Veritaa-IoT: A Distributed Public Key Infrastructure for the Internet of Things

Jakob Schaerer
Institute of Computer Science
University of Bern
Bern, Switzerland
jakob.schaerer@inf.unibe.ch

Severin Zumbrunn
Abilium GmbH
Bern, Switzerland

Torsten Braun
Institute of Computer Science
University of Bern
Bern, Switzerland

***Abstract*—With the growth of the Internet of Things (IoT) and improved interoperability between various IoT devices of different manufacturers and owners, verifying the authenticity and integrity of data becomes a challenge. Cryptographic signatures together with Distributed Public Key Infrastructures (DPKI) and Distributed Ledger Technology (DLT) are promising solutions to this challenge. In this work, we extended the Veritaa framework, a DLT-based DPKI with an immutable signature store, to support IoT applications and evaluate its applicability. Therefore, we propose the systems architecture and implement a low-power IoT client. We built a real-world testbed with different sensors to evaluate the proposed architecture and extensions. In the evaluation, we analyze the time required to secure sensor data on the DLT, the overhead introduced to assert authenticity, integrity, and immutability of the data, and the security of our proposed solution.**

***Index Terms*—Internet of things; sensor networks; trust, security, and privacy**

I. Introduction

With many new applications in the Industrial Internet of Things, Smart Home, autonomous driving, and more, the Internet of Things (IoT) grows quickly. These applications require different devices with distinct requirements. For example, some devices might be complex and connected to a powerful energy source, while others fulfill a simple purpose and run on a single coin cell battery for years. However, all IoT devices have in common that they eventually need to exchange information with other IoT devices, applications, or users. In the past, most IoT devices used to have a single purpose, like monitoring a physical parameter and representing the observed values on a dashboard or raising an alert when a threshold was exceeded. Therefore, the IoT devices were mostly controlled by a single application and its owner. However, recent improvements in interoperability enable new applications that rely on exchanging information between the IoT devices of different manufacturers (MFR) and owners. For example, a Smart City application that monitors a city's air quality should access and evaluate the measurement values of all publicly available sensors and forward alerts to various recipients. When different applications share the same IoT infrastructure and new sensors are only deployed to not yet covered regions, the carbon dioxide footprint is lower and fewer resources are required. However, when applications exchange information with devices beyond an entity's infrastructure, verifying the authenticity and integrity of data is a challenge. The quality and accuracy of different sensors that monitor the same parameter might differ, and some users or operators might misbehave. Therefore, it is essential to identify the source of information and evaluate its trustworthiness. Furthermore, to achieve auditability, some applications require an immutable log. An auditable system enables tracking of the system's state at any given time in history.

Asymmetric encryption can be used to assert the authenticity and integrity of data. Certificate Authorities (CA) use Public Key Infrastructure (PKI) to certify the public keys of authenticated entities. However, centralized PKIs are prone to a single point of failure [1]. Furthermore, the tree-shaped CA scheme does not represent the rather meshed relations between entities owning IoT devices. Distributed Ledger Technology (DLT) based Distributed Public Key Infrastructure (DPKI) is a promising approach to overcome these drawbacks of centralized PKI [2]. Additionally, the immutability property of DLTs makes information exchanged in the IoT auditable, which enables new applications. However, the limited available resources of IoT devices constrain the applicability of common DLTs, and lightweight solutions are required.

We propose a Distributed Public Key Infrastructure for the Internet of Things in this work. Therefore, we extend Veritaa [2] with statements that enable low-power IoT clients, propose the architecture, and design and implement a low-power IoT client. We built a real-world testbed with custom low-power hardware to evaluate the system's performance.

II. Related Work

Public Key Infrastructures have been around for several decades. However, with the trend of distributed ledger technology, the research of DLT based PKI has gained attention, and several solutions have been proposed. Some of the proposed solutions target the Internet of Things.

In Blockchain for IoT security and privacy Dorrie et al. show how their blockchain can be used in Smart Home applications [3]. Their solution comprises local blockchains,

ISBN 978-3-903176-48-5

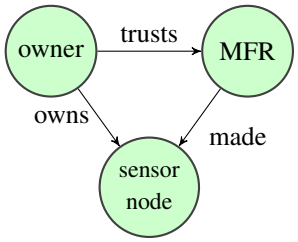

Fig. 1. Example of IoT device's real-world relations

a home miner, and local storage. The home miner generates shared keys for the devices and controls the access of the data.

Bouras et al. proposed a lightweight blockchain-based IoT identity management approach [4]. In their work, they suggest using a consortium (permissioned) blockchain to manage the identities of IoT devices. Their solution separates the membership service and the identity management protocol. The membership service is used to manage the blockchain nodes and network admins. Their identity management comprises registration, verification, and revocation of certificates. They have implemented a proof-of-concept prototype on hyper ledger fabric and evaluated their implementation.

Smart contracts of the Ethereum Blockchain have been used to build an Identity Management System for IoT devices [5]. The architecture comprises globally unique identifiers, identity management throughout the lifecycle, and ownership management of IoT devices. The proposed system relies on public-key cryptography. The manufacturer generates the key pair at production, requests a certificate from a CA, and then stores it on the created IoT device. Afterwards, the manufacturer registers the device identity on the Blockchain. Finally, the manufacturer uses ownership management to transfer the device ownership to the customer when the device is sold.

Blockchain-based PKI for smart meters has been proposed to provide trust in fuel dispensers [6]. In the proposed architecture, the certificate authority is replaced by a blockchain where permissioned endorsers directly can commit certificates. The permissioned endorsers are manufacturers' entities from society and institutions responsible for assuring the correct behavior of smart meters. The proposed model leaves it open how the permissioned endorsers are authenticated.

