2018 IEEE Confs on Internet of Things, Green Computing and Communications, Cyber, Physical and Social Computing, Smart Data, Blockchain, Computer and Information Technology, Congress on Cybermatics

Ensuring Proof-of-Authenticity of IoT Edge Devices using Blockchain Technology

Ujjwal Guin*, Pinchen Cui†, Anthony Skjellum◇
*Dept. of Electrical and Computer Engineering, Auburn University
†Dept. of Computer Science and Software Engineering, Auburn University
◇SimCenter & Dept. of Computer Science and Engineering, University of Tennessee at Chattanooga

***Abstract*—Imposter devices pose serious threats. The majority of low-cost edge devices can easily be counterfeited or cloned; the supply chain is insufficiently secure. Reliability of deployed devices can be called into question simply because they might be counterfeit or cloned. It is a must to identify edge devices' sourcing uniquely and verify their validity periodically at runtime.**

We integrate blockchain technology to authenticate resource-constrained, low-cost edge devices. We use SRAM-based physically unclonable functions (PUFs) to generate unique "digital fingerprints" (device IDs). Registered manufacturers upload a cryptographic hash of each device ID to a "globally accessible" blockchain instance (key-value store or smart contract). While registering/designating a device locally, the end-user verifies whether the hash is present in that blockchain. We utilize a "locally permissioned" blockchain infrastructure (which is still a globally managed blockchain or, in future, a sidechain) to authenticate edge devices for a defense-in-depth approach. Devices can authenticated periodically to prevent device cloning. Target environments can be large and have varied trust among users and lack a specific perimeter; this "local" blockchain methodology is thus pertinent, especially since blockchains gain security over time. Our approach reduces the potential for classes of information leakage and types of sabotage in a critical infrastructure or large-scale deployment (such as a smart city) arising from imposter devices. This methodology protects against such imposters in mobile settings within an IoT infrastructure too.

***Index Terms*—Internet of Things (IoT), Cyber-Physical Systems (CPS), Physically Unclonable Functions (PUF), Edge Device, Cloning, Blockchains, Lightweight Mining, Device Identity**

I. Introduction

The Internet of Things (IoT) is the collection of billions upon billions of devices ("things") connected to the Internet and purposed to enable direct interactions between the physical world as well as computer-based systems. It is estimated that there will between 20 and 50 billion such Internet-connected devices by 2020 [1]–[3]. 'Things" are a wide variety of electronic and electromechanical devices including smart thermostats, lights, watches, mobile phones, sensors, and actuators, as well as microcontrollers, among others [1]. Ensuring security and authenticity of these devices is challenging since they often have resource constraints such as low power requirements, low area budget, limited memory, and/or extremely low-cost. Some lack MAC addresses in their wireless protocols as well. Trappe et al. showed that the power constraint in IoT edge devices limits encryption functionality of sensor nodes, which leads to poorly encrypted communication or no encryption at all [4]. In a different study, HP revealed that almost 70% of IoT devices did not encrypt communications to the Internet or the local network [5]. These deficits open vulnerabilities for adversaries. Furthermore, encryption, if present, is no guarantee of identity.

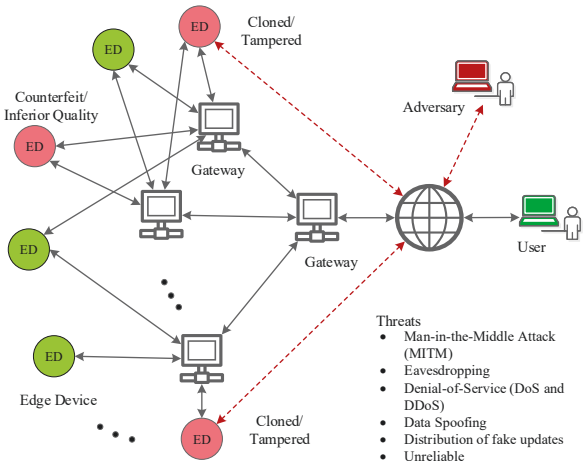

Figure 1: A standard IoT model with hardware vulnerabilities.

This paper focuses on ensuring the security of CPS/IoT systems through an information architecture that incorporates a globally managed blockchain dataset for integrity, availability, and identity purposes. Most edge devices are manufactured in limited-trust environments lacking relevant government regulations (e.g., to curtail counterfeiting and infiltration of threats at manufacture), move through supply chains without strong controls, and then are deployed in critical infrastructures worldwide. Thus, we need to develop solutions for protecting both hardware and software that take into account the manifold variety of attacks and threats inherent in such commodity-off-the-shelf devices. Attacks can originate from untrusted hardware of a CPS/IoT System and/or from the Internet by way of exploiting existing communication protocols and network traffic. Hardware attacks against a system can occur with physical tampering of a device and/or by the introduction of a cloned/counterfeit device [6]–[9] into the system. Software attacks against the system can be performed through network attacks such as Phishing, Denial of Service (DoS), and data spoofing [10], [11]. Our approach disallows classes of impersonation (cloning) and counterfeiting, which mitigate man-in-the-middle attacks against secure connections through unique identity.

Figure 1 shows a CPS/IoT System with various attacks that can originate from untrusted hardware. An adversary might create a backdoor and easily bypass existing security measures implemented in the software/firmware. An adversary could launch a wide variety of attacks (e.g., Man-in-the-Middle Attack (MITM), Eavesdropping, Denial-of-Service (DoS and DDoS), Data Spoofing, Distribution of fake updates, etc.) from the Internet or the internal network of an organization by exploiting

978-1-5386-7975-3/18/$31.00 ©2018 IEEE
DOI 10.1109/Cybermatics_2018.2018.00193

cloned, tampered, and/or compromised devices.

