2020

FPGA based Blockchain System for Industrial IoT

Lei Xu
The University of Texas Rio Grande Valley, lei.xu@utrgv.edu

Lin Chen

Zhimin Gao

Hanyee Kim

Taeweon Suh

See next page for additional authors

Recommended Citation

L. Xu, L. Chen, Z. Gao, H. Kim, T. Suh and W. Shi, "FPGA Based Blockchain System for Industrial IoT," 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 2020, pp. 876-883, doi: 10.1109/TrustCom50675.2020.00118.

FPGA based Blockchain System for Industrial IoT

Lei Xu, Lin Chen, Zhimin Gao, Hanyee Kim, Taeweon Suh, Weidong Shi

***Abstract*—Industrial IoT (IIoT) is critical for industrial infrastructure modernization and digitalization. Therefore, it is of utmost importance to provide adequate protection of the IIoT system. A modern IIoT system usually consists of a large number of devices that are deployed in multiple locations and owned/managed by different entities who do not fully trust each other. These features make it harder to manage the system in a coherent manner and utilize existing security mechanisms to offer adequate protection. The emerging blockchain technology provides a powerful tool for IIoT system management and protection because the IIoT nature of distributed deployment and involvement of multiple stakeholders fits the design philosophy of blockchain well. Most existing blockchain construction mechanisms are not scalable enough and too heavy for an IIoT system. One promising way to overcome these limitations is utilizing hardware based trusted execution environment (TEE) in the blockchain construction. However, most of existing works on this direction do not consider the characteristics of IIoT devices (e.g., fixed functionality and limited supply) and face several limitations when they are applied for IIoT system management and protection, such as high energy consumption, single root-of-trust, and low decentralization level. To mitigate these challenges, we propose a novel field programmable gate array (FPGA) based blockchain system. It leverages the FPGA to build a simple but efficient TEE for IIoT devices, and removes the single root-of-trust by allowing all stakeholders to participate in the management of the devices. The FPGA based blockchain system shifts the computation/storage intensive part of blockchain management to more powerful computers but still involves the IIoT devices in the block construction to achieve a high level of decentralization. We implement the major FPGA components of the design and evaluate the performance of the whole system with a simulation tool to demonstrate its feasibility for IIoT applications.**

***Index Terms*—Industrial IoT, FPGA, TEE, blockchain**

1. Introduction

Industrial IoT (IIoT) has become one of the fundamental building blocks of Industrial 4.0 and the backbone to support IT/OT convergence [1]. Due to its wide deployment and importance to the digitalized industrial infrastructure, it becomes more critical to provide adequate management and security protection of the IIoT systems. Compared with other IT systems, an IIoT system usually involves a large number of devices that are owned and managed by different entities. To adequately manage and protect the whole system, these entities need to collaborate closely and verify that all other entities are following the policies/rules to manage their IIoT devices. However, these entities may have their own interests and do not fully trust each other, and there is lack of a third party that all entities trust to coordinate them.

Blockchain, which was first proposed to build cryptocurrency systems without relying on a centralized third party [2], sheds new light on IIoT management and protection, and many related applications are proposed, such as nonrepudiation computing service for IIoT [3], IoT data authentication [4], access control of an IoT system [5], and accountability mechanism against information leakage [6]. While these techniques provide various useful features to improve the management and security of the IIoT system, their effectiveness heavily depends on the underlying blockchain backbone.

Most existing general blockchain construction mechanisms are designed for computers and do not consider the features of an IIoT system. Recently, several efforts have been made to leverage a trusted execution environment (TEE) for blockchain construction, such as proof-of-elapsed-time [7] and improved Byzantine fault tolerant using TEE [8]. These blockchain systems usually provide better performance without expensive computations, and are attractive for IIoT systems. Most TEE based blockchain designs use Intel SGX [9] as the underlying TEE, and have several limitations for an IIoT blockchain: (i) High energy consumption. Intel SGX processors are usually designed for laptops/desktops/servers, which are power-hungry and not suitable for IIoT devices powered by batteries. (ii) Single root-of-trust. For Intel SGX and most other processors supporting TEE, the vendor of the hardware is the root-of-trust and responsible for processor management. If the vendor's master secret is compromised or the vendor colludes with an attacker, the whole security design of such as system collapses. The vendor can even disable a processor without the agreement of the owner. (iii) Low decentralization level. It is possible to deploy existing TEE based blockchain construction scheme on computers and connect IIoT devices to these computers. However, this architecture prevents the IIoT devices from participating in the blockchain construction and causes a low decentralization level.

To overcome these limitations, we proposed a field programable gate array (FPGA) approach to enable TEE based blockchain for IIoT. Under the framework of the new blockchain system, each IIoT device is equipped with an FPGA as TEE to control the transaction generation, and a set of servers with more computation/storage/communication capacity work together to collected generated transactions to

build the blockchain. The new scheme also includes a novel FPGA design that removes the dependency on the vendor as the root-of-trust, and allows the owner of the hardware to fully control the system. Furthermore, the solution offers better energy efficiency because of the way FPGAs work.

In summary, our contributions of the paper include:

- We propose a framework of using FPGA to build a blockchain for IIoT applications, which utilizes the FPGA as TEE on IIoT devices;
- We design a decentralized secure device enrollment scheme that eliminates the dependency on the hardware vendor as the root-of-trust, and a two-stage blockchain construction mechanism that allows both the IIoT devices and connected computers to contribute; and
- We analyze the security features of the proposed scheme and conduct experiments on the resources cost, power consumption, and overall performance to demonstrate its practicability.

