LegIoT: Ledgered Trust Management Platform for IoT

Jens Neureither[1], Alexandra Dmitrienko[2], David Koisser[1], Ferdinand Brasser[1], and Ahmad-Reza Sadeghi[1]

[1] Technical University Darmstadt, Germany jens.neureither@gmail.com
{david.koisser,ferdinand.brasser,ahmad.sadeghi}@trust.tu-darmstadt.de
[2] University of Würzburg, Germany alexandra.dmitrienko@uni-wuerzburg.de

Abstract. We investigate and address the currently unsolved problem of trust establishment in large-scale Internet of Things (IoT) networks where heterogeneous devices and mutually mistrusting stakeholders are involved. We design, prototype and evaluate LegIoT, a novel, probabilistic trust management system that enables secure, dynamic and flexible (yet inexpensive) trust relationships in large IoT networks. The core component of LegIoT is a novel graph-based scheme that allows network devices (graph nodes) to re-use the already existing trust associations (graph edges) very efficiently; thus, significantly reducing the number of individually conducted trust assessments. Since no central trusted third party exists, LegIoT leverages *Distributed Ledger Technology* (DLT) to create and manage the trust relation graph in a decentralized manner. The trust assessment among devices can be instantiated by any appropriate assessment technique, for which we focus on remote attestation (integrity verification) in this paper. We prototyped LegIoT for Hyperledger Sawtooth and demonstrated through evaluation that the number of trust assessments in the network can be significantly reduced – e.g., by a factor of 20 for a network of 400 nodes and factor 5 for 1000 nodes.

Keywords: trust management, blockchain, remote attestation

1 Introduction

The Internet of Things (IoT) is a technology trend that promises to bring our life standard to a revolutionary new level. The key aspect of IoT are smart "things" that connect physical objects, such as home appliances, vehicles, and smart city objects (parking lots, traffic lights, etc.) to the digital world through various sensors and communication interfaces. In this context, there is a growing importance of applications where IoT devices from various parties need to collaborate with each other. Examples can be found in smart energy grids [50], smart factories allowing for the automatic assembly of customized products [23], up to sensors and actors enhancing various cases, like smart locks [44].

At the same time, the deployment of large IoT networks will lead to new threats. Malicious behavior and corrupted devices by one party can have devastating effects on the entire system. Attacks on smart meters [42] or smart

locks [18] show how liability is already a concern in these systems. Worse, attacks on Industrial Control Systems (ICS) are increasing in number and sophistication in recent years [26] and have grave security implications, like *Stuxnet* [35] demonstrated. Thus, it is essential to have appropriate security mechanisms in place, which allow connected devices to establish trust relationships in the network. However, such trust relations are challenging to build and maintain, as we explore in the following.

Challenge 1: The large number of devices and their heterogeneity in terms of capabilities—due to different hardware and software architectures—complicate any attempts to use unified schemes for trust establishment.
Challenge 2: IoT networks interconnect devices produced, owned and controlled by many different parties, which are typically mutually mistrusting. Hence, solutions should not solely rely on a central entity, which needs to be trusted by all stakeholders and represents a single point of failure.
Challenge 3: Establishing trust in a scalable manner without a central authority is difficult. For example, methods based on pairwise trust links do not scale to large networks. Furthermore, if devices handle their individual trust establishment processes independently, the results within their limited point of view are a lot less significant compared to a scenario where information is shared throughout the network.

Trust Establishment. Existing trust establishment methods enable entities to gain trust into other entities in the system through either *behavioral monitoring* and detection of abnormal activities [14, 25], or by means of a stronger type of trust indicator, such as cryptographic keys or *attestation evidence*—a primitive underlying Remote Attestation (RA), a key concept of Trusted Computing technology [41]. Behavioral monitoring can only detect malicious devices that deviate from the expected behavior, but are ineffective against malicious entities that play along established rules. Cryptographic keys help to tell apart malicious and benign devices; however, keys lose their benefits if the platform is compromised. In contrast, RA was specifically designed to allow an entity to remotely verify the state of another system and decide whether it is compromised in a process called *platform measurement* [3]. Typically, attestation assumes the existence of a trusted component, ranging from simple ROM to cryptographic co-processors.

However, RA is designed to work in a device-to-device fashion and does not scale well to large deployments. For instance, attempts to scale RA to large networks [30, 33, 34, 10] have limitations, such as the inability to build trust relationships between individual devices dynamically when they are needed. Instead, groups of devices can only be attested statically in their entirety. In swarm attestation schemes [11, 9, 16], this leads to high workloads on individual devices and requires a trusted central entity.

Trust Management. Trust management schemes rely on underlying trust establishment methods and manage available trust information in the system. Existing solutions can be divided into two categories: Recommendation and evidence-based [36]. The first category relies on recommendations from intermediaries to establish the trust relationship between two strangers [5, 51] and

is well-suited for decentralized systems. Nevertheless, they cannot provide trust establishment on demand between arbitrary nodes, since the existence of intermediate trust links is required but cannot be guaranteed. The second category of evidence-based systems can be used for trust establishment on demand. However, they either rely on trusted third parties [22, 27], or on out-of-band channels [47, 21] to disseminate trustworthy information, and hence, are not suitable for large-scale networks of devices controlled by mutually mistrusted parties.

To the best of our knowledge, state-of-the-art distributed trust management systems cannot satisfy both the decentralized setting and on-demand trust establishment at once—a limitation we aim to address in this paper.

Contributions. In particular, we make the following contributions:

- We propose LegIoT, a trust management framework that can leverage various evidence-based methods for trust establishment to enable interoperability among heterogeneous devices (addressing challenge 1). It builds and maintains trust information in a system-wide fashion by managing dynamic (recommendation-like) trust chains across the network. As such, LegIoT combines recommendation and evidence-based approaches in one system.
- LegIoT enables on-demand trust establishment between mutually mistrusted devices, even if no intermediate trust relations pre-exist. In its heart, LegIoT has a novel graph enlargement algorithm, which adds new trust links to the system, while maximizing their re-usability to benefit the entire network. Our evaluation results demonstrate that the overall number of trust establishment processes can be significantly reduced (challenge 3), e.g., by as much as a factor of 20 for a network size of 400 and a factor of 5 for 1000 nodes.
- LegIoT utilizes Distributed Ledger Technology (DLT) to store, manage and process trust information, which enables mutually mistrusting parties to participate in the network (challenge 2). DLT is the previously missing piece of the puzzle that allows us to blend evidence-based and recommendation-based approaches to inherit advantages of both worlds: On-demand trust establishment and a decentralized setting. We instantiated LegIoT using Hyperledger Sawtooth and published the implementation as open source[3].