We propose to use Distributed Ledger Technology (blockchain technology) to identify each edge device uniquely without the need for end-users to contact the original device manufacturer for provenance information (such as identity). Blockchains are modern distributed data structures designed originally for cryptocurrencies that achieve strong global consensus (via *mining*). Notable are Bitcoin [12] and Ethereum [13], [14], though there are many others. Key features include: block structure, lack of centralized control, and a consensus (mining) algorithm. For cryptocurrencies, this is typically a hard task, Proof-of-Work (PoW) or Proof-of-Stake (PoS) [15]. These computational proofs force miners to perform a significant task; the one finalizing the task first gets rewarded, and mines a new block. That block is then shared throughout the ledgers (the copies of the blocks in each miner's server). Because of the sequential, cryptographic hashing in a given block of the previous block, it is computationally infeasible to rewrite history. Stability is maintained in case of simultaneous mining; for instance, a longest-chain methodology is used in Bitcoin if two miners should both produce a valid block within a short time interval. Certain Blockchains support smart contracts [13], [14]; a smart contract is a script run across the Blockchain as a side-effect of mining that allows calculations suitable for creating escrow-like operations, key-value data sets, and many emerging and complex financial instruments. Some systems allow Turing-complete scripts.

One of the authors of this paper and others have introduced Scrybe [16], [17], a lightweight mining alternative to blockchains used for cryptocurrencies. As described below, there is no expensive computational task such as PoW/PoS in the Scrybe blockchain; for secure provenance, it is unneeded and wasteful of resources (we have no inflationary pressure in non-cryptocurrency applications, which is a principle driver for PoW/PoS).

The authenticity of a low-cost edge device can be ensured by verifying an unclonable device ID, which can be generated from an on-board SRAM memory (SRAM PUF [18], [19]) to avoid the cost of programmable non-volatile memory in low-cost edge devices. Recently, physically unclonable functions (PUFs) have received significant attention in the hardware security community because they can generate unique, unclonable bits for the identification and authentication of ICs. PUFs use inherently uncontrollable and unpredictable variations from a manufacturing process to produce random, unclonable bits. Several PUF architectures have been proposed over the years that include the arbiter PUF [20], ring oscillator PUF [21], and SRAM PUF [18], [19], among others. Since IoT edge devices have SRAM-based memory and embedded processors, SRAM PUFs offer a better solution to produce device IDs with no additional cost or complexity as compared to other options. In our identity approach, registered manufacturers of a given class of edge devices upload a cryptographic hash of the ID in the designated "globally accessible" Blockchain infrastructure for such provenance information. While registering a device in the IoT infrastructure, the end-user in turn needs to verify whether the hash is present in the "global infrastructure" Blockchain. We have also proposed to implement a "locally permissioned" Blockchain infrastructure for authenticating edge devices on a regular basis in their deployed environments primarily to prevent the threat of replacing an edge device with a cloned/tampered counterpart, or to allow duplicates to appear in different sectors of a large, mobile environment. In practice, only one Blockchain need hold both the "globally accessible" information and "locally permissioned" data per site.

The contributions of this paper are as follows:

1) *Architecture of Global Identity Blockchain Infrastructure (B_G):* We create a Global Identity Blockchain infrastructure in which manufacturers of edge devices register their devices. Once a device is registered, anyone can access its identity from anywhere (the Blockchain architecture supports both integrity and availability). It is unnecessary to contact the original device manufacturer for device authentication. Manufacturers must upload a cryptographic hash of the ID to prevent it from being cloned. During the registration of a device in the IoT infrastructure, it is necessary to verify whether the hash of device ID is present in the Global Identity Blockchain. This can completely eliminate the need for tracking manufacturers.

2) *Architecture of a Local Identity Blockchain Infrastructure (B_L):* We also designate a Locally Permissioned Blockchain infrastructure, which supports secure authentication for the local IoT system. A local administrator can register an authentic edge device into their data set stored in this Blockchain. Once devices are registered, the local system can authenticate them via this Local Identity Blockchain data set. However, before the Local Registration, the key management of an edge device should precede in which a secret key is generated and burned into the on-chip, one-time programmable memory. Once the key is programmed, direct access of device ID is prohibited to protect it from being copied, as many of these edge devices can be deployed in a hostile environment. Note that the local administrator is responsible for uploading both the cryptographic hash of the ID and the encrypted secret key into the Local Identity Blockchain data set.

3) *Secure Communication Protocol:* We also propose a secure communication protocol that can be invoked at a regular interval to authenticate all the edge devices in a CPS/IoT system. Note that the security of the architecture is ensured so long as the secret key of the given IoT device is safe.

4) *Scrybe – A Blockchain-Based Provenance Scheme:* The Scrybe blockchain can optionally be used to create the Global and/or Local Identity Blockchain data sets. This saves effort, power, and money at the global level by removing the need to use relatively expensive blockchain store for the identity information associated with new IoT devices. Proofs of resilience to attacks are given below and elsewhere. For using of Scrybe in the global data structure, a hypothetical trade association (management group) that maintains the service can use Ethereum smart contracts to front end this service specifically to manage the financial stability of the service (that is, charge registered vendors each time they wish to add new device identities). (That would represent a third, economic blockchain use in the architecture.)

For the local Blockchain data set, the same Scrybe blockchain described above can also be used. Scrybe is an efficient option because there is no need for a PoW/PoS blockchain to maintain strong consensus over the local identities. Without "hacking" a cryptocurrency blockchain (such as reducing its PoW effort), Scrybe can be used reliably here to deliver the

required blockchain B_L while maintaining security guarantee*.

The remainder of this paper is organized as follows. Section II describes our proposed authentication scheme, which can be adapted into both the global supply chain and to the local IoT infrastructure. We present the implementation of the proposed scheme in Section III. We perform a security evaluation in Section IV and we conclude the paper in Section V.

II. Proposed Authentication Scheme using Blockchain Technology

Bitcoin [12], the first Blockchain-based decentralized system proposed by Satoshi Nakamoto in 2008, was mainly used for digital currency transaction instead of data storage. Although users are able to use the `OP_RETURN` field to store up to 40 bytes of a chain within one transaction, the idea and performance of storing data into the Bitcoin Blockchain is still unrealistic [22]. Now, 10 years after 2008, a number of Blockchain-based platforms and services raised in this past decade and many of them can provide better on-chain data storage, such as Ethereum, Storj, Filecoin, Maidsafe, and DADI [23]–[27].

