Black-Box IoT: Authentication and Distributed Storage of IoT Data from Constrained Sensors

Panagiotis Chatzigiannis
George Mason University
pchatzig@gmu.edu

Foteini Baldimtsi
George Mason University
foteini@gmu.edu

Constantinos Kolias
University of Idaho
kolias@uidaho.edu

Angelos Stavrou
Virginia Tech
angelos@vt.edu

ABSTRACT

We propose Black-Box IoT (BBox-IoT), a new ultra-lightweight black-box system for authenticating and storing IoT data. BBox-IoT is tailored for deployment on IoT devices (including low-Size Weight and Power sensors) which are *extremely constrained* in terms of computation, storage, and power. By utilizing core Blockchain principles, we ensure that the collected data is immutable and tamper-proof while preserving data provenance and non-repudiation. To realize BBox-IoT, we designed and implemented a novel chain-based hash signature scheme which only requires hashing operations and removes all synchronicity dependencies between signer and verifier. Our approach enables low-SWaP devices to authenticate removing reliance on clock synchronization. Our evaluation results show that BBox-IoT is practical in Industrial Internet of Things (IIoT) environments: even devices equipped with 16MHz micro-controllers and 2KB memory can broadcast their collected data without requiring heavy cryptographic operations or synchronicity assumptions. Finally, when compared to industry standard ECDSA, our approach is two and three orders of magnitude faster for signing and verification operations respectively. Thus, we are able to increase the total number of signing operations by more than 5000% for the same amount of power.

1 INTRODUCTION

The commercial success of low Size Weight and Power (SWaP) sensors and IoT devices has given rise to new sensor-centric applications transcending traditional industrial and closed-loop systems [24, 67]. In their most recent Annual Internet Report [2], CISCO estimates that there will be 30 billion networked devices by 2023, which is more than three times the global population. While very different in terms of their hardware and software implementations, Industrial IoT (IIoT) systems share common functional requirements: they are designed to collect data from a large number of low-SWaP sensor nodes deployed at the edge. These nodes, which we refer to as edge *sensors*, are resource-constrained devices used in volume to achieve a broader sensing coverage while maintaining low cost. Thus, while capable of performing simple operations, low-SWaP sensors usually depend on battery power, are equipped with limited storage, and have low processing speed [19].

In practice, edge sensors are usually controlled by and report to more powerful gateway devices (which we refer to as *aggregators*) that process and aggregate the raw sensory data. For instance, in an Industrial (IIoT) environment, sensors are devices such as temperature sensors are broadcasting their measurements to the network router, which in turn submits it to the cloud through the Internet. Until recently, due to processing and storage constraints, many IoT designs were geared towards direct to cloud aggregation and data processing. However, latency, bandwidth, autonomy and data privacy requirements for IoT applications keep pushing the aggregation and processing of data towards the edge [43]. In addition, in most use cases, IoT devices need to be mutually *authenticated* to maintain system integrity and the data origin has to be verified to prevent data pollution attacks [45, 56] and in “model poisoning” where an attacker has compromised a number of nodes acting co-operatively, aiming to reduce the accuracy or even inject backdoors to the resulting analysis models [13, 31].

The use of distributed, immutable ledgers has been proposed as a prominent solution in the IoT setting allowing rapid detection of inconsistencies in sensory data and network communications, providing a conflict resolution mechanism without relying on a trusted authority [10]. A number of relevant schemes has been proposed in the literature [51, 54], integrating distributed ledgers (commonly referred to as *Blockchain*) with IoT.

The Challenge: One of the main roadblocks for using Blockchain-based systems as “decentralized” databases for sharing and storing collected data is their dependency on asymmetric authentication techniques. Typically, to produce authenticated data packets, sensors have to digitally sign the data by performing public key cryptographic operations, which are associated with expensive sign and verification computations and large bandwidth requirements. Although some high-end consumer sensor gateways and integrated sensors might be capable of performing cryptographic operations, a large number of edge sensors have limited computational power, storage and energy [16, 37]. To make matters worse, sensors try to optimize their power consumption by entering a “sleep” state to save power resulting in intermittent network connectivity and lack of synchronicity. Given such tight constraints, an important challenge is allowing low-SWaP devices being extremely constrained both in terms of computational power and memory (categorized as

---

IoTDI '21, May 18–21, 2021, Charlottesvle, VA, USA

ACM ISBN 978-1-4503-8354-7/21/05…$15.00
https://doi.org/10.1145/3450268.3453536

Class 0 in RFC 7228 [14] ref. Section 5.1), to authenticate and utilize a blockchain-based data sharing infrastructure.

Our Contributions: We design and implement BBox-IoT, a complete blockchain-based system for Industrial IoT devices aimed to create a decentralized, immutable ledger of sensing data and operations while addressing the sensor and data authentication challenge for extremely constrained devices. We aim to use our system as a "black-box" that empowers operators of an IIoT enclave to audit sensing data and operational information such as IIoT communications across all IIoT devices.

To perform sensor and data authentication operations *without* relying on heavy cryptographic primitives, we introduce a novel hash-based digital signature that uses an onetime hash chain of signing keys. While our design is inspired by TESLA broadcast authentication protocol [49, 50], our approach *does not* require any timing and synchronicity assumptions between signer and verifier. Overcoming the synchronicity requirement is critical for low-SWaP devices since their internal clocks often drift out of synchronization (especially those using low cost computing parts) [27, 57]. Our proposed scheme further benefits by the broadcast nature of the wireless communication. Indeed, in combination with the immutable blockchain ledger, we are able to ferret out man-in-the-middle attacks in all scenarios where we have more than one aggregators in the vicinity of the sensors. To bootstrap the authentication of sensor keys, we assume an operator-initiated device bootstrap protocol that can include either physical contact or wireless pairing using an operator-verified ephemeral code between sensors and their receiving aggregators. Our bootstrap assumptions are natural in the IoT setting, where sensors often “report" to specific aggregators and allows us to overcome the requirement for a centralized PKI. Note that our signature scheme is of independent interest, in-line with recent efforts by NIST for lightweight cryptography [58].

For the blockchain implementation where a *consensus* protocol is needed, we consider a *permissioned* setting, where a trusted party authorizes system participation at the aggregator level. Our system supports two main types of IoT devices: low-SWaP sensors who just broadcast data and self-reliant aggregators who collect the data and serve as gateways between sensors and the blockchain. While our system is initialized by a trusted operator, the operator is not always assumed present for data sharing and is only required for high-level administrative operations including adding or removing sensors from the enclave. We build the consensus algorithms for BBox-IoT using a modified version of Hyperledger Fabric [7], a well known permissioned blockchain framework, and leverage blockchain properties for constructing our protocols tailored for constrained-device authentication. However, BBox-IoT operations are designed to be lightweight and do not use public key cryptography based on the RSA or discrete logarithm assumptions, which are common, basic building blocks of popular blockchain implementations. We describe our system in details considering interactions between all participants and argue about its security.

We implemented and tested a BBox-IoT prototype in an IIoT setting comprising of extremely constrained sensors (Class 0 per RFC 7228). We employed 8-bit sensor nodes with 16MHz micro controllers and 2KB RAM, broadcast data every 10 seconds to a subset of aggregators (e.g. IIoT gateways) which in turn submit aggregated data to a cloud infrastructure. The evaluation shows that the IIoT sensors can compute our 64-byte signature in 50ms, making our signature scheme practical even for the least capable of IIoT platforms. Our evaluation section shows results by considering a sensor/gateway ratio of 10:1. When compared with ECDSA signing operations, our scheme is significantly more efficient offering two (2) and three (3) orders of magnitude speedup for signing and verification respectively. Our theoretical analysis and implementation shows that we can achieve strong chained signatures with half signature size, which permits accommodating more operations in the same blockchain environment. BBox-IoT is also over 50 times more energy-efficient, which makes our system ideal for edge cost-efficient but energy-constrained IIoT devices and applications.

Finally, we adopt the same evaluation for Hyperledger Fabric considered in previous work [7] and estimate the end-to-end costs of BBox-IoT when running on top of our Hyperledger modification, showing it is deployable in our considered use-cases.

2 BACKGROUND & PRELIMINARIES

2.1 Blockchain System Consensus

In distributed ledgers (or Blockchains), we can categorize the participants as follows: a) Blockchain maintainers (called also *miners*), who are collectively responsible for continuously appending valid data to the ledger, and b) clients, who are reading the blockchain and posting proposals for new data. While clients are only utilizing the blockchain in a read-only mode, the blockchain maintainers who are responsible for “book-keeping" must always act according to a majority's agreement to prevent faulty (offline) or Byzantine (malicious) behavior from affecting its normal functionality. This assumes that a *consensus* protocol takes place behind the scenes among these maintainers, which are distinguished to permissioned or permissionless, according to their participation controls.

Although the open nature of permissionless blockchains seems attractive, it does not really fit the membership and access control requirements for IoT deployments. In such settings, operators prefer to control the participation of IoT sensors and aggregators by means of authenticating them. Moreover, in permissioned settings consensus is computationally cheaper and thus better suited to nodes with limited capabilities.

Fundamental Consensus Properties: Informally, the ledger consensus problem [29] considers a number of parties receiving a common sequence of messages, appending their outputs on a public ledger. The two basic properties of a ledger consensus protocol are: (a) *Consistency*: An honest node's view of the ledger on some round j is a prefix of an honest node's view of the ledger on some round $j+\ell, \ell > 0$. (b) *Liveness*: An honest party on input of value x, after a certain number of rounds outputs a ledger view that includes x.

BBox-IoT Permissioned Consensus: The aforementioned fundamental properties are not sufficient for our system consensus. For instance, most “classical" consensus algorithms such as PBFT [20] have not been widely deployed due to various practical issues including lack of scalability. Taking the BBox-IoT requirements into account, the system's consensus algorithm needs to satisfy the following additional properties:

i. Dynamic membership: In BBox-IoT, there is no a priori knowledge of system participants. New members might want to join (or leave) after bootstrapping the system. We highlight that the vast majority of permissioned consensus protocols assume a static membership . Decoupling "transaction signing participants" from "consensus participants" is a paradigm that circumvents this limitation .

ii. Scalable: BBox-IoT might be deployed in wide-area scenarios (e.g. IIoT), so the whole system must support in practice many thousands of participants, and process many operations per second (more than 1000 op/s).

iii. DoS resistant: For the same reason above, participants involved in consensus should be resilient to denial-of-service attacks .

