symmetry

MDPI

Article

SDATA: Symmetrical Device Identifier Composition Engine Complied Aggregate Trust Attestation

Fajiang Yu * and Yanting Huang

Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China; huangyantingwhu@whu.edu.cn
* Correspondence: fjyu@whu.edu.cn

Abstract: Efficient safeguarding of the security of interconnected devices, which are often resource-constrained, can be achieved through collective remote attestation schemes. However, in existing schemes, the attestation keys are independent of the device configuration, leading to increased requirements for the trusted computing base. This paper introduces a symmetrical aggregate trust attestation that is compatible with devices adhering to the device identifier composition engine framework. The proposed scheme simplifies the trusted computing base requirements by generating an attestation key that is derived from the device configuration. Moreover, the scheme employs distributed aggregate message authentication codes to reduce both the communication volume within the device network and the size of the attestation report, thereby enhancing the aggregation efficiency. In addition, the scheme incorporates interactive authentication to accurately identify compromised devices.

Keywords: device identifier composition engine; remote attestation; aggregate message authentication code

Citation: Yu, F.; Huang, F. SDATA: Symmetrical Device Identifier Composition Engine Complied Aggregate Trust Attestation. *Symmetry* **2024**, *16*, 310. https://doi.org/10.3390/sym16030310

Academic Editor: Lorentz Jäntschi

Received: 2 February 2024
Revised: 29 February 2024
Accepted: 3 March 2024
Published: 6 March 2024

1. Introduction

The Internet of Things (IoT) has a broad range of applications, extending across sectors such as intelligent logistics and smart medical technology. Recently, there has been a significant surge in security incidents, primarily due to cybercriminals compromising the embedded devices within the IoT infrastructure [1–4]. Given the critical importance of securing these IoT embedded devices, various attestation techniques rooted in trusted computing have been proposed [5–12]. These techniques aim to verify the integrity of these devices. Considering that IoT embedded devices often operate within networks, researchers have developed Collective Remote Attestation (CRA) schemes [13–28]. These schemes are proficient at performing remote attestation for networks of IoT devices.

In each existing CRA, the attestation key is independent of the device configuration. However, to prevent impersonation of the attester and to ensure that the report is consistent with the device configuration, both the measurement code and the attestation code need to be stored in the Trusted Computing Base (TCB). This requirement imposes a higher demand on the TCB in these schemes. During the attestation process, the measurement code is responsible for assessing the device configuration and obtaining metric values, and the attestation code employs the attestation key to either sign the report or generate a Message Authentication Code (MAC).

According to the types of attestation keys, CRAs can be categorized into two types: symmetric [13–19] and asymmetric [20–24]. Certain schemes, including SEDA [25], DARPA [26], ERASMUS [27], and SALAD [28], offer support for both symmetric and asymmetric cryptography. However, asymmetric schemes such as SANA [20], US-AID [21], ESDRA [22], and SARA [23] demand substantial resources, making them unsuitable for scenarios with resource constraints. As a result, our study will primarily concentrate on symmetric schemes.

Furthermore, existing symmetric CRAs can be broadly divided into two categories: centralized and distributed verification. Centralized verification, exemplified by SAP [13] and ERASMUS [27], involves a remote verifier solely performing the integrity checks of all devices. In contrast, distributed verification involves either self-verification of device integrity (as in LISA [14], slimIoT [15], PADS [16], SCAPI [17], SALAD [28]) or verification by another device (as in SEDA [25], DARPA [26], SeED [18], HEALED [19]). This process necessitates a device to either store reference values or receive them from an attestation request, leading to increased Trusted Computing Base (TCB) requirements as these values must be securely stored in a write-protected area of the memory. Storing the reference values on the device can pose an inconvenience for updating the device. Conversely, if the device receives reference values from an attestation request, it can result in an increased volume of network communication.

We introduce SDATA, a Symmetrical Device Identifier Composition Engine (DICE) complied Aggregate Trust Attestation, with the following features: (1) **Adaptability to DICE-Equipped Devices**: SDATA is adaptable to DICE-equipped devices with minimal TCB requirements. The TCB of SDATA only needs to ensure a measurement of the device's initial configuration and generate a unique device identifier from this measurement. There is no need to store the attestation code and reference values in the TCB. A comparison of the TCB requirements of SDATA and other schemes [13–19,25–28] is presented in Table 1. (2) **Efficient Use of Symmetric Aggregate Message Authentication Codes**: SDATA uses symmetric aggregate message authentication codes for efficiency, reducing both the communication volume within the device network and the size of the attestation report. Additionally, it employs distributed aggregation to decrease aggregation time. (3) **Support Identification**: SDATA incorporates interactive authentication to identify compromised devices.

Table 1. Comparison of TCB requirements.

	SEDA, DARPA, LISA, SeED, PADS, slimIoT, HEALED	SCAPI, SALAD	ERASMUS, SAP	SDATA
measurement	✓	✓	✓	only measure layer 0
attestation code	✓	✓	✓	✗
reference value	store	receive	✗	✗

Outline: Section 2 provides some preliminaries, Section 3 outlines SDATA, and Section 4 describes the workflow of SDATA in detail. Section 5 discusses the security analysis of SDATA, and Section 6 elaborates on the performance evaluation of SDATA. Section 7 explores the extension of SDATA.

2. Preliminaries

2.1. DICE

DICE, an architecture proposed by the Trusted Computing Group (TCG) [29], offers robust security foundations for systems with minimal silicon requirements and without a TPM. It operates in a chain-like manner to generate a device identifier (di^l) for each layer using a one-way function ($\mathtt{OWF}$). This process is based on the unique device secret (uds) and the configuration measurement (ci^l) of each layer, which are safeguarded by hardware during secure booting. Assuming that the last layer is layer h, layer h can derive a symmetric attestation key k using the $\mathsf{di}^l(l = h)$ with a key derivation function ($\mathtt{KDF}$). The process is as follows:

$$\text{layer 0: } \mathsf{di}^0 = \mathtt{OWF}(\mathsf{ci}^0, \mathsf{uds}), \text{ layer l: } \mathsf{di}^l = \mathtt{OWF}(\mathsf{ci}^l, \mathsf{di}^{l-1})(l = 1, ..., h); \mathsf{k} = \mathtt{KDF}(\mathsf{di}^l)(l = h)$$

2.2. Aggregate Message Authentication Code

The aggregate Message Authentication Code (MAC) [30] is a technique that facilitates the consolidation of multiple message authentication codes, generated by a variety of

senders, into a more compact authentication code using XOR or hash operations. This condensed code can be verified by a recipient who holds the secret keys of the senders. The XOR aggregation and verification processes are outlined as follows (where id_i represents an identity, m_i is a message, t_i is an MAC tag, k_i is a secret key, n is the number of aggregations, and $\dot{T}$ is an aggregate MAC):