Veritaa is a DLT based DPKI with an immutable signature store [2]. Veritaa comprises the Graph of Trust (GoT), to immutably store signed declarations between identity claims of entities. In addition, a reputation system based on domain validation information and signed declarations between identity claims is used for the distributed certification of identity claims. Currently, Veritaa only supports full nodes, which require processing all transactions of the DLT and thus, are not suitable for the Internet of Things.

Multiple DLT based PKIs for the IoT have been proposed in the literature. In most of the related work, the public keys are certified by an entity, for example, the manufacturer of the device. However, to evaluate the quality of data measured by IoT, devices owned and operated by third parties, the validator requires more information than the authenticity of a public key. For example, the maintenance, and with that, the operator of an IoT device is essential for its well functioning. Therefore, multiple parties must be able to declare their relation to IoT devices to establish trust in IoT devices beyond their infrastructures. The Veritaa framework is a promising approach to build a DPKI where trust can be established beyond the own infrastructure. However, even if Veritaa relies on a lightweight BlockDAG, it is not applicable for the IoT. In this work, we propose and implement the architecture to support IoT clients in the Veritaa framework. Therefore, we extend the DLT with statements and pairing transactions. Furthermore, we design and implement a Veritaa IoT client running on low-power IoT devices. Our solution offloads the resource-demanding DLT operations to a more powerful Associated Full Node (AFN) to save resources on low-power IoT devices. Additionally, compared to other PKI, our proposed solution has the advantage that further declarations like made, measure, and pairing can be used besides the trust declarations. This makes the PKI more expressive than other solutions.

III. Distributed Public Key Infrastructure for the Internet of Things

The IoT comprises IoT devices that exchange information. While some IoT devices have plenty of resources, others are restricted in energy, computational capabilities, bandwidth, and memory. A DPKI for the IoT must be able to manage the spectrum of all these IoT devices.

A. Identity of IoT devices

When information is exchanged between IoT devices of different owners, operators, and manufacturers, it is essential that the receiver of the information can authenticate the source of the information. In order to authenticate IoT devices, each IoT device requires a unique identity to be distinguished from others. Merriam Webster defines identity as the condition of being the same with something described or asserted [7]. Consequently, identity consists of a set of properties that differentiate the distinct entities. If the identifying set of properties has the same values, they identify the same entity. Additionally, to prevent identity theft, the identifying properties must be hard to forge, which means it must only be possible for the distinct entity to obtain its identity. The required difficulty to forge an identity is often bound to the security requirement of an application. For example, today, in many applications, the identity of people is bound to their name, phone number, and address. Applications requiring higher security might use biometric information like fingerprints, retina, or the face as identifying properties.

In contrast to people, IoT devices do not have biometric information that can be used as identifying parameters. While many diverse types of IoT devices exist, many exact copies of each type exist. Therefore, an IoT device does not necessarily possess distinguishable properties, and the identifying properties must be set at manufacturing or later. Nevertheless, IoT devices have the advantage that they can use cryptographic

functions to build their identity. For example, a public key can be used as identifying property. The advantage of the public key as identifying property is that only the entity holding the private key can prove that it is this entity by signing some data. However, before the signature can be validated, the public key must be authenticated. The authenticating entity must verify that the public key belongs to the entity it claims. Since the IoT is growing quickly, it is impossible to authenticate the public keys manually, and PKIs are required. On a PKI, identity claims that map public keys to entities can be published, and third parties that have authenticated the identity claim can certify its validity. The certification can either be done with a CA model (PKI) or a distributed Web of Trust (WoT) like model (DPKI). In Veritaa, the Graph of Trust, a WoT like model, is used for the distributed certification of identity claims.

B. Graph of Trust and the Internet of Things

In Veritaa, all information is represented by the Graph of Trust (GoT). The GoT comprises identity claims, document identifiers, and their relations. The advantage of the GoT is that it inherently represents the structure of the relations between entities and data. With this information, it is possible to certify identity claims. Furthermore, it is possible to validate the authenticity and integrity of data [2]. This section shows how the GoT asserts authenticity and integrity of data in the IoT.

An identity claim of an IoT device cannot be treated as an identity claim of an individual or organization. Today, most IoT devices do not act and sustain independently, and individuals or organizations usually control them. Additionally, most IoT devices only interact with a few other entities and only receive a few trust votes. Consequently, it is difficult for an identity claim of an IoT device to gain enough reputation to be certified. Nevertheless, it is possible to represent the ownership relation between the owning entity and the IoT devices on the GoT. Therefore, it is possible to project the reputation of the owning entity to certify the identity claims of IoT devices. Figure 1 shows an example of relations between some entities related to an IoT device. All devices have a manufacturer and an owner. An owner usually trusts the manufacturer and owns the IoT device, and the manufacturer has built the IoT device. These relations and the reputation of the owner and manufacturer provide information about the trustworthiness of an IoT device.

The GoT is able to represent the relations between entities and their IoT devices. Figure 2 shows an example GoT where multiple companies have signed trust declarations to each other. Company B is a larger company with a hierarchy. Additionally, B owns several sensors and actuators. To represent the close hierarchy and ownership relations between the entities of identity claims, we introduced the pairing relationship. The pairing relationship comprises a signed request and a signed grant declaration. In this example, pairing is used to represent the organizational structure and the ownership of IoT devices. For example, a pairing between the identity claims $B1$ and $BS1$ is shown in Figure 2. Since the originating identity claims sign the relations, two identity claims can only be paired when both private key owners agree in the pairing. We elaborate the pairing in Section IV-C. All GoT readers can use this information to reason about the validity of the paired identity claims. When an entity interacts with a sensor or actuator, it only has to follow the paring relations to the root entity to validate its authenticity. The subgraph of an organization with all its paired identity claims can be considered a CA-like structure where the organization's identity claim is the root CA.