2.2 Modifying Hyperledger Fabric

Hyperledger [35], a well-known open-source blockchain platform in the permissioned model, satisfies a wide range of business blockchain requirements, and has developed several frameworks, supporting different consensus algorithms or even "pluggable" (rather than hardcoded) consensus like Hyperledger Fabric [7]. Its main components are categorized as follows:

(1) **Clients** are responsible for creating a transaction and submitting it to the peers for signing. After collecting a sufficient number of signatures (as defined by the system policy), they submit their transaction to the orderers for including it in a block. Client authentication is delegated to the application.
(2) **Peers** are the blockchain maintainers, and are also responsible for endorsing clients' transactions. Notice that in the context of Hyperledger, "Endorsing" corresponds to the process of applying message authentication.
(3) **Orderers** after receiving signed transactions from the clients, establish *consensus* on total order of a collected transaction set, deliver blocks to the peers, and ensure the consistency and liveness properties of the system.
(4) The **Membership Service Provider** (MSP) is responsible for granting participation privileges in the system.

Directly using Hyperledger in BBox-IoT is not possible, since we assume that lightweight devices (which for Hyperledger Fabric would have the role of "clients") are limited to only broadcasting data without being capable of receiving and processing. In Hyperledger Fabric, clients need to collect signed transactions and send them to the ordering service, which is an operation that lightweight devices are typically not capable of performing.

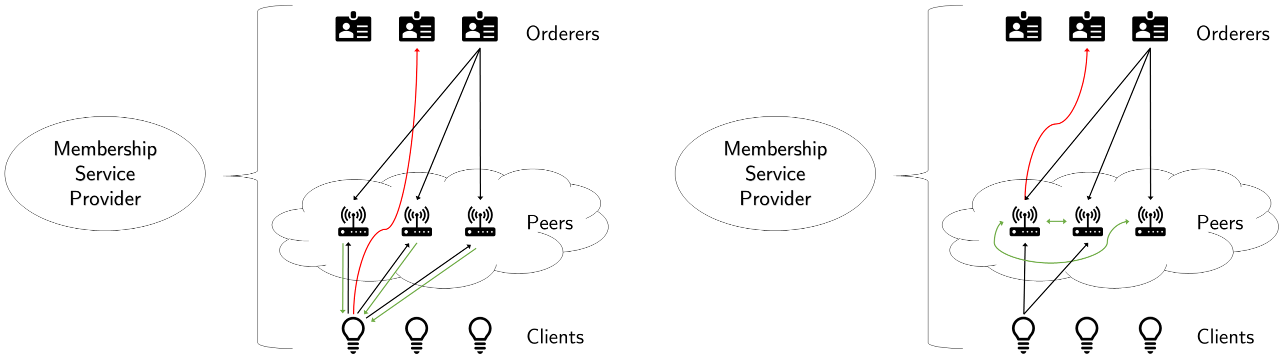
Our modification. To address this issue, we propose a modification in Hyperledger architecture. In our modified version, as shown in Figure 1b, a client broadcasts its "transaction" message to all nearby peer nodes. However, the transaction is handled by a *specific* peer (which are equivalent to an aggregator as we discuss in the next section), while peers not "responsible" for that transaction disregard it. That specific peer then assumes the role of the "client" in the original Hyperledger architecture simultaneously, while also continuing functioning as a peer node. As a client, it would be responsible for forwarding this transaction to other peers, and collecting the respective signatures, as dictated by the specified system policy, in a similar fashion to original Hyperledger Fabric. It would then forward the signed transaction to the ordering service, and wait for it to be included in a block. The ordering service would send the newly constructed block to all peers, which would then append it to the blockchain.

Security of our modifications. The proposed modifications of Hyperledger do not affect the established security properties (i.e. Consistency and Liveness), since a peer node simultaneously acting as a client, can only affect the signing process by including a self-signature in addition to other peers' signatures. However, because the signing requirements are dynamically dictated by the system policy, these could be easily changed to require additional signatures or even disallow self-signatures to prevent any degradation in security. We also note that while this modified version of Hyperledger effectively becomes agnostic to the original client, which otherwise has no guarantees that its broadcasted transaction will be processed honestly, our threat model discussed in the next section captures the above trust model.

3 BBOX-IOT SYSTEM PROPERTIES

In BBox-IoT there are five main types of participants, most of them inherited by Hyperledger Fabric: the MSP, orderers, local administrators, aggregators and sensors. Aggregators are equivalent to *peers* and sensors to *clients* in our modified Hyperledger Fabric architecture discussed in the previous section. We provide a high level description of each participant's role in the system and include detailed definitions in the full version of our paper [22].

- The MSP is a trusted entity who grants or revokes authorization for orderers, local administrators and aggregators to participate in the system, based on their credentials. It also initializes the blockchain and the system parameters and manages the system configuration and policy.
- Orderers (denoted by O) receive signed transactions from aggregators. After verifying the transactions as dictated by the system policy they package them into blocks. An orderer who has formed a block invokes the consensus algorithm which runs among the set of orderers O. On successful completion, it is transmitted back to the aggregators with the appropriate signatures.
- Local administrators (denoted by LAdm, are lower-level system managers with delegated authority from the MSP. Each LAdm is responsible for creating and managing a local device group G, which includes one or more aggregators and sensors. He grants authorization for aggregators to participate in the system with the permission of the MSP. He is also solely responsible for granting or revoking authorization for sensors in his group, using aggregators to store their credentials.
- Aggregators (denoted by Ag) are the blockchain maintainers. They receive blocks from orderers and each of them keeps a copy of the blockchain. They store the credentials of sensors belonging in their group and they pick up data broadcasted by sensors. Then they create blockchain "transactions" based on their data (after possible aggregation), and periodically collect signatures for these transactions from other aggregators in the system, as dictated by the system policy. Finally, they send signed transactions to the ordering service, and listen for new blocks to be added to the blockchain from the orderers.
- Sensors (denoted by S) are resource-constrained devices. They periodically broadcast signed data blindly without waiting for any acknowledgment. They interact with local administrators during their initialization, while their broadcasted data can potentially be received and authenticated by multiple aggregators.



(a) Original architecture. Clients collect signatures from peers for a transaction, then submit the signed transaction to the ordering service which then returns a block containing packaged transactions to peers.

(b) Modified architecture. Clients only broadcast the transaction to the peers, who are then responsible themselves for signing it before submitting it to the ordering service.

Figure 1: Modified Hyperledger Fabric architecture.

We then define the security and operational properties of BBox-IoT, in accordance with evaluation principles adopted in [15, 25, 30, 54].

3.1 Threat model & Assumptions

Physical layer attacks and assumptions. While our system cannot prevent physical tampering with sensors that might affect data correctness, any data discrepancies can be quickly detected through comparisons with adjacent sensors given the blockchain immutability guarantees. Similarly, any malicious or erroneous data manipulation by an aggregator will result in detectable discrepancies even when one of the aggregators is not compromised simultaneously. Of course, if all aggregators become compromised instantaneously, which is hard in a practical setting, our system will not detect any discrepancies. This raises the bar significantly for an adversary who might not be aware or even gain access to all aggregator nodes at the same time. Finally, attacks such as flooding/jamming and broadcast interception attacks are out of scope in this paper.

Trust Assumptions. We assume that MSP is honest during system bootstrapping only, and that device group participants (Local administrators, aggregators and sensors) may behave unreliably and deviate from protocols. For instance, they might attempt to statically or dynamically interfere with operations of honest system participants (e.g. intercept/inject own messages in the respective protocols), even colluding with each other to do so. This behavior is expected which our system is designed to detect and thwart.

Consensus Assumptions. As in Hyperledger, we decouple the security properties of our system from the consensus ones. For reference, this implies tolerance for up to 1/3 Byzantine orderer nodes, with a consensus algorithm satisfying at least the fundamental and additionally required properties discussed in Section 2.

Given the above adversarial setting, we define the following security properties:

S-1 Only authenticated participants can participate in the system. Specifically:
  a. An orderer non-authenticated by the MSP is not able to construct blocks (i.e., successfully participate in the consensus protocol). The ordering service can tolerate up to f malicious (byzantine) orderers.
  b. An LAdm non-authenticated by the MSP is not able to form a device group G.
  c. If an aggregator is not authenticated by the MSP, then its signatures on transactions cannot be accepted or signed by other aggregators.

S-2 *Sensor health:* Sensors are resilient in the following types of attacks:
  a. Cloning attacks: A non-authenticated sensor cannot impersonate an existing sensor and perform operations that will be accepted by aggregators.
  b. Message injection - MITM attack: A malicious adversary cannot inject or modify data broadcasted by sensors.

S-3 *Device group safety:* Authenticated participants in one group cannot tamper with other groups in any way, i.e.:
  a. An LAdm cannot manage another group, i.e. add or revoke participation of an aggregator or sensor in another device group, or interfere with the functionalities of existing aggregators or sensors at any time.
  b. An aggregator (or a coalition of aggregators) cannot add or remove any sensor in device group outside of their scope, or interfere with the functionalities of existing aggregators or sensors at any time.
  c. A sensor (or a coalition of sensors) cannot interfere with the functionalities of existing aggregators or other sensors at any time.

S-4 *Non-repudiation and data provenance:* Any BBox-IoT node cannot deny sent data they signed. For all data stored in BBox-IoT, the source must be identifiable.

S-5 *DoS resilient:* BBox-IoT continues to function even if MSP is offline and not available, or an adversary prevents communication up to a number of orderers (as dictated by the consensus algorithm), a number of aggregators (as dictated by the system policy) and up to all but one sensor. Also an adversary is not able to deny service to any system node (except through physical layer attacks discussed before).

S-6 *System policy and configuration security:* BBox-IoT policy and configuration can only be changed by MSP.

S-7 *Revocation:* The system is able to revoke authentication for any system participant, and a system participant can have its credentials revoked only by designated system participants.

4 CONSTRUCTIONS

We first set the notation we will be using throughout the rest of the paper. By λ we denote the security parameter. By $b \leftarrow B(a)$ we denote a probabilistic polynomial-time (PPT) algorithm B with input a and output b. By := we denote deterministic computation and by $a \to b$ we denote assignment of value a to value b. By $(\mathsf{pk}, \mathsf{sk})$ we denote a public-private key pair. We denote concatenation as $||$.