2 System Model

Our system model involves the following entities: (i) IoT devices $\mathcal{D}$ operating in a collaborative environment; (ii) device owners $\mathcal{O}$; (iii) device manufacturers $\mathcal{M}$, and (iv) distributed ledger $\mathcal{L}$. IoT devices $\mathcal{D}$ are manufactured by $\mathcal{M}$, and different devices $d_i, d_j \in \mathcal{D}$ can be manufactured by different $m \in \mathcal{M}$. We assume that $d_i \in \mathcal{D}$ has the means to establish trust relations with another $d_j \in \mathcal{D}$, e.g., through known characteristics of its benign behavior or via remote attestation, and fulfills the requirements of the deployed trust mechanism. The distributed ledger $\mathcal{L}$ is a tamper-proof append-only data structure, which is maintained cooperatively by a distributed network of ledger validators. We assume that $\mathcal{L}$

[3] Available under the link: https://github.com/legiot/LegIoT

has the ability to *store integrity-protected data* and *execute programs provably correct* through so-called smart contracts [15]. $\mathcal{L}$ is operated by a consortium of the stakeholders, which can include the owners $\mathcal{O}$, device manufacturers $\mathcal{M}$, as well as external entities like consumer advocacy organizations or government agencies. The parties maintaining $\mathcal{L}$ are mutually distrusting but we assume that only a minority of parties are malicious. New consortium members require a collective agreement by the consortium to participate, e.g., by vetting new members before admission, to prevent a party to gain disproportionate voting power.

Adversary Model. We assume that device manufacturers $\mathcal{M}$ are trusted to produce non-malicious hardware and firmware for IoT devices $\mathcal{D}$ and to provide correct information helping to verify that they are in the trusted state (e.g., benign software and hardware configurations). We also assume that devices $\mathcal{D}$ are initially trusted and are not infected prior deployment. After deployment, devices might be compromised. We denote an adversary by $\mathcal{A}$. At the network layer, we consider the *Dolev-Yao* model where $\mathcal{A}$ has unlimited access to the network [19, 17]. $\mathcal{A}$ is unable to break cryptographic primitives but can perform both passive attacks such as *eavesdropping* and active attacks like *message spoofing* and *replaying*. Through these capabilities, the adversary can further run arbitrary malware on devices $\mathcal{D}$. Furthermore, we inherit any assumptions from the underlying trust establishment mechanisms, e.g., many attestation schemes assume a trust anchor on the device.

Moreover, we rule out Denial-of-Service (DoS) attacks, as it is a common assumption in the context of trust management literature. Additionally, we assume an attacker is not able to compromise and leave devices without traces instantaneously. We refer to this phenomenon as a *fast roaming adversary* and argue, that an attacker at least needs the time $\mathcal{T}$ to restore the uninfected state of a device and hide his prior presence.

3 LegIoT Design

LegIoT is a trust management framework that enables network participants to query trust information on demand. In a nutshell, our system builds and uses *indirect trust relationships* between devices via a trust relation graph. The intuition is, if the trustworthiness of a device d_i was successfully verified by d_j, and another device $d_k \in \mathcal{D}$ wants to assess the trustworthiness of d_j shortly after, d_k can gain indirect trust into d_i. This concept can be extended to consider more devices, and thus can build trust chains, which in turn can be combined into a trust graph. This graph is calculated and managed by LegIoT using Distributed Ledger Technology (DLT). This avoids a single point of failure and enables mutually mistrusting parties to reach consensus about the trust graph without relying on a central authority.

Figure 1 shows an exemplary deployment of a new device d and its integration into the trust graph. In Step 1, a device manufacturer $m \in \mathcal{M}$ registers

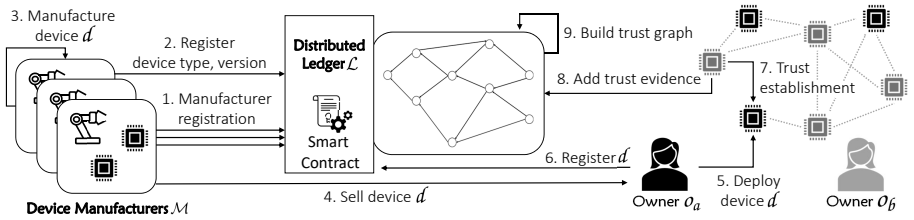

Fig. 1. Example of deploying a new device and integrating it into the trust graph in a collaborative environment.

itself on the distributed ledger $\mathcal{L}$. Additionally, m submits universal information about its devices required to publicly verify them by any party—namely, their trustworthy configurations or trustworthy behavioral characteristics (Step 2). After the owner o_a sets up and configures the device d acquired from m (Step 3, 4 and 5), o_a registers this individual deployment of d on $\mathcal{L}$ (Step 6), allowing any party to access the stored information to make their own trust assessments. The owners $\mathcal{O}$ are mutually mistrusting, so at some point there will be a request to establish trust in d by a device of another owner o_b in Step 7. After accessing all the necessary information from $\mathcal{L}$ to evaluate the trustworthiness of d, the result of this trust assessment is committed to $\mathcal{L}$ (Step 8). In Step 9, The ledger $\mathcal{L}$ then integrates this evidence into the trust graph via a smart contract.

Through the trust graph, $\mathcal{L}$ can serve queries regarding the trustworthiness of a device from any party. In such an instance we call the targeted device the prover $\mathcal{P}rv$ and the requesting party the verifier $\mathcal{V}rf$. The smart contract handling the trust query then tries to find a continuous *trust path* to the $\mathcal{P}rv$, which may possibly consist of multiple hops. Whenever a prover's trust score is requested and no path exists, the $\mathcal{V}rf$ will not perform a trust assessment of the device directly. Instead, it will follow suggestions of the trust management framework, which will find a $\mathcal{P}rv_{aux}$ as an optimal entry point into the trust graph. The trust link established from the suggested entry point will produce a valid trust path, while including as many devices as possible in between. Thus, a single trust evaluation process will result in gaining trust into several devices at once: the prover $\mathcal{P}rv$ at the end of the trust path and implicitly into all devices along the path from the prover node. We elaborate on the concept of indirect trust in Section 3.1. Hence, in the long-term, the number of individual trust establishment processes among devices can be largely reduced.

3.1 Theoretical Foundations

In the following, we define the notion of indirect trust relationships. Afterwards, we define a graph-based representation of trust relationships, elaborate on the idea of trust chains and propose methods for their valuations.

Indirect Trust. The notion of trust and indirect trust relationships was extensively studied in the context of social networks [31, 53]. Generally, the concept of trust is not transferable via multiple entities, which makes it non-trivial to apply the notion of indirect trust to a system. For this purpose we refer to Jøsang et al., who introduced the notions of *functional* and *referral* trust [31]. Both referral and functional trust can be *direct* an *indirect*. In addition, each trust link is assigned a scope σ. Figure 2 depicts how referral trust can be combined with direct functional trust to derive indirect functional trust. Here, *Alice* wants to buy bread from the baker *Charlie* and relies on the recommendation of her friend *Bob* regarding the trust scope, bread quality (σ). To enable indirect trust chains with multiple hops, two important requirements must be fulfilled. First, the *Functional Trust Derivation Criterion* states that "referral trust requires that the last trust arc represents functional trust and all previous trust arcs represent referral trust" [31]. Second, the *Trust Scope Consistency Criterion* states that a "trust path requires that there exists a trust scope which is a common subset of all trust scores in the path. The derived trust scope then is the largest common subset." [31].