By using one of these public Blockchain platforms plus a smart front-end to support the identity API, we can build and maintain a consortium-enabled, fully decentralized Blockchain, or directly use existing public chain as the infrastructure for the Global Blockchain data set. The ownership of the information relies on the owners of the smart contracts and private keys if, for instance, Ethereum is used. Manufacturers do not control the Blockchain; it lacks vulnerability to subversion from such organizations so long as the service is implemented by a neutral third party or group, whose smart contracts are audited. The Blockchain itself makes permanent records of the data.

Note that storing large amounts of data into public Blockchains is expensive and it also has a latency that depends on the particular system and the amount of reward provided to miners to add data. However, a consortium-enabled Blockchain can cut down the storage cost. Regardless of the type of Blockchain-based service used, the data stored in the Blockchain is always transparent and accessible as well as immutable and not susceptible to being forged or subverted. Though it has excellent durability and availability, there is no data privacy at all, namely the adversary is also able to read the data in the Blockchain. And, our proposed novel registration and authentication scheme can prevent sensitive identity data leaking from the chain.

A. Proposed Global Blockchain Infrastructure (B_G) for implementing Traceabilty of Edge Devices

The identification of a edge device for IoT applications is absolutely necessary because there are billions of devices already in the systems, and will continue to grow at an astonishing rate. These devices are produced by hundreds of different manufacturers, located across the globe. Traceabilty for an edge device is the key for verifying authentic hardware. We propose to implement a global Blockchain Infrastructure (B_G) that contains the necessary information to track the origin of an edge device.

Figure 2 shows our proposed Blockchain infrastructure to provide traceability for edge devices. Every device fabricated

*This is not a local blockchain copy; we use a real blockchain (in future, a sidechain) to retain the core value of a decentralized digital ledger.

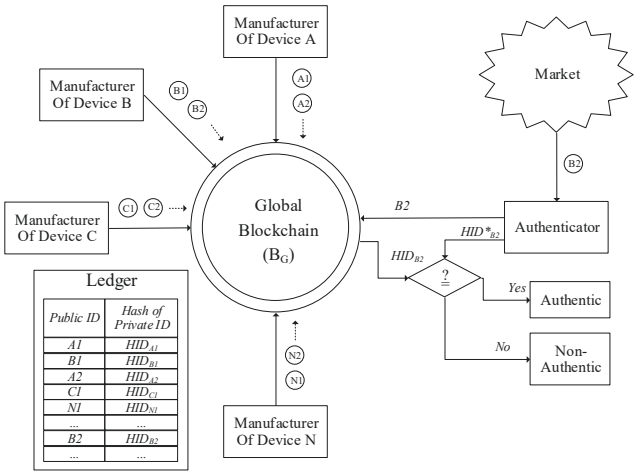

Figure 2: Global Blockchain instance (smart contract or key-value store) for verifying edge device identity.

by the authentic manufacturer will be equipped with a public ID (serial number) and a private ID, which can be generated from an SRAM PUF [18], [19]. Note that an edge device can also have other types of PUFs (such as an arbiter PUF [20], a ring oscillator PUF [21], or a butterfly PUF [28]) for generating the ID. All authentic edge devices must be registered on the global Blockchain B_G before they are sent to the market. Since the Blockchain ledger contains public entries, which can be accessed by anyone, an adversary can read the private ID and store this in a clone device. To prevent this, we propose to store a cryptographically secure hash [29] of the private ID in B_G, which prevents an attacker from constructing this private ID. The ledger for B_G contains the public ID, hash value of the private ID, plus relevant information (not shown in the figure) for identifying the manufacturer. Note that the small circles in the figure represent edge devices and we follow this notation throughout the paper.

The edge device registration process in B_G is as follows:

1) A manufacturer (the producer of device A) provides a challenge to the PUF of an edge device (e.g., A_1) to obtain its response. This response is the private ID (e.g., ID_{A1}) of that edge device. It then computes a cryptographically secure hash (e.g., HID_{A1}).
2) The manufacturer uploads the entry $\{A_1, HID_{A1}\}$ into (B_G) for device A_1, and finishes the registration for that device (e.g., A_1). Note that one can upload the challenge for a PUF along with its response to the Blockchain. However, this challenge can be stored internally, since only one challenge is required to produced an unclonable ID.
3) Perform Steps 1 and 2 to register other devices (e.g., A_2, A_3, ..., A_n).

A user or a distributor could use an Authenticator to access the global Blockchain B_G data set and thereby verify the authenticity of an edge device. The Authenticator can be an API (smart contract front end or blockchain accessor) or a complete program; that is, what a user needs to run during verification. Note that anyone can access an entry in the Global Blockchain dataset since it is public. The verification process can be described as follows:

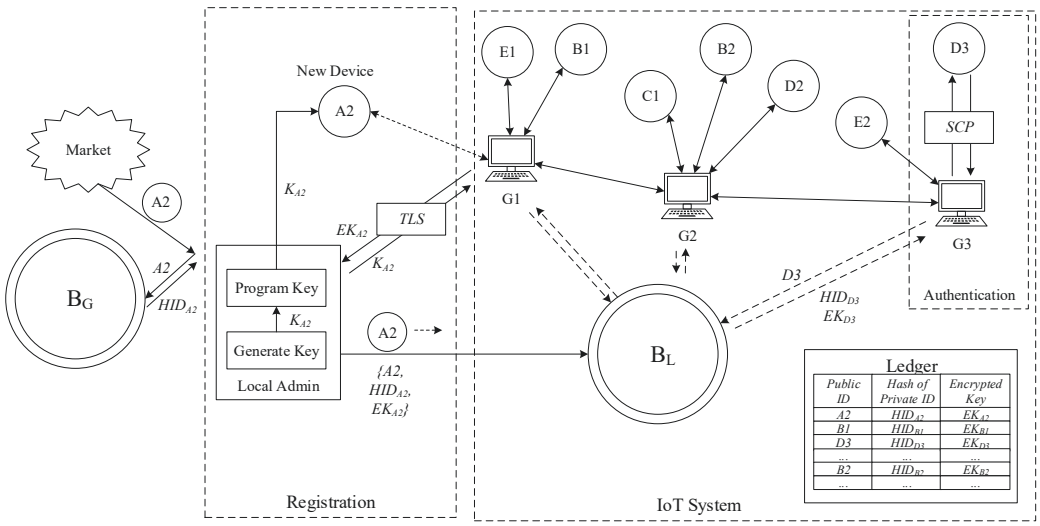

Figure 3: Local Identity Blockchain infrastructure for registering a new edge device and for authenticating edge devices periodically.

