Energy-Efficient Data Anchoring for IoT Devices

***Abstract*—With the advent of the Internet of Things (IoT), device-generated data has surged significantly. These ever-increasing volumes of data need to be trusted and relied on for various purposes. Data consumers heavily depend on the correctness and integrity of the generated data sets, whereas inadequate IoT security measures increase the risk of data contamination. While data quality problems due to technical issues like sensor faults, unstable communication links, and storage failures can be mitigated with simple measures like redundant nodes and checksums, protection against arbitrary attacks by hackers or insiders requires more sophisticated mechanisms.**

Blockchain technology enables such an integrity protection mechanism as data is stored in a distributed ledger that cannot be controlled by a single entity. However, the main challenge of this approach in the context of resource-constrained IoT devices lies in the potentially high computational overhead of the underlying cryptographic operations as well as additional storage and bandwidth requirements. Therefore, this paper proposes an energy-efficient data anchoring method tailored for IoT sensors that provides end-to-end security guarantees using a combination of advanced hash trees and digital signatures. By checking the corresponding ledger transaction, any interested party can verify the authenticity of arbitrary sensor data items, that were shared by the responsible data provider. Our experiments show a negligible energy overhead for sensor nodes compared to the unprotected protocol, thus proving the practical feasibility of the approach.

I. Introduction

Wireless sensors are used in different domains like industrial applications or home automation to provide a better insight into the current state of a machine or the environmental conditions of a building [1], [2]. These wireless devices communicate via a radio channel with a gateway, which may forward the data to other components connected via the Internet. This radio link represents the last mile of the communication to a higher-level system. The communication channel can also be used by a potential attacker [3]. In case that data is not directly consumed by the higher-level system but requested on-demand at a later stage (e.g., for an investigation after an accident), there must also be a way to prevent tampering with the produced sensor data at the corresponding storage layer (e.g., a database located in the cloud). Otherwise, involved parties (e.g., with database administration rights) could easily manipulate stored sensor data, thus changing the history of the system for external investigators. Similarly, hacking attacks on the storage layer or related components may cause the distribution of forged data.

Therefore, it is crucial to ensure the security of the entire system connected to the wireless sensors, in particular the integrity of the generated data. It must be ensured that this data really comes from the expected sensor devices and that it has not been manipulated between the time of measurement and the time of access by an interested internal or external party.

The measurement data have to be transmitted to the higher-level system via the wireless communication channel and potentially other network connections. It is important that all data be secured with respect to malicious modifications during the transmission or at a later stage. A simple concept is to encrypt the transmitted data or to use a cryptographic message authentication code (MAC). Typical System-on-Chip (SoC) solutions only support basic symmetric encryption algorithms, e.g., AES with 128 bit. This means that every participant in the network has to know this shared secret to participate in communication. Such a shared secret is a potential security threat and a secure method to establish the secret is complex [4].

On the other hand, asymmetric cryptographic algorithms could be used to sign messages and prevent malicious modifications, e.g., elliptic-curve cryptography algorithms. However, such algorithms are resource-intensive in terms of energy. Wireless sensor nodes are typically powered by batteries or energy harvesting systems (EHSs) [5]. Using batteries, the overall lifetime is mainly limited by the battery's capacity and the average power consumption. Using EHSs, the available energy is theoretically unlimited, but the average available power depends on the environmental conditions [6]. In both cases, the wireless sensor nodes must be classified as energy-constrained systems. Thus, energy-demanding algorithms can only be executed sparingly.

The amount of energy-demanding message signing operations can be reduced by applying optimized algorithms. This paper presents an energy-efficient method to secure continuous sensor data from energy-constrained IoT devices. Instead of signing each individual measurement, sensor devices can combine many measurements into a hash tree (a so-called Merkle tree [7]), whose root hash can be signed and transmitted in periodic intervals. At the gateway node, data from multiple sensors can be combined again using another hash tree.

Subsequently, the corresponding root hash needs to be stored in a reliable and secure fashion. One possible approach would be to simply use a trusted third party that promises to safeguard this metadata, but this would introduce a single point of failure and the possibility of collusion. Therefore, the suggested approach anchors measurement data on an append-only distributed ledger (i.e., a blockchain), which means that the root hash is included in a ledger transaction. Due to the one-way property of hash functions, it is unfeasible to manipulate the underlying sensor data without changing the hash value in the ledger transaction. Furthermore, as a distributed ledger is controlled by a peer-to-peer network of decentralized nodes that is secured via cryptographic mechanisms, confirmed transactions are basically immutable because they cannot be reversed without a network majority [8]. As each ledger transaction is reliably timestamped, this ensures that retroactive manipulations of sensor data are practically impossible even if an attacker gains complete control over the measurement system (including the sensor devices and the storage layer).

The feasibility of blockchain-based data anchoring in the IoT domain has already been demonstrated in the past [9]–[11]. However, previous approaches do not consider end-to-end data protection guarantees (i.e., from the sensor to the data consumer) in energy-constrained IoT scenarios. Data is typically not directly signed by the embedded sensor devices and attacks on the (remotely accessible) gateway node may compromise the security mechanism. Our main contribution is therefore the design and prototypical development of a data anchoring mechanism that is efficient enough to run on resource-constrained embedded sensor devices with minimal overhead.