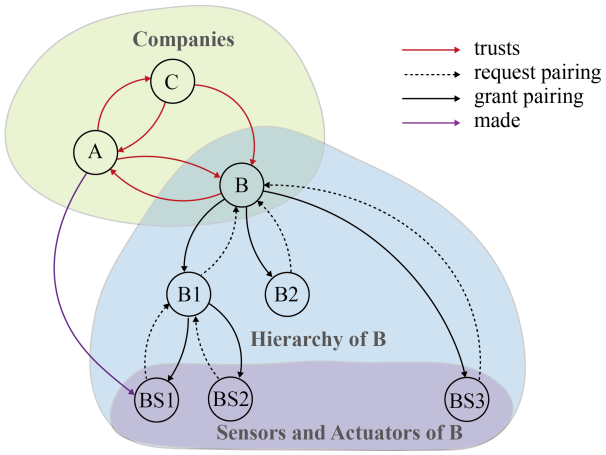

Fig. 2. Example declaration between identity claims on the GoT

C. Signing Measurements

Sensors' measurements can be secured with the GoT to assert their integrity, authenticity, and immutability. Therefore, the IoT device hashes the measurement and signs a declaration of type measurement towards this hash. In Figure 3 IoT device B has signed measurement declarations towards the measurement hashes M_1 and M_2. To achieve unique hashes for each measurement, we build the measurement hash from the following string: $|identity\_fingerprint|timestamp|measurement|$.

A validator can calculate a measurement sample's hash and search this hash in the GoT to verify the integrity and authenticity of the sample. When the validator has found the hash on the GoT, it can follow the declarations to see if a trusted identity has signed the declaration.

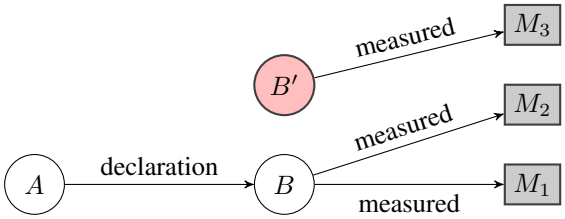

Fig. 3. Signing measurements and manipulated identity claim B'

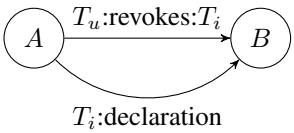

Fig. 4. Revocation of a declaration

D. Revocations

With a fast-growing number of IoT devices connected through the IoT, eventually, some of those devices might fail. The reason for failures is manifold. For example, a manufacturer could recognize a severe bug in the software and recall devices, or manipulation might result in the failure of the device. Therefore, an entity that has declared its trust on the GoT must be able to revoke the trust if the conditions for the trust are no longer given. On the GoT, an entity can revoke a signed declaration by signing a revoke transaction that contains the id of the revoked transaction. By declaring a revocation transaction, an entity states that the revoked transaction is no longer valid. Since all Veritaa full nodes maintain a copy of the GoT, they see that they should no longer use the revoked transaction in their calculations.

Figure 4 shows an example of a revoked transaction. With T_i entity A signed a declaration towards entity B. To revoke this transaction entity, A signed the revocation transaction T_u that references the revoked transaction T_i. Note that it is possible to revoke all kinds of transactions with this approach.

While the revocation transactions are helpful to update the reputation based on external conditions, they might fall short in case of manipulation. For example, when an attacker manipulates an IoT device, this might not be recognized by the entities that have signed trust declarations towards its identity claim. Hence, signing entities might not revoke their signed declarations, and their trust remains. To overcome the issue of trusting manipulated devices, the identity of the IoT device must change when an attacker manipulates the device. When the identity of the IoT devices changes on manipulation, the signed declarations still point to the old (not manipulated) identity. The new identity of the manipulated device starts over with no incoming trust declarations and, therefore, zero trust. Figure 3 shows an example, where the manipulation of the IoT device B changed its identity to B'. Since it has no incoming declarations, B' is not trusted.

To assure that an object's identity changes when the object has been manipulated, the identity must comprise at least one identifying property which always changes when an unauthorized change is made to the object. When an identifying property changes, also the identity changes. In terms of IoT devices, the device's private key must change on manipulation of the device to prevent a manipulated device from creating signatures in the name of the original device. Furthermore, the identity must be secured, not accessible, and unclonable. In Section IV-B we show how the tamper detection system is used to make an identity unusable in case of manipulation.

E. Associated Full Node (AFN)

Devices connected to the IoT are diverse in terms of functionalities and available resources. Infrastructure for the IoT must be designed to support all different types of devices. Since some IoT devices have limited computational capabilities, available energy, and storage, a DPKI for the IoT requires considering these constraints. For most IoT devices, it is not feasible to run a full node of a distributed ledger. In order to overcome this limitation, it is necessary to delegate the resource-demanding parts of the DLT to an Associated Full Node (AFN) and only manage the identities and signatures on the IoT device itself. The AFN is a normal Veritaa node that additionally manages an identity claim paired with the IoT device's identity claim. When the AFN is operated by the same party that operates an IoT device, the IoT device and the AFN can inherently trust each other. It must be asserted that information exchanged between the IoT device and the AFN cannot be dropped or manipulated. Since the AFN is paired with the IoT device, they have exchanged public keys and can verify each other's signatures. A valid signature asserts that a packet has not been manipulated. Acknowledgments (ACK) and negative acknowledgments (NACK) are used to assert that no information was dropped.