1) The authenticator queries the global Blockchain B_G data set with the public device ID (e.g., B_2 shown in Figure 2) of an edge device, and retrieves the hash of the private ID (e.g., HID_{B2}).
2) The authenticator provides a challenge to the PUF of the edge device under authentication, and collects the response, which is the secret private device ID (e.g., ID^*_{B2}).
3) The same secure hash is computed on the private ID (e.g., HID^*_{B2}).
4) The verification passes if the received hash from B_G, and computed hash of PUF response matches for an edge device i.

$$ver(HID_i, HID_i^*) = \begin{cases} pass & \text{if } HID_i = HID_i^* \\ fail & \text{otherwise;} \end{cases}$$

As mentioned above, economic viability of the global Blockchain infrastructure is important. If B_G is implemented using Ethereum, for instance, then smart contracts can be used to manage the key-value store added by manufacturers for each new device. A trade association (consortium, etc) registers vendors, and lets registered vendors add new device types. Then, vendors can use Ethereum smart contracts managed by the trade association to accomplish data registration for the B_G blockchain. The trade association makes the system viable by building payments into the Smart contracts to cover the overheads of data addition to the blockchain, as well as maintenance of the smart contracts and governance structure thereof. Unless a future technology such as sidechains becomes standard [30], the cost-per registration of this approach may be high, irrespective of fees that the trade association may level. Of course, the economics may become more favorable as data use cases grow.

For a global Blockchain B_G data set implemented via Scrybe, the Scrybe instance must be maintained across a set of permissioned miners. This set of miners could be distributed across several cloud infrastructures, and include trade association members as "miners" as well as arms-length third parties. Scrybe does not build in a payment system since it focuses on secure provenance. It always distributes the mining to active miners randomly and uniformly. Therefore, the proper economic model is simply a fee-for-service per miner running the service for the trade association. Since Scrybe uses lightweight mining, there is no arduous overhead for providing a miner. Ethereum smart contracts could be used as a the nominal payment method for such miners, as a function of their uptime (which can be measured by the percentage of blocks they mine).

B. Local Edge Device Registration and Authentication

Along with the threat from untrusted manufacturing, threats can also originate within the IoT/CPS infrastructures. Even though we screen all the new devices before adding them to the network, an authentic device can be compromised during its lifetime. For example, a rogue employee of an organization can replace an authentic device with its cloned counterpart. It is thus necessary to authenticate all edge devices in order to verify their authenticity on a regular basis. A local identity Blockchain data set, we denote as B_L, to hold all the public records of all the edge device connected to the CPS/IoT system, is necessary for such a purpose.

Figure 3 shows our proposed scheme that consists of two parts - (i) registration to add an new edge device into the IoT/CPS systems, and (ii) authentication to verify the identity of an existing device. Generally, trusted local administrator manage the IoT system and have the privilege to add an edge device into the network. In addition, a local administrator can hold one or multiple trusted devices, and only these devices are able to register an authentic edge device into the local identity Blockchain data set, B_L. We suggest that all trusted administrative devices remain offline after the local registration phase until a new edge device goes through the registration. This prevents a compromised device to register nonauthentic devices into B_L.

1) Registration of an Edge Device to the IoT/CPS infrastructure: Once the identity of a new edge device is verified (see Section II-A), the local admin performs the registration. The steps for the registration are as follows:

- The local admin first verifies whether the device is already registered in the local identity Blockchain data set B_L. No two devices have the same private ID.

- A random secret key is generated (e.g., K_{A2} for new edge device A_2 shown in Figure 3). The local admin programs that key into a one-time programmable (OTP) memory [31] of that edge device. Once the key is programmed into an edge device, direct access of the ID is blocked. One can only access the encrypted (see the communication protocol in Section II-C), which is required to prevent an adversary from copying this ID and launching an attack during authentication. Note that OTPs are resistant to tampering, and the contents remain unchanged once programmed. A one-time programming capability is provided primarily to prevent an attack arising when an adversary may gain direct access to an edge device (device capture).
- The secret device ID can be obtained from the SRAM-based PUF, and then the hash of that ID is computed.
- This new device is now added to the IoT system and directly communicates with a gateway (G_1 shown in Figure 3).
- The local admin communicates with G_1 using a secure communication protocol (e.g., TLS [32]). It sends the key K_{A2} to the gateway G_1. The gateway returns the encrypted key.
$$EK_{A2} = e_{K_{G1}}(K_{A2})$$
where $e()$ represents encryption process (e.g., AES [33]) and K_{G1} is the secret key of the gateway.
- The admin uploads the public ID, the hash of the secret ID and the encrypted key (e.g., $\{A_2, HID_{A2}, EK_{A2}\}$) into the B_L data set.

Once an edge device is registered into B_L, one gateway will be responsible to authenticate that device*. Note that we can do "damage control" if a gateway should be compromised. All edge device connected to it will be compromised, not all edge devices in the IoT environment. An adversary can access B_L through a compromised gateway. However, he/she can only read the hash of the private ID (e.g., HID_{B1}) and encrypted secret key (e.g., EK_{B1}). The identity of an edge device will be protected unless the secret key of the gateway (e.g., K_{G1}) is compromised. However, the access to these keys are often restricted from the software/firmware. It is thus safe to say that an adversary can impersonate a cloned device by compromising a gateway; however, the gateway will detect it once it returns to its normal state.

