamad Mansouri, Wafa Ben Jaballah, Melek Önen, Md Masoom Rabbani, and Mauro Conti. 2021. FADIA: Fairness-Driven Collaborative Remote Attestation. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '21)*. Association for Computing Machinery, New York, NY, USA, 60–71. https://doi.org/10.1145/3448300.3468284

[22] Satoshi Nakamoto. 2008. Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf Accessed: 2015-07-01.

[23] Sergei Popov. 2017. Iota whitepaper. *Technical White Paper, year* 2017 (2017).

[24] M. M. Rabbani, J. Vliegen, J. Winderickx, M. Conti, and N. Mentens. 2019. SHeLA: Scalable Heterogeneous Layered Attestation. *IEEE Internet of Things Journal* (2019), 10240–10250.

[25] Rob Waugh. 2016. Smart TV hackers are filming people having sex on their sofas – and putting it on porn sites. http://metro.co.uk/2016/05/23/smart-tv-hackers-are-filming-people-having-sex-on-their-sofas-and-putting-it-on-porn-sites-5899248.