The initial implementation of Veritaa stored the transactions directly in blocks. In order to delegate the resource-demanding DLT operations to an AFN, we introduce statements.

F. Statements

In order to enable power-constrained devices, we extend Veritaa with statements. A statement comprises a list of transactions, the id of the creator, and its signature. A statement's transactions are chained by referencing the previous transaction's hash, and the creator signs the hash of the last transaction. Additionally, the IoT devices link the statements by referencing the last transaction of the previous statement with the first transaction of the new statement. Therefore, each IoT device creates its own chain of transactions. This chain of

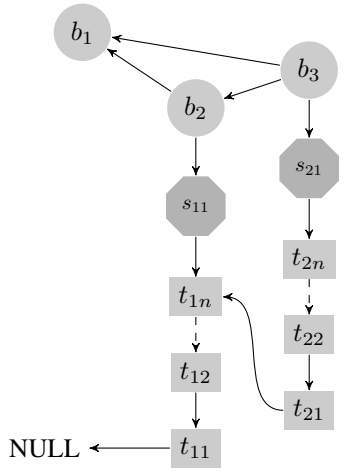

Fig. 5. Hash Tree extended with statements

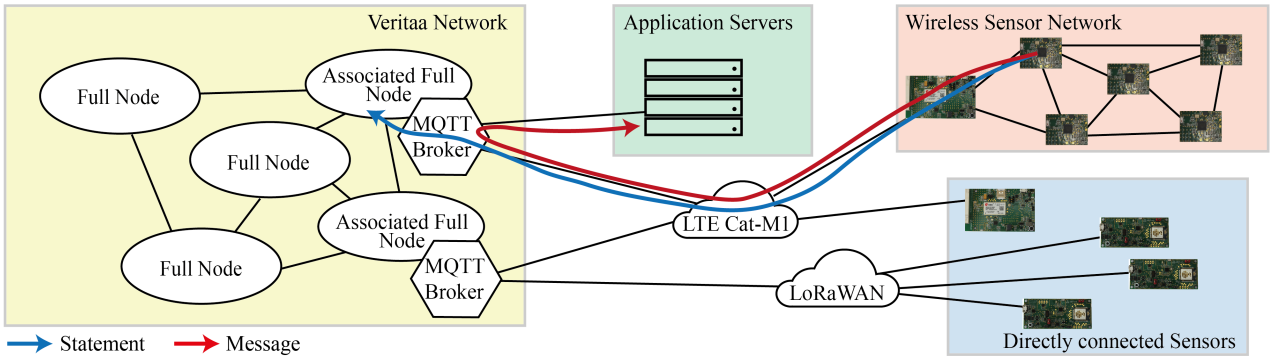

Fig. 6. Testbed and architecture overview

transactions asserts that statements cannot be dropped before they are written on the DLT. If a Veritaa node recognizes a missing referenced transaction, it sends a NACK to the corresponding IoT device, and the IoT device can resend the missing statement. Figure 5 shows an example hash tree of some blocks with two statements and their references. Each block contains a list of statements and each statement a signed list of transactions.

Compared to sending single transactions, the statement has the advantage that an IoT device does not have to sign each transaction individually. Consequently, the IoT device does not need to transmit the signature with each transaction, and, therefore, the energy required for this transmission can be saved. Furthermore, in contrast to committing complete blocks to a DLT, a statement has less overhead, and the AFN can perform the PoW required to protect the DLT from spam. Therefore, the statements enable offloading the block-overhead to the AFNs.

IV. Implementation

A. Testbed

To evaluate our proposed architecture and future algorithms and applications, we have built a flexible Veritaa IoT testbed. Figure 6 shows the architecture of our testbed that extends Veritaa with low-power IoT devices. The testbed comprises the Veritaa network, application servers, and IoT devices connected over LTE Cat-M1 or LoRaWAN. The Veritaa network is built out of several full nodes. Some of the full nodes operate as Associated Full Nodes (AFN), capable of handling IoT devices. The AFNs are equipped with an MQTT broker to exchange information between the Veritaa network, the IoT infrastructure, and the application servers. In our testbed, the Veritaa nodes are running in a Kubernetes cluster.

Veritaa assures the integrity and authenticity of messages by immutably storing their hashes and the creator's signature on a DLT. However, the messages are not stored on the DLT. While the message contains the data like sensor values, the signed statements ensure the message's authenticity, integrity, and immutability. Therefore, we distinguish between the signed statements directed to an AFN and the message directed to an application server. The message (red) and the statement (blue) are exchanged over an MQTT broker. The AFN and the application server subscribe to related topics to receive published statements and messages of their IoT devices of interest.

The architecture is designed to be flexible and support all kinds of IoT devices and IoT infrastructure. Sensor nodes with an Internet connection are connected directly to the corresponding MQTT broker. Gateways with an Internet connection are connected to the broker and translate the traffic to the given protocol for sensor nodes without a direct Internet connection. The gateways can map the traffic to different low-power protocols like MQTT-SN [8], SDNWise [9], or RPL [10].