4.1 Our Hash-based Signature Scheme

Our construction is a digital signature scheme that only requires hashing as the main operation. While inspired by the Lamport passwords [40] and TESLA [49, 50], it *avoids the need for any synchronization* between senders and receivers which is a strong assumption for the IoT setting. Instead, we assume the existence of a constant-sized state for both the sender and receiver between signing operations. Our scheme allows for a fixed number of messages to be signed, and has constant communication and logarithmic computation and storage costs under the following requirements and assumptions:

- There's *no* requirement for time synchronization, and a verifier should only need to know the original signer's pk.
- The verifier should immediately be able to verify the authenticity of the signature (i.e. without a "key disclosure delay" that is required in the TESLA family protocols.
- Network outages, interruptions or "sleep" periods can be resolved by requiring computational work from the verifier, proportional to the length of the outage.
- We do not protect against Man-in-the-Middle attacks in the signature level, instead, we use the underlying blockchain to detect and mitigate such attacks as we discuss later in Section 4.3.
- The signer has very limited computation, power and storage capabilities, but can outsource a computationally-intensive precomputation phase to a powerful system.

Our scheme, presented in Construction 1, is a chain-based one-time signature scheme, with each key derived from its predecessor as $k_i \leftarrow h(k_{i+1}), i \in \{n-1, n-2, \ldots, 0\}$ and h is a preimage resistant hash function. The keys when used in pairs (k_i, k_{i-1}) can be viewed as a public-private key pair for a one-time signature scheme, then forming a one-way hash chain with consecutive applications of h. The key k_n serves as the "private seed" for the entire key chain. In the context of integrity, a signer with a "public key" $k_{i-1} = h(k_i)$ would have to use the "private key" k_i to sign his message. Since each key can only be used once, the signer would then use $k_i = h(k_{i+1})$ as his "public key" and k_{i+1} as his "private key", and continue in this fashion until the key chain is exhausted.

For example as shown in Figure 2, we can construct a hash chain from seed k_5. For signing the 1st message m_1, the signer would use $(\mathsf{pk_1}, \mathsf{sk_1}) = (k_0, k_1)$ and output signature $\sigma = h(m_1||k_0)||k_1$. Similarly, for the 2nd message he would use $(\mathsf{pk_2}, \mathsf{sk_2}) = (k_1, k_2)$ and for the 5th message $(\mathsf{pk_5}, \mathsf{sk_5}) = (k_4, k_5)$.

Let $h : \{0,1\}^* \to \{0,1\}^\lambda$ be a preimage resistant hash function.

$(\mathsf{pk}, \mathsf{sk_n}, s_0) \leftarrow \mathsf{OTKeyGen}(1^\lambda, n)$

- sample a random "private seed" $k_n \leftarrow \{0,1\}^*$
- generate hash chain $\mathsf{pk} = k_0 = h(k_1) = h(h(k_2)) = \ldots = h^i(k_i) = h^{i+1}(k_{i+1}) = \ldots = h^{n-1}(k_{n-1}) = h^n(k_n)$
- hash chain creates n pairs of $(\mathsf{pk_i}, \mathsf{sk_i})$ where:
  $(\mathsf{pk_1}, \mathsf{sk_1}) = (k_0, k_1) = (h(k_1), k_1)$,
  $(\mathsf{pk_2}, \mathsf{sk_2}) = (k_1, k_2) = (h(k_2), k_2)$,
  ... ,
  $(\mathsf{pk_i}, \mathsf{sk_i}) = (k_{i-1}, k_i) = (h(k_i), k_i)$,
  ...,
  $(\mathsf{pk_n}, \mathsf{sk_n}) = (k_{n-1}, k_n) = (h(k_n), k_n)$
- initialize a counter $\mathsf{ctr} = 0$, store ctr and pairs as $[(\mathsf{pk_i}, \mathsf{sk_i})]_1^n$ to initial state s_0
- output $(\mathsf{pk} = \mathsf{pk_1}, \mathsf{sk_n}, s_0)$.

Note: Choosing to store only $(\mathsf{pk}, \mathsf{sk_n})$ instead of the full key lists introduces a storage-computation trade-off, which can be amortized by the "pebbling" technique we discuss in this section.

$(\sigma, \mathsf{sk_i}, s_i) \leftarrow \mathsf{OTSign}(\mathsf{sk_{i-1}}, m, s_{i-1})$

- parse s_{i-1} and read $\mathsf{ctr} \to i - 1$
- compute one-time private key $\mathsf{sk_i} = k_i$ from $n - i$ successive applications of the hash function h on the private seed k_n (or read k_i from $[\mathsf{sk}]_1^n$ if storing the whole list)
- compute $\sigma = h(m||\mathsf{pk_i})||\mathsf{sk_i} = h(m||k_{i-1})||k_i = h(m||h(k_i))||k_i$
- increment $\mathsf{ctr} \to \mathsf{ctr} + 1$, store it to updated state s_i

$\mathsf{OTVerify}(\mathsf{pk}, n, m, \sigma) := b$

- parse $\sigma = \sigma_1||\sigma_2$ to recover $\sigma_2 = k_i$
- Output $b = (\exists j < n : h^j(k_i) = \mathsf{pk}) \wedge (h(m||h(k_i)) = \sigma_1)$

Note: The verifier might choose to only store the most recent k_i which verified correctly, and replace pk with k_i above resulting in fewer hash iterations.

Construction 1: n-length Chain-based Signature Scheme

$$\begin{tikzcd} k_0 & k_1 \arrow[l, "h"'] & k_2 \arrow[l, "h"'] & k_3 \arrow[l, "h"'] & k_4 \arrow[l, "h"'] & k_5 \arrow[l, "h"'] \end{tikzcd}$$

Figure 2: Key generation for $n = 5$ and seed k_5. First signature uses as $\mathsf{pk} = k_0$ and $\mathsf{sk} = k_1$.

Constructing the one-way hash-chain described above, given the seed k_n, would require $O(n)$ hash operations to compute $k_0 = h^n(k_n)$, which might be a significant computational cost for resource-constrained devices, as the length of the hash chain n is typically large to offset the constraint of single-use keys. While we could precompute all the keys, which would cost a $O(1)$ lookup operation, we would then require $O(n)$ space, which is also a limited resource in such devices. Using efficient algorithms [36, 66], we can achieve logarithmic storage and computational costs by placing "pebbles" at positions $2^j = 1 \cdots \lceil log_2(n) \rceil$, which as shown in Section 5.3 makes our construction practical for resource-constrained devices. The verifier's cost is $O(1)$ when storing the most-recently-used k.

In the full version of the paper [22] we present formal definitions of chain-based signatures and prove unforgeability of our scheme.

Comparison and Discussion. Our scheme is directly comparable with the TESLA Broadcast Message Authentication Protocol [49, 50], which follows a similar chain-based paradigm but

Table 1: Hash-based scheme comparison.

Scheme	Architecture	NoSync	NoDelay
TESLA [49, 50]	Chain	✗	✗
μTESLA 2-level chain [44]	Chain	✗	✗
Sandwich, 1-level, light chain [32]	Chain	✗	✗
Comb Skipchain [32]	Chain	✓	✗
Short Hash-Based Signatures [23]	Chain	✓	✓
XMSS [18]	Tree	✓	✓
BPQS [21]	Chain	✓	✓
SPHINCS [11]	Tree	✓	✓
Our construction	Chain	✓	✓

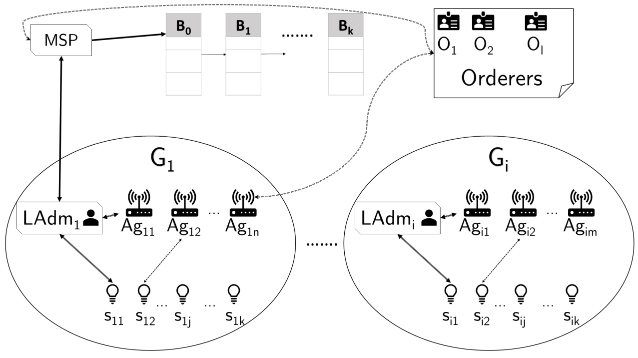


Figure 3: BBox-IoT construction overview

requires some synchronicity between the sender and receiver, and the receiver can only verify a message after some delay. Several other chain-based schemes have been proposed [23, 32, 44], forming a "hierarchy" of chains aiming to improve their efficiency in various aspects. However, most of them do not prevent the synchronicity requirement and delayed verification, in fact some even introduce additional requirements, e.g. special "commitment distribution" messages [44], where a verifier won't be able to verify a long series of signatures if those are lost. As our scheme is hash-based, we compare with another family of hash-based signatures schemes that follow a tree structure, e.g. XMSS [11] and SPHINCS [18]. While these schemes do not have any synchronicity assumptions, their performance is not suited for the low SWaP sensors we consider (even with resource-constrained device optimizations [34]). In Table 1 we compare with other hash-based schemes in terms of properties (i.e. no synchronicity or delays, denoted as NoSync and NoDelay respectively). In Table 2 we provide a concrete comparison with the rest of the schemes satisfying the above properties. In Section 6 we discuss some of the above schemes in more detail.

The caveat in our scheme is that it is susceptible to Man-in-the-Middle attacks. Specifically, an attacker might intercept a signature packet in transit (thus learning the "ephemeral" private key) and replace it with an arbitrary message and signature. Nevertheless such attacks are unlikely to be successful in our setting as discussed later in Section 4.3.

4.2 Overall BBox-IoT Construction

Our BBox-IoT system consists of the following components as shown in Figure 3 illustrating our modifications to the Hyperledger Fabric architecture.