The rest of the paper is organized as follows: Section II discusses a motivating use case and outlines the structure of the involved hardware. Section III presents the design of the data anchoring and verification processes. Section IV presents implementation details regarding the prototypical implementation of the approach, while Section V summarizes the benchmark results with respect to the optimization of power consumption. Section VI analyzes the feasibility of the suggested approach with regard to related work. Finally, Section VII concludes the paper and outlines directions for future work.

II. Motivating Use Case

As a real-world motivating use case, we consider a wireless sensor node intended to measure temperatures at automotive test beds [12]. There, it is necessary to be able to continuously transmit temperature data of certain motor parameters during certification processes. These data must not be changed by anybody, and the authenticity and integrity of the data must be maintained. To measure the temperature, wireless sensor nodes for high precision temperature measurements [13] are directly applied to a motor-under-test. Such a wireless sensor node is used for the prototype implementation and evaluation of the energy-constrained firmware. Besides the wireless sensor node, the hardware setup is composed of a Raspberry Pi embedded PC and a Particle ARGON board connecting them via a Bluetooth Low Energy (BLE) communication link.

Fig. 1 shows an overview of the investigated hardware setup. The wireless sensor node uses solar-based energy harvesting for supply. Thus, the implemented firmware must be implemented considering the power and energy constraints implied by energy harvesting as well as the computational and memory limits of the sensor device.

III. Data Anchoring Mechanism Design

In this section, we suggest an innovative method by which resource-constrained sensor nodes can continuously anchor measurement data into an append-only distributed ledger with

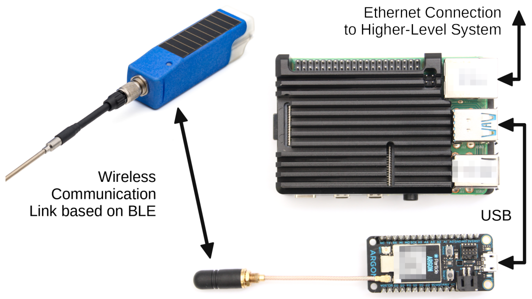

Fig. 1. Hardware structure overview showing the wireless sensor node (blue), the Particle ARGON board with a BLE antenna and the RaspberryPI embedded PC.

the help of edge nodes that act as gateways. It is worth mentioning that the data will be stored and ready to access in real time; however, the verification would be possible after a delay, which depends on the anchoring configuration. End-to-end security guarantees can be achieved as the sensors use digital signatures to protect data integrity. We leverage Merkle Tree [7] and Merkle Patricia Tries [14] to accumulate many sensor samples from different sensors in a single root hash, which is consequently published to a distributed ledger. Furthermore, the design considers the IoT limitations and constraints regarding energy consumption, computational power, memory usage, and communication bandwidth. In the following, we provide an in-depth description of this secure data anchoring mechanism.

In the proposed design, we recognize four separate entities: the sensors denoted by set S, the edge devices E, the distributed ledgers $\mathcal{L}$, and validators V. The sensors and edge devices must cooperate but can be governed by different individuals, as sensors consider edge devices as semi-trusted entities. The objective is to anchor the data records R generated by the sensors in the ledger/ledgers $\{\ell_0, ...\ell_m\} \in L$, which thus can be validated by the validator on the other end of the anchoring pipeline. Furthermore, in the occurrence of a dispute, these anchored records can be used to trace back and identify the underlying issue. Moreover, they can be utilized for data quality assessment. Fig. 2 depicts the bird's eye view of the proposed schema. In the following subsections, firstly, we will discuss the required background knowledge, then elaborate on preliminary solutions, and finally, explain our method in detail.

A. Data Anchoring

A record $r \in R$ is considered anchored if either directly, the record itself, or indirectly, i.e., the record's hash, is published at least in one secure append-only distributed ledger $\ell \in \mathcal{L}$. R is defined as a superset of all time series of sensors' samples, $R = (R_{s_1}, R_{s_2}, .., R_{s_{|S|}})$. Examples of aforementioned distributed ledgers can be Blockchain-based solutions like Bitcoin [8] and Ethereum [14], or Directed Acyclic Graph (DAG) based alternatives like IOTA [15].

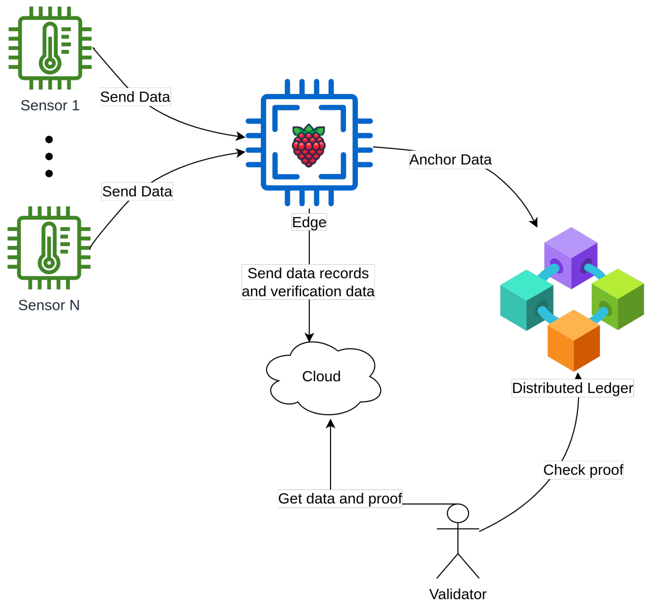

Fig. 2. The bird's eye view of our proposed schema

B. Data Validation

A record $r \in R$, generated at timestamp τ by sensor s will be considered valid if for time τ, it is uniquely anchored in a secure append-only distributed ledger and this anchored data is the only representative of the underlying record. It means that for the pair (r_s, τ), there is one and only one anchored value that directly or indirectly (via its hash) represents the record in the ledger. Furthermore, the representation should be unique or probabilistically unique to the data record. In other words, practically the chance of finding two different records with identical representations must be negligible.