2) Authentication of an Edge Device in the IoT/CPS infrastructure: The objective is to perform authentication on a registered device. This is to verify its recent identity as it can be replaced or tampered while in operation. We plan to authenticate every edge device at a regular interval primarily in order to verify their actual identity. Note that the whole IoT/CPS system automatically initiates the authentication depending on the type of application and its criticality. Figure 3 shows the overview of this authentication process, which can be described as follows:

- The gateway queries the local Blockchain data set B_L with the public ID of an edge device (e.g., D_3 shown in Figure 3). B_L returns the hash of the private ID (e.g., HID_{D3}) and encrypted key (e.g., EK_{D3}).
- Gateway recovers the secret key (e.g., K_{D3}) by using the following equation:
$$K_{D3} = e^{-1}_{K_{G3}}(EK_{D3})$$
where, $e^{-1}()$ is the decryption function and K_{G3} is the secret key of the gateway G_3.

*We will loosen this restriction in future.

- The gateway uses a secure communication protocol, SCP (see the following Section II-C), to authenticate an edge device.

C. SCP: Secure Communication Protocol for Edge Device Authentication

We use a secure communication protocol [34] to transfer the secret ID of the edge device to the gateway for authentication. Figure 4 shows the secure communication protocol. Note that the gateway must be equipped with a cryptographically secure random number generator (CSPRNG) [35], [36] for generating random nonces (n). While implementing the protocol, it is necessary to use lightweight encryption for transferring the ID. The gateway and the edge device use a one-time-pad (OTP) [37], [38] for such purposes. Note that we need to provide the cryptographic hash computation support at the software level at the edge device.

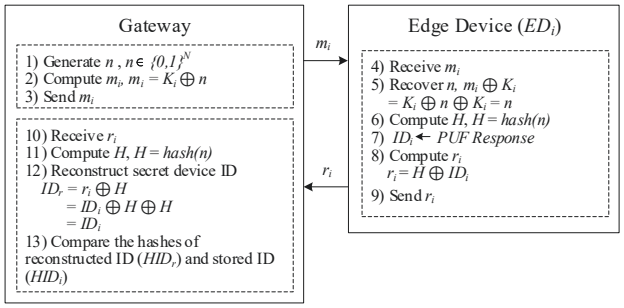

Figure 4: SCP: Secure communication protocol for edge device authentication [34].

The communication protocol for authenticating an edge device is described as follows:

- At the gateway, an on-chip CSPRNG generates a unique nonce (n). The gateway stores this in an on-chip memory for decrypting the secret device ID. A one-time pad (OTP) now encrypts this nonce with the key (K_i). The gateway then sends this encrypted nonce ($n \oplus K_i$) (depicted as (m_i) in the figure) to the i^{th} edge device, ED_i, to request for its identification (Steps 1-3).
- The unique nonce (n) is recovered at the edge device by XORing the m_i with the shared secret key (K_i). A cryptographically secure hash (*e.g.,* SHA-2 or SHA-3 [29]) is computed on this nonce (n) to produce a 256/512 bits hash output (H). We recommend using the existing hardware resources (embedded processor and memory [39], [40]) of the edge devices to compute the hash (Steps 4-6).
- The edge device first reconstructs its secret ID by querying the PUF, and then encrypts this ID, ID_i using the hash, H (Steps 7-8).
$$r_i = ID_i \oplus H$$
The encrypted edge device ID, r_i, is sent to the gateway for authentication (Step 9).
- After receiving r_i, the gateway computes the same hash (SHA-2 or SHA-3) using the stored random nonce, n. The gateway now reconstructs the secret device ID (Steps 10-12).
$$r_i \oplus H = ID_i \oplus H \oplus H = ID_i$$
- Finally, the gateway computes the hash of ID_i and compares it with the hash received from the local Blockchain data set

B_L. The verification passes if these two hashes match. Upon failure, the gateway evicts this edge device from the network.

III. Scrybe: A Blockchain-based Provenance System

This section overviews Scrybe, the Clemson-UTC-Auburn secure provenance system [16], [17], [41]. Subsequently, Section IV-E provides an overview and further references about how Scrybe supports non-repudiation and is also robust against distributed denial of service (DDoS) attacks.

There are two main components of Scrybe: blocks and transactions [17]. A blockchain is simply a sequence of linked blocks, where the current block contains the hash of the previous block.

a) Blocks: Each block contains the hash of the previous block, which makes the blockchain *immutable*. Blocks are added to the blockchain by *miners*, entities responsible for maintaining the integrity of the blockchain. Scrybe only allows authorized entities to mine blocks through the secure LWM algorithm (comprising the Scrybe "consortium"), which will be discussed further in Section IV-E. Miners are responsible for aggregating a list of transactions and calculating the Merkle root. The Merkle root allows other miners quickly to verify that every transaction is actually included in the block. When a miner is selected to add a block to the blockchain, the block is broadcast to all the other miners, and the data are verified (previous hash, Merkle root, and the miner's signature). At this stage, other miners will be able to detect if a transaction is omitted from the block, if an unauthorized miner broadcasts a block, and if the miner's signature is invalid.

b) Transactions: Transactions are the backbone of provenance [17]. Transactions can reference previous transactions, providing a chain of custody, or they can be *genesis events*, which register the acquisition of new data. References to other transactions use the *input* fields, while genesis events use *output* fields. The output fields contain persistent URLs (PURLs) that point to the data, along with the SHA-3 hash of the data, ensuring its validity. Additionally, the output fields contain PURLs that point to XML provenance of the data along with the SHA-3 hash of the XML provenance. Note that all transactions have an output while genesis events only have an output, with no input; normal transactions have both.

By storing the SHA-3 hash of the transaction instead of the original transaction, one can drastically reduces the size of the blockchain, and there will be no penalty for an extensive number of inputs and outputs in any given transaction. The original transaction will be stored on a *transaction server*, which will be locally maintained, along with the *data server* and the *metadata server*. In the current Scrybe implementation, the transactions and metadata coreside in the Blocks; for future Scrybe releases, a fully realized version that segregates the metadata server and transaction server is planned. However, our present application described here actually prefers the "lumped model."

c) Lightweight Mining: Scrybe introduces a novel way to mine new blocks in the blockchain [17], which is not a difficult proof-of-work (PoW) required in cryptocurrency applications. The lightweight mining algorithm (LWM) introduced in Scrybe is presented in Algorithm 1.