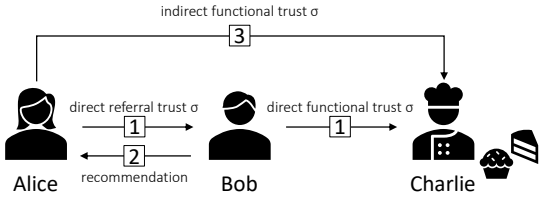

Fig. 2. Example regarding the process of gaining indirect functional trust.

Yu et al. define $T_i(j)^t$ as the "trust rating assigned by agent i to agent j at time t" [53]. We extend this definition in a way, that it also includes the trust scope σ between agent i and agent j: $T_i(j,\sigma)^t$. Further, we constrain:

$$0 \leq T_i(j,\sigma)^t \leq 1 \; and \; T_i(j,\sigma)^0 = 0. \tag{1}$$

Regarding trust propagation, the following rule holds for an indirect trust chain $x \implies y \implies z$ [53]:

$$(T_x(z,\sigma)^{t+1} \leq T_x(y,\sigma)^t) \wedge (T_x(z,\sigma)^{t+1} \leq T_y(z,\sigma)^t) \tag{2}$$

To satisfy this rule, multiplication can be used for determining indirect trust [53]:

$$T_x(z,\sigma)^{t+1} = T_x(y,\sigma)^t \; T_x(z,\sigma)^t \tag{3}$$

Trust Scope. The trust scope σ plays an important role in trust paths as it defines what the subject of a relationship actually is. Of particular interest is

the question of whether evidences enable referral trust, functional trust or both. Devices may or may not provide referral trust, and the actual scope depends on the underlying scheme that is used for trust establishment. Generally, LegIoT can rely on various trust establishment methods.

Let us take remote attestation as an example, which can be classified into four groups. (i) software-based attestation schemes [45, 3] aim to not rely on specific hardware components, while (ii) hybrid architectures [20, 32], (iii) hardware-based architectures [41, 43], and (iv) control flow attestation [2, 4] leverage hardware trust anchors. Hence, they provide different levels of resilience against compromise and vary in trust scope. We provide a more thorough discussion on the various attestation schemes in Appendix A.

Trust Graph and Trust Chains. To represent the trust graph of a system, we consider a *directed and weighted graph* $(\mathcal{D}, \mathcal{E})$. $\mathcal{D}$ is the set of all devices participating in the network. $\mathcal{E}$ represents the set of all established trust links which create the edges in the graph[4]. The idea behind trust chains implies that devices do not necessarily need to be assessed directly. Instead, it suffices that a verifier evaluates any vertex $\mathcal{d} \in \mathcal{D}$ where a path $p_{\mathcal{d} \rightarrow \mathcal{P}rv}$ exists, if the following constraints apply:

1. *Path Scope:* $p_{\mathcal{d} \rightarrow \mathcal{P}rv}$ fulfills the scope criteria σ (cf. Section 3.1 *Trust Scope*).
2. *Trust Scope:* $\mathcal{d}$ must be eligible for referral trust.
3. *Temporal Edge Validity:* All edges in the resulting path $p_{\mathcal{V}rf \rightarrow \mathcal{P}rv}$ must still be considered valid at the time of calculation.

The validity of edges is strongly bound to the temporal proximity τ of the trust evaluation process and the usage of the trust evidence $\tau = \mathcal{T}_{use} - \mathcal{T}_{eval}$. $\mathcal{T}_{use}$ is the time regarding the usage of the evidence and $\mathcal{T}_{eval}$ its inception). After a device was evaluated successfully, it might be compromised by adversaries. Thus, the *resilience* and *significance* of edges vanishes over time. We define a time threshold $\mathcal{T}_{min}$, assuming an adversary requires some time to progressively compromise devices in the network. As long as $\tau < \mathcal{T}_{min}$ the evidence is fully valid. Afterwards, the probability of an attack increases the longer the device was not checked. Therefore, the validity decreases according to a time function $v(\tau)$. We also define the expiration time $\mathcal{T}_{exp}$. After this time evidences are considered invalid.

Trust Chain Valuation. As a first step of a path valuation, the *weight* of a single edge is calculated. In addition to temporal validity, the valuation is also influenced by the resilience of an underlying trust evaluation scheme (cf. Section 3.1 *Trust Scope*). The reliability function r for a trust evaluation method type $\mathcal{E}val$ produces a probabilistic result $r(\mathcal{E}val) \in [0, 1]$. Together, the combined edge reliability $f(\tau, \mathcal{E}val)$ can be calculated as follows:

$$f(\tau, \mathcal{E}val) = v(\tau) * r(\mathcal{E}val) \tag{4}$$

[4] An edge is equivalent to a direct trust rating $T_i(j)$ of two nodes; yet, we simply use $e \in \mathcal{E}$ for better readability.

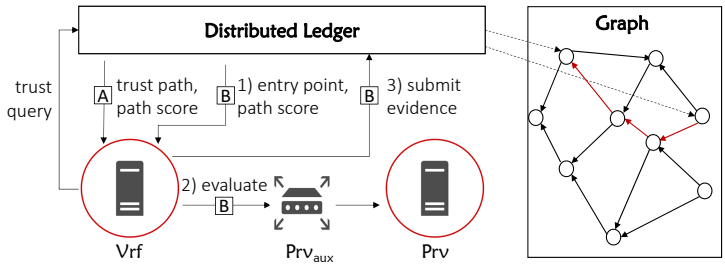

Fig. 3. Illustration of an exemplary trust query, sent by $\mathcal{V}rf$.

The resulting value expresses the *probability* that the targeted node is in a *trustworthy condition* from the point of view of the verifying node. To chain the reliability of multiple sequential edges on a path together, we use the findings from Equation (3). Based on this, the probability weights can be multiplied for each edge e in path p:

$$\phi(p) = \prod_{e \in p} f\Big(\tau(e), \mathcal{E}val(e)\Big) \tag{5}$$

3.2 Protocols and Algorithms

We now proceed to describe protocols and algorithms based on the concepts described in the previous section.

System Initialization and Administration. Prior to deployment, multiple databases containing administrative data need to be initialized. In particular, every manufacturer $\mathcal{M}$ maintains a Policy DB on the ledger $\mathcal{L}$, which includes information about their devices as well as information to evaluate their state. Furthermore, an owner $\mathcal{O}$ needs to submit four databases to the ledger $\mathcal{L}$: (i) System Settings DB, (ii) Device DB, (iii) Verifier DB, and (iv) Trust Evaluation Methods DB.