We developed a gateway, a LoRaWAN based sensor, and a WSN sensor as low-power IoT devices for our testbed and designed and implemented the Veritaa IoT client to connect them to their AFN. In this testbed, all IoT devices are based on the EFR32MG21 microcontroller and custom circuit boards. The EFR32MG21 is a system on a chip with up to 96k RAM, up to 1 MB Flash memory, a multi-protocol radio module, and a security processor that enables the SecureVault technology of Silicon Labs [11]. To connect the IoT devices with the AFNs and application servers, we use the u-Blox SARA R510 for LTE Cat-M1 and the Microchip WLR089U0 for LoRaWAN. For the communication between the IoT devices, the EFR32MG provides a multi-protocol 2.4GHz module that supports 802.15.4 and Bluetooth Low Energy.

B. Identity with Microcontrollers

Veritaa, as a DPKI, manages the distributed certification of identity claims. The security of a private key is not in the scope of the Veritaa framework but in the scope of the applications that use the Veritaa framework to publish, authenticate, and certify identity claims. In this work, we use the EFR32MG21 microcontrollers that contain SecureVault, a dedicated security CPU that isolates cryptographic functions and data from the host processor core [12].

Figure 7 shows how we use Silicon Labs SecureVault to build the identity claim of an IoT device. All cryptographic functions are executed on the dedicated security CPU of SecureVault. The elliptic curve cryptography key pair is generated in the SecureVault, and only an encrypted wrapped key

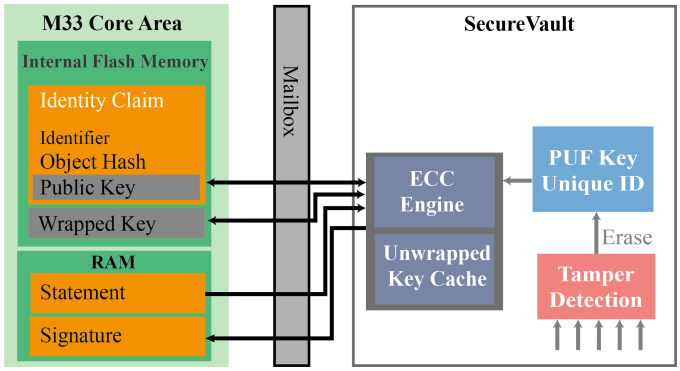

Fig. 7. Identity claim using Silicon Labs SecureVault

is stored in the internal flash memory of the microcontroller. Therefore, it is not possible to access the private key from the IoT device's firmware. However, the public key can be accessed by loading the wrapped key into SecureVault and exporting the public key. This public key can then be used to build the identity claim of the IoT device. The wrapped key is loaded from the flash memory into SecureVault over the mailbox when the private key is needed. Messages that must be signed are sent over the mailbox to the elliptic curve cryptography engine, and the signature is then returned over the mailbox.

The tamper detection system of SecureVault monitors several system parameters. When the tamper detection system detects a tamper attempt and a certain threshold is exceeded, the tamper detection erases the physical unclonable function (PUF) key. When the PUF key is erased, the wrapped keys cannot be encrypted anymore, and, therefore, the private keys are not usable anymore. Consequently, SecureVault's tamper detection can be used to prevent manipulated IoT devices from signing further transactions. SecureVault also provides Secure Boot with Root of Trust and Secure Loader to ensure that the firmware is not exchanged with malicious code.

Note that SecureVault is one way to implement identity claim management on an IoT device. The key pair could also be created using the arm TrustZone, and the trusted execution environment [13] or just plain keys. When the transactions are valid and the statements correctly signed, it is impossible to decide how the private keys for the signature were managed. However, the manufacturer can sign a "manufactured" declaration towards each IoT device it has made. Therefore, all who validate a signed message can see which manufacturer has created a device and use the manufacturer's reputation to reason about the quality of its security implementation. A responsible manufacturer can also transparently sign a declaration towards devices with specific security vulnerabilities. This transparency provides information that can be used to decide if a specific IoT device should be trusted or not.

C. Pairing

Figure 8 shows a sequence diagram of the pairing process. The pairing is conducted by a provisioner, an entity that is able to initiate signatures of the IoT client and an identity claim managed on the IoT client's AFN. Typically, the provisioner is the owner of the IoT device or the operator of the IoT device's infrastructure. The provisioner can use a smartphone application to pair an IoT device with its AFN. The following actions are performed to pair the IoT device (A) with its AFN (B): 1) The provisioner forwards the fingerprint and public key of B's identity to A and initiates the pairing request. 2) A creates a request pairing declaration that points towards the fingerprint of B provided by the provisioner, signs it in a statement, and forwards the statement to B. 3,4) B wraps the statement in a block and commits it to the DLT. 5) B confirms to A that the statement was successfully written to the DLT. 6) B reports to the provisioner that it has an open incoming request pairing declaration. 7) The provisioner validates that A signed the open pairing request and, if so, forwards the fingerprint and public key of A to the B and initiates the grant pairing. 8,9) B creates a grant pairing declaration towards A's identity claim, wraps it in a statement, and block and commits it to the DLT. Finally, 10) B sends acknowledgments to A and the provisioner to notify that the pairing was successful.

After pairing, the IoT device and the AFN have exchanged their public keys and, therefore, can validate each other's signatures and exchange encrypted messages. Furthermore, the pairing is declared on the GoT, and all Veritaa members can see that the IoT device is paired with the AFN and, therefore, reason about the authenticity data signed by the IoT device.

V. Evaluation

A. Security Analysis

Attackers might try to manipulate, repudiate, or forge measurement values. In this security analysis, we assume that the used cryptographic functions (hash functions and signatures) and that the tamper detection system of the microcontroller are secure. Furthermore, we assume that the DLT is secure, and therefore, all blocks committed to the network are immutably stored.