$\mathtt{AggMAC}(\{(\mathsf{id}_1, m_1, t_1), \ldots, (\mathsf{id}_n, m_n, t_n)\}) \rightarrow \dot{T}$: take bitwise XOR of MAC tags $\dot{T} = \bigoplus_{i \in (1,\ldots,n)} t_i$

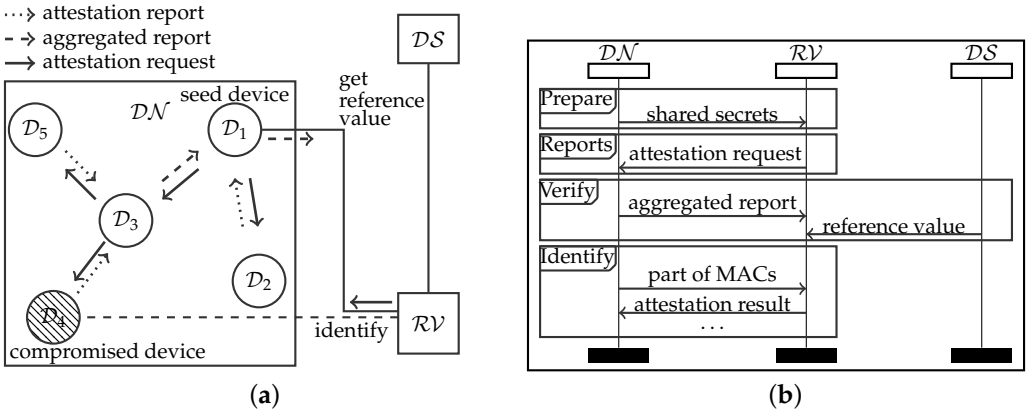
$\mathtt{AggVerify}(\{(\mathsf{id}_1, m_1, k_1), \ldots, (\mathsf{id}_n, m_n, k_n)\}, \dot{T}) \rightarrow \mathsf{result}$: for each $(\mathsf{id}_i, m_i, k_i)$, recompute $\ddot{t}_i = \mathtt{MAC}(k_i, m_i)$ and then recalculate $\ddot{T} = \bigoplus_{i \in (1,\ldots,n)} \ddot{t}_i$, if $\dot{T} = \ddot{T}$ holds, $\mathsf{result} = \mathsf{ACCEPT}$, otherwise, $\mathsf{result} = \mathsf{REJECT}$

3. SDATA

3.1. System Model

We consider a network, denoted as $\mathcal{DN}$, which consists of interconnected, resource-constrained devices. This network is composed of numerous embedded devices, represented as $\mathcal{D}_i$, equipped with DICE. The devices in $\mathcal{DN}$ undergo verification by a remote verifier ($\mathcal{RV}$), with the device supplier ($\mathcal{DS}$) providing reference values to facilitate this process. During an attestation, it is assumed that only a seed device, denoted as $\mathcal{D}_1$, exists within $\mathcal{DN}$. The primary responsibilities of this seed device include receiving an attestation request from the remote verifier $\mathcal{RV}$, forwarding this request to other devices within the network, generating an aggregate report of $\mathcal{DN}$, and subsequently transmitting this report to $\mathcal{RV}$. This process is depicted in Figure 1a. The devices within $\mathcal{DN}$ are capable of identifying and interacting with their immediate neighbors and communicating with $\mathcal{RV}$, and possess the computational ability to compute the MAC.

The objectives of SDATA are as follows: (1) Completeness: Assuming that all devices in $\mathcal{DN}$ are benign, $\mathcal{RV}$ should consistently produce a positive attestation outcome. (2) Scalability: The system should facilitate the remote verification of the integrity of a large $\mathcal{DN}$ as a whole. (3) Unforgeability: The system must be capable of detecting and declaring the network as untrustworthy if some devices are remotely compromised, or if some attestation reports are intentionally falsified. (4) Efficiency: The system should provide superior efficiency compared to individual attestations. (5) Identifiability: The system should possess the capability to identify compromised devices.



(**a**) (**b**)

Figure 1. (**a**) SDATA system architecture. (**b**) SDATA entity interaction.

3.2. Work Flow

SDATA consists of four system processes: prepare, reports, verify, and identify. The interactions between entities in these processes are depicted in Figure 1b.

Prepare. Before the deployment of any device, the identity of layer 0 for each device must be shared with $\mathcal{RV}$. Subsequently, during the booting process, each device generates a symmetric attestation key derived from the chained identity of its last layer.

Reports. Upon receiving an attestation request, each device employs its attestation key to produce an MAC for reporting its integrity. A device may also collect reports from its child devices, aggregate these accumulated MACs with its own, and then relay the consolidated outcome to its parent.

Verify. The $\mathcal{RV}$ verifies $\mathcal{DN}$ by reconstructing the attestation keys, which are based on the shared identities of devices at layer 0 and the device reference values from $\mathcal{DS}$, and subsequently verifies the aggregate MAC.

Identify. If the verification is unsuccessful, the devices in $\mathcal{DN}$ will attempt to send their stored aggregate reports to $\mathcal{RV}$ for individual verification. If some of these aggregate reports also fail the verification, the aforementioned procedure is repeated to identify compromised devices.

3.3. Security Assumptions and Threats

This section outlines the security assumptions and adversary model.

Security Assumptions. (1) TCB assumption: The TCB of SDATA is limited to the DICE layers in the devices of $\mathcal{DN}$. (2) It is assumed that the hash function, MAC algorithm, `OWF`, and `KDF` employed in SDATA are secure. (3) It is assumed that $\mathcal{RV}$ is honest and secure.

Adversary Model. (1) It is assumed that an adversary can access a device's data and code, with the exception of the DICE layer. And, the cold boot attack is not considered. (2) It is assumed that an adversary can eavesdrop, intercept, tamper with, and replay all messages exchanged between devices and between devices and $\mathcal{RV}$. (3) DoS attacks are not considered.

Given these security assumptions, SDATA is designed to thwart these potential attacks and achieve the objectives outlined in Section 3.1.

4. SDATA Design

This section explores the design of SDATA. For the purpose of clarity in the subsequent content, each device in the network $\mathcal{DN}$ will be denoted as $\mathcal{D}_i$, with the last layer of $\mathcal{D}_i$ referred to as layer h.

4.1. Prepare

During the secure booting process, the DICE layer of $\mathcal{D}_i$ serves as its TCB. It measures the layer 0 component and combines it with a uds to generate a unique identifier, di_i^0, for layer 0. A potential TCB implementation within $\mathcal{D}_i$ could include a processor boot Read-Only Memory (ROM) that houses both the uds_i and the DICE layer code. Additionally, a simple Memory Protection Unit (MPU) could be employed to restrict access to uds_i exclusively to the DICE layer.

