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

•••

Presented at ACM/IEEE International Conference on Internet of Things Design and Implementation (IoTDI)

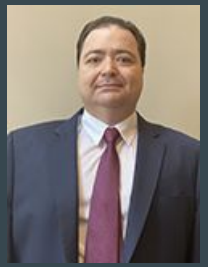
Authors

Panagiotis Chatzigiannis: PhD student George Mason University

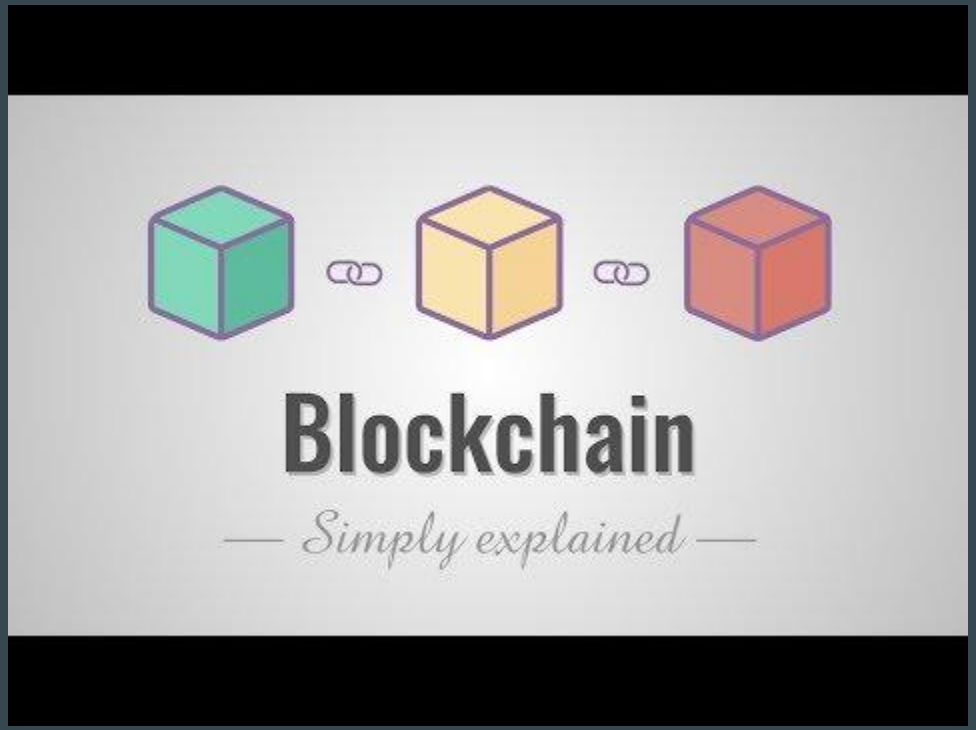
Foteini Baldimtsi: Assistant Professor George Mason University

Constantinos Kolias: Assistant Professor University of Idaho

Angelos Stavrou: Professor Virginia Tech



What is Blockchain?