The rest of the paper is organized as follows: In Section 2, we briefly review the background of blockchain and FPGA. We present an overview of the FPGA based blockchain construction scheme for IIoT in Section 3, and the detailed design is given in Section 4. Section 5 analyzes the security features of the scheme and gives the experiment results. We review related works in Section 6 and conclude the work in Section 7.

2. Background

In this section, we briefly introduce the background of blockchain and FPGA.

Blockchain and its construction. A blockchain is a data structure maintained by multiple parties together, where information is organized as linked blocks and determined by all participants together [2]. From the way of participant management, a blockchain can be classified as permissionless (public) or permissioned (private). A public blockchain is open to everyone to join and there is no mechanism to control the membership. On the other hand, a private blockchain only allows authorized parties to participate and needs administrators to manage all participants. Since an IIoT system is not an open system, we consider private blockchain in this paper.

FPGA. Field programmable gate arrays (FPGAs) are semiconductor devices that are built with a matrix of configurable logic blocks (CLBs) connected via programmable interconnections. CLBs are configured by bitstreams, which define the functions of the FPGA. Once the FPGA is configured, the functions are fixed and it works like an application specific integrated circuit (ASIC). Since the functions of an IIoT device are relatively fixed, the FPGA does not need to be re-configured frequently.

Modern FPGAs also provide some useful security features such as bitstream encryption [10] and anti-tamper capability [11], which are utilized to build the FPGA based blockchain for IIoT.

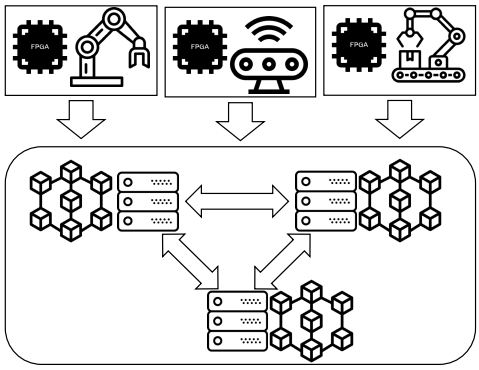

Figure 1. Overview of the FPGA based blockchain for IIoT systems. The FPGA attached to an IIoT device converts the data to transactions and sends to servers to build the blockchain. In practice, the IIoT device can also query the blockchain to obtain information.

3. Overview of FPGA based Blockchain for IIoT

This section gives an overview of the FPGA based blockchain architecture for IIoT, and discusses the security model and design goals.

3.1. System Overview

Three groups of participants are involved in the FPGA based blockchain for IIoT:

- The set of IIoT devices $\mathcal{D}$. An IIoT device $D \in \mathcal{D}$ generates data and utilizes the equipped FPGA to convert the original data to transactions and attaching metadata for the transaction, which is used in blockchain construction. In the remainder of the paper, we use the term device or FPGA for an IIoT device equipped with an FPGA.
- The set of servers $\mathcal{S}$. Servers are responsible for most of block construction and blockchain maintenance works. A server $S \in \mathcal{S}$ receives transactions from devices, packs them to blocks, and linked to the blockchain. Multiple servers communicate with each other and use a pre-defined consensus protocol to determine the order of the newly created blocks.
- The set of administrators $\mathcal{A}$. Administrators are responsible for adding and removing a participant (IIoT device or server) to/from the system. The owners of the devices/servers can play the role of administrators. Each administrator $A \in \mathcal{A}$ is equipped with a public/private key pair (pk_A, sk_A), and they know each other's public key. Therefore, they can communicate with each other securely by establishing encrypted and authenticated channels.

Figure 1 depicts the high-level architecture of the FPGA based blockchain for IIoT and the workflow. A typical IIoT system is not open to the public, and all participants need to be enrolled in the system before they can interact with the system. As the whole system is owned by multiple entities,

corresponding administrators work together to decide whether a device/server can participate the blockchain construction by assigning public/private key pairs to them, i.e., a device/server can participate the system only if it has a private key with corresponding public key endorsed by all administrators.

IIoT devices are the data source of the blockchain system. For an IIoT device, the FPGA board converts the data into transactions and controls the submission of these transactions to a connected server. The server then collaborates with other peer servers to pack the transactions into a block and connects it to the blockchain. There are different ways to utilize this framework for blockchain construction, and we borrow the idea of PoET for a concrete implementation in this paper.

3.2. Security Assumptions and Design Goals

Security assumptions. We assume each IIoT device is equipped with an FPGA, and the FPGA is properly built without any backdoor or trojan. Both the pre-built functions of and the bitstream loaded into the FPGA are tamper-resistant, i.e., an attacker cannot affect the execution of a function or extract extra information. We also require that the IIoT device and the FPGA board is securely connected that the attacker cannot alter the data send from the IIoT device to the FPGA. Note that we do not require a third party to pre-install any secret value to the FPGA.

A server is not fully trusted, i.e., it may deviate from the pre-defined protocols and try to add blocks to the blockchain according to their preferences. We also assume at least one administrator is honest and follows all pre-defined protocols.

Design goals. The major design goals of the FPGA based blockchain for IIoT systems include: (i) Fully controllable by the administrators of the system. Only administrators of the IIoT devices and corresponding FPGAs can set up the system without relying on any trusted third party, including the vendor of the devices. (ii) Preserving desirable blockchain features. The system should be built in a decentralized manner and provide an immutability feature. (iii) Re-configurable for different scenarios. The administrators should be able to add and remove participants, and change the configuration parameters of each participant to accommodate different application scenarios.