An adversary might try to manipulate, repudiate, or forge sensor values on the application servers after they have been measured. Suppose an attacker has manipulated or forged a sensor value on an application server. Then a validator can build the hash of the data and check on the GoT if the alleged IoT device has signed the data. If a signed declaration exists on the GoT, then data authenticity and integrity are proven.

An adversary that operates an application server might try to delete some data of an IoT device. Since the DLT only secures the data hashes, it can only prove that a data point existed at a given time, but the DLT cannot recreate it. However, if entities that do not trust each other collaborate on an application, all can store the data in their own infrastructure, and in case of a dispute, the DLT can be used to resolve the conflict. Assume an entity deletes a datapoint of its own copy of the data set and claims that the data point never existed. Then all members of Veritaa can see the signed declaration towards a hash with no corresponding data point. With this information, they can prove that the data point was deleted. Furthermore,

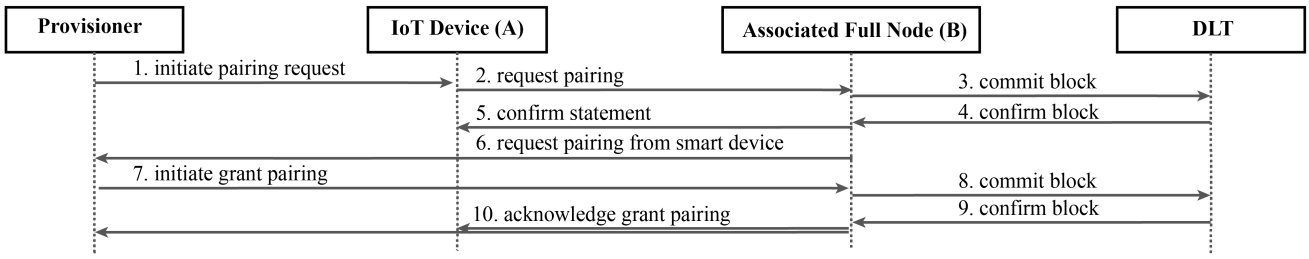

Fig. 8. The sequence of pairing an IoT device with its AFN

let us assume another entity also has a copy of the data set. Therefore, it is able to build the hashes of all data points and select the data point that corresponds with the disputed signed declaration on the GoT. Since the hash function is secure, two data points with the same hash are equal. Consequently, the solution is able to detect missing data, and the original data can be restored if non-trusting parties make their own copy of the data.

An adversary might physically manipulate the IoT device and abuse its identity to forge signatures. Let us assume that the attacker has manipulated the hardware or firmware of the sensor node. Then the tamper detection of the microcontroller will detect the manipulation. As discussed in Section IV-B, the detection of this manipulation erases the PUF key, and therefore, the identity of the IoT device is destroyed. Consequently, it is impossible to manipulate an IoT device and use its identity to sign forged declarations.

An adversary might try to manipulate a statement before it is written to the DLT. Let us assume that an adversary has inserted, removed, or manipulated a transaction in a statement. Then the hash of the transaction changes, and since the following transaction references the hash of the manipulated transaction, the hash of the following transactions would also change. Therefore, also the hash of the last transaction changes. Therefore, the signature of the statement, which signs the hash of the last transaction, is invalid after the manipulation. The DLT only accepts blocks that only contain valid statements. Consequently, the statement cannot be manipulated. Furthermore, assume an adversary dropped a statement before it was written to the DLT. Then the first transaction of the following statement would reference the transaction hash of the last transaction of the dropped statement. The AFN would recognize that the first transaction is referencing a non-existing transaction and can send a NACK to initiate retransmission. Consequently, it is not possible to drop a statement unrecognized.

B. Transmission time to DLT

Between an event that initiates a new transaction and the time the transaction is written immutably on a distributed ledger, time elapses. In IoT applications, this time depends on the underlying network architecture and technology. For example, LoRaWAN has a limited throughput, 5G enables high bandwidth with low latency, and connections through Wireless Sensor Networks have relatively high packet loss rates. Consequently, in IoT applications, the time until a transaction is written to the DLT can deviate.

This experiment evaluated the time until a statement's transactions have immutably been written from an IoT device to the DLT. In this experiment, we connected the IoT devices over LTE Cat-M1 and LoRaWAN. Note that the radio might longer be active than the time required to commit a transaction on the DLT. This is because the radio remains active after the transmission and waits for ACKs.

Figure 9a shows the time required to commit a new statement from the IoT devices to the DLT. The experiment shows that for LTE Cat-M1, times below 1s are feasible. Note that these times include the whole process, especially the wake-up times of the microcontroller and the SARA-R510 module.

Figure 9b shows the performance when the statements are transmitted over LoRaWAN. Compared to LTE Cat-M1, the LoRaWAN network has a lower bandwidth and a lower MTU size to reach larger distances. However, the hashes of transactions and the signature of the statement require additional space that might exceed the MTU when multiple transactions are sent. Therefore, we split the statement, and each transaction is sent separately to the AFN. When the last transaction has been sent, the signature for the complete statement is transmitted. The AFN validates the received transactions and signatures and assembles the parts as a statement. Due to the low bandwidth, the sensor nodes connected over

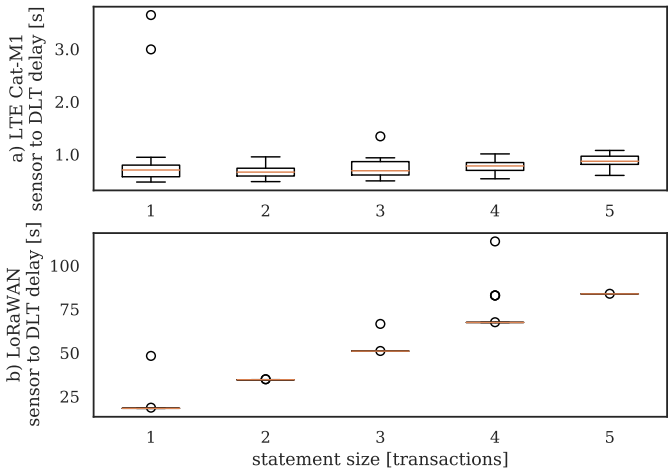

Fig. 9. Required time to send a statement to the DLT.