- A (trusted) Membership Service Provider[1] MSP, which resembles a Trusted Party, and is responsible for authorizing participation in the system. The MSP bootstraps the system and forms the genesis block, which contains hardcoded information on its public key and the consensus algorithm. The genesis block also initializes the authorized system participants and the system policy (denoted by Pol), both of which can be changed later.
- A permissioned blockchain BC, which consists of normal "transaction" blocks and special "configuration" blocks.
- A configuration Config for BC, containing membership information for local administrator, orderer and aggregators, as well as system policy data. As in Hyperledger Fabric, Config is stored in the configuration blocks.
- A set of orderer nodes $\mathcal{O} : \{\mathsf{O}_1, \mathsf{O}_2, ..., \mathsf{O}_\ell\}$, responsible for achieving consensus on forming new blocks. These nodes are assumed static, although it can be extended to handle dynamic membership.
- A set of device groups $\mathcal{G} : \{\mathsf{G}_1, \mathsf{G}_2, ..., \mathsf{G}_\mathsf{n}\}$. On each group G_i there exist:
  - A local administrator LAdm_i, responsible for its group membership, which includes a set of aggregators and sensors. In order for LAdm_i to add or remove an aggregator in the system must also have consent from the MSP, however he does not need permission to handle sensor membership.
  - A set of aggregators $\mathcal{AG}_i : \{\mathsf{Ag}_{\mathsf{i}1}, \mathsf{Ag}_{\mathsf{i}2}, ..., \mathsf{Ag}_{\mathsf{im}}\}$, which have also the role of *peers* in Hyperledger Fabric. We assume aggregators can perform regular cryptographic operations and aggregate data received from sensors. As discussed in our modified Hyperledger, they also briefly take the role of a "client".
  - A set of sensors $\mathcal{S}_i : \{\mathsf{S}_{\mathsf{i}1}, \mathsf{S}_{\mathsf{i}2}, ..., \mathsf{S}_{\mathsf{ik}}\}$, which are assumed to be resource-constrained devices. These would be the equivalent of *clients* in the original Hyperledger Fabric architecture, but here they are assumed to only broadcast their data to nearby group aggregators, without expecting a confirmation. The only step where interaction occurs is during initial setup, where they exchange their public key and other initialization data with the group administrator. We also assume that sensors can only perform basic cryptographic operations (i.e. hashing), meaning they can't perform public key cryptography operations that use exponentiations.

We first describe the initialization process for the system's MSP and genesis block B_0. After generating its keys, MSP bootstraps the system with pre-populated participation whitelists of orderers, local group administrators, and aggregators (denoted by OL, LL and PL respectively) and a pre-defined system policy. Sensors do not need to be tracked from the MSP, as participation authorization for sensors is delegated to the group local administrators. Local administrators control authorization privileges with a respective sensor whitelist denoted by SL, and they also keep a whitelist of group aggregators denoted by AL.

Furthermore, we detail the functionality of reading or updating the system's configuration, including the permissioned participants and the system policy. Orderers and local administrators can only be authorized for participation by the MSP, while aggegators need their local administrator's approval as well. As discussed above,

---

[1] The MSP also includes the system administrator.

Table 2: Hash-based scheme comparison for 256-bit messages and 256-bit security parameter. Sizes in bytes. $\mathbb{M}$,$\mathbb{F}$ and $\mathbb{H}$ denote MAC, PRF and hash operations respectively. n denotes length of chain-based schemes.

Scheme	$\lvert\sigma\rvert$	$\lvert pk \rvert$	$\lvert sk \rvert$	Sign()	Verify()
Short Hash-Based Signatures [23]	$128 + log_2 n$	32	$64(\lceil log_2(n) \rceil + 1)$	$(\lceil log_2(n) \rceil + 3)\mathbb{H} + 3\mathbb{F}$	$\lceil log_2(n) \rceil$
XMSS [18]	2692 (4963)	1504 (68)	64	$747\mathbb{H} + 10315\mathbb{F}$	$83\mathbb{H} + 1072\mathbb{F}$
BPQS [21]	2176	68	64	$1073\ \mathbb{H}$	$1073\ \mathbb{H}$
SPHINCS [11]	41000	1056	1088	$386\mathbb{F}$, 385 PRGs, $167519\ \mathbb{H}$	$14060\ \mathbb{H}$
Our Construction	32(64)	32	32	$\lceil log_2(n) \rceil\ \mathbb{H}$	$1\ \mathbb{H}$

sensor participation is handled by the local administrators, however, group aggregators also keep track of group participation for sensors in a passive manner. The local administrators are also responsible for revoking participation rights for aggregators and sensors belonging in their group. In general, granting or revoking participation privileges is equivalent to adding or removing the participant's public key from the respective whitelist.

Furthermore, on a high-level, sensors "blindly" broadcast their data as signed transactions. Nearby aggregators (belonging to the same device group) receive and verify the data and collect the required amount of signatures from other aggregators in the system (as defined by the system policy), and then submit the signed transaction to the ordering service. The orderers then by running the consensus protocol, "package" the collected transactions to form a blockchain block. Finally, the block is sent back to the aggregators, who as the blockchain "maintainers", append it to the blockchain. The core system functionalities are shown in Construction 2. .

Sensor join: Defined by SensorJoin() protocol between a sensor and a Local administrator. This is the only phase when a sensor is interacting with the system, as the LAdm generates a new hash chain and its associated pebbles in a powerful device. The pebbles are then loaded to the sensor, and LAdm updates the group aggregators with the new sensor's public key.

Sensor broadcast: Defined by SensorSendData() protocol between a sensor and group aggregators. For some data m, the sensor computes the one-time hash-based signature using OTSign() and the signed data m, σ is broadcasted to all group aggregators. If there are any aggregator who receives a different signed message m', σ, the message is discarded, else it remains in the aggregator's pending memory for processing.

Aggregator transaction: Defined by AggrSendTx() protocol between aggregators and orderers. For an aggregator to submit aggregated data to the blockchain, it first needs to collect the needed signatures from other aggregators. Then it submits the signed transaction to the ordering service, which in turn executes the Consensus() algorithm to construct a block with a set of signed transactions. Finally, the block is transmitted to the aggregators, who append the block as the blockchain maintainers.

Sensor transfer: Defined by SensorTransfer algorithm, executed when a sensor is transferred to a new location or device group. The handing-over aggregator saves its state of our signature scheme w.r.t. that sensor and encrypts it on the blockchain under the receiving aggregator's public key. After sensor transfer, the receiving aggregator decrypts that state and resumes message verification.

Optionally in our construction, a symmetric group key K_G can be shared between each group's local administrator, aggregators and sensors for confidentiality purposes. However, the additional encryption operations have an impact mainly on sensors, which have constrained computational and storage resources. Note that using such key for authentication or integrity would be redundant since these properties are satisfied using public keys existing in the

SensorJoin
- Sensor generates a seed uniformly at random, and generates hash chain through OTKeyGen algorithm. (computation is outsourced to a powerful device)
- Sensor stores hash chain "pebbles" in its memory and outputs the last element of the chain as public key to the LAdm

SensorSendData
- Sensor computes signature σ for broadcasted data m using OTSign algorithm
- S_{ij} broadcasts σ to aggregators in group.
- Each aggregator after verifying the signature through OTVerify, checks if any other aggregator received a conflicting message. It adds the message - signature pair in its local state, pending for blockchain submission.

AggrSendTx
- Aggregator parses its local state for pending blockchain operations as a transaction.
- Aggregator computes signature on transaction and sends it to other aggregators.
- Each aggregator after verifying signature and sender membership in the system, signs the transaction.
- The sending aggregator submits signed transaction to ordering service after reaching necessary number of signatures, as dictated by system policy.
- Each orderer after verifying signatures, runs consensus algorithm which outputs a blockchain update operation.
- The blockchain operation is received by orderers who update the blockchain state.

SensorTransfer
- Aggregator encrypts the state for the sensor under the reveiving aggregator's pk (i.e. the most recent received sk_i) and submits it to the blockchain using AggrSendTx. Sensor is removed from the device group and is transferred to new group.
- Receiving aggregator decrypts state from the blockchain and resumes verification of received data from sensor.

Construction 2: BBox-IoT core algorithms and protocols

appropriate membership lists and revocation operations can still be performed at an equivalent cost using those lists.

4.3 Security Analysis

Given the threat model discussed in Section 3.1, most of the security properties (all but S-2 and S-5) rely on the security of the underlying signature scheme and consensus properties. As it is straightforward to prove security for these, we focus on *S-2 sensor health* security property (which includes resilience to MITM attacks) and *S-5* (resilience to DoS attacks).

In order for an adversary $\mathcal{A}$ to impersonate/clone a sensor, it would either have to break the unforgeability of our signature scheme, or launch a MITM attack which is a potential attack vector as discussed in Section 4.1.

As discussed in Section 3.1, we consider jamming attacks at the physical layer outside the scope of this paper. Given the nature of our setting where a sensor's broadcast has typically short range, we consider MITM and message injection attacks hard and unlikely to launch but we still consider them as part of our threat model. Even in these unlikely scenarios, MITM attacks can be easily mitigated in BBox-IoT. A first approach for detecting such attacks is to leverage blockchain properties, where aggregators can compare data received from a sensor in the blockchain level. Our assumption here is that sensor data can be received by more than one aggregators in the vicinity of the sensor which is a reasonable senario for typical dense IoT deployments. If there's even one dissenting aggregator, probably victim of a MITM attack, all the associated data would be considered compromised and disregarded and the operator will be notified of the data discrepancy detected. The above approach while simple, still permits a MITM attacker to "eclipse" a sensor from the system using a jamming attack.

An alternative approach is to make a proactive check in a group level, where each aggregator would verify the validity of its received data by comparing it with other aggregators before even submitting it to the blockchain. In both above strategies, the attacker's work increases significantly because he would need to launch simultaneous MITM attack between the sensor and all aggregators in the vicinity.

Additionaly, we argue that our system is DOS resilient (S-5) in the following scenarios:

- MSP offline or not available: The core system functionality is not affected, although there can be no configuration changes in the system. All algorithms and protocols (except those involving adding or revoking orderers, local administrators or aggregators or those involving system policy changes) perform authentication through the configuration blocks and not the MSP itself.
- Orderers unavailable: Reduces to tolerance properties of the consensus algorithm.
- LAdm unavailable: The core system functionality is not affected, although there can be no administrative operations in the respective group.
- Ag unavailable: Transactions are not processed only in the respective groups. However if the number of unavailable aggregators exceeds a certain threshold, no transactions can be processed in the whole system.

Table 3: Classes of Constrained Devices in terms of memory capabilities according to RFC 7228.