C. Accumulators

This section introduces the basic data structures that will be used for anchoring in the subsequent solutions. Accumulators are space- and time-efficient data structures to examine the membership in a set [16]. Three different types of accumulators are relevant to the suggested anchoring protocol.

A *Concatenator* is an elementary accumulator that concatenates all members with a separator placed between elements. All members must be scanned to check membership.

A *Merkle Tree (MT)* is an accumulator tree data structure that provides efficient and secure membership validation for leaf node data. The tree is built bottom-up from the leaf nodes, linking them through a hash function. Each data point first goes through a hash function, to preserve data privacy in the generated proofs, to construct the leaves. The concatenation of two neighboring leaf nodes (left and right) is fed again to a hash function to build an intermediate node, which is the parent of both leaf nodes. This process continues until the tree's root node, the so-called Merkle root, is created. This value can then be anchored to a distributed ledger to ensure the integrity of all included data records. Using different hash functions for leaves and intermediate nodes is advised due to security concerns. Otherwise, the MT cannot distinguish between a multi-leveled structure of a tree and a tree containing only two leaves with the same hashes.

To inspect the membership of a data record in an MT, the party that holds the tree and its underlying data should supply a Merkle proof. The proof is created by providing the requested data record, the immediate neighbor leaf, and neighbors of any intermediate node traversing from the questioned data record to the root node. Using the requested data record and the Merkle proof, anyone can recompute the Merkle root and compare it with the corresponding anchored value on a distributed ledger, thus validating that the data record was included in the original Merkle tree.

Merkle Trees are straightforward to build and efficient in proof generation while preserving leaves' data privacy. MTs are parallelizable, scalable, and cryptographically ensuring the integrity of underlying data records.

A *Merkle Patricia Trie (MPT)* is an advanced tree-based accumulator. Unlike a Merkle Tree, a Merkle Patricia Trie is a key-value storage, which means only one value can be presented for each unique key. Each leaf in the MPT holds a value and the longest uncommon suffix of the key with neighboring leaves. The intermediate nodes are either extension nodes or branch nodes. Extension nodes store the longest common prefix of the tree traversing from the root and a pointer to a branch node. The branch nodes keep nibble/s (half-byte) of a key, which is shared among its children. Branch nodes point to either extension nodes or leaves.

The membership in MPT is checked against an MPT proof provided by the tree and data holder. Proofs are generated similarly to the MTs', however for MPTs it is essential to capture the order pairs of key-value plus the internal data structure of branch and extension nodes.

D. Identity Management

For the sake of simplicity, we assume a straightforward identity and user management for the sensors and edge nodes. It is assumed that each node has a public-private key pair denoted by Pk and Sk, respectively, which is configured manually in the bootstrapping phase by system administrators. Additionally, identities are known to the validator in advance; therefore, the validator expects a particular signer (i.e., with a specific public key) for a specific record.

A natural extension of this approach would be to use smart contracts on the distributed ledger to manage the public keys of sensors and edge devices as well as their allocation to specific test sites. This would increase the system's flexibility regarding identity revocation and dynamic attachment of new nodes. However, open issues regarding key management and trust in the responsible system administrator remain, which are beyond the scope of the current paper.

E. Solutions

This part will discuss a couple of possible solutions and their limitations. In the end, our approach will be described.

1) Direct solution: This simple and naive solution for secure data anchoring would store sensors' data values or their hash on the ledger. If the hash function is denoted as $Hash(\cdot)$, signing as $Sign(\cdot)$, building a transaction as $BuildTx(\cdot)$, and broadcasting as $Send_{ledger}(\cdot)$, for a data record in the time series of sensor n in $r \in R_{s_n}$ the anchoring function, $Anchor(\cdot)$, can be defined as Eq.1.

$$Anchor(r) \leftarrow Send(BuildTx(Sign(Hash(r)) \parallel Hash(r))) \tag{1}$$

For validation, the validator needs to fetch the data point for sensor s, timestamp τ, and its reference on the ledgers from the cloud then does the reverse process of anchoring Eq. 2.

$$\begin{aligned} Validation(r) \leftarrow &(Hash(Fetch_{data}(s,\tau)) == Fetch_{ledger}(s,\tau)) \\ &\wedge Validate_{signature}(Fetch_{ledger}(s,\tau)) \end{aligned} \tag{2}$$

However, on the sensor side, this approach would consume extensive resources on hashing and signing, consequently increasing the power consumption and the requirements on the sensors' processing power. In a bigger picture, assumingly, the sensors generate data at a rate of one record per second; with dozens of sensors, we would easily reach the ledgers' limitation on scalability and face potentially very high transaction costs.

2) Trusted Edge Solution: This is a solution in which we assume that the same organization manages the sensor and the edge device. In this setup, the sensors, S, send the data records, R, in plaintext or over a secure connection to the edge device. Afterwards, the edge device would use an MT to aggregate the received data records of each sensor to its respective tree. Whenever the tree reaches its capacity, the root hash for the tree, along with metadata, will be saved on the distributed ledger. Although this approach has low energy consumption and minimal sensor requirements, its biggest drawback is the need for trust in the edge device, which is often unrealistic in real-world situations. Additionally, claiming end-to-end security under this assumption is impossible. Another weakness of this method lies in its linear correlation between sensor and transaction numbers.