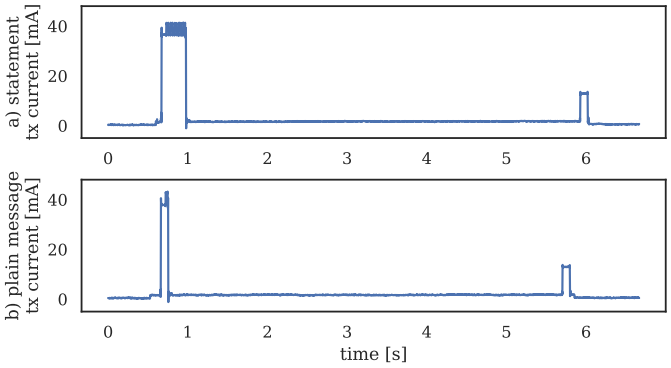

Fig. 10. LoRaWAN tx current consumption

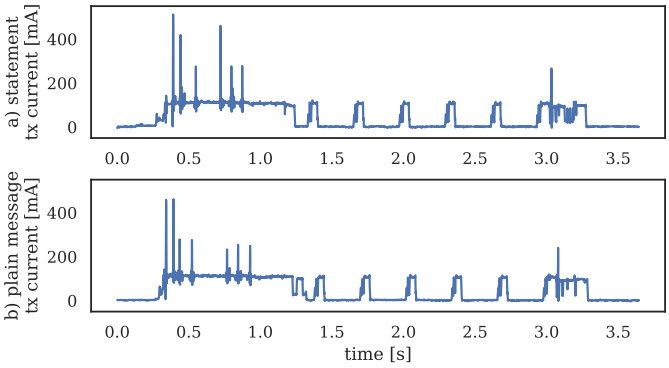

Fig. 11. LTE Cat-M1 tx current consumption

LoRaWAN also occupy the medium for a longer time. The range, data rate, and MTU of LoRa depends on the spreading factor. In this experiment, we used the spreading factor of 7, which results in a maximum payload size of 222 bytes. Note that for spreading factors > 9, the MTU is lower than a transaction, and therefore, a further split of the transaction would be required for the transmission. After transmitting a packet, the sensor node must remain silent to meet the maximum duty cycle requirement. With the given LoRa setup, it is possible to send up to approximately 200 uplink messages per hour (depending on the payload size) [14]. Note that the fair access policy of some LoRaWAN networks might even have lower limits. The results include the sleep time to meet the maximum duty cycle between each statement packet. The LoRaWAN experiment shows that it requires 20 s to commit a transaction to the DLT.

C. Energy Consumption

The statement comprises hashes, creator id, and creator signature to ensure the transmitted data's immutability, integrity, and authenticity. We have measured the current and energy consumption required to transmit statements and plain messages to evaluate the statement overhead. Figure 11 shows the current consumption when the message is transmitted over the LoRaWAN using spreading factor 7 and 125 kHz bandwidth. With this configuration, the bit rate is 5.47 kbps. In this experiment, we transmitted a 32 bytes long plain message, which resulted in a 170 bytes long signed statement. Due to the relatively low data rate of LoRa, the transmission of data consumes most of the energy.

Figure 11 shows the current consumption when the message is transmitted over LTE Cat-M1. LTE Cat-M1 has an uplink bit rate of up to 1.2 Mbps. Therefore, the air time of the data is around 200 µs for the plain message and 1.1 ms for the statement. However, after the SARA module has transmitted a packet, it remains awake for around 3 s and listens to incoming messages and ACKs. Consequently, the size of the transmitted information does not contribute much to the overall energy consumption.

We calculated the energy to transmit a statement and plain message based on the current consumption. Figure 12 shows the energy required to transmit a statement and plain messages for LoRaWAN and LTE Cat-M1.

Equation 1 shows how overhead is calculated. The overhead sets the direct costs in relation with the indirect costs, where $c(x)$ is the cost function.

$$overhead = \frac{c(statement) - c(message)}{c(message)} \cdot 100 \quad [\%] \quad (1)$$

Table I shows the signature overhead when messages are secured with our proposed solution. A signature overhead of 431.25% seems high when using the required bytes as a cost function. However, the signature overhead is lower if we use the energy required to transmit the message as a cost function. Since establishing the connection and listening for acknowledgments also requires energy, the signature overhead does not contribute that much to the overall energy consumption.

The experiment shows that securing measurements with our proposed solution requires 73 % more energy with LoRaWAN and 5.16 % more energy with LTE Cat M-1 than when the measurements are not secured. However, this overhead is affordable for applications that require authenticity, integrity, and immutability of sensor data. If the sensor depends on a power supply like a solar cell, the solar cell must be dimensioned accordingly, and if the sensor depends only on batteries, they need to be replaced more frequently. Note that the quality-relevant sensors that benefit from this additional security also require frequent maintenance like calibrations.