The purpose of LWM is to provide randomization in miner selection. In a Denial of Service (DoS) attack against Scrybe, we assume that a malicious miner targets a particular user by excluding the victim's transactions from the block he or she creates. The randomization offered by LWM guarantees that the victim's transactions will always be integrated sooner or later, as long as there is at least one honest miner. Formal proofs of this property will be published separately by the Scrybe authors.

Algorithm 1: Lightweight Mining Algorithm (LWM)

Input : The number of miners N
for *each miner* m_i, $0 \leq i < N$, **do**
  m_i generates a random number r_i ;
  m_i broadcasts the SHA-3 hash of the r_i, denoted by $H(r_i)$;
  Once m_i has collected all N hashes $\{H(r_0), H(r_1), \cdots, H(r_{N-1})\}$, m_i broadcasts the random number r_i ;
  Once m_i has collected all N random numbers $\{r_0, r_1, \cdots, r_{N-1}\}$, m_i calculates $l = \sum_j r_j \mod N$;
  m_l is the selected miner to create the next block from the collected transactions. (Without loss of generality, we map $m_i = i, 0 \leq i < N$ as a simple rank ordering for the registered miners.) ;
end

The core idea of LWM is "sharing-hash-first" [16]. If every miner only sends out the random number without sharing the hashes first, a miner can hold his/her own number until he or she has received everyone else's random number. This allows a malicious miner to manipulate the miner-selection by choosing a number that produces a m_l that is in favor of a particular miner or deliberately excludes a particular miner. "Sharing-hash-first" ensures that every miner has to share his/her own number (in the form of hash) with others before they see others' choices. Since hash values are considered impossible to invert in practice, a miner cannot change the random number after the fact. Thus, LWM can tolerate up to $N - 1$ malicious miners who collude. As long as there is one miner generating a random number, the modulo operation is randomized.

d) Servers : Locally maintained servers hold the raw data comprising the ledgers (the blockchains are held in the ledgers) [17]. The integrity of the transaction server can be verified by generating a list of all the transactions on the blockchain and comparing that to all the transactions on the transaction server. If there is any discrepancy, then the transaction server is deemed disreputable. The integrity of the data and metadata can be verified by comparing the SHA-3 hash of the data to the SHA-3 hash stored in the transaction—if these hashes differ, the relevant server is considered disreputable. The method for storing data on these servers is configurable, and left to the end-user's discretion.

IV. Threat Analysis

We analyze various attacks and solutions to prevent them.

A. Illegitimate Registration

Although we assume that any trusted manufacturer can register any device in the global Blockchain data set B_G in the description of our proposed scheme, we need to provide authorization unless there may be a potential risk of illegitimate registration on the B_G. Suppose, manufacturer A produces devices $\{A_1, A_2, ..., A_i\}$, are only allowed to add these device, similarly another manufacture B can only add devices $\{B_1, B_2, ..., B_i\}$ into data set B_G. We need to provide restrictions for manufacturer A to add devices B_is, and vice versa.

– *Prevention of Illegitimate Registration:* In order to prevent A from adding device B_is into data set B_G, apparently we should hold another list that stores the devices that each manufacture

can add into the Blockchain. Before adding a device into B_G, permissions will be first checked by querying the list. Compared to the device adding stream on the B_G, this permission list is only a tiny piece of data that needs less frequently update, and it is necessary to make it visible, accessible, and secure. Thus, storing this list in another Blockchain, or storing it as a part of the Ledger in the data set B_G could be taken into consideration.
– *Detection of Illegitimate Devices:* Suppose A has a piece of cloned B_2, we name it B_2^*, and then A registers B_2^* into data set B_G as A_2. This registration does not violate the permission list, however, the fake device has been registered into the B_G and cannot be detected by the Authenticator. Instead of changing the data set B_G's configuration, we suggest that the Authenticator should not only return whether a device is authentic but also return detailed registration info of a device. For instance, when the end user verifies this B_2^*, the Authenticator should return: "This device is an authentic device registered as A_2". So the user should know that he/she is holding a B_2 but registered as A_2. Also, all the registration on the data set B_G is traceable, the record of who added a device via illegitimate registration can be found in the transaction history. The objective of the above discussion aims to provide a baseline for prevention and detection.

B. Replay Attack on SCP

The objective of an attacker is to pass the authentication based on prior communication. In this attack, we assume that an adversary does not have access to the secret key (K_i), which is programmed in a tamper-proof memory in an edge device. Let us assume that an adversary observes two prior communications. First, he/she observes $n_1 \oplus K_i$ from the gateway and $H_{n_1} \oplus ID_i$ from the edge device. From this observation, the attacker can compute $K_i \oplus ID_i \oplus n_1 \oplus H_{n_1}$, which is shown below:

$$(n_1 \oplus K_i) \oplus (H_{n_1} \oplus ID_i) \tag{1}$$

From the second communication, the attacker observes $n_2(\neq n_1) \oplus K_i$ from the gateway and $H_{n_2} \oplus ID_i$ from the edge device, and can compute $K_i \oplus ID_i \oplus n_2 \oplus H_{n_2}$.

Now the attacker can perform the following operations:

$$(n_1 \oplus K_i) \oplus (n_2 \oplus K_i) = n_1 \oplus n_2 \tag{2}$$

$$(H_{n_1} \oplus ID_i) \oplus (H_{n_2} \oplus ID_i) = H_{n_1} \oplus H_{n_2} \tag{3}$$

From Equation 3, it is evident that an adversary successfully replays a prior communication if it becomes zero, when $H_{n_1} = H_{n_2}$. This contradicts the collision property of a secure hash [29]. Note that $n_1 \neq n_2$. Thus, the communication protocol becomes resistant to replay attack, when a system designer implements a secure hash function (SHA-2 or SHA-3).