System Settings DB includes system-wide settings, such as a security parameter denoting the maximum path distance between $\mathcal{P}rv$ and $\mathcal{V}rf$. The Device DB maps a device identity (represented by public keys) to a device described in Policy DB. Verifier DB contains entities that are not included in Device DB; yet, are permitted to act as verifiers. These could be, for instance, external entities such as device owners. Trust Evaluation Methods DB stores parameters for each trust evaluation method, such as reliability score and timing characteristics, which denotes how fast the validity of an established trust evidence vanishes over time.

Trust Query. The trust query protocol offers an interface to query the state of another device. The protocol is depicted in Figure 3. It is called by a verifier with

an identity $\mathcal{ID}_{\mathcal{V}rf}$ that requests to establish trust into a prover with $\mathcal{ID}_{\mathcal{P}rv}$. The query is represented by the transaction $trustQuery(\mathcal{ID}_{\mathcal{V}rf}, \mathcal{ID}_{\mathcal{P}rv}, minReliability)$. With $minReliability$ the verifier can specify the lower bound of the final trust score the path is allowed to have. LegIoT continues with the following steps: (i) Verify that $\mathcal{ID}_{\mathcal{P}rv}$ and $\mathcal{ID}_{\mathcal{V}rf}$ are found in Device DB or Verifier DB; (ii) run a graph search algorithm (cf. Section 3.2 *Graph Search Algorithm*) with parameters ($\mathcal{ID}_{\mathcal{V}rf}$, $\mathcal{ID}_{\mathcal{P}rv}$, $minReliability$, $secParameter$) where the $secParameter$ denotes the maximum permitted path length; (iii) if a path exists with the desired $minReliability$, it is returned to the verifier (case A in Fig. 3).

If no path exists, an optimal entry point is returned according to the Graph Enlargement Policy (GEP) presented in Section 3.2 *Optimal Entry Point* (case B, Step 1 in Fig. 3). $\mathcal{V}rf$ then proceeds to execute the trust evaluation process (Step 2) and submits the collected evidences (Step 3). Included in the submission is the obtained trust score $\mathcal{S}_{\mathcal{V}rf \rightarrow \mathcal{P}rv}$, the trust evaluation method $\mathcal{E}val$, and the prover's *device class* as well as its *version* regarding the Policy DB.

Graph Search Algorithm. To search the graph for an existing path in between $\mathcal{P}rv$ and $\mathcal{V}rf$, LegIoT uses limited-depth breadth-first search with the parameters $\mathcal{ID}_{\mathcal{V}rf}$, $\mathcal{ID}_{\mathcal{P}rv}$, $minReliability$, and $secParameter$. The functionality is described in Algorithm 1. The algorithm searches the graph for a direct path between $\mathcal{V}rf$ and $\mathcal{P}rv$. The *expand* operation in line 10 denotes the listing of all predecessors for a given node that were not visited, yet. *Fringe* denotes the frontier of nodes that were already visited by a search algorithm. Nodes in the fringe directly border to nodes that were not yet visited and will be expanded next. Any valid path found (multiple may exist) will result in a positive trust assessment and the found path is returned. Otherwise, Algorithm 2 is called in line 22 to calculate the best entry point (described in Section 3.2 *Optimal Entry Point*).

Optimal Entry Point. If a trust query is issued and no path between $\mathcal{P}rv$ and the $\mathcal{V}rf$ exists, the system needs to integrate $\mathcal{V}rf$ to the trust graph in a way that it will be able to reach $\mathcal{P}rv$ through a valid path. We use a *GEP* to determine a node, which the verifier is supposed to evaluate. The idea behind this policy is to create and use the synergy effects of individual trust evaluation processes. The longer the path between verifier and prover is, the more likely it is that a path already exists for subsequent queries and no new trust evaluation is necessary. The algorithm for the entry point calculation is described in Algorithm 2. The task of the GEP is to find an entry point in the graph that meets the following conditions:

- *Validity:* The entry point enables at least one valid path to the prover.
- *Distance:* The entry point has the maximal hop distance to the prover that is permitted by the *secParameter*.
- *Optimality:* For multiple entry points at a given distance, the option with the highest resulting trust score is chosen.

Algorithm 1 *BuildPath* function for establishing a path between $\mathcal{Vrf}$ and $\mathcal{Prv}$ (shortened)

Input: $\mathcal{Vrf}$, $\mathcal{Prv}$, $minReliability$, $secParameter$
Output: $pathFound$, $path$, $pathReliability$, $entrypoint$
$fringe\{\} \leftarrow \mathcal{Prv}$
$newFringe\{\} \leftarrow \emptyset$
$maxDepth \leftarrow (secParameter - 1)$
$visited[] \leftarrow \emptyset$
$currentDepth \leftarrow 0$
$path[] \leftarrow \emptyset$
$rating[] \leftarrow \emptyset$
while $currentDepth \leq maxDepth$ **do**
  for $node$ in $fringe$ **do**
    $expanded \leftarrow$ **expand**$(node)$
    for $(newEdge, parent)$ in $expanded$ **do**
      $edgeReliability \leftarrow$ **reliability**$(newEdge)$
      $path[parent] \leftarrow path[node] \cup newEdge$
      $rating[parent] \leftarrow rating[node] \cdot edgeReliability$
      $visited \leftarrow visited \cup parent$
      $newFringe \leftarrow newFringe \cup parent$
    if $\mathcal{Vrf} \in expanded$ **then**
      return $true, path[\mathcal{Vrf}], rating[\mathcal{Vrf}], \emptyset$
  $currentDepth \leftarrow currentDepth + 1$
  $fringe \leftarrow newFringe$
  $newFringe \leftarrow \emptyset$
$entrypoint \leftarrow$ **CalculateEntryPoint**$(visited, minReliability)$
return $false, path[entrypoint], rating[entrypoint], entrypoint$

Algorithm 2 *CalculateEntryPoint* sub-algorithm to determine the best possible graph entry point

Input: $visited$, $minReliability$
Output: $entrypoint$
for $candidate$ in $visited$ **do**
  if reliability$(candidate) < minReliability$ **then**
    delete $candidate$
sortByReliability$(visited, descending)$
sortBySearchDepth$(visited, descending)$
$entrypoint \leftarrow visited[0]$
return $entrypoint$

4 Prototype Implementation

We implemented LegIoT based on Hyperledger Sawtooth, a framework for distributed ledgers [28]. The Hyperledger functionality is implemented in Python and consists of about 2900 LoC. It is included in the *Transaction Processors (TP)*—Sawtooth's equivalents of smart contracts in public blockchains. Three of them are provided by Sawtooth framework itself. Settings TP implements a voting-based administration of system settings among validators. Furthermore, Identity and Settings TPs manage permissioning of both validators and clients. Block-Info TP enables access to historic ledger-specific data such as *latest block number* and *current timestamp* in the context of transaction processing. Our implementation is included in the next two TPs described. Trust TP builds the core component of LegIoT and includes all application logic for handling sub-

mitted evidences and calculating paths for trust queries (Algorithm 1 and 2). Administration TP handles updates of the databases described in Section 3.2.