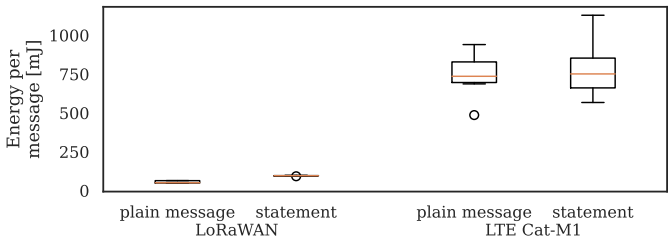

Fig. 12. Required energy to transmit a message

	LoRaWAN		LTE Cat-M1	
	statement	plain message	statement	plain message
Data used [bytes]	170	32	170	32
Data overhead [%]	***431.25***		***431.25***	
Energy used [mJ]	95.67 ±1.66	55.30 ±7.35	784.65 ±160.69	746.12 ±116.23
Energy overhead [%]	**73.00**		**5.16**	

TABLE I
SIGNATURE OVERHEAD

VI. CONCLUSION AND FUTURE WORK

In this work, we built a Distributed Public Key Infrastructure for the Internet of Things with an integrated signature store based on the Veritaa framework. Therefore, we extended the GoT with new pairing capabilities, the ABCG with statements to offload DLT functions to an AFN and enable low-power IoT devices, designed network architecture, and implemented an IoT client for Veritaa. We have built a real-world testbed to evaluate the extensions and enable future experiments and application testing. The testbed comprises custom-made low-power IoT devices using LoRaWAN and LTE Cat-M1 for communication. As a proof-of-concept, we used the testbed to perform several experiments.

We evaluated the delay until messages are secured on the DLT and the overhead introduced by securing the messages on the DLT. Our evaluation shows that securing the information introduces some overhead. However, there is always a tradeoff between energy consumption and security in low-power applications. For many applications that require this level of security, the additional required energy is affordable. Furthermore, the security analysis showed that the proposed architecture can ensure IoT device data's authenticity, integrity, and immutability.

We showed how the GoT is able to establish trust in the authenticity of the public keys of IoT devices. However, compared to other PKIs, the GoT is more expressive and allows signed declarations beyond trust, made, measured, and pairing. In future work, we will investigate how additional types of declarations can assert data quality.

REFERENCES

[1] Ankush Singla and Elisa Bertino. "Blockchain-Based PKI Solutions for IoT". In: 2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC). Philadelphia, PA, Oct. 2018, pp. 9–15. DOI: 10.1109/CIC.2018.00-45.

[2] Jakob Schaerer, Severin Zumbrunn, and Torsten Braun. "Veritaa: A distributed public key infrastructure with signature store". In: *International Journal of Network Management* (Sept. 6, 2021). DOI: 10.1002/nem.2183.

[3] Ali Dorri et al. "Blockchain for IoT security and privacy: The case study of a smart home". In: 2017 IEEE International Conference on Pervasive Computing and Communications: Workshops (PerCom Workshops). Kona, HI, Mar. 2017, pp. 618–623. DOI: 10.1109/PERCOMW.2017.7917634.

[4] Mohammed Amine Bouras et al. "A Lightweight Blockchain-Based IoT Identity Management Approach". In: *Future Internet* 13.2 (Jan. 22, 2021), p. 24. DOI: 10.3390/fi13020024.

[5] Ahmad Sghaier Omar and Otman Basir. "Identity Management in IoT Networks Using Blockchain and Smart Contracts". In: 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData). July 2018, pp. 994–1000. DOI: 10.1109/Cybermatics_2018.2018.00187.

[6] Wilson Melo et al. "Public-Key Infrastructure for Smart Meters using Blockchains". In: 2020 IEEE International Workshop on Metrology for Industry 4.0 IoT. June 2020, pp. 429–434. DOI: 10.1109/MetroInd4.0IoT48571.2020.9138246.

[7] *identity*. In: vol. Merriam-Webster.com. 2021. URL: https://www.merriam-webster.com/dictionary/identity (visited on 08/25/2021).

[8] Andy Stanford-Clark and Hong Linh Truong. *MQTT For Sensor Networks (MQTT-SN) Protocol Specification*. International Business Machines Corporation (IBM), 1999, p. 28.

[9] Laura Galluccio et al. "SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks". In: 2015 IEEE Conference on Computer Communications (INFOCOM). ISSN: 0743-166X. Apr. 2015, pp. 513–521. DOI: 10.1109/INFOCOM.2015.7218418.

[10] Tim Winter et al. "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks." In: *rfc* 6550 (2012), pp. 1–157.

[11] Silicon Labs. *EFR32xG21 Wireless Gecko Reference Manual*. Sept. 8, 2020. URL: https://www.silabs.com/documents/public/reference-manuals/efr32xg21-rm.pdf (visited on 09/14/2021).

[12] *UG103.05: IoT Endpoint Security Fundamentals*. Silicon Labs, p. 13. URL: https://www.silabs.com/documents/public/user-guides/ug103-05-fundamentals-security.pdf (visited on 11/12/2021).

[13] Arm Ltd. *TrustZone*. Arm Developer. URL: https://developer.arm.com/ip-products/security-ip/trustzone (visited on 09/14/2021).

[14] *Airtime calculator for LoRaWAN*. URL: https://avbentem.github.io/airtime-calculator/ttn/eu868/90,12 (visited on 09/10/2021).