4. Detailed Design

In this section, we present the detailed design of the FPGA based blockchain for IIoT.

4.1. Cryptography Tools

The FPGA based blockchain for IIoT systems utilizes several cryptography primitives to achieve the security goals. In this paper, we use hash function SHA256, elliptic curve based signature scheme ECDSA, and encryption scheme ECIES [12]. A key derivation function (KDF) based on SHA256 HMAC [13] is also utilized. The elliptic curve based cryptographic primitives are selected because the public key

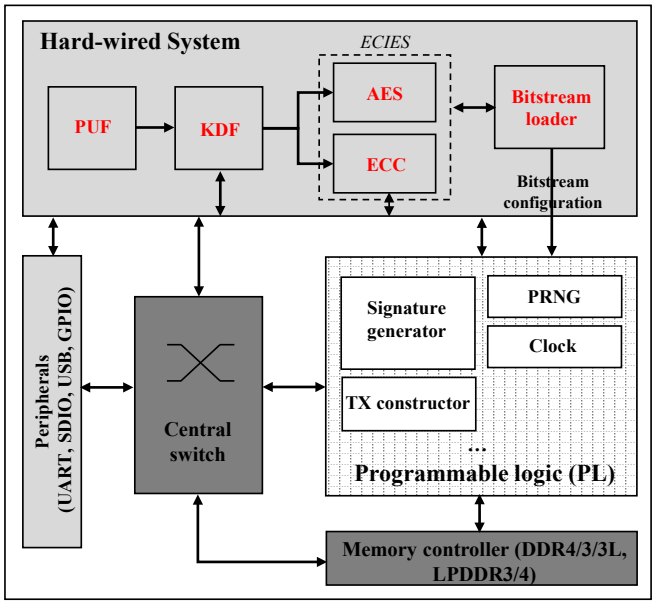

Figure 2. The FPGA design. Components with red names are newly introduced. These components work together with the bitstream to facilitate the blockchain construction. The value generated by the PUF is specific to the FPGA device and can only be fed into the KDF component as one of its inputs. The design does not require the FPGA vendor to place a secret value into the FPGA device, so end users do not need to trust the vendor for proper secret values management.

can be easily generated from the private key. Specifically, the KDF output is treated as an integer and used as a private key directly, and the corresponding public key is calculated by a scalar multiplication with the base point, which is a public parameter. Another benefit of using the elliptic curve cryptography suite is that ECIES uses a symmetric encryption scheme such as AES as an integrated part to handle concrete data encryption and decryption, which makes the scheme as a whole is efficient. A multi-signature scheme [14] is also used in the design, which aggregates multiple signatures on the same message to save space. For design consistency, we adopt an elliptic curve based multi-signature scheme [14].

4.2. The Design of the FPGA

The FPGA provides a TEE to prepare transactions and enforce part of the blockchain construction mechanism that is executed on the IIoT device side. Figure 2 depicts the diagram of the FPGA design, and the main components work as follows:

- PUF. The PUF can generate a secret value x that is unique to the device and the generation process is hard to be cloned [15]. x can only be accessed by several predefined components of the FPGA that are tamper resistant, such as the KDF. These components will never disclose x, and only use it as part as their inputs. In other words, x serves as the identity of the device in an indirect way as it can only be generated by the specific device itself.

- KDF. The KDF accepts more than one inputs and derives a pseudo random number. In this work, we use the HMAC based KDF, where the PUF output x is used as the key, and the user needs to provide another input. The one-wayness feature of KDF guarantees that even if an adversary obtains the derived value, he/she cannot recover the PUF generated value. The KDF output can be used directly as a symmetric key or a seed of a random number generator. To construct an elliptic curve key pair, the KDF output can also be treated as an integer to serve as a private key, and the corresponding public key is calculated using a scalar multiplication with the base point of the pre-defined elliptic curve.
- Bitstream loader. This component accepts a bitstream encrypted using ECIES, and decrypts it using a corresponding private key that is generated by KDF utilizing PUF. Since ECIES already provides authenticity protection, this component also guarantees the integrity of the loaded bitstream. Using asymmetric encryption for bitstream encryption is different from the current practice of using symmetric encryption such as AES and MAC to protect bitstream because multiple parties are involved in setting up the bitstream in the blockchain scenario and these parties do not fully trust each other to share a common secret (i.e., the symmetric encryption and MAC keys).

These components are built into the FPGA in a secure manner, i.e., they are only accessible through pre-determined interfaces and an adversary cannot look into or affect the execution processes of these components. Note that this design does not require the vendor to embed a root secret to an FPGA before delivering it to the end user, which is essential for building a decentralized system. Under the framework of FPGA based blockchain for IIoT, a KDF generates two types of public/private key pairs: device key and blockchain-specific key. There is only one key pair that serves as the device key, which is unique to the device and used to protect the communication with the outside world, such as loading encrypted bitstream into the device. There can be multiple key pairs as blockchain-specific keys, and each key pair is unique to a blockchain running on the device. If a device is involved in multiple blockchain instances, it uses a dedicated key pair for each blockchain.

Figure 3 summarizes the use of PUF and KDF in device enrollment and blockchain construction. More details are discussed in the remainder of this section.