For validation, the validator must know the location of the anchored records and the associated proof, which proves that a particular record exists in an MT [17]. Initially, the validator queries the ledger to verify the existence of the tree's root hash. Then, using the root hash and the Merkle proof, it would validate the queried data records.

3) Trusted Edge Solution with Aggregation of Multiple Sensors: The previous approach can be improved by using an MPT as an additional accumulator to mitigate the scalability problem. This method follows a nearly analogous procedure as the previous method; however, in the last step, instead of anchoring the Merkle root hash of each sensor, they are added to an MPT with the sensor ID and the associated time frame as the key and its root hash as value. The advantage of using an MPT in the last step instead of an MT lies behind the fact that the MPT creates a one-to-one relation between the sensor ID and captured timestamp with its underlying MT root value. As a result, multiple records for the same sensor ID and timestamp cannot be placed in one MPT.

Through the verification process, after fetching the anchored MPT root hash from the ledger, it is necessary to validate the existence of the Merkle root hash inside the MPT using proof made for the MPT. Then, the inclusion of requested data records will be checked against the MT using its proof. While this method relieves the scalability problem, it is still challenged by the trusted edge device assumption.

4) Our method: We tackle the mentioned problems using three different accumulator data structures to make anchoring less frequent and more affordable, therefore making the solution feasible to implement and operate in real-world scenarios while keeping the solution secure. Fig. 3 depicts the anchoring process. As it has been shown, the data records are aggregated using two different accumulators, concatenator and MT, in the sensor. Then, the digest of the accumulators built in separate sensors is sent to the edge device to get further added to another accumulator, MPT. In the following, the method is described in greater detail.

In the proposed method, inside the sensor s_n, each data record in R_{s_n} will be added to a concatenator C on their arrival order until it reaches accumulator capacity, creating a chain $c_k^{k+|C|} = r_{s_n}^k \parallel ... \parallel r_{s_n}^{k+|C|} = \sum_{i=k}^{k+|C|} r_{s_n}^i$, whereas k is the starting index of the sensor's data record, indicated by the red circle indexed 1 in Fig. 3. Meanwhile, it sends each sample to the edge device as well. When a concatenator is filled up, it will be fed to an MT along with its corresponding first (τ_k) and last ($\tau_{k+|C|}$) sample timestamp, $l_k^{k+|C|} = \tau_k \parallel \tau_{k+|C|} \parallel c_k^{k+|C|}$. Concatenating the timestamp with the data records makes it unattainable for the sensor, edge device, and cloud to put overlapping time intervals in one tree. Gradual insertion of concatenators to the MT will continue to the point that MT is filled to its capacity (Eq. 3), indicated by the red circle indexed 2 in the figure. To construct intermediate nodes of MT, besides the left and right leaves, the lower τ_{start} and upper τ_{end} timestamp's bound of aggregating leaves have been concatenated.

$$mt_k^{k+|MT|\times|C|} := \begin{cases} Hash(mt_k^{\lfloor k+\frac{(\frac{|MT|}{2})\times|C|}{2}\rfloor}, mt_{\lfloor\frac{(\frac{|MT|}{2})\times|C|}{2}+k+1\rfloor}^{k+(\frac{|MT|}{2})\times|C|}) \parallel \tau_{start} \parallel \tau_{end} & \textit{if } |MT| \neq 1, \\ leaf & \textit{Otherwise} \end{cases} \tag{3}$$

It is worth noting that, MT can be built step-by-step and bottom-up in practice. Therefore, it is unnecessary to keep the whole tree in the memory. In that case, the memory usage of building MT is $O(\log_2 n)$.

Subsequently, the sensor will sign the tree's root hash and send it along with the root hash to the edge device (Eq. 4), denoted by the red circle indexed 3.

$$\begin{aligned} \lambda(k, k+|MT|\times|C|) \leftarrow &Send(Sign(Root(mt_k^{k+|MT|\times|C|})) \\ &\parallel Root(mt_k^{k+|MT|\times|C|}) \parallel \tau_k \parallel \tau_{k+|MT|\times|C|}) \end{aligned} \tag{4}$$

On the edge device side, upon arrival of the new root hash, the Merkle Tree, $mt_k^{k+|MT|\times|C|}$, will be rebuilt using the received data records, and the resulting root hash will be compared against the obtained one. If they match, the accompanied signature will be validated. Consequently, after confirmation of the received tree, the new root hash, root hash signature, and the covered timestamp will be added to a Merkle Patricia Tree on the edge with the corresponding sensor identity as the key, marked by the red circle indexed 4 in the figure. As a result, the MPT will hold different sensors' MT root hash. When the MPT has reached its maximum allowed size, a transaction containing the root hash will be created and broadcast to the chosen distributed ledger, shown by the red circle indexed 5. Furthermore, the edge device will send the MTs, MPTs, and anchoring information to the cloud as well; with the provided information the cloud can create the proofs.

In the validation process, the validator queries the cloud to get the requested data records, their anchoring references on one or more distributed ledgers, and their proofs on the Merkle Patricia Trie and Merkle Tree. Initially, it will verify if the MPT root hash is present in the anticipated ledgers. Afterwards, it is essential to confirm the correctness of the MT that is saved inside MPT, which represents the sensor data records that have been requested, provided that the previous step has been successful. This step uses the proof of MPT targeting the MT root hash generated by the cloud. Finally, it is essential that the existence and uniqueness of the records during the requested period will be verified against the MT; in order to do that, the validator should utilize the cloud-generated MT's proof for requested data records, which have been stored in the MT. The records will only be considered valid after completing the last verification step.