The preparation process for device $\mathcal{D}_i$ is illustrated in Figure 2 and involves two scenarios: (1) **Pre-Deployment**: Prior to deployment, $\mathcal{D}_i$ shares the TCB-generated di_i^0 with $\mathcal{RV}$. (2) **Post-Deployment**: Upon deployment, $\mathcal{D}_i$ generates an identifier, $\mathsf{di}_i^l (l = h)$, in a chain-like manner, derived from di_i^0. Ultimately, the last layer, layer h, contains id_i (computed based on di_i^0 by layer 0), $\mathsf{di}_i^l (l = h)$, and cd_i. The cd_i includes the component information of layers $1, \ldots, h$. Subsequently, $\mathcal{D}_i$ generates an attestation key, k_i, based on $\mathsf{di}_i^l (l = h)$.

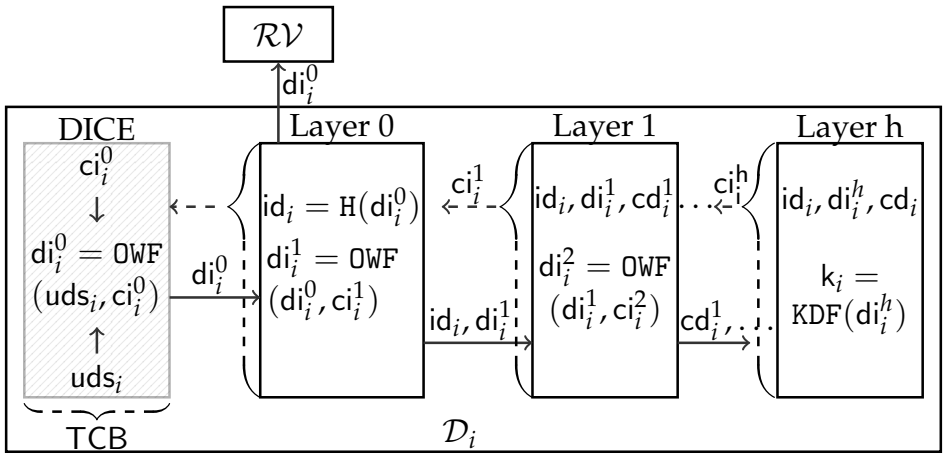


Figure 2. Device preparation process.

4.2. Reports

The attestation request, which includes a random number denoted as vn, is initiated by $\mathcal{RV}$. The seed device receives this request and broadcasts it throughout the network. Each device designates the first device that transmits the request as its parent and further disseminates the request. This dissemination culminates in a logical tree structure where each device represents a node.

4.2.1. Generate Individual Report

When an attestation request is received by the device $\mathcal{D}_i$, it generates its own random number, dn_i. $\mathcal{D}_i$ then constructs m_i by concatenating vn and dn_i. Following this, $\mathcal{D}_i$ employs k_i to calculate the MAC tags, denoted as $t_i = \mathtt{MAC}(k_i, m_i)$.

In conclusion, the individual report of $\mathcal{D}_i$ is $(\mathsf{id}_i, \mathsf{dn}_i, t_i, \mathsf{cd}_i)$.

4.2.2. Aggregate Reports

In the case where $\mathcal{D}_i$ is a non-leaf device, it first produces its own attestation report and then waits for a certain duration to gather reports from its child devices.

Suppose that $\mathcal{D}_i$ has gathered u individual reports from its leaf children, each denoted as $(\mathsf{id}_j, \mathsf{dn}_j, t_j, \mathsf{cd}_j)$, where $1 \leq j \leq u$. $\mathcal{D}_i$ then recalculates $m_j = \mathsf{vn} \| \mathsf{dn}_j$ for $j = 1, \ldots, u, i$. Following this, $\mathcal{D}_i$ aggregates $(\mathsf{id}_j, \mathsf{m}_j, t_j, \mathsf{cd}_j)$ $(j = 1, \ldots, u, i)$ according to $T_i^{'} \leftarrow \mathtt{AggMAC}(\{(\mathsf{id}_1, m_1, t_1), \ldots, (\mathsf{id}_u, m_u, t_u), (\mathsf{id}_i, m_i, t_i)\})$ as described in Section 2.2. Subsequently, $\mathcal{D}_i$ merges $(\mathsf{id}_j, \mathsf{dn}_j, \mathsf{cd}_j)$ $(j = 1, \ldots, u, i)$ into $(\hat{\mathsf{id}}_i^{'}, \hat{\mathsf{dn}}_i^{'}, \hat{\mathsf{cd}}_i^{'})$, resulting in an aggregate report $(T_i^{'}, \hat{\mathsf{id}}_i^{'}, \hat{\mathsf{dn}}_i^{'}, \hat{\mathsf{cd}}_i^{'})$. If $u = 0$, indicating that $\mathcal{D}_i$ has no leaf children, then $T_i^{'} = t_i, \hat{\mathsf{id}}_i^{'} = \{\mathsf{id}_i\}, \hat{\mathsf{dn}}_i^{'} = \{\mathsf{dn}_i\}, \hat{\mathsf{cd}}_i^{'} = \{\mathsf{cd}_i\}$.

Suppose that $\mathcal{D}_i$ has gathered v aggregate reports from its non-leaf children, each denoted as $(T_j, \hat{\mathsf{id}}_j, \hat{\mathsf{dn}}_j, \hat{\mathsf{cd}}_j)$, where $1 \leq j \leq v$. D_i aggregates each $(T_j, \hat{\mathsf{id}}_j, \hat{\mathsf{dn}}_j, \hat{\mathsf{cd}}_j)$ with $(T_i^{'}, \hat{\mathsf{id}}_i^{'}, \hat{\mathsf{dn}}_i^{'}, \hat{\mathsf{cd}}_i^{'})$:

$$T_i = T_i^{'} \bigoplus T_j, \hat{\mathsf{id}}_i = \hat{\mathsf{id}}_i^{'} \cup \hat{\mathsf{id}}_j, \hat{\mathsf{dn}}_i = \hat{\mathsf{dn}}_i^{'} \cup \hat{\mathsf{dn}}_j, \hat{\mathsf{cd}}_i = \hat{\mathsf{cd}}_i^{'} \cup \hat{\mathsf{cd}}_j \tag{1}$$

Subsequently, $\mathcal{D}_i$ forwards the aggregate report $(T_i, \hat{\mathsf{id}}_i, \hat{\mathsf{dn}}_i, \hat{\mathsf{cd}}_i)$ to its parent.