4.3. Bitstream Structure

A bitstream is loaded into an FPGA to help the IIoT device prepare transactions that are submitted to servers to build the blockchain. All administrators need to achieve agreement on the bitstream contents, and a bitstream has the following functions/data sections:

- Digital signature generation. This function is further divided into two sub-functions. The first sub-function is obtaining the private key from the KDF. Secret values from administrators are provided to KDF for key generation, the process of which is discussed later. The second sub-function is to use the private key to sign a message and return the signature.
- Pseudo Random number generator (PRNG). This function utilizes the seed generated by the KDF and derives a sequence of random values. This function also maps the generated random value to a desired range, which is defined in the metadata section of the bitstream.
- Clock. This function measures the past time from a given point, and triggers other functions when a predefined condition is met.
- Metadata. Metadata stores information that is specific to the device and the blockchain, including values that are used to derive the blockchain specific private key, parameters for the random number generator, and other information related to the configuration of the device.
- Transaction construction. This function embeds information collected by devices into a transaction and adds necessary information, such as the digital signature of the transaction.

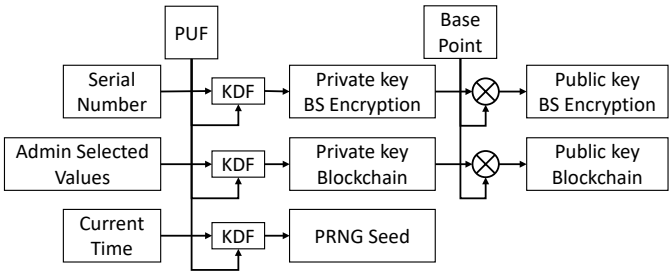

Figure 3. The role of PUF in the FPGA based blockchain for IIoT. The output of the PUF is used for multiple purposes, including the bitstream encryption key pair generation (device key), the blockchain key pair generation (this key pair is specific for the bitstream, which defines a blockchain instance), and the random number generator seed generation.

The construction of the bitstream and details on how these functions/data are used in the blockchain construction are discussed later.

4.4. Decentralized Device and Server Enrollment

One major advantage of the FPGA based blockchain for IIoT systems is that the FPGA vendor is only responsible for providing properly manufactured hardware that does not have backdoors or trojans, but does not work as the root-of-trust to put secret values in devices. After the device is delivered, the administrator has full control of the device and the vendor cannot impact the operation of the device.

Before a device can be used in the system, the administrators need to enroll it into the system by provisioning the bitstream and corresponding key materials in a decentralized manner, i.e., they need to work together and achieve consensus on the task. For a device $D \in \mathcal{D}$, all administrators in $\mathcal{A}$ need to achieve an agreement to enroll it to the system, and on the contents of the bitstream *BS* that will be loaded into D. Three features should be guaranteed in this process:

- Binding. *BS* can and can only be loaded into the specific device D so that if one believes a message is generated by *BS*, he/she can safely assume that the message is created by D.
- Involvement verifiability. Each administrator $A \in \mathcal{A}$ can verify that he/she has agreed to add D into the system.
- Non-malleability. An adversary (who can be an administrator) cannot alternate an existing bitstream or generate a new bitstream from existing ones without being detected by an honest server that processes transactions created by the compromised bitstream.

Device enrollment protocol. A device without a bitstream cannot participate in the blockchain construction, so the device enrollment process also includes the bitstream construction/loading process. The device enrollment process works as follows:

- The device D utilizes internal PUF and KDF to generate a public/private key pair (pk^D_{BSP}, sk^D_{BSP}) for bitstream protection, i.e., the device key for bitstream encryption and decryption. Here the KDF takes two inputs, the PUF value and the serial number of the device. pk^D_{BSP} also serves as the device key and is released to all administrators. Note that it is the administrators' responsibility to guarantee that pk^D_{BSP} is generated by the device and there is no man-in-the-middle attack. This can be done by physically connecting to the device to obtain the public key.
- Each administrator $A_i \in \mathcal{A}$ selects a random value s_{A_i} for the bitstream *BS* and device D. If the same bitstream is used for another device, the administrator needs to select another random value.
- A_i encrypts s_{A_i} using ECIES encryption algorithm and pk^D_{BSP}, the resulted is denoted as $c_{s_{A_i}}$. ECIES provides both confidentiality and authenticity protection.
- All administrators work together to attach metadata to the bitstream *BS*, and the metadata field includes three parts: (i) the list of identities of administrators in the form of their public keys; (ii) the ciphertexts of random values $c_{s_{A_i}}$; and (iii) an aggregated signature σ_{BS} on updated *BS*, which is generated by all administrators together using the multi-signature scheme. The *BS* is then encrypted with ECIES using pk^D_{BSP} and sent to the device.
- D loads the encrypted bitstream *BS* using the bitstream loader, where *BS* is decrypted using sk^D_{BSP} and the result is loaded into the program logic part of D.
- D uses all random values s_{A_i} included in *BS* metadata together with the internal PUF generated value as inputs to KDF to generate another public/private key pair (pk^D_{BS}, sk^D_{BS}) (the application specific key), which is used for the blockchain construction. pk^D_{BS} is released to all administrators.
- D generates a proof to each A_i to prove that his/her selected random value s_{A_i} is involved in the generation of pk^D_{BS}. Specifically, D generates a signature σ_{c_s} on $(pk^D_{BS}, \{c_{s_{A_i}}\})$ using private key sk^D_{BSP} and shares the signature with administrators.
- All administrators verify the signature σ_{c_s} using the public key pk^D_{BSP}, which they obtained directly from the device D. Then they endorse the public key pk^D_{BS} by signing it using their private keys and the multisignature scheme, and D can use sk^D_{BS} for interaction with other components of the system for authentication and other attestation purposes.