The Client side of LegIoT is implemented in two modules. (1) AdminClient serves administrative tasks, such as initialization of system parameters and populating databases (cf. Section 3.2 *System Initialization and Administration*). (2) IoTclient is used to execute the trust query protocol as described in Section 3.2 *Trust Query*. Both clients are written in Python and consist of 1200 and 900 LoC, respectively, and are available as open source[3].

Additionally, we ported IoTclient to C to run it on the LPC55S69-EVK evaluation board from NXP. The board is based on ARM Cortex-M33 processor and includes the ARM TrustZone security architecture [8], which provides the necessary hardware features to support hybrid RA methods such as TrustLight [32] and SMART [20]. Furthermore, we implemented a middlebox to translate Message Queuing Telemetry Transport (MQTT)[5] requests issued by the board into HTTP requests processed by Hyperledger. The C version of the client contains some proprietary libraries, which prohibits us to open source it. We are willing to share it upon request for non-commercial use.

5 Evaluation

Within this section, we evaluate how LegIoT is able to enhance the *trust evaluation* performance compared to a scenario without trust management. If a trust path from $\mathcal{V}rf$ to $\mathcal{P}rv$ already exists, *zero* trust evaluation processes are needed to establish trust, which we call a *trust query hit*. If there is no valid path, *one* trust evaluation process must be carried out to the entry point, called a *trust query miss*. The goal is to maximize the hit percentage, which we define as:

$$HitPercentage = \frac{TrustQueryHits}{(TrustQueryHits + TrustQueryMisses)} * 100$$

Parameters. Several factors influence the hit percentage. Unless otherwise specified we use the following parameter values. The network size N is set to 200 and the *minReliability* to 0.80. The edge reliability function is set to:

$$f(\tau, \mathcal{E}val) = -0.0006666667 * x + 1.2$$

Further, we use $\mathcal{T}_{min} = 300s$ and $\mathcal{T}_{exp} = 600s$. Attacks in the wild like on Industrial Control Systems (ICS) [26] last over weeks, with a stealthy adversary trying to take over large parts of a network. Coupled with the fact that the adversary will not know when the next attestation towards compromised devices will take place, the system can certainly protect the network in its entirety with the chosen time parameters.

Note that LegIoT has an *initialization phase* before entering a normal *operation phase*. To ensure that the operation phase is reflected within the evaluation, *simulation duration* $= 1200s$ and *query rate* $= \frac{N}{2}$ are set accordingly.

[5] http://mqtt.org/

Additionally, simulations are repeated five times by default. Between simulation repetitions, the trust graph is not reset in order to reflect long-time operation.

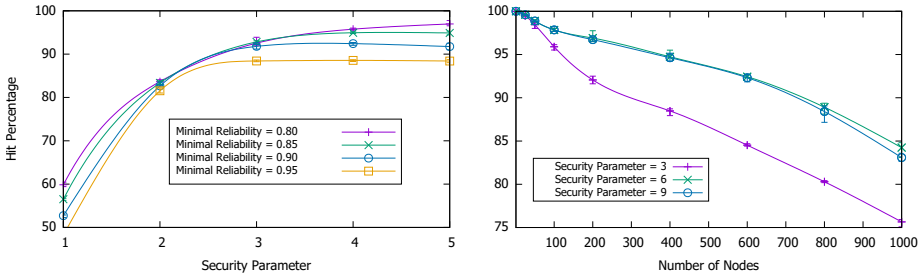

Fig. 4. Effects on the hit percentage for different minimal reliability parameters *(left)* and for increasingly larger networks with different security parameters *(right)*.

Security Parameter Variation. The security parameter denotes the maximum permitted number of hops between $\mathcal{Vrf}$ and $\mathcal{Prv}$ on a trust path. In the following, we vary the security parameter from zero to five, where zero matches a case without using our system. Figure 4 on the left shows the influence of a larger security parameter. A $minReliability = 0.80$ and a security parameter of *three* already boosts the hit percentage to approximately 93%. This means that in 100 trust queries only 7 fresh trust assessments are needed. It is impossible for the hit percentage to reach 100% because evidences expire and need to be renewed. Thus, we state that a *horizontal asymptote* exists which the curve approaches. This asymptote changes depending on the other parameters.

Minimal Reliability Variation. In Figure 4 on the left we display graphs for four different minimal reliabilities ranging from 0.80 to 0.95. Generally, lower required reliabilities result in a higher hit percentage. It can be observed that the higher the minimal reliability is, the faster the function approaches an asymptote. While the curve of '0.95' does not improve from security parameter 3 on, the curves '0.9' and '0.85' still profit from increasing the value. This phenomenon results from the fact, that the individual edge reliabilities are multiplied for the total reliability score. Thus, for a path that uses a higher security parameter the product of edge probabilities is lower than for a path that is only expanded regarding a lower security parameter.

Network Size Variation. To model different network sizes, we vary the number of nodes from $N = 25$ to $N = 1000$. Additionally, three different security parameters are tested with all network sizes. Results are depicted in Figure 4 on the right. With the security parameter set to 6 and $N = 400$ a hit percentage of 95% is achieved (a reduction by a factor of 20) and 80% for $N = 1000$ (factor 5).

All curves have in common that a network enlargement lowers the hit percentage. This can be explained by the fact that if the graph grows, it is less probable that a random verifier and prover are located in the same area. To overcome the increasing distance, a higher security parameter has a positive impact. The curves with less-restrictive hop limitations (Security Parameter 6 and 9) descend more significantly at higher network sizes than the curve with Security Parameter 3. The curves for parameters 6 and 9 are nearly equal. This phenomenon is explained by the fact that the minimal reliability of '0.8' also limits search depth indirectly, due to multiplication of individual edge reliabilities. There is a high probability that the overall path reliability falls below '0.8' at a certain length. In order to make use of a greater security parameter, the minimal reliability must be lowered or a better suited path reliability calculation method must be used. While we stated that LegIoT does not profit from security parameter increases at a certain point in the context of minimal reliability limitations, it can still be useful to compensate for network size effects.

Edge Reliability Function Variation. Any arbitrary function can be used to describe desired time validity behavior. For exemplary purposes, we assume that the supported trust evaluation methods are different types of attestation schemes, which we described in Appendix A. They can be mapped to linear functions with different validity periods. Functions are noted in the format $[Timefunction, \mathcal{T}_{min}, \mathcal{T}_{exp}]$. All functions degrade linearly from '1' between $\mathcal{T}_{min}$ and $\mathcal{T}_{exp}$. They vary in their *gradient* as well as in *validity periods*. Short-lived functions like $[-0.003333333 * x + 1.2, 60, 120]$ reflect attestation types with weaker resilience such as *software-based attestation*. On the contrary schemes like *control-flow attestation* with stronger resilience against compromise might use functions with high validity for a long duration such as $[-0.000333333 * x + 1.2, 600, 1200]$.