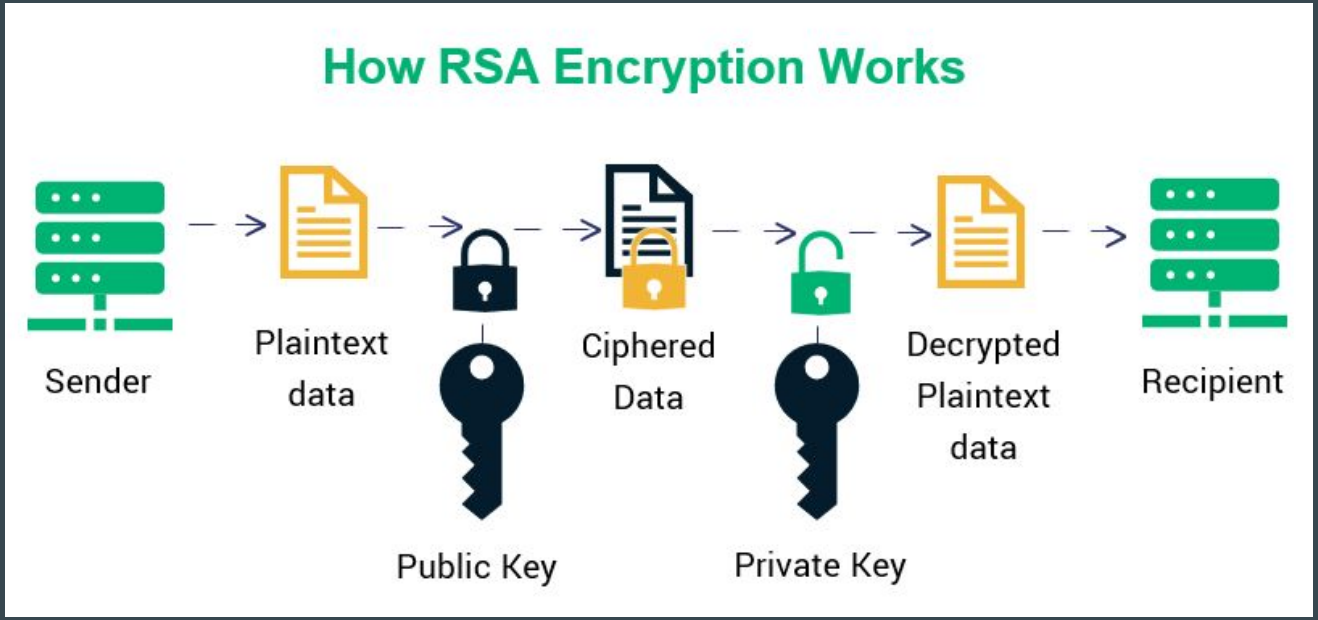
What was the problem?

IoT applications are moving more data processing to the edge. So...?

- Data poisoning
- Injecting backdoors

What does industry currently do?

“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.”



RSA vs ECDSA Encryption

ECDSA is elliptic curve digital signature algorithm also known as ECC or elliptic curve cryptography.

Both asymmetric encryption algorithms.

Security (In Bits)	RSA Key Length Required (In Bits)	ECC Key Length Required (In Bits)
80	1024	160-223
112	2048	224-255
128	3072	256-383
192	7680	384-511
256	15360	512+

RSA	ECDSA
One of the earliest methods of public-key cryptography, standardized in 1995.	Comparatively new public-key cryptography method compared to RSA, standardized in 2005.
Today, it's the most widely used asymmetric encryption algorithm.	Compared to RSA, ECDSA is a less adopted encryption algorithm.
It works on the principle of the Prime Factorization method.	It works on the mathematical representation of Elliptical Curves.
RSA is a simple asymmetric encryption algorithm, thanks to the prime factorization method.	The complexity of elliptical curves makes ECDSA a more complex method compared to RSA.
RSA is a simpler method to implement than ECDSA.	Implementing ECDSA is more complicated than RSA.
RSA requires longer keys to provide a safe level of encryption protection.	Compared to RSA, ECDSA requires much shorter keys to provide the same level of security
As it requires longer keys, RSA slows down the performance.	Thanks to its shorter key lengths, ECDSA offers much better performance compared to RSA.

The TESLA Broadcast Authentication Protocol

ABSTRACT:

One of the main challenges of securing broadcast communication is source authentication, or enabling receivers of broadcast data to verify that the received data really originates from the claimed source and was not modified en route. This problem is complicated by mutually un- trusted receivers and unreliable communication environments where the sender does not retransmit lost packets.

This article presents the TESLA (Timed Efficient Stream Loss-tolerant Authentication) broadcast authentication protocol, an efficient protocol with low communication and computation overhead, which scales to large numbers of receivers, and tolerates packet loss. TESLA is based on loose time synchronization between the sender and the receivers.