Note that all the operations carried out by D is determined by the bitstream *BS*, which is verified by all administrators. The encryption of *BS* can be done by any administrators, and others can repeat the encryption process to verify that the cipher-text of *BS* is correctly constructed before contributing to the aggregated signature σ_{BS}.

Security analysis of the enrollment protocol. The enrollment protocol guarantees the three features described at the beginning of Section 4.4: (i) Binding. The bitstream protection key pair (pk^D_{BSP}, sk^D_{BSP}) is generated by the device D and only D has access to the private key. Since the bitstream is encrypted by pk^D_{BSP}, D and the bitstream are bound. (ii) Involvement verifiability. An administrator only endorses the public key pk^D_{BS} that is bound with the bitstream after verifies that his/her secretly selected value is involved in the generation of pk^D_{BS}, so he/she can verify his/her involvement in the generation of the key pair (pk^D_{BS}, sk^D_{BS}), which is essential to participate the blockchain operations. (iii) Non-malleability. An adversary cannot modify an existing bitstream as its integrity is protected by a multi-signature generated by all administrators. If an adversary manages to generate a bitstream by him/herself and submits to the device, he/she cannot endorse the public key pk^D_{BS} returned by the device on behalf of other administrators. Therefore, the non-malleability feature is guaranteed. After the device is enrolled in the system, it can use sk_{BS} to generate digital signatures to authenticate itself and generated messages.

Server enrollment. The server enrollment process is simpler. Administrators generate a multi-signature on the server selected public key and distribute both the key and corresponding signature to other servers.

4.5. The Consensus Protocol

The FPGA based blockchain for IIoT system has both the devices and servers involved in the blockchain construction. We present a concrete implementation utilizing the idea of proof-of-elapsed-time [7], [16]. Under this framework, the IIoT devices do not participate blockchain construction directly because they do not have the capability to collect transactions from other devices to form blocks and store the blockchain. Instead, IIoT devices only prepare transactions for their own data and leave the blockchain construction and storage job to the servers connected to IIoT devices. Correspondingly, the execution of the consensus protocol is divided into two phases: transaction construction and block construction/storage. Therefore, the original design of PoET cannot be applied directly.

Transaction construction. The FPGA controls the transaction creation and submission of a device as follows:

- Based on the configuration parameters embedded in the bitstream, the FPGA uses the PRNG to select a waiting time t_w which follows a predefined distribution U, and the parameters of U are defined in the metadata section of the bitstream. The seed for the random number generator is provided by the KDF, which takes as input both the PUF and the current time. Since the device can only support limited time precision, U is a discrete distribution. Although the random number generator only gives uniformly distributed values, it is possible to construct arbitrary waiting time distribution using inverse transform sampling [17], which is also implemented as part of the bitstream and does not affect the FPGA based blockchain for IIoT framework.
- The clock function maintains the elapsed time since t_w is generated. When the elapsed time reaches t_w, the device can submit a transaction to the system. The device constructs a transaction tx_D using the transaction construction function included in the bitstream. The transaction has three fields: (i) The payload, which is used to store the IIoT device generated data. (ii) The waiting time t_w, which is created by the PRNG. (iii) The authentication tag, which stores a digital signature on the other fields and is generated using sk_{BS}^{D}.
- The constructed transaction tx_D is then sent to connected servers for further processing and being included in the blockchain. This step does not require a device to interact with others.

Block construction. Servers are responsible for putting transactions onto the blockchain, and they disseminate received transactions to other peer servers. After collecting a certain number of transactions from devices, a server packs them together to create a new block and links it to its local blockchain. The server also broadcasts the new block to other servers to include it in their local copies of the blockchain. The number of transactions in a block is determined by a predetermined parameter. Within the block, multiple transactions are ordered based on their creation and waiting time. Transactions with the same creation and waiting time can be ordered by other metrics such as its hash value. Besides the transactions, the constructed block also has the following two fields: (i) The hash value of the previous block. The purpose of the hash is to establish the link between the new block and its predecessor block; (ii) The digital signature. The server signs the newly created block with its private key sk_S.

Block acceptance. After receiving a new block, a server checks whether it is correctly formed: (i) All transactions are generated and signed by legitimate devices and correctly constructed; (ii) There is no replayed transactions in the block, i.e., transactions that have been included in existing blocks do not appear in the new block; (iii) It is linked to the most recent block with the right hash value; and (iv) The block signature is valid and generated by a known server. If the received block passes all the tests, the server accepts it by adding it to the end of his/her local copy of the blockchain.

Branch elimination. Due to the network latency and the way devices and servers are connected, servers may receive different transactions and produce different blocks, which causes multiple branches in the system. In order to eliminate extra branches in an efficient manner, the FPGA based blockchain for IIoT adopts a multi-criteria strategy. If one branch is just a predecessor of the other branch, the latter is selected, which includes more blocks. If two branches are different at the last block, the following criteria is applied: (i) A block with a larger sum of waiting time has higher priority; (ii) If two blocks have the same sum of waiting time, the one with a transaction that has the earliest creation time has higher priority; (iii) If the earliest creation time is also the same, the one with a smaller hash value has higher priority. By adopting this strategy, branches can be eliminated at their early stages as one does not need to wait for more blocks to be added to make the decision.

4.6. Detection of Compromised Devices

In an extreme case, an adversary may manage to break the TEE of the device, and leverages the compromised device as a stepping stone to attack the system. Since we use waiting time to control the submission of transactions, the adversary can manipulate the waiting time to gain advantages over other devices on submitting transactions. To mitigate this risk, servers who are responsible for putting transactions into blocks run a statistical test to check whether the pattern of a device submitting transactions follow the predefined waiting time distribution. If the submission rate of a certain device deviates too much from the distribution and exceeds a predefined threshold, a server flags the event and removes the device from the list of enrolled devices.