Name	RAM	Flash
Class 0	<<10 KiB	<<100 KiB
Class 1	10 KiB	100KiB
Class 2	50KiB	250KiB

Also, an adversary might attempt to flood an aggregator by broadcasting messages and arbitrary signatures. In this scenario, the aggregator would be overwhelmed since by running OTVerify for each message-signature pair separately, it would have to check the signature against all hash chain values up to the first public key. To mitigate this, we propose checking only for a few hashes back to the chain specified by a parameter (defined by a system parameter "maxVerifications" as shown in Algorithm 2). This parameter can be set by the local administrator but should be carefully selected. A small value might generate the need of frequent re-initializations for the sensors - if a long network outage occurs between a sensor and an aggregator and they lose "synchronization", the local administrator should reinitialize the sensor in the device group. On the other hand, a large value would amplify the impact of DoS attacks.

Algorithm 1 Sensor send data

```
tempkey ← $k_0$
initPebbles()
while True do
    m ← readSensor()
    output.type ← "payload"
    output.data ← m
    transmit(output)
    T1.start()
    hashedData ← $h$(m||tempkey)
    output.type ← "hash"
    output.data ← hashedData
    transmit(output)
    tempkey ← computePebbles() {as in [36]}
    output.type ← "secretKey"
    T1.end()
    output.data ← tempkey
    transmit(output)
end while
```

5 PERFORMANCE EVALUATION & MEASUREMENTS

5.1 The IIoT Setting With Constrained Devices

IIoT environments are complex systems comprising of heterogeneous devices that can be tracked at different organizational layers, namely (a) computational, (b) network, (c) sensor/edge layers [63]. Devices at the higher levels are powerful servers dedicated to the analysis of data, storage, and decision making. They frequently reside outside the factory premises, i.e., in cloud infrastructures. On the other hand, on-site and at the edge layer, a myriad of low-SWaP devices such as sensors and actuators reside, assigned with the tasks of posting their data or reconfiguring their status based on received instructions. On typical real-life IIoT deployments, the

Algorithm 2 Aggregator receive data

```
publickey ← $k_0$
verifications ← 0
while verifications < maxVerifications do
    check1 ← False
    check2 ← False
    read ← input()
    if read.type = “payload" then
        T3.start()
        m ← read.data
    else if read.type = “hash" then
        $s_1$ ← read.data
    else if read.type = “secretKey" then
        $s_2$ ← read.data
        tempkey ← $s_2$
        $i$ ← 0
        T2.start()
        while $i$ < maxVerification ∧ doWhile = True do
            if $h$(tempkey) = publickey then
                check1 ← True
                if $h$(m||publickey) = $s_1$ then
                    check2 ← True
                    publickey ← secretkey
                end if
                doWhile = False
            else
                tempkey ← $h$(tempkey)
                i++
            end if
        end while
        if check1 ∧ check2 = True then
            print(“Payload m is valid")
            verifications++
            T2.end()
        else
            print("Verification failed")
        end if
        T3.end()
    end if
end while
```

processing speed of such devices ranges from tens (e.g., Atmel AVR family) to hundreds of Mhz (e.g., higher-end models of ARM Cortex M series). Diving even deeper, at the lower end of the spectrum, one may observe sensor-like devices that are severely constrained in memory and processing capabilities.

Such extremely constrained devices have been considered by RFC 7228 [14] which underlines that “most likely they will not have the resources required to communicate directly with the Internet in a secure manner”. Thus, the communication of such nodes must be facilitated by stronger devices acting as gateways that reside at the network layer. In Table 3 we provide a taxonomy of constrained devices residing at the edge of IIoT according to RFC 7228.

In this work, we consider a generic IIoT application scenario that involves Class 0 devices which are connected to more powerful IoT gateways in a sensor/gateway ratio of 10:1. The chosen platforms and all experimental decisions were made to provide a realistic scenario under the following assumptions: (a) devices severely constrained in terms of computational power and memory resources (Class 0) and (b) moderately demanding in terms of communication frequency (i.e. transmission once every 10 seconds).

5.2 Evaluation Setup

Our testbed consists of Arduino UNO R3 [1] open-source microcontroller boards equipped with ATmega328P 16 MHz microcontroller and 2KB SRAM fitted with a Bluetooth HC-05 module. These devices are really constrained and they represent the minimum of capabilities in all of IoT sensors utilized in our experimental scenarios (Class 0 in Table 3). For the gateways, we use Raspberry Pi 3 Model B devices equipped with a Quad Core 1.2GHz BCM2837 64bit CPU and 1GB RAM.

We first focus on evaluating our system in a device group level[2]. We use the one-time signature scheme outlined in Construction 1 and SHA256 as the hash function $h()$. The length of the hash chain sets the upper bound on the number of one-time signatures each sensor $\mathsf{S_i}$ can generate. In the case where the sensor's available signatures are depleted, it would enter an “offline" state and the Local Administrator LAdm would need to manually renew its membership in the system through the $\mathsf{SensorJoin}$ protocol. In a large-scale deployment of our system however, frequent manual interventions are not desirable, so our goal is to pick a sufficiently large n such that the available one-time signatures to the sensor last for the sensor's lifetime. As discussed above and taking similar schemes' evaluations into account [6], we consider a frequency of one (1) signing operation per 10 seconds for simplicity. We consider sensor lifetimes between 4 months as an lower and 21 years as a upper estimate (as shown in Table 5), which imply a hash chain between 2^{20} and 2^{26} elements respectively.

In the setup phase, we pre-compute the hash-chain as needed by the pebbling algorithm [36] and load the initial pebble values into the sensor. We first measure the actual needed storage on the sensor for various values of n. Note that for $n = 2^{26}$, the lower bound for needed storage using a 256-bit hash function is about $26 \cdot 256 = 832$ bytes of memory. Then we set the sensor device to communicate with the aggregator through Bluetooth in broadcast-only mode and measure the maximum number of signing operations that can be transmitted to the aggregator for various values of n, as well as the verification time needed on the aggregator side since it will need to verify a large number of sensor messages. The fact that we are able to run BBox-IoT on Class 0 devices demonstrates the feasibility of our approach for all low-SWaP sensors.

5.3 Signing and Verification

We run our experiments under different scenarios and multiple times. Our evaluation results, which are shown in Table 5, represent the statistical average across all measurements. Note that for measuring the average signature verification time on the aggregator side, we assume that the aggregator is able to receive all the data broadcasted by the sensor. If a network outage occurs between them (and the sensor during the outage keeps transmitting), the aggregator after reestablishing connection would have to verify

---

[2] Our code is available at https://github.com/PanosChtz/Black-Box-IoT

Table 4: Evaluation for sensor-aggregator protocol - Average verification times

	T2 (μsec)				T3 (msec)			
maxV	20	22	24	26	20	22	24	26
100	28.12	31.18	31.34	28.95	42.83	42.84	42.91	43.08
500	30.78	31.94	30.31	30.63	51.25	51.23	51.37	51.39
1000	31.39	30.96	31.14	30.74	55.27	55.35	55.36	55.41
2500	30.57	30.97	32.39	30.86	60.61	60.65	60.7	60.78
5000	33.26	31.7	31.66	31.43	64.66	64.74	64.79	64.83
10000	33.34	33.38	33.6	31.41	68.68	68.75	68.78	68.86

the signature by traversing the hash chain back up to the last received secret key, which incurs additional computation time (in Figure 4 we show the associated verification cost in such occasions). As expected, the verification time is relatively constant in all measurements, about 0.031ms on average. This suggests that such an aggregator could still easily handle 10^5 sensors transmitting data for verification (as we considered one transmission every 10 seconds for each sensor).

Table 5, shows that the pebbles data stucture consumes most of the required memory storage in our implementation, while the remaining program requires a constant amount of memory for any number of pebbles. We also observe a slight impact of the number of pebbles on the total verification time, which is mainly affected by the sensor's capability to compute the signature on its message and the next secret key. For example, the sensor needs 50ms to compute the next signature with $n = 2^{26}$ and 49.95ms for $n = 2^{24}$. Also by comparing the total verification time with the signature computation time, we conclude the extra 14.3 msec are needed for transmitting the signature.

In Table 4 we provide a series of measurement results for the average verification time of 1 signature on the aggregator. By T2 we denote the verification time of a signature and by T3 the total verification time by an aggregator (as shown in Algorithm 2). The average total verification time (denoted by maxV) increases significantly as we require more verification operations from the Arduino device. This happens because of dynamic memory fragmentation as the pebbling algorithm updates the pebble values.

Comparison with ECDSA. We compare our lightweight scheme with ECDSA, which is commonly used in many blockchain applications. We assume IoT data payloads between 50 and 220 bytes, which can accommodate common data such as timestamps, attributes, source IDs and values. In Table 6 we show that our scheme is more efficient compared to ECDSA by 2 and 3 orders of magnitude for signing and verification respectively. Even when considering larger payload sizes which impact hash-based signature operations, our scheme remains much more efficient. However, verification cost for our scheme increases linearly during network outages, and as shown in Figure 4 it might become more expensive than ECDSA when more than 2400 signature packets are lost.

Another metric we consider is energy efficiency, which is of particular importance in IoT applications that involve a battery as power source. Our experiments depicted in Figure 5 show that our ATmega328P microcontroller can perform more than 50x hash-based signing operations compared to the equivalent ECDSA operations for the same amount of power. Finally, while our hash-based signature normally has a size of 64 bytes (as shown in Table 5), we can "compress" consecutive signatures along a hash chain to 32 bytes by only publishing the most recent k_i. The verifier would then generate the previous hash chain values at a minimal computational cost. This makes possible to store more authenticated data in the blockchain, as we show below.

Table 5: Evaluation for sensor-aggregator protocol (average values for 5000 verifications)

Hash Chain length n	2^{20}	2^{22}	2^{24}	2^{26}
Sensor lifetime for 1sig/10sec (m: months, y: years)	4 m	16 m	5 y	21 y
Pebble Gen time (seconds)	1.62	6.49	24.57	95.33
Verification time per signature (msec)	0.031			
Signature size (bytes)	64+ $\|m\|$			
Total dynamic memory usage (bytes)	1436	1520	1604	1678
Pebble struct memory usage (bytes)	840	924	1008	1082
Program memory usage (bytes)	596			
Signature computation time (msec)	49.82	49.88	49.95	50.00
Average total verification time per signature (msec)	64.15	64.25	64.26	64.32
Communication cost (msec)	14.3			

5.4 Consensus Performance

Considering the use-case scenario discussed in Section 5.1, we discuss the performance of our BBox-IoT system as a whole. We show that the most important metric in the system is the transaction throughput which heavily depends on the ability of the SWaP sensors to transmit data in a group setting. Of course, the scalability of the system overall is also directly proportional to the number of system active participants it can support simultaneously.