It is important to emphasize that $\mathcal{D}_i$ is required to temporarily hold u collected individual reports, its own individual report, and v collected aggregate reports, as well as the aggregate report $(T_i^{'}, \hat{\mathsf{id}}_i^{'}, \hat{\mathsf{dn}}_i^{'}, \hat{\mathsf{cd}}_i^{'})$ produced by $\mathcal{D}_i$, for a certain period. This storage facilitates the subsequent identification of potentially compromised devices.

Finally, the seed device generates an aggregate report for $\mathcal{DN}$, represented as $(T, \hat{\mathsf{id}}, \hat{\mathsf{dn}}, \hat{\mathsf{cd}})$, and forwards this report to $\mathcal{RV}$.

4.3. Verify

Upon receiving the aggregate report $(T, \hat{\mathsf{id}}, \hat{\mathsf{dn}}, \hat{\mathsf{cd}})$, which consolidates n individual reports, $\mathcal{RV}$ undertakes the following steps:

(1) For $(\hat{\mathsf{id}}, \hat{\mathsf{dn}}, \hat{\mathsf{cd}})$, $\mathcal{RV}$ filters out duplicate layer component information from $\hat{\mathsf{cd}}$, resulting in a set of distinct layer component information $\{\mathsf{lc}_1, \ldots, \mathsf{lc}_x\}$. $\mathcal{RV}$ then retrieves the component reference values from $\mathcal{DS}$ based on $\{\mathsf{lc}_1, \ldots, \mathsf{lc}_x\}$. For each $(\mathsf{id}_i, \mathsf{dn}_i, \mathsf{cd}_i)$ where $1 \leq i \leq n$, $\mathcal{RV}$ maps the obtained values $\mathsf{ci}_1, \ldots, \mathsf{ci}_x$ to each layer's integrity reference value $\tilde{\mathsf{ci}}_i^{\,l}$ $(l = 1, \ldots, h)$.

(2) Subsequently, $\mathcal{RV}$ reconstructs the attestation key $\mathsf{k}_i^{'}$ of $\mathcal{D}_i$ based on the shared identifier di_i^0 using the following formulas:

$$\mathsf{k}_i^{'} = \mathtt{KDF}(\tilde{\mathsf{di}}_i^{\,l})(l = h), \tilde{\mathsf{di}}_i^{\,l} = \mathtt{OWF}(\tilde{\mathsf{di}}_i^{\,l-1}, \tilde{\mathsf{ci}}_i^{\,l})(l = 2, \ldots, h), \tilde{\mathsf{di}}_i^{\,1} = \mathtt{OWF}(\mathsf{di}_i^0, \tilde{\mathsf{ci}}_i^{\,1}) \tag{2}$$

Following this, $\mathcal{RV}$ recalculates $m_i = \mathsf{vn} \| \mathsf{dn}_i$.

(3) $\mathcal{RV}$ then verifies the aggregate MAC T according to $\mathsf{result} \leftarrow \mathtt{AggVerify}(\{(\mathsf{id}_1, m_1, k_1^{'}), \ldots, (\mathsf{id}_n, m_n, k_n^{'})\}, T)$, as described in Section 2.2. If the value of result is ACCEPT, this signifies that $\mathcal{DN}$ is deemed trustworthy. Conversely, $\mathcal{DN}$ is untrustworthy.

4.4. Identify

In the event that the outcome in Section 4.3 is REJECT, a subsequent procedure can be employed to identify compromised devices. Let $\mathbb{I}$ represent the set of devices that have not been verified, and $\mathbb{J}$ denote the set of compromised devices. Both sets are initially assigned the value $\hat{\mathsf{id}}$.

(1) Upon receiving an identification request from $\mathcal{RV}$ or the parent, a device $\mathcal{D}_i$ ($i \in \{1, \ldots, n\}$, where the initial value of i is 1) selects an aggregate report from its temporary storage, denoted as $(T_{\mathbb{S}}, \hat{\mathsf{id}_{\mathbb{S}}}, \hat{\mathsf{dn}_{\mathbb{S}}}, \hat{\mathsf{cd}_{\mathbb{S}}})$, and sends it to $\mathcal{RV}$.

(2) If the verification result from $\mathcal{RV}$ is ACCEPT, $\mathbb{J} \leftarrow \mathbb{J} \backslash \hat{\mathsf{id}_{\mathbb{S}}}, \mathbb{I} \leftarrow \mathbb{I} \backslash \hat{\mathsf{id}_{\mathbb{S}}}$. If $\mathbb{I} \neq \emptyset$, the device goes to step (1) with another aggregate report.

(3) If the verification result is REJECT, there are two potential scenarios:

If $(T_{\mathbb{S}}, \hat{\mathsf{id}_{\mathbb{S}}}, \hat{\mathsf{dn}_{\mathbb{S}}}, \hat{\mathsf{cd}_{\mathbb{S}}})$ represents the aggregate report from itself and its leaf child nodes, it sends $u + 1$ stored individual reports to $\mathcal{RV}$. Then, set $\mathbb{I} \leftarrow \mathbb{I} \backslash \hat{\mathsf{id}_{\mathbb{S}}}$, and the device IDs that were successfully verified are removed from $\mathbb{J}$. If $\mathbb{I} \neq \emptyset$, the device goes to step (1) with another aggregate report.

If $(T_{\mathbb{S}}, \hat{\mathsf{id}_{\mathbb{S}}}, \hat{\mathsf{dn}_{\mathbb{S}}}, \hat{\mathsf{cd}_{\mathbb{S}}})$ is one of the other aggregate reports, the device notifies the corresponding non-leaf child node. This non-leaf child node then goes to step (1).

The final output is a set $\mathbb{J}$ consisting of the IDs of the compromised devices. Based on the identification results, the administrator of $\mathcal{DN}$ has the ability to repair these compromised devices.

In conclusion, Figure 3 illustrates the comprehensive attestation process of SDATA.