5. Analysis and Experiments

In this section, we analyze the performance and security features of the proposed FPGA based blockchain for IIoT.

5.1. Security Analysis

Besides the basic security features of a blockchain like immutability, the FPGA based blockchain for IIoT also provides extra security features such as full user control and high resilience.

Immutability. Immutability means an adversary cannot alter transactions that have been added to the blockchain. Since a legitimate private key used for signing transactions is safely kept inside the FPGA TEE, an adversary cannot generate fake transactions. He/she can only manipulate existing transactions, e.g., changing the order of transactions/blocks and removing an existing transaction. In order to achieve this goal, the adversary needs to show to other servers that the branch he/she builds has a higher priority than the legitimate one. According to the strategy of branch elimination, the likelihood of success is low due to two reasons: (i) Removing one or more transactions only makes the branch priority even lower; (ii) Reordering all the transactions into new

blocks can be easily detected. The probability that a server is able to submit a block depends on the device waiting time distribution and it is easy to detect if a server builds too many blocks.

Full user control. A device is identified by a public/private key pair (pk^D_{BSP}, sk^D_{BSP}), which is generated by the KDF component using the onboard PUF as one of the inputs. The private key is only available to some internal components of the device and the vendor cannot extract it. Therefore, the vendor cannot impersonate or revoke the device after it is delivered to the end user. To guarantee the genuineness of the corresponding public key, administrators physically obtain public keys from devices and then endorse them using a multi-signature scheme. Since these administrators own the system, they do not have the incentive to cheat and other participants can safely trust these certificates.

High resilience. The FPGA based blockchain for IIoT enforces full agreement on participants' membership, and is resilient to rogue administrators as long as there is at least one honest administrator in the system. Specifically, if a subset of administrators is not cooperative and wants to manipulate the system, they can try to add new servers and/or new devices to the system that they can fully control without the agreement of other administrators. In this case, legitimate servers (i.e., servers that are added to the system following the protocol) will not accept blocks created by these servers controlled by rogue administrators as their public keys are not endorsed by all administrators. The situation is the same for IIoT devices. If a subset of administrators collude and try to add a faked device to the system, all transactions submitted by this device will be rejected by an honest server as the signature of the transaction cannot be verified by a correctly endorsed device public key.

The only way for an adversary to pass the signature verification process to compromise the system is to break the FPGA to manipulate the device, i.e., put fake information into transactions and submit transactions at a faster rate. Detection of fake transactions is out of the scope of the FPGA based blockchain for IIoT but it can detect abnormal submission rate using statistic test as discussed in Section 4.

5.2. Consensus Protocol Analysis

According to the design of the FPGA based blockchain for IIoT, devices do not need to interact with others to submit a transaction. The only limitation for a device is that it has to wait for a randomly generated time period to submit a new transaction. This method can be treated as a simulation of PoW without the expensive mining process. In a PoW scheme, one needs to wait until it successfully produces a valid magic number to submit a block to the system. One can usually assume this waiting time follows a uniform distribution if each node randomly select their mining scope, and the distribution parameter is determined by the level of mining hardness.

For the FPGA based approach, the randomly generated waiting time is not applied to control block submission directly. Instead, a server waits for a certain number of transactions to arrive and packs them into a block to submit. The distribution of this waiting time depends on several factors, including the distribution of device waiting time, the number of devices in the system, the way devices and servers are connected, and network latency. If servers' waiting times are too concentrated, i.e., a large number of servers submit their blocks in a short time period, the probability of collision increases, which means different blocks are added after the same predecessor but finally only one of them will be accepted by the whole system. A high collision rate will decrease the performance of the system as more blocks are abandoned during the process of adding new blocks. When the server waiting time follows a uniform distribution, one can expect a lower collision rate. We simulate this scenario and present the experimental results in Section 5.4. We refer the readers to [7] for a detailed analysis of a more complicated distribution used in the original PoET, which also applies here.

TABLE 1. RESOURCE UTILIZATION OF THE MAJOR COMPONENTS ON THE FPGA.

Component	#BRAMs (18Kbit)	#DSPs (48E)	#FFs	#LUTs	Exec Time
PUF based RNG	0 (0%)	0 (0%)	315 (0.06%)	128 (0.05%)	1 Cycle (5ns)
Timer	0 (0%)	0 (0%)	35 (0.01%)	138 (0.05%)	N/A
256-bit public key generator	0 (0%)	288 (11.42%)	125,164 (22.83%)	84,368 (30.78%)	N/A
Signature generator	0 (0%)	320 (12.70%)	144,044 (26.28%)	99,189 (36.19%)	N/A

5.3. FPGA Bitstream Implementation and Cost

We implement major components of the bitstream using Xilinx ZCU102 board and Vivado 2018.3 tool. All the components are generated for targeting 200Mhz clock cycle. TABLE 1 summarizes the resource utilization and execution time.

- Random number generator. A strong or unpredictable random number generator is essential for the security of the system, otherwise, a catastrophe of security penetration such as private key leakage may happen [18]. To prevent such kinds of attacks, we implemented a PUF based True Random Number Generator (TRNG) proposed by Sadr and Zolfaghari-Nejad [19].
- Timer: We implement a simple timer for waiting time. A decrement counter and a control register are utilized for the purpose.
- ECC 256-bit public key and ECDSA signature generator. We select secp256r1 curve which has 256-bit private key. Both public key generator and ECDSA signature generator use the private key generated by the KDF component.