Sensors. Our measurements indicate that the aggregator - which is a relatively powerful device - is not the bottleneck in the protocol execution. Based on the measurements in Table 5, we can safely assume that a single aggregator can verify over a thousand sensors' data being continuously broadcasted, since the signature computation time by a sensor is three (3) orders of magnitude larger than the verification time by an aggregator. This is still a pessimistic estimation, since we previously assumed that a sensor broadcasts (and signs) data every 10 seconds, which implies that the aggregator can accommodate even more sensors.

Orderers. Since orderers only participate in the consensus protocol to sign blocks, we only need a few orderers such that our system remains resilient to attacks at the consensus level should a subset of orderers become compromised. Orderers can be strategically distributed over a geographical area to minimize the network latency between an aggregator and the ordering service, controlled by the main organization (which also controls the MSP). Evaluations performed in previous works have shown that by having 3 orderers, 3000 transactions/second can be easily achieved using the

	BBox-IoT		ECDSA	
Message length	Sensor Sign	Aggr Vrfy	Sensor Sign	Aggr Vrfy
50	50.43	0.0339	4200	42.55
100	53.47	0.0349		
150	56.40	0.0357		
202	59.33	0.03687		
218	60.06	0.0369		
Signature size	32		64	

Table 6: Signing and verification costs (in milliseconds) compared with message and signature sizes (in bytes). Note we assume hash-based signatures are aggregated as discussed in Section 5.3. Signer is ATmega328P microcontroller and verifier is RPi 3.

consensus protocol used in the current version of Hyperledger Fabric (with a potential of further improvement in a future adoption of BFT-SMART), and even considering up to 10 orderers in the system does not greatly affect its performance [7, 55].

Aggregators. The expected number of aggregators in the system depends on the use case as it is expected. As discussed in Section 5.1, where gateways play the role of BBox-IoT aggregators, we consider a sensor/gateway ratio of 10:1 for our evaluation purposes. To our knowledge, no evaluation of Hyperledger Fabric has ever been performed to consider such a great number of peers, which would require a great amount of resources to perform. However, by adopting the evaluation performed in [7] which measured the throughput in terms of number of peers up to 100 (which as discussed, are the aggregators in our system), we can extrapolate this evaluation to the order of thousands, which shows that with the aid of a "peer gossip" protocol, the system remains scalable if the peers are in the same approximate geographical area which implies low average network latency.

Blockchain operations. As discussed, aggregators' role is to aggregate sensor data into blockchain transactions. Assuming that aggregators perform no "lossy" operations (such as averaging techniques), they would just package many collected sensor data along with the respective signatures into a transaction which in turn would be submitted to the ordering service. If we assume as in [7] a block size of 2MB, we can estimate how much signed sensor data a block can hold. Given the discussion in Section 5.3, a Hyperledger block could hold (at most) about 15800 signed sensor data using our hash-based scheme vs. 12700 using ECDSA.

Latency. We also wish to estimate the time from a value being proposed by an aggregator until consensus has been reached on it (assuming the block contains a single transaction). Again we can adopt previous evaluations in Hyperledger Fabric [7], which show an average of 0.5 sec for the complete process. Finally, considering that the previous evaluations mentioned above were all preformed on the original Hyperledger Fabric (while our architecture requires a slight modification as discussed in Section 2.2), for our purposes we assume that the expected performance of aggregators (which are essentially Hyperledger peers also having client application functionalities) is not affected by this additional functionality, since the main affecting factor that can potentially become a bottleneck for the scalability of the whole system is network latency and not computational power.

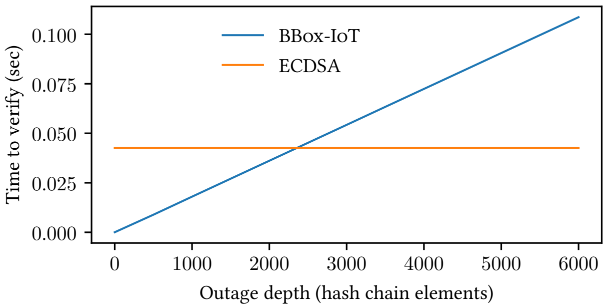


Figure 4: Aggregator verification costs in network outages. BBox-IoT is more expensive when more than about 2400 signature packets are lost.

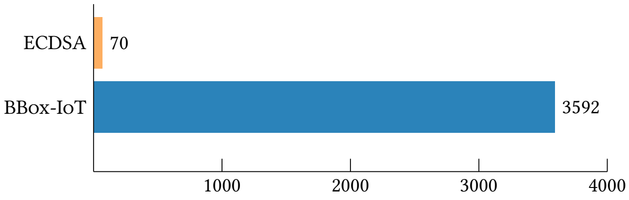


Figure 5: Number of signing operations for a 20mWh battery.

6 RELATED WORK

We now discuss a number of works that connect IoT to the blockchain setting or works which build cryptographic primitives to optimize different parts of computation for resource-constrained IoT devices. Note that none of these works addresses the problem of authentication for extremely constrained (Class 0) devices.

6.1 IoT and Blockchain

Shafagh et al. [54] presented an architecture aiming to handle IoT data in a decentralized manner while achieving confidentiality, authenticity and integrity. This proposed system defines itself as "IoT compatible" being append-only by a single writer and can be accessed by many readers, and consists of a layered design on top of an existing public blockchain to store access permissions and hash pointers for data, while storing the actual data off-chain using decentralized P2P storage techniques. Other approaches [8, 47, 59] also used a similar "layering" paradigm. While these approaches are simpler than ours, they ultimately rely heavily on the performance and properties of the underlying public blockchain and are not specifically tailored to handle resource-constrained IoT devices.

Dorri, Kanhere, and Jurdak [25] considered a "local" private blockchain maintained by a capable device, managed by the on-site owner and containing the local IoT device transactions. These lower-tier elements would be overlaid by a shared blockchain that can handle hashed data originating from the local blockchain and stored in a cloud storage service, and can enable access to local data. The above approach also offers confidentiality and integrity for submitted data and is suitable for resource-constrained IoT devices, however it is more complex than BBox-IoT and requires managing and replicating data over several points in the system.

More recently, AlTawy and Gong [5] presented a blockchain-based framework in the supply chain setting using RFIDs. This model considered blockchain smart contracts interacting with an overlay application on the RFID readers and a centralized server that handles membership credentials. This framework offers anonymity for the participating entities, which prove their membership in zero-knowledge, while their anonymity remains revocable by the server. It also provides confidentiality for its transactions and enforces a notion of "forward secrecy" which enables future product owners in the supply chain to access its entire history. BBox-IoT differs from the above work in several ways, since it is tailored to handle resource-constrained devices. Our work does not have confidentiality or anonymity as a main goal, although it can be added as an option using symmetric keys. We also do not require any smart contract functionality from the blockchain, and we operate exclusively in the permissioned setting.

IoTLogBlock [51] shares a common goal with our work: enabling the participation of low-power devices in a distributed fashion, and similarly uses Hyperledger as a "cloud service" in a IoT setting. The crucial difference with our work, is that IoTLogBlock is evaluated on a Class 2 device using ECDSA signatures, which are far more expensive than our proposed hash-based signature and could not have been supported at all by a Class 0 device, while having much larger power consumption (Fig 5). Our proposed signature scheme is a key component for efficient implementations of blockchain-based systems in the IIoT setting.

Several more approaches have been presented which augmented an IoT infrastructure with a blockchain, focusing on providing two-factor authentication [62], managing or improving communication among IoT devices [46, 53], implementing a trust management system in vehicular networks [65], providing edge computing services [64], data resiliency [42], providing secure and private energy trade in a smart-grid environment [3] and implementing a hierarchical blockhain storage for efficient industrial IoT infrastructures [60] and all of which are orthogonal to our work. We point the reader to [4, 28] for extensive reviews on the related literature.

6.2 Hash-based Signatures

Lamport's One-Time Signatures (OTS) [39] was the first scheme to allow the use of a hash function to construct a signature scheme. Then, Winternitz OTS and WOTS+ [17][33]enabled a time-memory tradeoff by signing messages in groups, used in turn by XMSS [18] in a Merkle tree construction. Other works such as HORS [52] enabled signing more than once, and more recently SPHINCS and SPHINCS+ [11, 12] enabled signing without the need to track state. Using HORS [52] as a primitive combined with a hash chain, Time Valid One-Time Signature (TV-HORS) [61] improves in signing and verification computational efficiency, but assuming "loose" time synchronization between the sender and the verifier.All of the aforementioned schemes, while only involving hash-based operations, still incur large computational and/or space costs and cannot be implemented in Class 0 resource-constrained devices we consider.

TESLA [49, 50] constructs a "one-way" hash chain to generate temporal MAC keys for specified time intervals, disclosing each chain element with some time delay Δ. While "pebbling" algorithms [36, 66] enable logarithmic storage and computational costs as discussed in Section 4.1, it requires "loose" time synchronization between the sender and the receiver for distinguishing valid keys. In an IIoT setting this would require the frequent execution of an interactive synchronization protocol since such devices are prone to clock drifting [27, 57]. Several modifications and upgrades to TESLA have been proposed, but most of them still require time synchronization [32, 44].

6.3 Cryptographic Operations in IoT

In the context of improving cryptographic operations in the IoT setting, Ozmen and Yavuz [48] focused on optimizing public key cryptography for resource-constrained devices. This work exploited techniques in Elliptic Curve scalar multiplication optimized for such devices and presented practical evaluations of their scheme on a low-end device. Even though the device used in this work is can be classified as a Class 1 or Class 2 device, our construction signing is more efficient both in terms of computation cost and storage by at least an order of magnitude.

Hülsing, Rijneveld and Schwabe [34] showed a practical evaluation of the SPHINCS hash-based signature scheme [11] on a Class 2 device. At first glance this implementation could also serve our purposes, however our proposed construction, while stateful, is much cheaper in terms of runtime, storage and communication costs, without such additional assumptions.