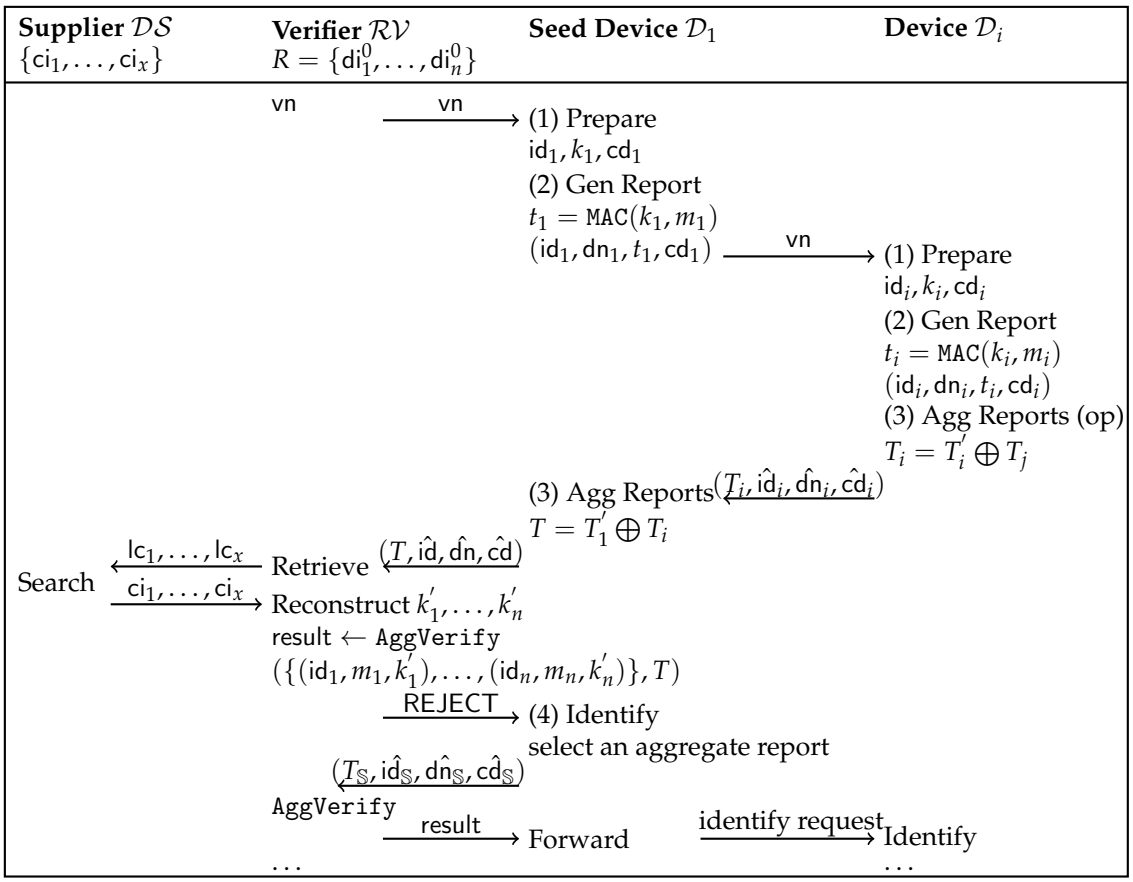


Figure 3. SDATA attestation process.

5. Security Analysis

This section will provide a proof of completeness and an elaboration on unforgeability.

5.1. Completeness

As described in Figure 2 in Section 4.1, the attestation key of $\mathcal{D}_i$ is given by $\mathsf{k}_i = \mathtt{KDF}(\mathsf{di}_i^l)$ $(l = h)$, where $\mathsf{di}_i^l = \mathtt{OWF}(\mathsf{di}_i^{l-1}, \mathsf{ci}_i^l)(l = 1, \ldots, h)$, $\mathsf{di}_i^0 = \mathtt{OWF}(\mathsf{uds}_i, \mathsf{ci}_i^0)$. Assuming that $\mathcal{RV}$ is honest, $\mathcal{D}_i$ is benign, and cd_i in the aggregate report remains constant, the $\tilde{\mathsf{ci}}_i^l$ $(l = 1, \ldots, h)$ retrieved by $\mathcal{RV}$ from $\mathcal{DS}$ and the corresponding ci_i^l will be identical. This ensures that the reconstructed key $\mathsf{k}_i^{'}$ by $\mathcal{RV}$ will match k_i. If each reassembled key is consistent with the attestation key of its respective device, and dn_i and the individual random number of each device $\mathcal{D}_i$, which is part of the message m_i $(i = 1, \ldots, n)$, along with the tag T and id contained in the aggregate report, remain constant, then $\mathcal{RV}$ will execute the AggVerify algorithm and invariably receive an ACCEPT outcome. As a result, $\mathcal{RV}$ will consistently yield a positive attestation outcome for $\mathcal{DN}$.

5.2. Unforgeability

In SDATA, the attestation code is not part of the TCB, yet it can still prevent report forgery attacks. Consider a scenario where a device $\mathcal{D}_i$ is compromised and at least one layer l $(l \in \{0, \ldots, h\})$ has been breached. In this case, $\mathcal{D}_i$ can exhibit three main types of report forgery abnormalities:

(1) $\mathcal{D}_i$ attempts to generate a valid report: If an adversary tries to alter a layer l while causing $\mathcal{D}_i$ to generate a valid report, the measurements ci_i^l computed by layer $l-1$ should match the corresponding $\tilde{\mathsf{ci}}_i^l$ $(l = 0, \ldots, h)$ provided by $\mathcal{DS}$. However, this is not the expected behavior of layer $l-1$. Consequently, this would necessitate modifying the code

in layer $l-1$. Following the same reasoning, it would also require altering the DICE layer. But, this contravenes the TCB assumption outlined in Section 3.3.

(2) $\mathcal{D}_i$ attempts to obtain the correct attestation key k_j to impersonate $\mathcal{D}_j$:

$\mathcal{D}_i$ tries to generate a k_j on its own. $\mathsf{k}_j = \mathtt{KDF}(\mathsf{di}_j^l)(l = h)$, where $\mathsf{di}_j^l = \mathtt{OWF}(\mathsf{di}_j^{l-1}, \mathsf{ci}_j^l)(l = 1, \ldots, h), \mathsf{di}_j^0 = \mathtt{OWF}(\mathsf{uds}_j, \mathsf{ci}_j^0)$. Since uds_j cannot be accessed outside of the DICE layer of $\mathcal{D}_j$, di_j^0 cannot be leaked by $\mathcal{RV}$, and di_j^l $(l = 0, \ldots, h)$ cannot be leaked by $\mathcal{D}_j$'s layer l (as explained in the following sub-case). Therefore, $\mathcal{D}_i$ cannot generate a k_j.

$\mathcal{D}_i$ tries to obtain secrets from $\mathcal{D}_j$. If the compromised layer l $(l = 0, \ldots, h-1)$ of $\mathcal{D}_j$ attempts to transmit di_j^l or di_j^{l+1}, or the compromised layer h attempts to transmit k_j to $\mathcal{D}_i$, as described in (1), di_j^l, …, di_j^h, or k_j will be incorrect.

This implies that, under any circumstances, $\mathcal{D}_i$ cannot obtain the correct attestation key of $\mathcal{D}_j$. Therefore, $\mathcal{D}_i$ cannot successfully impersonate $\mathcal{D}_j$.

(3) $\mathcal{D}_i$ attempts to perform replay attacks: Each attestation request includes a random number vn and each individual report contains a device random number $\mathsf{dn_i}$. If $\mathcal{D}_i$ attempts to replay the report, $\mathcal{RV}$ will detect it during the computation of m_i.