Deliberate or unintended disruption can occur during or after the anchoring and validation process. These disruptions can have internal or external sources. The attacker may try to cause a Denial-of-Service (DoS) attack for the external scenario. While it is possible to carry out such an attack by bombarding the cloud, it won't stop the anchoring process; as a result, there won't be any interruption in the correctness of newly anchored data records. An external or internal attacker may want to alter previously anchored data records; however, since the distributed ledger is assumed to be secure and append-only, this attack is infeasible. Another internal attack may initiated by a collusion between the edge device and the sensors' owner. One possible scenario is that they try to insert two records in the MPT using a slightly different timestamp. Nevertheless, since they share the same prefix, the sensor ID, they would fall inside the same branch node of MPT; consequently, a validator will figure it out effortlessly using the generated proof of MPT. A more sophisticated scenario would be anchoring on two records in multiple MPTs. The attacker tries to put inconsistent data records in consecutive or far apart MPTs, but this will cause a conflict between the sensor timestamp and the anchoring timestamp of the distributed ledger, resulting in a sensor timestamp being ahead of the ledger timestamp. It is impractical for an attacker to alter the sensor's data in an attack where only the edge device is compromised because every MT root hash must be signed by the sensor. Nonetheless, the attacker can perform a DoS against the anchoring process.

IV. Prototype Implementation

The prototype implementation of the wireless sensor node's firmware is designed to be executed on the nRF52840 SoC from Nordic Semiconductor since this is the main processing and communication unit of the investigated sensor nodes. As presented in Section II, the sensor nodes measure the temperature of a device-under-test and transmit the measurement data towards a gateway. In the use case, the sampling rate and the data transmission rate are 2 Hz. The sensor data are transmitted together with the current state and other information in one so-called sensor data block.

We have evaluated the influence of the change of two different parameters: the Merkle tree (MT) size and the leaf concatenator size. The MT size is the number of leaf elements that can be included. Each leaf represents a single sensor data block if the leaf size is one. After the completion of a tree, the signing process will start, and the resulting signature will be transmitted together with the Merkle root hash. We use the SHA256 algorithm for hashing, and for signing, we apply the Elliptic Curve Digital Signature Algorithm (ECDSA) algorithm with a resulting signature length of 64 bytes. This implies that an increase in the MT size leads to an increase in covered sensor data blocks. Since the sampling rate is fixed, the interval between two consecutive signing processes will expand. Thus, the average power consumption can be reduced. By increasing the leaf size, each leaf would contain more than one sensor data block. As a result, one leaf can hold multiple sensor data blocks, reducing the number of hashing and signing processes. Therefore, multiple sensor data blocks are concatenated using a so-called leaf-accumulator, before being hashed and becoming a first-level child-node. At first glance, this method can be seen as a trivial way to reduce the amount of signing processes per time frame. However, this approach is not scalable because it requires keeping all sensor data in memory to calculate the hash and perform the signing process. It might be meaningful for small leaf sizes since it is very simple for small sensor data blocks. To get an overview of potential savings, we have implemented and compared different MT sizes with two different leaf sizes (one and two). Section V shows the results of our evaluation.

Besides the firmware for the wireless sensor node, an overview of the other functional blocks of the prototype implementation is shown in Fig. 4. All important functionalities are currently executed on the Raspberry Pi edge device. For a first evaluation and uncomplicated demonstration, this seems to be a reasonable solution. The database should be deployed in a cloud environment for an in-production system.

The wireless sensor node and the gateway service are communicating with Bluetooth Low Energy (BLE) technology. The Particle ARGON board provides connectivity to the Raspberry Pi with a BLE antenna. The measurement task can be started via the gateway service, and all data from the wireless sensor node are provided by it. The measurement data, as