Kumar et al. [38] propose an integrated confidentiality and integrity solution for large-scale IoT systems, which relies on an identity-based encryption scheme that can distribute keys in a hierarchical manner. This solution also uses similar techniques to our work for signature optimization for resource-constrained devices, however, it requires synchronicity between the system participants. Portunes [41] is tailored for preserving privacy (which is not within our main goals in our setting), and requires multiple rounds of communication (while we consider a "broadcast-only" setting)

Finally we mention an extensive IoT authentication survey [26]. In this work, our authentication scheme is comparable to [9] which utilizes hashing for one-way authentication in a distributed architecture, however our scheme is more storage-efficient, suited for low-SWaP (Class 0) sensors.

7 CONCLUSIONS

In this paper we designed and implemented BBox-IoT, a blockchain inspired approach for Industrial IoT sensors aiming at offering a transparent and immutable system for sensing and control information exchanged between IIoT sensors and aggregators. Our approach guarantees blockchain-derived properties to even low-Size Weight and Power (SWaP) devices. Moreover, BBox-IoT acts as a "black-box" that empowers the operators of any IoT system to detect data and sensor tampering ferreting out attacks against even SWaP devices. We posit that enabling data auditing and security at the lowest sensing level will be highly beneficial to critical infrastructure environments with sensors from multiple vendors.

Finally, we envision that our approach will be implemented during the sensor manufacturing stage: having industrial sensors shipped with pre-computed pebbles and their key material labeled using QR-code on the sensor body will allow for a seamless and practical deployment of BBox-IoT.

REFERENCES

[1] 2019. Arduino Uno Rev3. https://store.arduino.cc/usa/arduino-uno-rev3

[2] 2020. Cisco Annual Internet Report . https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[3] N. Z. Aitzhan and D. Svetinovic. 2018. Security and Privacy in Decentralized Energy Trading Through Multi-Signatures, Blockchain and Anonymous Messaging Streams. *IEEE Transactions on Dependable and Secure Computing* 15, 5 (Sep. 2018), 840–852. https://doi.org/10.1109/TDSC.2016.2616861

[4] M. Ali, M. Vecchio, M.Pincheira, K. Dolui, F. Antonelli, and M. Rehmani. 2019. Applications of Blockchains in the Internet of Things: A Comprehensive Survey. *IEEE Communications Surveys and Tutorials* 21, 2 (2019), 1676–1717. https://doi.org/10.1109/COMST.2018.2886932

[5] Riham AlTawy and Guang Gong. 2019. Mesh: A Supply Chain Solution with Locally Private Blockchain Transactions. *PoPETs* 2019, 3 (2019), 149–169. https://doi.org/10.2478/popets-2019-0041

[6] Dorian Amiet, Andreas Curiger, and Paul Zbinden. 2018. FPGA-based Accelerator for SPHINCS-256. *IACR TCHES* 2018, 1 (2018), 18–39. https://doi.org/10.13154/tches.v2018.i1.18-39 https://tches.iacr.org/index.php/TCHES/article/view/831.

[7] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolic, S. Weed Cocco, and J. Yellick. 2018. Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. *CoRR* abs/1801.10228 (2018). http://arxiv.org/abs/1801.10228

[8] L. Bai, M. Hu, M. Liu, and J. Wang. 2019. BPIIoT: A Light-Weighted Blockchain-Based Platform for Industrial IoT. *IEEE Access* 7 (2019), 58381–58393. https://doi.org/10.1109/ACCESS.2019.2914223

[9] Omaimah Omar Bamasag and Kamal Youcef-Toumi. 2015. Towards Continuous Authentication in Internet of Things Based on Secret Sharing Scheme. In *Proceedings of the 10th Workshop on Embedded Systems Security, WESS 2015, Amsterdam, The Netherlands, October 8, 2015*, Stavros A. Koubias and Thilo Sauter (Eds.). ACM, 1. https://doi.org/10.1145/2818362.2818363

[10] Jonathan Bell, Thomas D. LaToza, Foteini Baldimtsi, and Angelos Stavrou. 2017. Advancing Open Science with Version Control and Blockchains. In *12th IEEE/ACM International Workshop on Software Engineering for Science, SE4Science@ICSE 2017, Buenos Aires, Argentina, May 22, 2017.* IEEE, 13–14. https://doi.org/10.1109/SE4Science.2017.11

[11] Daniel J. Bernstein, Daira Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Michael Schneider, Peter Schwabe, and Zooko Wilcox-O'Hearn. 2015. SPHINCS: Practical Stateless Hash-Based Signatures. In *EUROCRYPT 2015, Part I (LNCS, Vol. 9056)*, Elisabeth Oswald and Marc Fischlin (Eds.). Springer, Heidelberg, 368–397. https://doi.org/10.1007/978-3-662-46800-5_15

[12] Daniel J. Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. 2019. The SPHINCS$^+$ Signature Framework. In *ACM CCS 2019*, Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz (Eds.). ACM Press, 2129–2146. https://doi.org/10.1145/3319535.3363229

[13] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin B. Calo. 2019. Analyzing Federated Learning through an Adversarial Lens. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 634–643. http://proceedings.mlr.press/v97/bhagoji19a.html

[14] C. Bormann, M. Ersue, and A. Keranen. 2014. *Terminology for Constrained-Node Networks.* RFC 7228. RFC Editor.

[15] Mic Bowman and Camille Morhardt. 2018. Blockchain Must Adapt to Build Trust in the Internet of Things. https://www.coindesk.com/blockchain-must-adapt-build-trust-internet-things/ Retrieved June 24, 2018.

[16] Kyle Boyer, Laura Brubaker, Kyle Everly, RIchard Herriman, Paul Houston, Sean Ruckle, Rory Scobie, and Ian Ulanday. 2017. A distributed sensor network for an off-road racing vehicle. International Foundation for Telemetering.

[17] Johannes Buchmann, Erik Dahmen, Sarah Ereth, Andreas Hülsing, and Markus Rückert. 2011. On the Security of the Winternitz One-Time Signature Scheme. Cryptology ePrint Archive, Report 2011/191. http://eprint.iacr.org/2011/191.

[18] Johannes A. Buchmann, Erik Dahmen, and Andreas Hülsing. 2011. XMSS - A Practical Forward Secure Signature Scheme Based on Minimal Security Assumptions. In *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011*, Bo-Yin Yang (Ed.). Springer, Heidelberg, 117–129. https://doi.org/10.1007/978-3-642-25405-5_8

[19] Joseph Bugeja, Paul Davidsson, and Andreas Jacobsson. 2018. Functional Classification and Quantitative Analysis of Smart Connected Home Devices. In *2018 Global Internet of Things Summit, GIoTS 2018, Bilbao, Spain, June 4-7, 2018.* IEEE, 1–6. https://doi.org/10.1109/GIOTS.2018.8534563

[20] Miguel Castro and Barbara Liskov. 1999. Practical Byzantine Fault Tolerance. In *OSDI '99* (New Orleans, Louisiana, USA). USENIX Association, Berkeley, CA, USA. http://dl.acm.org/citation.cfm?id=296806.296824

[21] Konstantinos Chalkias, James Brown, Mike Hearn, Tommy Lillehagen, Igor Nitto, and Thomas Schroeter. 2018. Blockchained Post-Quantum Signatures. Cryptology ePrint Archive, Report 2018/658. https://eprint.iacr.org/2018/658.

[22] Panagiotis Chatzigiannis, Foteini Baldimtsi, Constantinos Kolias, and Angelos Stavrou. 2021. Black-Box IoT: Authentication and Distributed Storage of IoT Data from Constrained Sensors (full version). https://cs.gmu.edu/~pchatzig/BBoxIoT.pdf

[23] Erik Dahmen and Christoph Krauß. 2009. Short Hash-Based Signatures for Wireless Sensor Networks. In *CANS 09 (LNCS, Vol. 5888)*, Juan A. Garay, Atsuko Miyaji, and Akira Otsuka (Eds.). Springer, Heidelberg, 463–476.

[24] Hasan Derhamy, Jens Eliasson, Jerker Delsing, and Peter Priller. 2015. A survey of commercial frameworks for the internet of things. In *ETFA 2015*. IEEE.

[25] Ali Dorri, Salil S. Kanhere, and Raja Jurdak. 2016. Blockchain in internet of things: Challenges and Solutions. *CoRR* abs/1608.05187 (2016). arXiv:1608.05187 http://arxiv.org/abs/1608.05187

[26] Mohammed El-hajj, Ahmad Fadlallah, Maroun Chamoun, and Ahmed Serhrouchni. 2019. A Survey of Internet of Things (IoT) Authentication Schemes. *Sensors* 19, 5 (2019), 1141. https://doi.org/10.3390/s19051141

[27] A. Elsts, X. Fafoutis, S. Duquennoy, G. Oikonomou, R. J. Piechocki, and I. Craddock. 2018. Temperature-Resilient Time Synchronization for the Internet of Things. *IEEE Trans. Industrial Informatics* 14, 5 (2018), 2241–2250. https://doi.org/10.1109/TII.2017.2778746

[28] Mohamed Amine Ferrag, Makhlouf Derdour, Mithun Mukherjee, Abdelouahid Derhab, Leandros A. Maglaras, and Helge Janicke. 2019. Blockchain Technologies for the Internet of Things: Research Issues and Challenges. *IEEE Internet Things J.* 6, 2 (2019), 2188–2204. https://doi.org/10.1109/JIOT.2018.2882794

[29] Juan A. Garay and Aggelos Kiayias. 2020. SoK: A Consensus Taxonomy in the Blockchain Era. In *CT-RSA 2020 (LNCS, Vol. 12006)*, Stanislaw Jarecki (Ed.). Springer, Heidelberg, 284–318. https://doi.org/10.1007/978-3-030-40186-3_13

[30] O. Garcia-Morchon, R. Rietman, S. Sharma, L. Tolhuizen, and J.L. Torre-Arce. 2015. A Comprehensive and Lightweight Security Architecture to Secure the IoT Throughout the Lifecycle of a Device Based on HIMMO. In *ALGOSENSORS 2015* (Patras, Greece). Springer-Verlag, Berlin, Heidelberg, 112–128. https://doi.org/10.1007/978-3-319-28472-9_9

[31] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2019. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access* 7 (2019), 47230–47244.