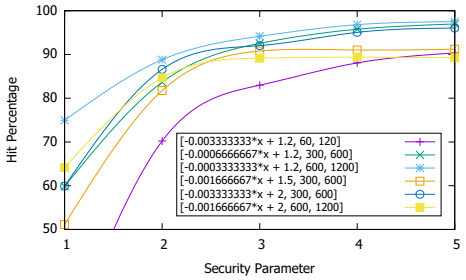

Fig. 5. Hit percentage effects regarding selected edge reliability functions.

In Figure 5 we show how variations of these function parameters influence hit percentages with different security parameters. The first three functions that all degrade from '1' to '0.8' within their degeneration phase offer similar characteristics. As expected, the function with the longest validity achieves the highest

hit percentages. It is noticeable, that the slowly descending functions profit from higher security parameters. Altogether, function differences in most cases have less impact the higher the security parameter is. Even though validity times are varied by the factor of ten, the hit percentage difference is less than 10%.

6 Security Analysis

In this section, we informally analyze possible attack vectors on LegIoT, explain their impact, and if needed, their mitigation. Our analysis focuses on LegIoT itself without taking into account potential security weaknesses of underlying technologies (e.g., distributed ledgers or crypto primitives).

Evidence Spoofing. The adversary may submit faulty evidence as he impersonates a verifier. Nevertheless, these edges are only used if another (honest) device evaluated the adversary's device to complete a trust path. As the device is infected, the trust evaluation method reports a lack of trust and no path is established. Thus, spoofed evidence is never used when considering the present attacker model that excludes *fast roaming adversaries*.

Roaming Adversaries. A roaming adversary is the one that infects the device to forge the trust evidence for a malicious node and leaves the device with no traces. If positively evaluated by another honest verifier while the forged evidence is still valid, trust is gained in the malicious node and in all other nodes connected to it. An attacker could use this strategy to create a great number of trust links to devices infected by him via one single device. Our system provides probabilistic protection against this attack vector. First, our attacker cannot leave devices without traces in no time (cf. Section 2). Second, he cannot know for sure when the next evaluation towards his device will take place, limiting his chances to succeed. Third, the estimated minimal compromise time $\mathcal{T}_{min}$ and the *temporal edge validity function* further restrict the motion range an adversary has.

Device Infection in Validity Period. A device can possibly be infected during the validity period of evidence created for it as a prover. Therefore, evidences towards the device might still exist even if it was corrupted in the meantime. The uncertainty of whether an infection happens in the validity period of an evidence is again reflected by $\mathcal{T}_{min}$, $\mathcal{T}_{exp}$ as well as the *temporal edge validity function*. Thus, the risk can be controlled by setting these parameters.

Timestamp Attacks. If timestamps can be forged or manipulated, system correctness is highly endangered. One important precaution is that timestamps are added by the Validator nodes that run the ledger. Using time from verifying devices as a timestamp would cause issues since this would require them to have a synchronized real-time clock, which is hard to implement in practice in the targeted setting. However, a malicious verifier could carry out a trust evaluation process for an honest prover and withhold this trust evidence. He might keep the evidence, infect the prover device and then submit the valid evidence. As a result, a valid evidence for a malicious device exists in the graph. Similar to *evidence spoofing*, the trust edge alone is not sufficient to convince other devices of the

correctness of a prover. At the point where other verifiers try to build trust into the prover, the path leads via the corrupted verifier that uploaded the evidence. Another attack may be compromising a single Validator node. Here, in order to enforce deterministic transaction results, other Validators are unable to check the timestamps generated by the block creator since they never match exactly. To prevent this, Sawtooth offers rules that enforce timestamp monotonicity and a roughly synchronized network time [29].

Sneaky Hopping Attack. With *sneaky hopping*, an adversary tries to predict the next system actions to stay undetected. He always tries to compromise devices that will most likely not be attested in the near future. This also requires the adversary to be *roaming* but he has more time to infect, leave and hide traces (*slow roaming*). For LegIoT this is not applicable since it is unpredictable which evidence will be added or trust score queried next. Both *evidence submission* and *trust query* are initiated by entities whose decisions are out of range of the attacker.

7 Related Work

Trust Management. Trust Management is an important issue in distributed systems, which aims to track the trustworthiness of all individual devices in an up-to-date fashion. A key work in this field was proposed by Blaze et al. [13], which first tackled the problem of decentralized trust beyond the verification of certificates. The work of Abdul-Rahman and Hailes [1] then introduced the notion of a recommendation-based trust model, in which recommendations help to establish trust between two strangers. Another class of trust models derive trust based on evidence, such as cryptographic keys, and typically rely on trusted parties (e.g., for key distribution). Examples are the resurrecting duckling model [47], Pretty Good Privacy (PGP) [21] and the X.509 certificate [27] systems. Our approach is a combination of both recommendation- and evidence-based approaches. It relies on attestation evidences and indirect trust links built using recommendations, while enabling dynamic and on-demand trust establishment in networks between heterogeneous devices controlled by mutually mistrusting parties. Furthermore, it can leverage a stronger type of trust indicator, such as *attestation evidence*, which enables the detection of compromised devices. Our trust establishment is also directly applicable from a single request, in contrast to systems that need to converge over multiple recommendations, like bayesian trust models [49].

Another line of work in the area of trust management targets ad-hoc networks [37, 14], which are dynamic and do not typically include trusted third parties. Such systems work by nodes disseminating trust information about other nodes through the network. Trust judgments are made by monitoring neighboring nodes and identifying suspicious behavior. In contrast, LegIoT relies on a stronger type of evidence that can detect compromised devices, even if they behave correctly. Furthermore, trust information is stored and managed on a

distributed ledger, and thus nodes do not need to constantly disseminate information among peers.

DLT-based Intrusion Detection. Research papers [25, 46, 7, 38] deal with the question of whether distributed ledgers can be leveraged for detecting anomalies and isolating untrusted devices. All these systems rely on behavioral monitoring for trust establishment and maintain reputation scores for network nodes, while LegIoT can leverage stronger underlying trust establishment methods, such as *remote attestation*, and provides means to utilize indirect trust relationships.

DLT-based Trust Management. Recently, DLT-based trust management systems were developed to support authentication systems [6, 39]. Similarly to LegIoT, they use distributed ledgers to manage trust information in the system, but rely on evidence-based trust establishment methods based on cryptographic keys. In particular, Alexopoulos et al. [6] show how to enhance a public-key-to-id binding for PGP [21] and X.509 public key infrastructure (PKI) [27]. Their system can build and leverage indirect trust relationships; however, neither provides on-demand trust establishment nor maximizes benefits of pre-established trust connections. The solution by Moinet at al. [39] is intended to substitute PGP and X.509 PKI in Wireless Sensor Networks (WSN). Their approach to trust management is different, as it relies on the concept known as Human-like Knowledge based Trust (HKT), and does not include the notion of indirect trust.