C. Denial of Service (DoS) Attack

In this scenario, an attacker intentionally makes a genuine device fail during its authentication. For example, an attacker disables a security camera in a particular area by invalidating its registration. An attacker can eavesdrop on the communication between an edge device and the gateway. It seems possible to use IP spoofing to send incorrect responses to the gateway to disrupt authentication. During authentication, a gateway (G) generates a random number n_1, then send $m_1 = K \oplus n_1$ to an edge device (E). The attacker intercepts $r_1 = H(n_1) \oplus ID$ transmitted by the edge device E, and returns $r_1^*(\neq r_1)$ to the gateway G. As a result, the gateway will fail to reconstruct the edge device ID, and the authentication will fail. Fortunately, this can be prevented by a minor modification in the secure communication protocol, SCP described in Section II-C.

In order to prevent a DoS attack, SCP does not directly reject a device upon a failed authentication. The gateway needs to authenticate one more time for confirmation. Upon failed authentication, the gateway G will request the edge device E for another authentication and will verify the ID received from previous communication. The two stage verification process can be described as follows: *1) First Authentication:*

- The gateway G generates a random number n_1, then sends $m_1 = K \oplus n_1$ to the edge device E.
- E returns $r_1 = H(n_1) \oplus ID$ to G. Attackers intercepts r_1, and replaces it with $r_1^*(\neq r_1)$.
- G recovers the edge device ID, $ID^{(1)}$ by computing $ID^{(1)} = r_1^* \oplus H(n_1)$.
- Authentication fails as $ID \neq ID^{(1)}$ due to $r_1^*(\neq r_1)$.

2) Second Authentication:

- G generates a random number $n_2(\neq n_1)$, then sends $m_2 = K \oplus n_1$ to the edge device E.
- E returns $r_2 = H(n_2) \oplus ID$ to G. Attackers intercepts r_2, and need to replace it with $r_2^*(\neq r_2)$ to lunch the DoS attack again.
- G recovers the edge device ID, $ID^{(2)}$ by computing $ID^{(2)} = r_2^* \oplus H(n_2)$.
- Authentication fails as $ID \neq ID^{(2)}$ due to $r_2^*(\neq r_2)$.
- Gateway now verifies $ID^{(1)}$ and $ID^{(2)}$.

For an attacker, the probability of matching these IDs from two communications lies on the security strength of the hash function. Finding a collision for a secure hash (SHA-2 or SHA-3) is a hard problem. As a result, the attacker will have out of luck to find a r_2^* that maps $ID^{(1)}$.

D. Physical Attacks

The security of our proposed scheme depends largely on the shared secret key between the gateway and the edge devices that should be stored in tamper-proof memory and the SRAM PUF based device IDs for edge devices. These information can be stolen through sophisticated physical attacks or reverse engineering. Today's optical microscopes can produce 3D images of a microchip with superfine resolution. Scanning Electron Microscopes (SEM) and Transmission Electron Microscopes (TEM) can generate images of different inner layers of a microchip. Chipworks (now TechInsights) has successfully performed such experiments legitimately for the purpose of competitive analysis and patent research. The physical layout of a chip can be reconstructed through destructive physical attacks as well. Data stored in a non-volatile memory (NVM) can be reconstructed through infrared backside imaging, which can be used to directly look at the memory contents. All these physical attacks can definitely be used to find the secret key or the device ID. However, an adversary can impersonate only one device through such physical attacks, which does not make any financial motivation for performing such attacks.

E. Scrybe Security Verification

Since Scrybe replaces the resource-intensive Proof of Work (PoW) mining with LWM, a comprehensive security analysis must be conducted. The analysis shows that Scrybe provides data integrity, non-repudiation, and, more importantly, strong

resistance to Distributed Denial of Service (DoS) attacks resulting from insider threats. A formal verification of the LWM is given in [42]. A Petri Net [43] is used to model the DoS attack, which is then transformed into a Markov chain [44]. Further explanation of Scrybe security and robustness is given in a forthcoming publication as well that builds on [17], [42].

V. Conclusion

We presented a blockchain-based architecture for unique identification of edge devices through registration of hashed PUF attributes of a given device at manufacture. By accessing our Global Blockchain data set, a registered device can be verified by anyone anywhere without tracking the specific manufacturer. The Local Identity Blockchain data set (scoped to each site, however large) allows the local IoT/CPS to authenticate all edge devices robustly; this local approach supports a defense-in-depth architecture. This dual-use blockchain approach counters the counterfeit and clone problems, enhances the reliability and usability of the supply chain, and ensures the authenticity of edge devices. We introduced a novel, low-cost secure communication protocol for local device authentication. Optionally, we chose to utilize Scrybe, a new blockchain architecture, with a unique lightweight mining algorithm, which makes the whole scheme more efficient. Overall, blockchains gain security with the addition of data blocks, unlike databases, ensuring that greater security with greater utilization. As immediate future work, we will provide the means for N gateways to register access to a given device when locally provisioned. This justifies the use of the local blockchain data set since such a generalization can expand to large-scale deployments.

ACKNOWLEDGEMENT

The authors wish to acknowlege Drs. Richard R. Brooks and Yu Lu of Clemson University, and Mr. Carl Worley of Auburn University for their helpful advice and recommendations.