Despite using purely symmetric cryptographic functions (MAC functions), TESLA achieves asymmetric properties. We discuss a PKI application based purely on TESLA, assuming that all network nodes are loosely time synchronized.

Their Solution

- Does not require any timing and synchronicity assumptions between signer and verifier
- 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
- Evaluation section shows results by considering a sensor/gateway ratio of 10:1

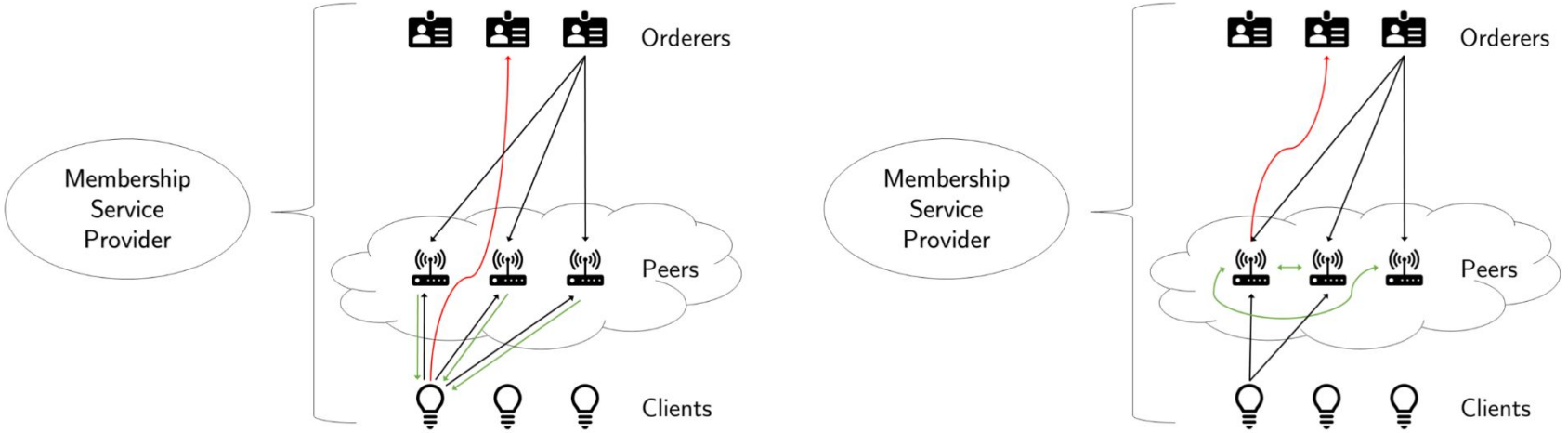
Consensus Algorithm

1. Dynamic membership: In BBox-IoT, there is no a prior 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 .
2. 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).
3. DoS resistant: For the same reason above, participants involved in consensus should be resilient to denial-of-service attacks .

Hyperledger Fabric

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.

Hyperledger Fabric Visual



(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.

BBOX-IOT SYSTEM PROPERTIES

- 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.

BBOX-IOT SYSTEM PROPERTIES

- 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.

BBOX-IOT SYSTEM PROPERTIES

- 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.

BBOX-IOT SYSTEM PROPERTIES

- 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.

BBOX-IOT SYSTEM PROPERTIES

- 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.

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

$$k_0 \xleftarrow{h} k_1 \xleftarrow{h} k_2 \xleftarrow{h} k_3 \xleftarrow{h} k_4 \xleftarrow{h} k_5$$