6. Performance Evaluation

In our experiment, each device node is simulated by a process, and devices within $\mathcal{DN}$ utilize the gossip protocol [31] for communication. The hash and HMAC functions are facilitated by the libsecp256k1 library [32].

The evaluation experiments are carried out on Ubuntu 20.04 in a VMware (VM) environment. The VM is hosted on a physical machine outfitted with a 13th Gen Intel(R) Core(TM) i7-13700H @ 2.40 GHz processor and 32 GB of RAM.

All existing symmetric CRAs utilize Trusted Execution Environment (TEE) architectures, such as SMART [33], TrustLite [34], and TyTan [35]. These architectures are distinct from SDATA; hence, a comparison with them has not been conducted. A series of experiments were conducted, involving varying numbers of devices within $\mathcal{DN}$, to assess both its communication volume and aggregation times.

6.1. Report Size and Communication Volume

Firstly, the primary communication data between the seed device and $\mathcal{RV}$ are the aggregate report. In our experiment, the sizes of the attestation key, random number, and MAC tag are all 32B. According to the specification [36], it is assumed that the size of the detailed information for each layer's component is 200 B, although this may not be necessary in practical applications. It is also assumed that the number of device layers is 3, and the number of devices in $\mathcal{DN}$ is n. An individual attestation report of a device $\mathcal{D}_i$ is represented as $(\mathsf{id}_i, \mathsf{dn}_i, t_i, \mathsf{cd}_i)$, as described in Section 4.2, and its size is 496B. Without aggregation, the total reports of $\mathcal{DN}$ are represented as $(\hat{t}, \hat{\mathsf{id}}, \hat{\mathsf{dn}}, \hat{\mathsf{cd}})$, and their size is $496n$B. However, with aggregation, the aggregate report of $\mathcal{DN}$ is represented as $(T, \hat{\mathsf{id}}, \hat{\mathsf{dn}}, \hat{\mathsf{cd}})$, requiring only $(464n + 32)$B. In practical applications, the network between the seed device and $\mathcal{RV}$ is typically a high-load network, and the size of the report is not a significant concern.

Secondly, $\mathcal{DN}$ is a resource-constrained network. The hop-byte metric is employed to evaluate the communication volume within $\mathcal{DN}$, referring to a byte transmitted between a child node and its parent node. It is important to note that the communication volume that we subsequently consider only includes MAC tags in reports during the attestation response process, excluding identification. Consider a device, $\mathcal{D}_i$, a non-leaf node in a spanning tree generated during a complete attestation, with x descendants. Without aggregation, $\mathcal{D}_i$ would directly merge $x + 1$ MAC tags, including its own report and the individual reports of all its descendants, and transmit a total of $32(x + 1)$ hop-bytes to its parent. In contrast, with aggregation, as described in Section 4.2.2, the aggregate report of $\mathcal{D}_i$ would only transmit an aggregate MAC of 32 hop-bytes to its parent. This aggregation leads to a reduction in the communication volume within $\mathcal{DN}$.

Figure 4a illustrates the communication volume within $\mathcal{DN}$ under both aggregation and non-aggregation scenarios, across varying numbers of devices. The height of an entire bar represents the communication volume within $\mathcal{DN}$ for a specific number of devices. In the case of aggregation, the total communication volume remains consistent for a given number of devices, eliminating the need for stacked bars. On the other hand, in the non-aggregation stacked bar, the series y $(y = 1, \ldots, 7)$ represents the average hop-bytes transmitted from level y to level $y - 1$ over 20 runs. Within a spanning tree, the root node is at level 0, its child nodes are at level 1, and so on. As depicted in the figure, the communication volume with aggregation is only equal to the hop-bytes of level 1 without aggregation. As the number of devices increases, the communication volume with aggregation decreases significantly.

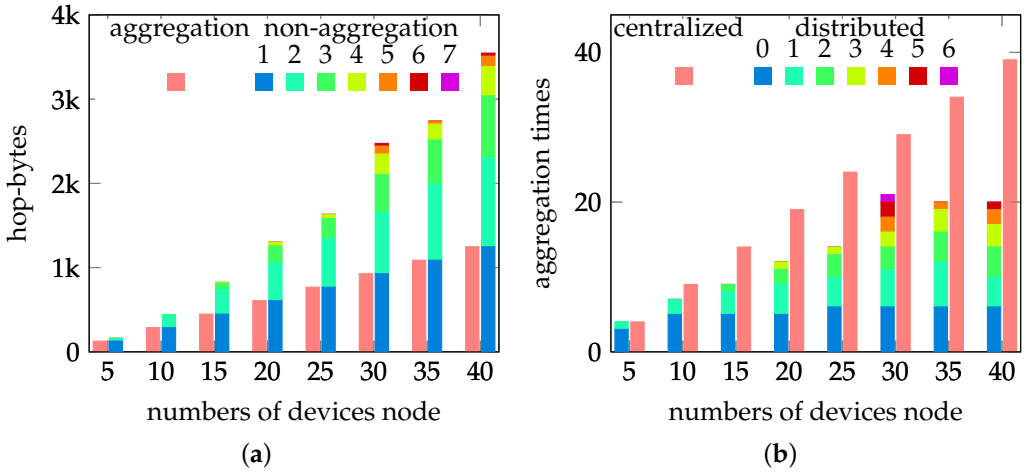


Figure 4. Evaluation of communication volume and aggregation times. (**a**) Communication volume. (**b**) Aggregation times.

6.2. Distributed Aggregation

In the SDATA system, which is inherently distributed, aggregation is performed concurrently across multiple non-leaf devices. The series y $(y = 0, \ldots, 6)$ in the stacked bar of Figure 4b represents the maximum aggregation times of level y nodes in a spanning tree, averaged over 20 runs. The total height of a bar signifies the time taken to generate the complete report of $\mathcal{DN}$. In the case of centralized aggregation, the seed device sequentially consolidates all individual reports from its descendants, and the aggregation times remain consistent for a given number of devices. As depicted in Figure 4b, distributed aggregation is more efficient than centralized aggregation. Notably, as the number of devices increases, the efficiency of aggregation significantly improves.

7. Discussion

The related work [13–28] is illustrated in the introduction. Existing attestation methods are classified into symmetric and asymmetric categories. Within the symmetric category, further division is made into centralized and distributed verification. A common characteristic across these attestation schemes is that the attestation key is independent of the device configuration. This aspect distinguishes them from SDATA. Further details will not be elaborated on in this section.