DLT and Trusted Computing. TM-Coin [40] deals with "Trustworthy Management of TCB Measurements in IoT". The authors eliminate the redundancy of measurements by letting miners attest the Trusted Computing Base (TCB) of IoT devices. Afterwards, a remote verifier can query these results from the distributed ledger. In contrast, LegIoT does not involve miners into the attestation process—instead, it is carried out by other clients, which simplifies deployment. Furthermore, unlike TM-Coin, LegIoT utilizes attestation results submitted by other nodes to reduce the number of overall attestation processes in the system and can combine various attestation methods in one solution.

Banerjee et al. [12] propose a conceptual design for a blockchain-based security layer for identification, auditing, and isolation of misbehaving nodes in IoT networks, which relies on remote attestation to detect compromised entities. The attestation process in the system takes place between two clients and the result is recorded in a smart contract in a form of a *whitelist*. In contrast, LegIoT builds and manages a system trust graph and provides the means to reuse results of previously conducted trust evaluation processes. Furthermore, LegIoT supports various attestation schemes in one solution, while the system by Banerjee et al. is limited to only one method.

Last, Xu et al. [52] propose a security model for remote attestation for V2X applications with focus on anonymity of attested vehicles. In contrast to LegIoT, it does not focus on trust management and only supports *hardware-based* remote attestation that requires support of Trusted Platform Module (TPM). This limits heterogeneity and to a large extend, rules out IoT use-cases.

8 Conclusion

With LegIoT, we present a new approach to trust management in IoT networks, in which recommendation-like trust links provided by intermediate nodes are combined with evidence-based trust establishment methods. The design leverages distributed ledger technology to store, manage and distribute trust information in the network—the missing piece of the puzzle that made the combination of both approaches possible. The resulting solution inherits benefits of both worlds: On-demand trust management from evidence-based systems as well as the dynamics and trustless setting from reputation-based solutions. As a result, LegIoT can provide dynamic on-demand trust management in decentralized networks and can combine various underlying trust establishment methods to accommodate the various needs of heterogeneous IoT platforms. The novel algorithm for trust graph management in the heart of LegIoT maximizes the benefits of newly established trust links for the entire network, which improves the scalability of the system by multiple factors.

Acknowledgment

This research has been funded by the Federal Ministry of Education and Research of Germany (BMBF) in the framework KMU-innovativ-Verbundprojekt: Secure Internet of Things Management Platform - SIMPL (project number 16KIS0852), by BMBF within the project iBlockchain, by the European Space Operations Centre with the Networking/Partnering Initiative, and by the Intel Collaborative Research Institute for Collaborative Autonomous & Resilient Systems (ICRI-CARS).

References

1. Abdul-Rahman, A., Hailes, S.: A distributed trust model. In: Proceedings of the 1997 workshop on New security paradigms. pp. 48–60 (1998)
2. Abera, T., Asokan, N., Davi, L., Ekberg, J.E., Nyman, T., Paverd, A., Sadeghi, A.R., Tsudik, G.: C-FLAT: Control-flow attestation for embedded systems software. In: ACM SIGSAC CCS (2016)
3. Abera, T., Asokan, N., Davi, L., Koushanfar, F., Paverd, A., Sadeghi, A.R., Tsudik, G.: Things, trouble, trust: On building trust in iot systems. In: ACM DAC (2016)
4. Abera, T., Bahmani, R., Brasser, F., Ibrahim, A., Sadeghi, A.R., Schunter, M.: DIAT: Data integrity attestation for resilient collaboration of autonomous systems. NDSS (2019)
5. Aberer, K., Despotovic, Z.: Managing trust in a peer-to-peer information system. In: ACM CIKM (2001)
6. Alexopoulos, N., Daubert, J., Mühlhäuser, M., Habib, S.M.: Beyond the hype: On using blockchains in trust management for authentication. In: IEEE Trustcom/BigDataSE/ICESS (2017)
7. Alexopoulos, N., Vasilomanolakis, E., Ivánkó, N.R., Mühlhäuser, M.: Towards blockchain-based collaborative intrusion detection systems. In: CRITIS (2017)

8. Alves, T., Felton, D.: TrustZone: Integrated hardware and software security (2004)
9. Ambrosin, M., Conti, M., Ibrahim, A., Neven, G., Sadeghi, A.R., Schunter, M.: SANA: Secure and scalable aggregate network attestation. In: ACM SIGSAC CCS (2016)
10. Ammar, M., Washha, M., Ramabhadran, G.S., Crispo, B.: Slimiot: Scalable lightweight attestation protocol for the Internet of Things. In: IEEE DSC (2018)
11. Asokan, N., Brasser, F., Ibrahim, A., Sadeghi, A.R., Schunter, M., Tsudik, G., Wachsmann, C.: SEDA: Scalable embedded device attestation. In: ACM SIGSAC CCS (2015)
12. Banerjee, M., Lee, J., Chen, Q., Choo, K.R.: Blockchain-based security layer for identification and isolation of malicious things in IoT: A conceptual design. In: ICCCN (2018)
13. Blaze, M., Feigenbaum, J., Lacy, J.: Decentralized trust management. In: IEEE S&P (1996)
14. Buchegger, S., Le Boudec, J.Y.: Performance analysis of the CONFIDANT protocol. In: ACM MOBIHOC (2002)
15. Buterin, V.: A next-generation smart contract and decentralized application platform. Whitepaper (2014), https://blockchainlab.com/pdf/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf
16. Carpent, X., ElDefrawy, K., Rattanavipanon, N., Tsudik, G.: Lightweight swarm attestation: A tale of two LISA-s. In: ACM AsiaCCS (2017)
17. Cervesato, I.: The Dolev-Yao intruder is the most powerful attacker. In: ACM/IEEE LICS (2001)
18. Dardaman, C.: Breaking & entering with Zipato SmartHubs (2019), https://blackmarble.sh/zipato-smart-hub/
19. Dolev, D., Yao, A.: On the security of public key protocols. IEEE Transactions on Information Theory (1983)
20. Eldefrawy, K., Tsudik, G., Francillon, A., Perito, D.: Smart: Secure and minimal architecture for (establishing dynamic) root of trust. In: NDSS (2012)
21. Elkins, M., Torto, D.D., Levien, R., Roessler, T.: MIME Security with OpenPGP, IETF RFC 3156 (2001), www.ietf.org/rfc/rfc3156.txt
22. Eschenauer, L., Gligor, V., Baras, J.: On trust establishment in mobile ad-hoc networks. In: Security Protocols Workshop (2002)
23. Forum, W.E.: This is how a smart factory actually works (2019), https://www.weforum.org/agenda/2019/06/connectivity-is-driving-a-revolution-in-manufacturing/
24. Francillon, A., Nguyen, Q., Rasmussen, K.B., Tsudik, G.: A minimalist approach to remote attestation. In: DATE (2014)
25. Golomb, T., Mirsky, Y., Elovici, Y.: CIoTA: Collaborative IoT anomaly detection via blockchain. CoRR (2018)
26. Hemsley, K., Fisher, R.: History of industrial control system cyber incidents (2018)
27. Housley, R., Ford, W., Polk, W., Solo, D.: Internet X.509 Public Key Infrastructure, Certificate and CRL Profile, IETF RFC 2459 (1999), www.ietf.org/rfc/rfc2459.txt
28. Hyperledger: Hyperledger Sawtooth – a modular platform for building, deploying, and running distributed ledgers. (2018), https://www.hyperledger.org/projects/sawtooth
29. Hyperledger: Hyperledger Sawtooth v1.1.4 documentation (2019), https://sawtooth.hyperledger.org/docs/core/releases/1.1.4/
30. Ibrahim, A., Sadeghi, A.R., Tsudik, G., Zeitouni, S.: DARPA: Device attestation resilient to physical attacks. In: 9th ACM WiSec (2016)