We also measure the energy consumption of these components. The random number generator consumes about

0.837W and the timer consumption is negligible. The public key cryptography operations are relatively more expensive, with the public key generator for 1.265W and the signature generator for 1.358W.

5.4. Performance Simulation

For scalability, we use SimBlock [20] to simulate the performance of FPGA based blockchain construction. SimBlock is an open-source blockchain simulator developed by Tokyo Institute of Technology. SimBlock is configurable by adjusting parameters, including node locations, node latency, block rate, block size, etc. We conduct our simulation on an Amazon EC2 t2.large type instance. We set the number of servers to 600, distributed in 6 different regions. The average latency between any two regions is 200 ms approximately. The down/upload bandwidth for each server is simulated as 25Mbps to 600 Mbps. We change the source code to adopt our consensus protocol as follows, which is based on the block waiting time and earliest transaction time in a block.

```
Collections.sort(blockList, new
    Comparator<Block>(){
@Override
public int compare(Block a, Block b){
    //First: check the block waiting
        time
    int order = Long.signum(a.
        getElapsedTime() - b.
        getElapsedTime());
    if(order != 0) return order;
    //Second: check the earliest
        transaction time in a block
    order = Long.signum(a.
        getEarliestTxTime() - b.
        getEarliestTxTime());
    if(order != 0) return order;
    //Compare hash value if both values
        above are same
    order = System.identityHashCode(a) -
         System.identityHashCode(b);
    return order;
    }
});
```

The results are summarized in Figure 4. Success rate indicates the rate where a block is added to the main chain successfully without a collision. We can observe that when the block rate is between 1 block per 2 minutes and 1 block per 10 minutes (Bitcoin's block rate), our system can maintain a high block success rate. The results show that our system is scalable because when the servers receive blocks at a higher rate, they can still achieve agreement on the order of all received blocks. In the simulation, the block size is set to 0.5 MB. We can further improve the throughput by increasing the block size. Furthermore, the situation we consider here is relatively extreme as a typical IIoT system is usually deployed within a small area and has a smaller number of servers.

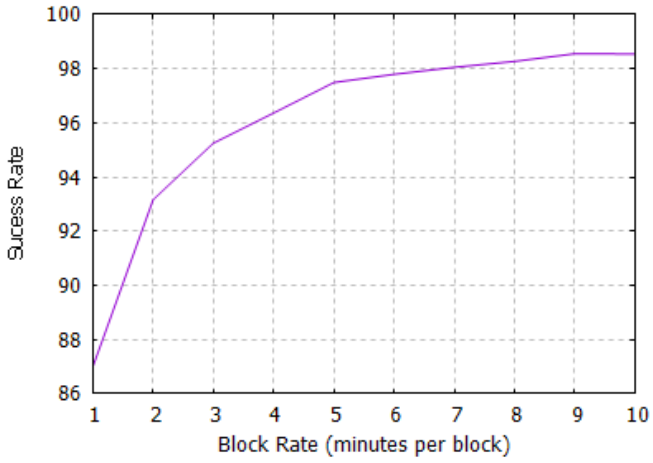

Figure 4. Simulation results on the performance of FPGA based blockchain for IIoT.

6. Related Works

In this section, we review related works on hardware facilitated blockchain construction for IIoT.

Proof-of-work acceleration hardware. Proof-of-work is still one of the major blockchain construction methods. To accelerate the mining speed and gain advantages in the competition with other miners, a variety of application-specific integrated circuits (ASICs) are developed for cryptocurrencies such as Bitcoin [21], [22]. FPGAs are also used to accelerate mining [23]. This line of research focuses on PoW performance improvement and has a different goal than the scheme presented in this paper, which aims at removing the single root-of-trust and offering a new way of hardware based blockchain construction for IIoT.

Hardware based blockchain security enhancement. CONFIDE utilizes a TEE to offer transaction confidentiality in the consortium blockchain environment [24]. Yuan *et al.* proposed a design to implement private smart contract on public blockchain, which used a TEE to build a confidential and secure off-chain platform for private contracts storage and execution [25]. Ayoade *et al.* designed a secure IoT data management scheme combining TEE and blockchain technology [26]. These works use SGX as the underlying TEE [9], and do not work directly on IIoT devices. But they can be integrated with the proposed scheme and run on the servers. Therefore, they are complimentary to the FPGA based blockchain system.

Hardware based blockchain construction. Several blockchain construction schemes utilizing hardware, especially hardware based TEE, have been proposed. Intel PoET utilizes the SGX to generate random waiting time to control the priviladge of block producing [7]. The concept of proof-of-luck also utilizes the SGX as a trusted random source to [27]. Hardware based TEE is also used to improve existing blockchain construction schemes such as proof-of-stake [28],

and BFT [8]. These works do not consider the initialization of the TEE and rely on the vendor as the root-of-trust. The decentralized enrollment scheme can be applied to these schemes to remove the dependency on the vendor. The other major difference is that the proposed scheme divides the blockchain construction into two parts and allows the IIoT devices to participate the process. This feature further improves the decentralization level of the blockchain system.

7. Conclusion

Using hardware based TEE provides a new choice for constructing an efficient blockchains for IIoT system management and protection. The proposed FPGA based blockchain framework leverages power efficient FPGAs to build TEE for IIoT devices to support blockchain operation, which does not rely on any third party as the root-of-trust. We also provide one concrete blockchain construction mechanism under this framework using the concept of proof-of-elapsed-time. Besides the desirable feature of fully user controllability, the FPGA is power efficient and fits the energy constrained IIoT environment. The cost and efficiency of the FPGA based blockchain for IIoT are also verified by experiments and simulation.