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

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	✓	✓

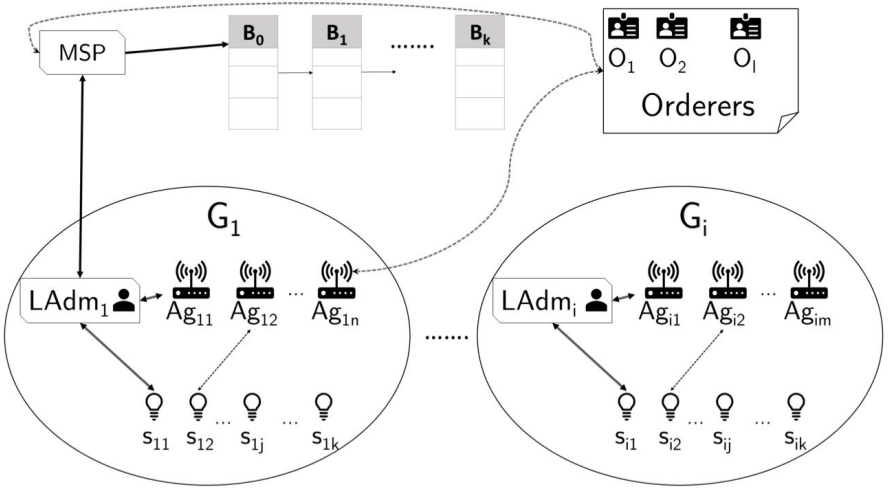


Figure 3: BBox-IoT construction overview

Results

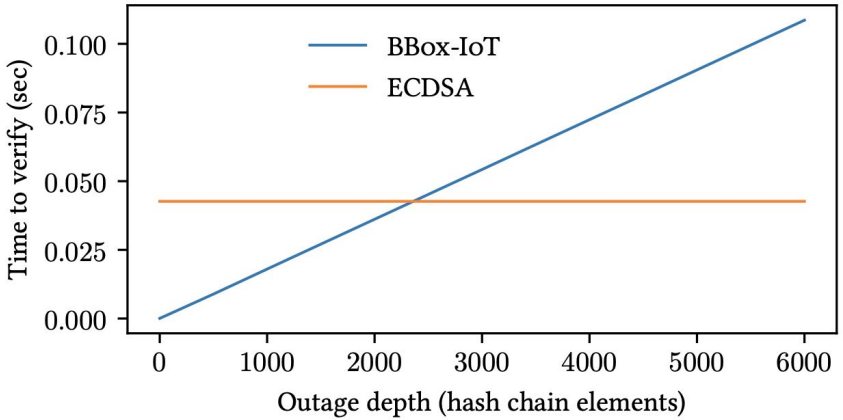


Figure 4: Aggregator verification costs in network outages. BBox-IoT is more expensive when more than about 2400 signature packets are lost.

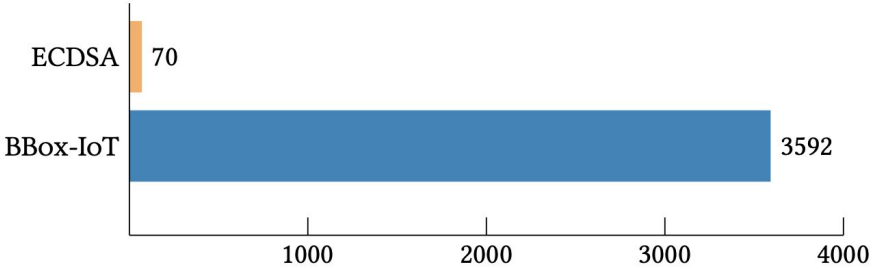


Figure 5: Number of signing operations for a 20mWh battery.

Sources

https://sectigostore.com/blog/ecdsa-vs-rsa-everything-you-need-to-know/

How does a blockchain work - Simply Explained

Panagiotis Chatzigiannis - GMU CS Department - George ...https://cs.gmu.edu › ~pchatzig

TwitterFoteini Baldimtsi (@FBaldimtsi) | Twitter

Constantinos (Costas) Kolias, Ph.D. - University of Idahohttps://www.uidaho.edu › engr › our-people › faculty

https://ece.vt.edu/people/profile/angelos

The TESLA Broadcast Authentication Protocol - People ...https://people.eecs.berkeley.edu › ~tygar › papers

https://www-users.cselabs.umn.edu/classes/Fall-2021/csci8980-ec/papers/BlackBoxAuth.pdf