The extension of the SDATA system is being explored, with the identification process being the initial focus for refinement. The proposed identification process of SDATA in this paper is interactive. Devices select the next group for testing after receiving the verification result from the previous group, a feature that is characteristic of adaptive group testing. An adaptive group testing protocol [37] could be employed to strike a balance between the number of interactions and the volume of communication. This could involve the use of binary search, the rake-and-winnow algorithm, Li's multi-stage algorithm, and the

digging algorithm. Alternatively, the identification process could be transformed into a non-interactive method using non-adaptive group testing [38]. This method would use a disjunct matrix to divide the attestation device into multiple test sets, and then send the attestation results of multiple test sets to the verifier at once, thereby reducing the number of interactions.

This section also includes a discussion on the attestation-key-sharing aspect of SDATA. In SDATA, each device is required to share its layer 0 identifier with $\mathcal{RV}$, necessitating $\mathcal{RV}$ to store identifiers of all devices. For a closed scenario, an alternative strategy could be to share only the di_1^0 of the seed device with $\mathcal{RV}$, while other devices use di_1^0 as their unique device secrets. Then, $\mathcal{RV}$ can still reconstruct the attestation keys of the devices using di_1^0 and the reported components' detailed information.

However, symmetric attestation secret sharing is not suitable for open scenarios, such as the remote attestation of confidential container clusters or other systems in cloud computing [39–43]. Instead, aggregate remote attestation based on asymmetric cryptography could be considered for confidential container clusters, as it eliminates the need for secret sharing.

8. Conclusions

We present SDATA, an efficient symmetric aggregate trust attestation that is compliant with DICE and is designed for a network of interconnected low-end devices. The TCB of SDATA only needs to accurately generate an identifier for device layer 0. Subsequently, the device derives an attestation key associated with its configuration, eliminating the need to store the attestation code and reference values. This feature simplifies the TCB requirements compared to other schemes while still offering protection against forgery attacks. Moreover, the use of an aggregate MAC significantly reduces both the communication volume within the device network and the size of the attestation report. The distributed aggregation further enhances the aggregation efficiency. SDATA employs interactive authentication to identify compromised devices.

Author Contributions: Conceptualization, F.Y. and Y.H.; methodology, F.Y.; software, Y.H.; validation, F.Y. and Y.H.; formal analysis, F.Y.; investigation, F.Y. and Y.H.; resources, F.Y. and Y.H.; data curation, Y.H.; writing—original draft preparation, Y.H.; writing—review and editing, F.Y. and Y.H.; visualization, Y.H.; supervision, F.Y.; project administration, Y.H.; funding acquisition, F.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China under Grant No. 61772384.

Data Availability Statement: The data used in the experiment are detailed in the performance evaluation section.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Margolis, J.; Oh, T.T.; Jadhav, S.; Kim, Y.H.; Kim, J.N. An in-depth analysis of the mirai botnet. In Proceedings of the 2017 International Conference on Software Security and Assurance (ICSSA), Altoona, PA, USA, 24–25 July 2017; pp. 6–12.
2. A Bug in Smart Meters in Spain Could Cause Widespread Blackouts. Available online: http://www.freebuf.com/news/47634.html (accessed on 1 December 2023).
3. Overview of IoT Threats in 2023. Available online: https://securelist.com/iot-threat-report-2023/110644/ (accessed on 1 December 2023).
4. The 2023 IoT Security Landscape Report. Available online: https://www.bitdefender.com/files/News/CaseStudies/study/429/2023-IoT-Security-Landscape-Report.pdf (accessed on 1 December 2023).
5. Ammar, M.; Crispo, B.; Tsudik, G. SIMPLE: A Remote Attestation Approach for Resource-constrained IoT devices. In Proceedings of the 2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS), Sydney, NSW, Australia, 21–25 April 2020; pp. 247–258.
6. Kuang, B.; Fu, A.; Susilo, W.; Yu, S.; Gao, Y. A survey of remote attestation in Internet of Things: Attacks, countermeasures, and prospects. *Comput. Secur.* **2022**, *112*, 102498. [CrossRef]