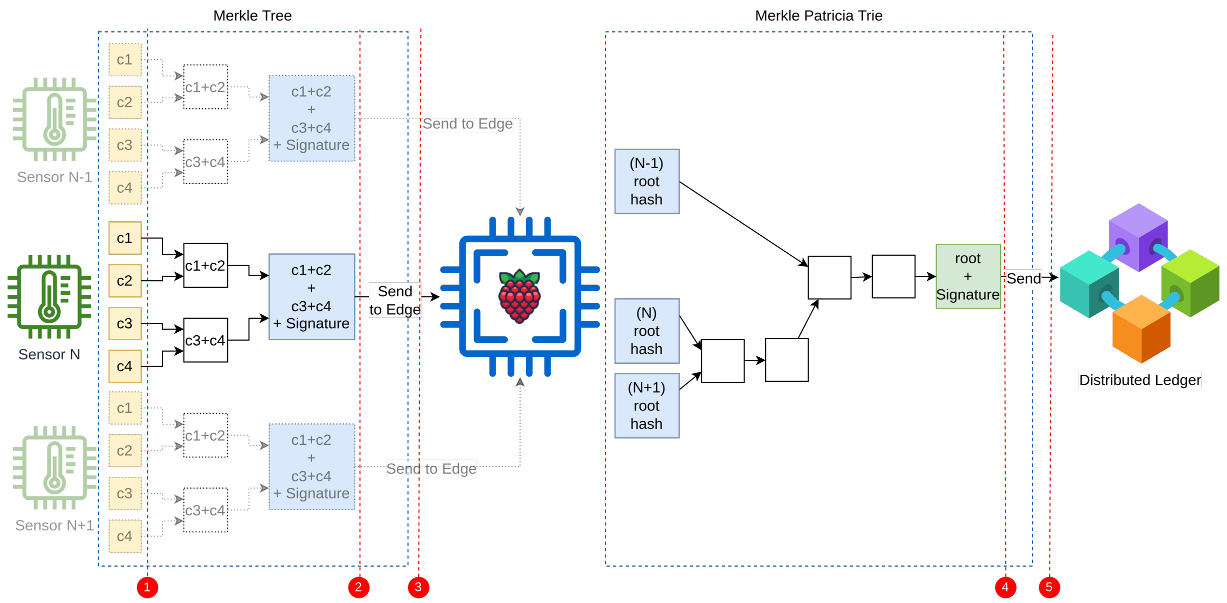

Fig. 3. Detailed depiction of anchoring process in our method

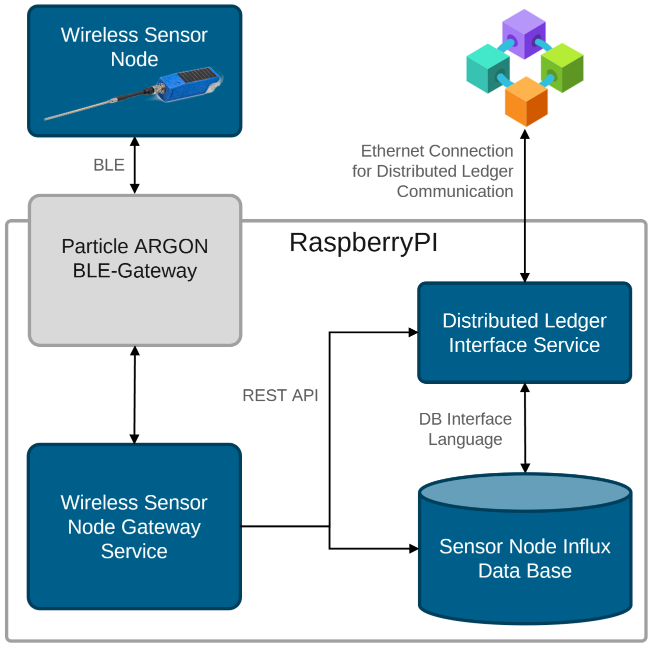
Wireless Sensor Node
BLE
Particle ARGON BLE-Gateway
RaspberryPI
Ethernet Connection for Distributed Ledger Communication
Distributed Ledger Interface Service
REST API
DB Interface Language
Wireless Sensor Node Gateway Service
Sensor Node Influx Data Base

Fig. 4. Functional blocks of the prototype implementation. Besides the functionality of the wireless sensor node and the distributed ledger, the main tasks are executed on the Raspberry PI-based edge devices.

well as the hash tree data, is provided. The data is forwarded to the distributed ledger interface service for anchoring and to the time series database for storage and verification. For the time series database, we have employed InfluxDB[1], an open-source time series database (TSDB) capable of handling IoT-generated data efficiently.

As the MT root hashes arrive at the distributed ledger interface service, they are added to an MPT as a value with the corresponding sensor identity as the key. When the tree has filled up, the MPT root hash are anchored using the Streams framework[2] of IOTA. It should be considered that our method is ledger agnostic; however, we used IOTA as it employs a data structure called Tangle, which provides a robust and feeless distributed ledger. The miners, responsible for producing new blocks in other Blockchains like Bitcoin, have been removed in IOTA from the design. As a result, transaction issuers must perform a manageable Proof-of-Work as a trade-off for paying no transaction fee to prevent network spamming. We used Streams to publish and manage the chain of anchored blocks. Furthermore, through Streams, it is possible to manage the subscription of who is able to read this chain.

In our implementation, the distributed ledger interface contains both cloud and edge device logic. This implies the edge device is responsible for anchoring data and generating proof associated with it. In a production environment, the anchoring part should remain on the edge device, while the proof generation and time series database should be migrated to the cloud. Moreover, an SQLite database interacting with the distributed ledger interface is used to store the association between the MT, MPT, and the data they contain.

V. Benchmarking

Benchmarking is based on current consumption measurements during the execution of the sensor firmware in the course of sensor data acquisition. The current consumption has been measured with the Power Profiling Kit 2 from

[1]https://www.influxdata.com/influxdb/

[2]https://github.com/iotaledger/streams

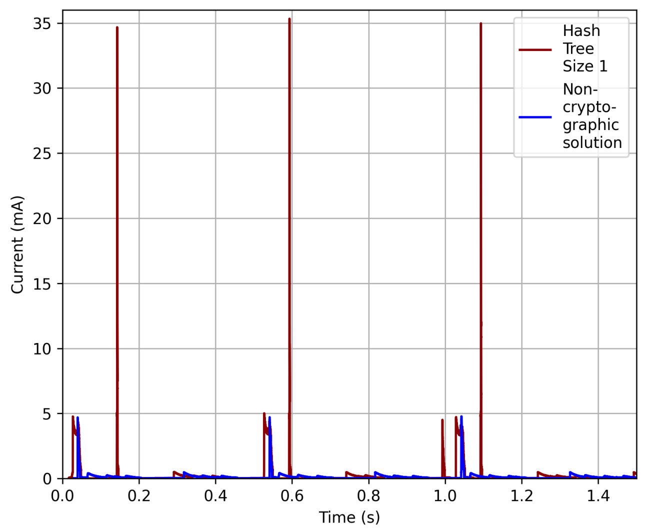

Fig. 5. Measurement of the current consumption of the wireless sensor node without any cryptographic algorithms enabled (blue) and with hashing and signing every message which represents a hash tree size of one (red).