This material is based upon work supported by the National Science Foundation under Grants Nos. 1755733, 1547164, 1547245, and 1821926. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] Gartner Research, "Gartner says 6.4 billion connected "things" will be in use in 2016, up 30 percent from 2015," 2015.
[2] S. Smith, "Internet of Things connected devices to triple by 2021, reaching over 46 billion units," in *Juniper Research*, 2016.
[3] D. Evans, "The Internet of Things: How the Next Evolution of the Internet Is Changing Everything," 2011.
[4] W. Trappe, R. Howard, and R. S. Moore, "Low-energy security: Limits and opportunities in the Internet of things," *IEEE Security & Privacy*, vol. 13, no. 1, pp. 14–21, 2015.
[5] K. Rawlinson. Hp study reveals 70 percent of internet of things devices vulnerable to attack. [Online]. Available: http://www8.hp.com/us/en/hp-news/press-release.html?id=1744676#.WUrrwWgrKM8
[6] M. M. Tehranipoor, U. Guin, and S. Bhunia, "Invasion of the hardware snatchers," *IEEE Spectrum*, vol. 54, no. 5, pp. 36–41, 2017.
[7] M. M. Tehranipoor, U. Guin, and D. Forte, *Counterfeit Integrated Circuits: Detection and Avoidance*. Springer, 2015.
[8] U. Guin, K. Huang, D. DiMase, J. Carulli, M. Tehranipoor, and Y. Makris, "Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1207–1228, Aug 2014.
[9] U. Guin, D. DiMase, and M. Tehranipoor, "Counterfeit integrated circuits: Detection, avoidance, and the challenges ahead," *Journal of Electronic Testing*, vol. 30, no. 1, pp. 9–23, 2014.
[10] T. Borgohain, U. Kumar, and S. Sanyal, "Survey of security and privacy issues of Internet of things," *arXiv preprint arXiv:1501.02211*, 2015.
[11] H. He, C. Maple, T. Watson, A. Tiwari, J. Mehnen, Y. Jin, and B. Gabrys, "The security challenges in the IoT enabled cyber-physical systems and opportunities for evolutionary computing & other computational intelligence," in *Evolutionary Computation (CEC), 2016 IEEE Congress on*. IEEE, 2016, pp. 1015–1021.
[12] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008, https://bitcoin.org/bitcoin.pdf.
[13] Anonymous. White paper: Next-generation smart contract and decentralized application platform. Last accessed: March 17, 2018. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper
[14] G. Wood. Yellow paper: Ethereum: A secure decentralised generalised transaction ledger. Last accessed: March 17, 2018. [Online]. Available: https://github.com/ethereum/yellowpaper
[15] U. Mukhopadhyay, A. Skjellum, O. Hambolu, J. Oakley, L. Yu, and R. Brooks, "A brief survey of cryptocurrency systems," in *Annual Conference on Privacy, Security and Trust (PST)*, 2016.
[16] Richard Brooks and Anthony Skjellum, "Using the blockchain to secure provenance meta-data (a CCoE webinar presentation)," June 2017, technical presentation. NCCoE seminar.
[17] C. Worley, L. Yu, R. R. Brooks, J. Oakley, O. Hambolu, A. Skjellum, A. Altarawneh, J. S. Obeid, L. Lenert, K. Wang, and U. Mukhopadhyay, "Scrybe: A 2nd-generation Blockchain technology with Lightweight Mining for secure provenance and related applications," April 2018.
[18] W. Wang, A. Singh, U. Guin, and A. Chatterjee, "Exploiting power supply ramp rate for calibrating cell strength in SRAM PUFs," in *IEEE Latin-American Test Symposium*, 2018.
[19] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, "Fpga intrinsic pufs and their use for ip protection," in *International workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2007, pp. 63–80.
[20] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas, "Silicon physical random functions," in *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*. ACM, 2002, pp. 148–160.
[21] G. Suh and S. Devadas, "Physical Unclonable Functions for device authentication and secret key generation," in *Proc. of ACM/IEEE on Design Automation Conference*, 2007, pp. 9–14.
[22] bitcoin.org, "Bitcoin core version 0.9.0 released," 2014, https://bitcoin.org/en/release/v0.9.0#opreturn-and-data-in-the-block-chain.
[23] "Ethereum - a decentralized platform that runs smart contracts," https://www.ethereum.org/.
[24] "Storj - decentralized cloud storage," https://storj.io/.
[25] "Filecoin - a decentralized storage network," https://filecoin.io/.
[26] "Maidsafe - the new decentralized internet," https://maidsafe.net/.
[27] "Dadi - decentralized web services," https://dadi.cloud/en.
[28] S. S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls, "The butterfly PUF protecting IP on every FPGA," in *IEEE International Workshop on Hardware-Oriented Security and Trust.*, 2008, pp. 67–70.
[29] NIST, "FIPS PUB 180-4: Secure Hash Standard (SHS)," August 2015.
[30] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timn, and P. Wuille. Enabling blockchain innovations with pegged sidechains. Last accessed: March 17, 2018. [Online]. Available: https://blockstream.com/sidechains.pdf
[31] B. Stamme, "Anti-fuse memory provides robust, secure NVM option," *EE Times*, July 2012.
[32] T. Dierks, "The transport layer security (TLS) protocol version 1.2," 2008.
[33] J. Daemen and V. Rijmen, *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.
[34] U. Guin, A. Singh, M. Alam, J. Canedo, and A. Skjellum, "A Secure Low-Cost Edge Device Authentication Scheme for the Internet of Things," in *International Conference on VLSI Design*, 2018.
[35] D. E. Holcomb, W. P. Burleson, and K. Fu, "Initial SRAM state as a fingerprint and source of true random numbers for RFID tags," in *In Proceedings of the Conference on RFID Security*, 2007.
[36] B. Sunar, W. Martin, and D. Stinson, "A provably secure true random number generator with built-in tolerance to active attacks," *Computers, IEEE Transactions on*, vol. 56, no. 1, pp. 109–119, Jan 2007.
[37] G. S. Vernam, "Secret signaling system," 1919, US Patent 1,310,719.
[38] J. Katz and Y. Lindell, *Introduction to modern cryptography*. CRC press, 2014.
[39] T. Eisenbarth, S. Heyse, I. von Maurich, T. Poeppelmann, J. Rave, C. Reuber, and A. Wild, "Evaluation of sha-3 candidates for 8-bit embedded processors," in *The Second SHA-3 Candidate Conference*, 2010.
[40] "Atmel AVR232: Authentication Using SHA-256," Tech. Rep. [Online]. Available: http://www.atmel.com/Images/doc8184.pdf
[41] R. R. Brooks, K. Wang, L. Yu, J. Oakley, A. Skjellum, J. S. Obeid, L. Lenert, and C. Worley. (2018) Scrybe: A Blockchain ledger for clinical trials. IEEE Blockchain in Clinical Trials Forum: Whiteboard challenge winner.
[42] O. Hambolu, "Maintaining anonymity and trust," Ph.D. dissertation, Clemson University, 2018.
[43] R. David and H. Alla, "Petri nets and grafcet: tools for modelling discrete event systems," 1992.
[44] L. Yu, J. M. Schwier, R. M. Craven, R. R. Brooks, and C. Griffin, "Inferring statistically significant hidden markov models," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 7, pp. 1548–1558, 2013.