31. Jøsang, A., Hayward, R., Pope, S.: Trust network analysis with subjective logic. In: Australasian Computer Science Conference (2006)
32. Koeberl, P., Schulz, S., Sadeghi, A.R., Varadharajan, V.: TrustLite: A security architecture for tiny embedded devices. In: EuroSys (2014)
33. Kohnhäuser, F., Büscher, N., Gabmeyer, S., Katzenbeisser, S.: Scapi: A scalable attestation protocol to detect software and physical attacks. In: ACM WiSec (2017)
34. Kohnhäuser, F., Büscher, N., Katzenbeisser, S.: Salad: Secure and lightweight attestation of highly dynamic and disruptive networks. In: ACM AsiaCCS (2018)
35. Langner, R.: Stuxnet: Dissecting a cyberwarfare weapon. IEEE S&P (2011)
36. Li, H., Singhal, M.: Trust management in distributed systems. Computers (2) (2007)
37. Marti, S., Giuli, T.J., Lai, K., Baker, M.: Mitigating routing misbehavior in mobile ad hoc networks. In: MobiCom (2000)
38. Meng, W., Tischhauser, E.W., Wang, Q., Wang, Y., Han, J.: When intrusion detection meets blockchain technology: A review. IEEE Access **6** (2018)
39. Moinet, A., Darties, B., Baril, J.L.: Blockchain based trust & authentication for decentralized sensor networks. CoRR (2017)
40. Park, J., Kim, K.: TM-Coin: Trustworthy management of TCB measurements in IoT. In: PerCom Workshops. IEEE (2017)
41. Pearson, S., Balacheff, B.: Trusted Computing Platforms: TCPA Technology in Context. Prentice Hall Professional (2003)
42. Rayner, G.: Smart meters could leave british homes vulnerable to cyber attacks, experts have warned (2018), https://www.telegraph.co.uk/news/2018/02/18/smart-meters-could-leave-british-homes-vulnerable-cyber-attacks/
43. Sabt, M., Achemlal, M., Bouabdallah, A.: Trusted execution environment: What it is, and what it is not. In: IEEE Trustcom/BigDataSE/ISPA (2015)
44. Scout, S.L.: Guide on Airbnb smart locks (2019), https://www.postscapes.com/airbnb-smart-lock/
45. Seshadri, A., Perrig, A., van Doorn, L., Khosla, P.: Using software-based attestation for verifying embedded systems in cars. In: ESCAR Workshop (2004)
46. Signorini, M., Pontecorvi, M., Kanoun, W., Di Pietro, R.: Bad: Blockchain anomaly detection. CoRR (2018)
47. Stajano, F., Anderson, R.: The resurrecting duckling: Security issues for ad hoc wireless networks. In: Security Protocols Workshop (1999)
48. TCG: Trusted computing group, https://trustedcomputinggroup.org/
49. Wang, Y., Vassileva, J.: Bayesian network-based trust model. In: IEEE/WIC WI 2003. pp. 372–378. IEEE (2003)
50. World, T.: IoT in utilities market forecasted to grow to $53.8 billion by 2024 (2020), https://www.tdworld.com/grid-innovations/article/21120887/iot-in-utilities-market-worth-538-billion-by-2024
51. Xiong, L., Liu, L.: Building trust in decentralized peer-to-peer electronic communities. In: ICEC (2002)
52. Xu, C., Liu, H., Li, P., Wang, P.: A remote attestation security model based on privacy-preserving blockchain for v2x. IEEE Access **6** (2018)
53. Yu, B., Singh, M.P.: A social mechanism of reputation management in electronic communities. In: CIA Workshop. Springer (2000)

A Attestation Schemes and Trust Scope

Remote attestation originally came to prominence as a feature of the TPM [41], the standard defined by the Trusted Computing Group (TCG) [48]. Many approaches to attestation have been developed, which differ in underlying requirements and security guarantees provided. Generally, they provide different levels of *resilience*, which refers to the general robustness of the underlying architecture against compromise. In the following we discuss attestation approaches of four categories, and suggest what resilience level and trust scope they can provide.

Hardware-based architectures. include strong cryptographic co-processors like TPMs [41]. A different approach are Trusted Execution Environments (TEEs) that use an isolated processing environment [43]. Usually, they offer complex attestation mechanisms with arbitrary cryptographic functionality. Since cryptographic co-processors are well studied and strongly protected, hardware-based architectures generally have a *high* resilience. Thus, this architecture is able to attest other devices and create functional as well as referral trust.

Hybrid architectures. generally include minimal security features like Read Only Memory (ROM) and Memory Protection Unit (MPU) for secure storage [24]. Generally, hybrid schemes such as *SMART* [20] and *TrustLite* [32] attest a defined area of code only. Their limitations are less significant compared to software-based attestation schemes. Thus, their resilience is considered to be *medium*. If the attested code contains the segment that handles device functionality, functional trust is gained. In contrast, referral trust requires the *attestation component* of the prover to be attested.

Software-based attestation. Generally, secure co-processors are not available on low-end embedded devices due to minimal cost requirements. Thus, purely software-based approaches were developed [45]. They do not assume any secrets on the prover's device, since there is no secure storage available at the prover side. Instead, these schemes are based on using side-channel information to decide whether an attestation result is valid. However, this approach poses many assumptions on the network topology and adversarial capabilities. For instance, the verifier needs to have direct communication with the prover with no intermediate hops [3]. We consider *resilience* of this attestation type as *low* because the potential attack surface is comparatively high. As attestation statements made by such attestations about other parties cannot be trusted, they can only provide functional trust.

Control-flow attestation. is a relatively recent development in the attestation landscape [2]. Static attestation, to which previously discussed attestation categories belong to, is not able to capture misbehavior of software during runtime. This is where runtime attestation comes into play by monitoring an application's control flow and detecting all deviations from the expected flow (documented in the security policy). This approach enables the highest trust guarantees of all attestation schemes. Runtime attestation schemes like *DIAT* [4] offer a *very high* resilience because they also protect against runtime adversaries, and thus can provide both referral and functional trust.