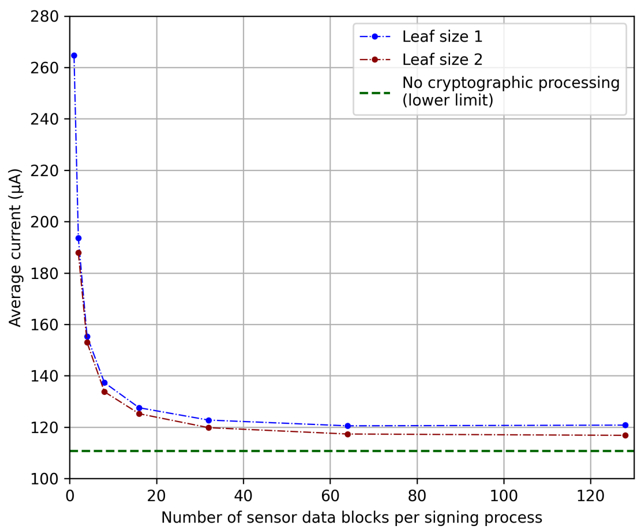

Fig. 6. Average current consumption of different scenarios with different hash tree sizes and leaf sizes.

Nordic Semiconductor. Fig. 5 shows the current consumption of the wireless sensor node in two different configurations: without and with encryption and signing of each measurement. The diagram shows three consecutive measurement and data transmission events with a time period of $0.5\,\mathrm{s}$. The first configuration is considered as the most power saving solution using no encryption and signing at all. This configuration is shown by the blue trace at the diagram. The maximum current consumption remains below of $5\,\mathrm{mA}$ and the average current consumption is $110.8\,\mu\mathrm{A}$. The second configuration is represented by the red trace. It shows the current consumption with enabled cryptographic algorithms and signing with a hash tree size of one. This means that every message is signed, and signatures are transmitted immediately for each measurement. This represents the naive implementation where every message is individually signed with the highest power consumption. It can be seen that the maximum current consumption is around $35\,\mathrm{mA}$ produced by high peaks during the signing process. The increased power consumption after the measurement is caused by the execution of the hashing of the measured data. Thus, the resulted average current consumption is $264.7\,\mu\mathrm{A}$. For resource constrained devices, this increase ($139\,\%$) is very significant and could potentially lead to implementations of more insecure solution with simple cryptographic algorithms.

The latter scenario indicates the one with the most power consumption. However, in order to demonstrate the benefits of our approach, we have to consider cases where multiple sensor values are combined in a hash tree. Therefore, we have performed additional measurements with different hash tree sizes as well as different leaf sizes, which results in a current consumption that lies between both extreme values. The results are shown in Fig. 6. A detailed presentation of the data is given in Table I. The evaluation shows the expected results of

TABLE I
DETAILED LISTING OF THE EVALUATED SCENARIOS WITH THE NUMBER OF SENSOR DATA BLOCKS PER SIGNING PROCESS AND THE RESULTING AVERAGE CURRENT CONSUMPTION.

Number of sensor data blocks per signing process	Hash tree size	Leaf size	Average current consumption	Increase compared to non-crypto
1	1	1	264,7	138,9%
2	2	1	193,6	74,7%
2	1	2	187,9	69,6%
4	4	1	155,3	40,2%
4	2	2	153	38,1%
8	8	1	137,3	23,9%
8	4	2	133,8	20,7%
16	16	1	127,5	15,1%
16	8	2	125,2	13%
32	32	1	122,7	10,8%
32	16	2	119,8	8,1%
64	64	1	120,5	8,8%
64	32	2	117,3	5,9%
128	128	1	120,8	9%
128	64	2	116,8	5,4%

a significant decrease of the average current consumption with increased hash tree size and leaf size. However, the average current consumption can not be reduced significantly any more after a certain hash tree size. The reason is the power needed to operate the sensor node without cryptographic functions. This means a good trade-off between power consumption and hash tree size should be chosen depending on the actual application. Finally, the results show the real-world applicability by only a very low increase of current consumption compared to the non-cryptographic solution considering a reasonable leaf size and hash tree size.

VI. Related Work

In this section, a couple of blockchain-based data integrity solution, which has targeted IoT systems, has been investigated. In [10], an integrated IoT solution has been proposed that provides data integrity for sensing data. This approach uses Raspberry Pi as an IoT server and Hyperledger Fabric as a Blockchain to store data; however, this method has yet to be optimized for sensor energy usage. Furthermore, since the solution uses a private blockchain, an external validator out of the consortium cannot check the data integrity. In [11], the authors proposed a method to provide data integrity under different Merkle Tree structures. Nevertheless, the approach uses a single-level accumulation of data to anchor on the Blockchain, and the Blockchain is contacted by the client directly, which is infeasible for sensors. In [18], a method to enforce data provenance and integrity has been proposed using Physical Unclonable Functions (PUFs). This method uses an Ethereum-like Blockchain to anchor the data records using a smart contract. Nonetheless, This method has not considered the limitations of IoT devices and their massive data load. The authors in [19] have proposed a blockchain-based privacy-preserving data integrity scheme for outsourced data in a semi-trusted cloud environment. In [20] the authors proposed a method to securely and fairly share Industrial IoT data. They used a fog-based design in which the computationally heavy task would be off-loaded to the fog node. However, in this method sensor data wouldn't be aggregated in the sensors, consequently increasing the energy consumption.