References

[1] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, "The industrial internet of things (iiot): An analysis framework," *Computers in Industry*, vol. 101, pp. 1–12, 2018.

[2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[3] Y. Xu, J. Ren, G. Wang, C. Zhang, J. Yang, and Y. Zhang, "A blockchain-based nonrepudiation network computing service scheme for industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3632–3641, 2019.

[4] L. Xu, L. Chen, Z. Gao, X. Fan, T. Suh, and W. Shi, "Diota: Decentralized-ledger-based framework for data authenticity protection in iot systems," *IEEE Network*, vol. 34, no. 1, pp. 38–46, 2020.

[5] M. A. Khan and K. Salah, "Iot security: Review, blockchain solutions, and open challenges," *Future Generation Computer Systems*, vol. 82, pp. 395–411, 2018.

[6] Y. Xu, C. Zhang, Q. Zeng, G. Wang, J. Ren, and Y. Zhang, "Blockchain-enabled accountability mechanism against information leakage in vertical industry services," *IEEE Transactions on Network Science and Engineering*, 2020.

[7] L. Chen, L. Xu, N. Shah, Z. Gao, Y. Lu, and W. Shi, "On security analysis of proof-of-elapsed-time (poet)," in *International Symposium on Stabilization, Safety, and Security of Distributed Systems*. Springer, 2017, pp. 282–297.

[8] J. Behl, T. Distler, and R. Kapitza, "Hybrids on steroids: Sgx-based high performance bft," in *Proceedings of the Twelfth European Conference on Computer Systems*. ACM, 2017, pp. 222–237.

[9] V. Costan and S. Devadas, "Intel sgx explained," *IACR Cryptology ePrint Archive*, vol. 2016, no. 086, pp. 1–118, 2016.

[10] S. Trimberger, J. Moore, and W. Lu, "Authenticated encryption for fpga bitstreams," in *Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays*, 2011, pp. 83–86.

[11] E. Peterson, "Developing tamper resistant designs with xilinx virtex-6 and 7 series fpgas," *Application Note. Xilinx Corporation*, 2013.

[12] D. Hankerson and A. Menezes, *Elliptic curve cryptography*. Springer, 2011.

[13] L. Chen, "Recommendation for key derivation using pseudorandom functions," National Institute of Standards and Technology, Tech. Rep., 2008.

[14] A. Boldyreva, "Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme," in *International Workshop on Public Key Cryptography*. Springer, 2003, pp. 31–46.

[15] J.-L. Zhang, G. Qu, Y.-Q. Lv, and Q. Zhou, "A survey on silicon pufs and recent advances in ring oscillator pufs," *Journal of computer science and technology*, vol. 29, no. 4, pp. 664–678, 2014.

[16] A. Corso, "Performance analysis of proof-of-elapsed-time (poet) consensus in the sawtooth blockchain framework," Master's thesis.

[17] C. R. Vogel, *Computational methods for inverse problems*. Siam, 2002, vol. 23.

[18] E. DeBusschere and M. McCambridge, "Modern game console exploitation," *Technical Report, Department of Computer Science, University of Arizona*, 2012.

[19] A. Sadr and M. Zolfaghari-Nejad, "Physical unclonable function (puf) based random number generator," *arXiv preprint arXiv:1204.2516*, 2012.

[20] Y. Aoki, K. Otsuki, T. Kaneko, R. Banno, and K. Shudo, "Simblock: a blockchain network simulator," *arXiv preprint arXiv:1901.09777*, 2019.

[21] M. B. Taylor, "The evolution of bitcoin hardware," *Computer*, vol. 50, no. 9, pp. 58–66, 2017.

[22] J. Barkatullah and T. Hanke, "Goldstrike 1: Cointerra's first-generation cryptocurrency mining processor for bitcoin," *IEEE micro*, vol. 35, no. 2, pp. 68–76, 2015.

[23] S. Oliveira, F. Soares, G. Flach, M. Johann, and R. Reis, "Building a bitcoin miner on an fpga," in *South Symposium on Microelectronics*, vol. 15, 2012.

[24] Y. Yan, C. Wei, X. Guo, X. Lu, X. Zheng, Q. Liu, C. Zhou, X. Song, B. Zhao, H. Zhang *et al.*, "Confidentiality support over financial grade consortium blockchain," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 2227–2240.

[25] R. Yuan, Y.-B. Xia, H.-B. Chen, B.-Y. Zang, and J. Xie, "Shadoweth: Private smart contract on public blockchain," *Journal of Computer Science and Technology*, vol. 33, no. 3, pp. 542–556, 2018.

[26] G. Ayoade, V. Karande, L. Khan, and K. Hamlen, "Decentralized iot data management using blockchain and trusted execution environment," in *2018 IEEE International Conference on Information Reuse and Integration (IRI)*. IEEE, 2018, pp. 15–22.

[27] M. Milutinovic, W. He, H. Wu, and M. Kanwal, "Proof of luck: An efficient blockchain consensus protocol," in *proceedings of the 1st Workshop on System Software for Trusted Execution*, 2016, pp. 1–6.

[28] W. Li, S. Andreina, J.-M. Bohli, and G. Karame, "Securing proof-of-stake blockchain protocols," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Springer, 2017, pp. 297–315.