[32] Yih-Chun Hu, Markus Jakobsson, and Adrian Perrig. 2005. Efficient Constructions for One-Way Hash Chains. In *ACNS 05 (LNCS, Vol. 3531)*, John Ioannidis, Angelos Keromytis, and Moti Yung (Eds.). Springer, Heidelberg, 423–441. https://doi.org/10.1007/11496137_29

[33] Andreas Hülsing. 2013. W-OTS+ - Shorter Signatures for Hash-Based Signature Schemes. In *AFRICACRYPT 13 (LNCS, Vol. 7918)*, Amr Youssef, Abderrahmane Nitaj, and Aboul Ella Hassanien (Eds.). Springer, Heidelberg, 173–188. https://doi.org/10.1007/978-3-642-38553-7_10

[34] Andreas Hülsing, Joost Rijneveld, and Peter Schwabe. 2016. ARMed SPHINCS - Computing a 41 KB Signature in 16 KB of RAM. In *PKC 2016, Part I (LNCS, Vol. 9614)*, Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang (Eds.). Springer, Heidelberg, 446–470. https://doi.org/10.1007/978-3-662-49384-7_17

[35] Hyperledger. 2018. Hyperledger Architecture Volumes 1 and 2. https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf

[36] Markus Jakobsson. 2002. Fractal hash sequence representation and traversal. In *Information Theory, 2002.* IEEE, 437.

[37] Fatma Karray, Mohamed Wassim Jmal, Alberto García Ortiz, Mohamed Abid, and Abdulfattah Mohammad Obeid. 2018. A comprehensive survey on wireless sensor node hardware platforms. *Comput. Networks* 144, 89–110. https://doi.org/10.1016/j.comnet.2018.05.010

[38] Sam Kumar, Yuncong Hu, Michael P. Andersen, Raluca Ada Popa, and David E. Culler. 2019. JEDI: Many-to-Many End-to-End Encryption and Key Delegation for IoT. *CoRR* abs/1905.13369 (2019). arXiv:1905.13369 http://arxiv.org/abs/1905.13369

[39] Leslie Lamport. 1979. *Constructing Digital Signatures from a One-way Function.* Technical Report SRI-CSL-98. SRI International Computer Science Laboratory.

[40] Leslie Lamport. 1981. Password Authentication with Insecure Communication. *Commun. ACM* 24, 11 (Nov. 1981), 770–772. https://doi.org/10.1145/358790.358797

[41] Hongyang Li, György Dán, and Klara Nahrstedt. 2014. Portunes: Privacy-preserving fast authentication for dynamic electric vehicle charging. In *2014 IEEE International Conference on Smart Grid Communications, SmartGridComm 2014, Venice, Italy, November 3-6, 2014.* IEEE, 920–925. https://doi.org/10.1109/SmartGridComm.2014.7007766

[42] X. Liang, J. Zhao, S. Shetty, and D. Li. 2017. Towards data assurance and resilience in IoT using blockchain. In *MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM).* 261–266. https://doi.org/10.1109/MILCOM.2017.8170858

[43] Li Lin, Xiaofei Liao, Hai Jin, and Peng Li. 2019. Computation offloading toward edge computing. *Proc. IEEE* 107, 8 (2019), 1584–1607.

[44] Donggang Liu and Peng Ning. 2003. Efficient Distribution of Key Chain Commitments for Broadcast Authentication in Distributed Sensor Networks. In *NDSS 2003*. The Internet Society.

[45] Jing Liu, Yang Xiao, and Jingcheng Gao. 2014. Achieving Accountability in Smart Grid. *IEEE Systems Journal* 8, 2 (2014), 493–508. https://doi.org/10.1109/JSYST.2013.2260697

[46] D. Miller. 2018. Blockchain and the Internet of Things in the Industrial Sector. *IT Professional* 20, 3 (May 2018), 15–18. https://doi.org/10.1109/MITP.2018.032501742

[47] O. Novo. 2018. Blockchain Meets IoT: An Architecture for Scalable Access Management in IoT. *IEEE Internet of Things Journal* 5, 2 (April 2018), 1184–1195. https://doi.org/10.1109/JIOT.2018.2812239

[48] Muslum Ozgur Ozmen and Attila A. Yavuz. 2017. Low-Cost Standard Public Key Cryptography Services for Wireless IoT Systems. In *Proceedings of the 2017 Workshop on Internet of Things Security and Privacy, IoT S&P@CCS,*, Peng Liu, Yuqing Zhang, Theophilus Benson, and Srikanth Sundaresan (Eds.). ACM, 65–70. https://doi.org/10.1145/3139937.3139940

[49] A. Perrig, R. Canetti, J. D. Tygar, and D. Song. 2000. Efficient authentication and signing of multicast streams over lossy channels. In *IEEE S & P 2000*. 56–73. https://doi.org/10.1109/SECPRI.2000.848446

[50] Adrian Perrig, Ran Canetti, J. D. Tygar, and Dawn Song. 2002. The TESLA Broadcast Authentication Protocol.

[51] Christos Profentzas, Magnus Almgren, and Olaf Landsiedel. 2019. IoTLogBlock: Recording Off-line Transactions of Low-Power IoT Devices Using a Blockchain. In *44th IEEE Conference on Local Computer Networks, LCN 2019, Osnabrueck, Germany, October 14-17, 2019*, Karl Andersson, Hwee-Pink Tan, and Sharief Oteafy (Eds.). IEEE, 414–421. https://doi.org/10.1109/LCN44214.2019.8990728

[52] Leonid Reyzin and Natan Reyzin. 2002. Better than BiBa: Short One-Time Signatures with Fast Signing and Verifying. In *ACISP 02 (LNCS, Vol. 2384)*, Lynn Margaret Batten and Jennifer Seberry (Eds.). Springer, Heidelberg, 144–153. https://doi.org/10.1007/3-540-45450-0_11

[53] M. Samaniego and R. Deters. 2017. Internet of Smart Things - IoST: Using Blockchain and CLIPS to Make Things Autonomous. In *IEEE ICCC 2017*. 9–16. https://doi.org/10.1109/IEEE.ICCC.2017.9

[54] Hossein Shafagh, Lukas Burkhalter, Anwar Hithnawi, and Simon Duquennoy. 2017. Towards Blockchain-based Auditable Storage and Sharing of IoT Data. In *Proceedings of the 2017 on Cloud Computing Security Workshop* (Dallas, Texas, USA) *(CCSW '17)*. ACM, New York, NY, USA, 45–50. https://doi.org/10.1145/3140649.3140656

[55] J. Sousa, Al. Bessani, and M. Vukolic. 2018. A Byzantine Fault-Tolerant Ordering Service for the Hyperledger Fabric Blockchain Platform. In *DSN 2018*. IEEE Computer Society, 51–58. https://doi.org/10.1109/DSN.2018.00018

[56] P. Tenti, H. K. M. Paredes, and P. Mattavelli. 2011. Conservative Power Theory, a Framework to Approach Control and Accountability Issues in Smart Microgrids. *IEEE Transactions on Power Electronics* 26, 3 (March 2011), 664–673. https://doi.org/10.1109/TPEL.2010.2093153

[57] Francisco Tirado-Andrés, Alba Rozas, and Álvaro Araujo. 2019. A Methodology for Choosing Time Synchronization Strategies for Wireless IoT Networks. *Sensors* 19, 16, 3476. https://doi.org/10.3390/s19163476

[58] Meltem Sönmez Turan, Kerry A McKay, Çağdaş Çalık, Donghoon Chang, and Larry Bassham. 2019. Status report on the first round of the NIST lightweight cryptography standardization process. (2019).

[59] J. Wan, J. Li, M. Imran, and D. Li. 2019. A Blockchain-Based Solution for Enhancing Security and Privacy in Smart Factory. *IEEE Transactions on Industrial Informatics* (June 2019). https://doi.org/10.1109/TII.2019.2894573

[60] Gang Wang, Zhijie Shi, Mark Nixon, and Song Han. 2019. ChainSplitter: Towards Blockchain-Based Industrial IoT Architecture for Supporting Hierarchical Storage. In *IEEE International Conference on Blockchain, 2019*. IEEE, 166–175. https://doi.org/10.1109/Blockchain.2019.00030

[61] Qiyan Wang, Himanshu Khurana, Ying Huang, and Klara Nahrstedt. 2009. Time Valid One-Time Signature for Time-Critical Multicast Data Authentication. In *INFOCOM 2009. 28th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 19-25 April 2009, Rio de Janeiro, Brazil*. IEEE, 1233–1241. https://doi.org/10.1109/INFCOM.2009.5062037

[62] L. Wu, X. Du, W. Wang, and B. Lin. 2018. An Out-of-band Authentication Scheme for Internet of Things Using Blockchain Technology. In *2018 International Conference on Computing, Networking and Communications (ICNC)*. 769–773. https://doi.org/10.1109/ICCNC.2018.8390280

[63] Yulei Wu, Hong-Ning Dai, and Hao Wang. 2020. Convergence of Blockchain and Edge Computing for Secure and Scalable IIoT Critical Infrastructures in Industry 4.0. *IEEE Internet of Things Journal* (2020).

[64] Z. Xiong, Y. Zhang, D. Niyato, P. Wang, and Z. Han. 2018. When Mobile Blockchain Meets Edge Computing. *IEEE Communications Magazine* 56, 8 (August 2018), 33–39. https://doi.org/10.1109/MCOM.2018.1701095

[65] Z. Yang, K. Yang, L. Lei, K. Zheng, and V. C. M. Leung. 2019. Blockchain-Based Decentralized Trust Management in Vehicular Networks. *IEEE Internet of Things Journal* 6, 2 (April 2019), 1495–1505. https://doi.org/10.1109/JIOT.2018.2836144

[66] Dae Hyun Yum, Jae Woo Seo, Sungwook Eom, and Pil Joong Lee. 2009. Single-Layer Fractal Hash Chain Traversal with Almost Optimal Complexity. In *CT-RSA 2009 (LNCS, Vol. 5473)*, Marc Fischlin (Ed.). Springer, Heidelberg, 325–339. https://doi.org/10.1007/978-3-642-00862-7_22

[67] Han Zou, Yuxun Zhou, Jianfei Yang, and Costas J Spanos. 2018. Towards occupant activity driven smart buildings via WiFi-enabled IoT devices and deep learning. *Energy and Buildings* 177 (2018), 12–22.

ACKNOWLEDGMENTS

Foteini Baldimtsi and Panagiotis Chatzigiannis were supported by NSA 204761.