VII. Conclusion and Future Work

In this paper, we have presented an energy-efficient mechanism for protecting the data integrity of sensor data in a wireless sensor network. The suggested mechanism provides end-to-end security guarantees as data hashes are digitally signed by the originating sensor device. The prototypical implementation shows the practical feasibility of the approach, with an energy overhead of less than 5% that can be further minimized by increasing leaf and tree sizes depending on the concrete use case and its security requirements. This blockchain-based approach can greatly increase trust in the validity of data from wireless sensor networks because it reliably prevents forged data streams and retroactive manipulations of data records (even by insiders). This is achieved via efficient hashing and signing processes combined with the immutable history provided by a distributed ledger.

In future work, we plan to investigate additional blockchain-based security features by utilizing smart contracts. This includes a secure identity management of sensors as well as automatized access control and data marketplaces. In addition, we plan to apply our concepts to different hardware with even stricter resource constraints.

References

[1] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial internet of things: Challenges, opportunities, and directions," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 11, pp. 4724–4734, 2018.

[2] M. Stoyanova, Y. Nikoloudakis, S. Panagiotakis, E. Pallis, and E. K. Markakis, "A survey on the internet of things (iot) forensics: Challenges, approaches, and open issues," *IEEE Communications Surveys and Tutorials*, vol. 22, no. 2, pp. 1191–1221, 2020.

[3] A. Karakaya and S. Akleylek, "A survey on security threats and authentication approaches in wireless sensor networks," in *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, March 2018, pp. 1–4.

[4] S. Dey and A. Hossain, "Session-key establishment and authentication in a smart home network using public key cryptography," *IEEE Sensors Letters*, vol. 3, no. 4, pp. 1–4, 2019.

[5] L. B. Hörmann, A. Berger, A. Pötsch, P. Priller, and A. Springer, "Estimation of the harvestable power on wireless sensor nodes," in *2015 IEEE International Workshop on Measurements & Networking (M&N)*, 2015, pp. 1–6.

[6] L. B. Hörmann, P. M. Glatz, K. B. Hein, M. Steinberger, C. Steger, and R. Weiss, "Towards an On-Site Characterization of Energy Harvesting Devices for Wireless Sensor Networks," in *IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops 2012)*, 2012, pp. 415–418.

[7] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Conference on the theory and application of cryptographic techniques*. Springer, 1987, pp. 369–378.

[8] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[9] H. Shafagh, L. Burkhalter, S. Ratnasamy, and A. Hithnawi, "Droplet: Decentralized authorization and access control for encrypted data streams," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 2469–2486.

[10] L. Hang and D.-H. Kim, "Design and implementation of an integrated iot blockchain platform for sensing data integrity," *sensors*, vol. 19, no. 10, p. 2228, 2019.

[11] D. Yue, R. Li, Y. Zhang, W. Tian, and C. Peng, "Blockchain based data integrity verification in p2p cloud storage," in *2018 IEEE 24th international conference on parallel and distributed systems (ICPADS)*. IEEE, 2018, pp. 561–568.

[12] H.-P. Bernhard, A. Springer, A. Berger, and P. Priller, "Life cycle of wireless sensor nodes in industrial environments," in *13th IEEE Int. Workshop Factory Commun. Sys.*, Trondheim, Norway, May 2017.

[13] A. Berger, T. Hölzl, L. B. Hörmann, H.-P. Bernhard, A. Springer, and P. Priller, "An Environmentally Powered Wireless Sensor Node for High Precision Temperature Measurements," in *2017 IEEE Sensors Applications Symposium (SAS)*. IEEE, 2017, pp. 1–6.

[14] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum project yellow paper, 2014, https://ethereum.github.io/yellowpaper/paper.pdf.

[15] S. Müller, A. Penzkofer, N. Polyanskii, J. Theis, W. Sanders, and H. Moog, "Tangle 2.0 leaderless nakamoto consensus on the heaviest dag," *IEEE Access*, vol. 10, pp. 105 807–105 842, 2022.

[16] I. Ozcelik, S. Medury, J. Broaddus, and A. Skjellum, "An overview of cryptographic accumulators," *arXiv preprint arXiv:2103.04330*, 2021.

[17] A. Lackner, S. A. M. Mirhosseini, and S. Craß, "Securing file system integrity and version history via directory merkle trees and blockchains," in *International Conference on Database and Expert Systems Applications*. Springer, 2022, pp. 294–304.

[18] U. Javaid, M. N. Aman, and B. Sikdar, "Blockpro: Blockchain based data provenance and integrity for secure iot environments," in *Proceedings of the 1st Workshop on Blockchain-enabled Networked Sensor Systems*, 2018, pp. 13–18.

[19] Q. Zhao, S. Chen, Z. Liu, T. Baker, and Y. Zhang, "Blockchain-based privacy-preserving remote data integrity checking scheme for iot information systems," *Information Processing & Management*, vol. 57, no. 6, p. 102355, 2020.

[20] J. Sengupta, S. Ruj, and S. D. Bit, "Fairshare: Blockchain enabled fair, accountable and secure data sharing for industrial iot," *IEEE Transactions on Network and Service Management*, 2023.