7. De Oliveira Nunes, I.; Jakkamsetti, S.; Rattanavipanon, N.; Tsudik, G. On the TOCTOU problem in remote attestation. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, 15–19 November 2021; pp. 2921–2936.
8. Helble, S.C.; Kretz, I.D.; Loscocco, P.A.; Ramsdell, J.D.; Rowe, P.D.; Alexander, P. Flexible Mechanisms for Remote Attestation. *Assoc. Comput. Mach.* **2021**, *24*, 2471–2566. [CrossRef]
9. Tan, H.; Tsudik, G.; Jha, S. MTRA: Multiple-tier remote attestation in IoT networks. In Proceedings of the 2017 IEEE Conference on Communications and Network Security (CNS), Las Vegas, NV, USA, 9–11 October 2017; pp. 1–9.
10. De Oliveira Nunes, I.; Eldefrawy, K.; Rattanavipanon N.; Steiner M.; Tsudik G. VRASED: A Verified Hardware/Software Co-Design for Remote Attestation. In Proceedings of the 28th USENIX Security Symposium (USENIX Security 19), Santa Clara, CA, USA, 14–16 August 2019; pp. 1429–1446.
11. Román, R.; Arjona, R.; Baturone, I. A lightweight remote attestation using PUFs and hash-based signatures for low-end IoT devices. *Future Gener. Comput. Syst.* **2023**, *148*, 425–435. [CrossRef]
12. Cao, J.; Zhu, T.; Ma, R.; Guo, Z.; Zhang Y.; Li, H. A Software-Based Remote Attestation Scheme for Internet of Things Devices. *IEEE Trans. Dependable Secur. Comput.* **2023**, *20*, 1422–1434. [CrossRef]
13. De Oliveira Nunes, I.; Dessouky, G.; Ibrahim, A.; Rattanavipanon, N.; Sadeghi, A.; Tsudik, G. Towards Systematic Design of Collective Remote Attestation Protocols. In Proceedings of the 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), Dallas, TX, USA, 7–10 July 2019; pp. 1188–1198.
14. Carpent, X.; ElDefrawy, K.; Rattanavipanon, N.; Tsudik, G. Lightweight swarm attestation: A tale of two lisa-s. In Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, Abu Dhabi, United Arab Emirates, 2–6 April 2017; pp. 86–100.
15. Ammar, M.; Washha, M.; Ramabhadran, G.S.; Crispo, B. SlimIoT: Scalable Lightweight Attestation Protocol for the Internet of Things. In Proceedings of the 2018 IEEE Conference on Dependable and Secure Computing (DSC), Kaohsiung, Taiwan, 10–13 December 2018; pp. 1–8.
16. Ambrosin, M.; Conti, M.; Lazzeretti, R.; Rabbani, M.M.; Ranise, S.PADS: Practical Attestation for Highly Dynamic Swarm Topologies. In Proceedings of the 2018 International Workshop on Secure Internet of Things (SIoT), Barcelona, Spain, 6 September 2018; pp. 18–27.
17. Kohnhäuser, F.; Büscher, N.; Gabmeyer, S.; Katzenbeisser, S. SCAPI: A Scalable Attestation Protocol to Detect Software and Physical Attacks. Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks, Boston, MA, USA, 18–20 July 2017; pp. 75–86.
18. Ibrahim, A.; Sadeghi, A.; Zeitouni, S. SeED: Secure Non-Interactive Attestation for embedded device. In Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks, Boston, MA, USA, 18–20 July 2017; pp. 64–74.
19. Ibrahim, A.; Sadeghi, AR.; Tsudik, G. HEALED: HEaling & Attestation for Low-End Embedded Devices. In Proceedings of the Financial Cryptography and Data Security, Frigate Bay, St. Kitts and Nevis, 18–22 February 2019; pp. 627–645.
20. Ambrosin, M.; Conti, M.; Ibrahim, A.; Neven, G.; Sadeghi, A.; Schunter, M. SANA: Secure and Scalable Aggregate Network Attestation. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, New York, NY, USA, 24–28 October 2016; pp. 731–742.
21. Ibrahim, A.; Sadeghi, A.; Tsudik, G. US-AID: Unattended Scalable Attestation of IoT Devices. In Proceedings of the 2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS), Salvador, Brazil, 2–5 October 2018; pp. 21–30.
22. Kuang, B.Y.; Fu, A.; Yu, S.; Yang, G.M.; Su, M.; Zhang, Y.Q. ESDRA: An Efficient and Secure Distributed Remote Attestation Scheme for IoT Swarms. *IEEE Internet Things J.* **2019**, *6*, 8372–8383. [CrossRef]
23. Dushku, E.; Rabbani, M.M.; Conti, M.; Mancini, L.V.; Ranise, S. SARA: Secure Asynchronous Remote Attestation for IoT Systems. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 3123–3136. [CrossRef]
24. Khurshid, A.; Raza, S. AutoCert: Automated TOCTOU-secure digital certification for IoT with combined authentication and assurance. *Comput. Secur.* **2023**, *124*, 102952. [CrossRef]
25. Asokan, N.; Brasser, F.; Ibrahim, A.; Sadeghi, A.; Schunter, M.; Tsudik, G.; Wachsmann, C. Seda: Scalable embedded device attestation. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, New York, NY, USA, 12–16 October 2015; pp. 964–975.
26. Ibrahim, A.; Sadeghi, A.; Tsudik, G.; Zeitouni, S. DARPA: Device Attestation Resilient to Physical Attacks. In Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks, Darmstadt, Germany, 18–20 July 2016; pp. 171–182.
27. Carpent, X.; Tsudik, G.; Rattanavipanon, N. ERASMUS: Efficient remote attestation via self-measurement for unattended settings. In Proceedings of the 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, 19–23 March 2018; pp. 1191–1194.
28. Kohnhäuser, F.; Büscher, N.; Katzenbeisser, S. SALAD: Secure and Lightweight Attestation of Highly Dynamic and Disruptive Networks. In Proceedings of the 2018 on Asia Conference on Computer and Communications Security, Incheon, Republic of Korea, 4–8 June 2018; pp. 329–342.
29. DICE-Layering-Architecture. Available online: https://trustedcomputinggroup.org/wp-content/uploads/DICE-Layering-Architecture-r19_pub.pdf (accessed on 14 November 2023).
30. Aggregate Message Authentication Schemes for Internet of Things Environment (Study Group 17). Available online: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.1366-202009-I!!PDF-E&type=items (accessed on 13 November 2023).

31. Pittacus. Available online: https://github.com/izeigerman/pittacus (accessed on 10 August 2023).
32. secp256k1. Available online: https://github.com/bitcoin-core/secp256k1(accessed on 10 August 2023).
33. Eldefrawy, K.; Tsudik, G.; Francillon, A.; Perito, D. Smart: Secure and minimal architecture for (establishing dynamic) root of trust. In Proceedings of the Network and Distributed System Security (NDSS) Symposium, San Diego, CA, USA, 5–8 February 2012; pp. 1–15.
34. Koeberl, P.; Schulz, S.; Sadeghi, A.; Varadharajan, V. TrustLite: A security architecture for tiny embedded devices. In Proceedings of the Ninth European Conference on Computer Systems, Amsterdam, The Netherlands, 13–16 April 2014; pp. 1–14.
35. Brasser, F.; El Mahjoub, B.; Sadeghi, A.; Wachsmann, C.; Koeberl, P. TyTAN: Tiny Trust Anchor for Tiny Devices. In Proceedings of the 52nd Annual Design Automation Conference, New York, NY, USA, 7–11 June 2015; pp. 1–6.
36. DICE Attestation Architecture. Available online: https://trustedcomputinggroup.org/wp-content/uploads/TCG_DICE_Attestation_Architecture_r22_02dec2020.pdf (accessed on 13 November 2023).
37. Sato, S.; Shikata, J. Interactive Aggregate Message Authentication Scheme with Detecting Functionality. In *Advanced Information Networking and Applications*; Springer International Publishing: Cham, Switzerland, 2019; pp. 1316–1328.
38. Hirose, S.; Shikata, J. Non-adaptive Group-Testing Aggregate MAC Scheme. In *Advanced Information Networking and Applications*; Springer International Publishing: Cham, Switzerland, 2018; pp. 357–372.
39. Benedictis M.D.; Lioy, A. Integrity verification of Docker containers for a lightweight cloud environment. *Future Gener. Comput. Syst.* **2019**, *97*, 236–246. [CrossRef]
40. Sun, Y.; Safford, D.; Zohar, M.; Pendarakis, D.; Gu, Z.S.; Jaeger, T. Security namespace: Making linux security frameworks available to containers. In Proceedings of the 27th USENIX Security Symposium (USENIX Security 18), Baltimore, MD, USA, 15–17 August 2018; pp. 1423–1439.
41. Arnautov, S.; Trach, B.; Gregor, F.; Knauth, T.; Martin, A.; Priebe, C.; Lind, J.; Muthukumaran, D.; O'keeffe, D.; Stillwell, M.L.; et al. SCONE: Secure linux containers with intel SGX. In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), Savannah, GA, USA, 2–4 November 2016; pp. 689–703.
42. Lebedev, I.; Hogan, K.; Devadas, S. Secure Boot and Remote Attestation in the Sanctum Processor. In Proceedings of the 2018 IEEE 31st Computer Security Foundations Symposium (CSF), Oxford, UK, 9–12 July 2018; pp. 46–60.
43. Ba, H.; Zhou, H.; Mei, S.; Qiao, H.; Hong, T.; Wang, Z.; Ren, J. Astrape: An efficient concurrent cloud attestation with ciphertext-policy attribute-based encryption. *Symmetry* **2018**, *10*, 425. [CrossRef]