BlockPro: *Block*chain based Data *Pro*venance and Integrity for Secure IoT Environments

Uzair Javaid
National University of Singapore
Singapore
uzair.javaid@u.nus.edu

Muhammad Naveed Aman
National University of Singapore
Singapore
naveed@comp.nus.edu.sg

Biplab Sikdar
National University of Singapore
Singapore
bsikdar@nus.edu.sg

ABSTRACT

Data provenance and data integrity are among the key concerns in IoT based environments such as smart cities, smart grids, and vehicular networks etc. Many IoT devices suffer from both impersonation and data tampering attacks due to their architectural and computational limitations, which are unable to provide adequate level of security. This paper aims to provide and enforce data provenance and data integrity in IoT environments by using Physical Unclonable Functions (PUFs) and Ethereum, a blockchain variant with smart contracts. PUFs provide unique hardware fingerprints to establish data provenance while Ethereum provides a decentralized digital ledger which is able to withstand data tampering attacks.

CCS CONCEPTS

• **Computer systems organization** → **Peer-to-peer architectures**; **Embedded and cyber-physical systems**; • **Security and privacy** → *Security protocols*; Security in hardware;

KEYWORDS

Internet of things (IoT), impersonation attack, data provenance, physical unclonable function (PUF), blockchain, smart contract, and Ethereum.

ACM Reference Format:
Uzair Javaid, Muhammad Naveed Aman, and Biplab Sikdar. 2018. BlockPro: *Block*chain based Data *Pro*venance and Integrity for Secure IoT Environments. In *The 1st Workshop on Blockchain-enabled Networked Sensor Systems (BlockSys'18), November 4, 2018, Shenzhen, China.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3282278.3282281

1 INTRODUCTION

The Internet of Things (IoT) is a technological advancement aimed at the integration of physical devices in a wide range of environments through the Internet. This ranges from vehicles to bicycles, smart homes, industries, CCTV cameras etc. These devices can communicate and share data/information autonomously with minimal human intervention. With the ongoing developments, IoT is expected to help and facilitate in resource management, intelligent spaces, smart cities and industry automation etc. [9, 11].

An overview of traditional IoT based system designs is shown in Figure 1 where different kinds of devices (e.g. sensors and cameras) interact with a central server through a communication link. It is worth noting that cyber threats exist from device level to the communication level. Moreover, the resource constrained nature of IoT devices exacerbates the security challenges [9, 10, 12]. Thus, it may not be feasible to apply classical security techniques to IoT systems. Therefore, new protocols and frameworks are needed for the IoT.

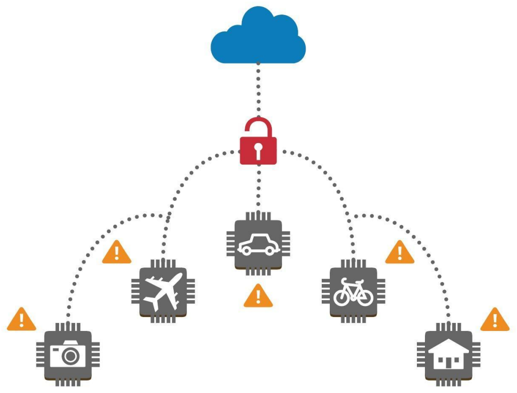


Figure 1: Security loopholes in a conventional IoT architecture.

Among the various security requirements, data provenance and data integrity remain major concerns for the IoT. This paper proposes the use of Physical unclonable functions (PUFs) for providing data provenance. A PUF can be formally described as a system that maps a set of challenges to a set of responses based on the physical micro-structure of a device. This way, it makes it nearly impossible to modify, clone or tamper with a PUF [3], thereby providing a unique hardware fingerprint for each IoT device.

This paper uses blockchain as its main platform. A blockchain is a decentralized online ledger consisting of a genesis block with all the succeeding blocks linked together through hashes. Every block contains a hash of a timestamp and the previous block, making it highly resistant to data tampering. Moreover, blockchains use distributed digital ledgers and decentralized storage for providing increased robustness and transaction transparency. The most widespread use of blockchains is in crypto-currencies such as Bitcoin [7]. However, many applications have adopted blockchains to provide decentralized and trust-free solutions. Another use of blockchains is with computer programs (*smart contracts*) such as Ethereum [4]. One of the advantages of using blockchains with IoT environments is that it allows IoT devices to transact freely without relying on third parties [7]. However, scalability in blockchains remains one of the key concerns [5].

A typical IoT environment involves varying security features, and relies on the security capabilities of the devices, servers, and other associated components. This paper proposes BlockPro, which integrates PUFs with blockchain for a safe and secure IoT environment that not only ensures data provenance, but also enforces data integrity by providing an immutable storage platform. The framework proposed in this paper provides the following security features:

(i) A blockchain based solution for preserving data integrity and blocking unregistered devices.
(ii) A decentralized network model for control and trust-free operation of IoT environments.
(iii) Provide unique identities to IoT devices.

This paper is organized as follows. Section 2 introduces PUFs. Section 3 describes the network and threat models. Section 4 presents the proposed system design and Section 5 presents the security and performance analysis. Section 6 describes the implementation and evaluation. Section 7 discusses the related work and we finally conclude the paper in Section 8.

2 PHYSICAL UNCLONABLE FUNCTIONS

The proposed technique to provide data provenance and integrity in IoT systems is based on PUFs. Therefore, we present a brief introduction to PUFs in this section.

A PUF is characterized by a challenge response pair (CRP). Mathematically, a PUF can be represented as:

$$R^i = P(C^i). \tag{1}$$

If a challenge is input to a PUF a number of times, the PUF will always produce the same response with high probability. On the other hand, if the same challenge is input to a different PUF it will produce a different response with high probability [1].

3 NETWORK MODEL, ASSUMPTIONS, AND THREAT MODEL

3.1 Network model

Figure 2 describes the network model of BlockPro. This model consists the following units:

- **IoT device:** This unit comprises of "things" (IoT nodes/devices).
- **Smart contract:** This is the computer program which works together with the blockchain to ensure data provenance and data integrity. The following two smart contracts are coded for BlockPro:
  i. SmartContract_1: This contract interacts with the IoT devices and makes sure they are legit and the data being uploaded is coming from a known and trusted origin, i.e., establishing data provenance.
  ii. SmartContract_2: This contract is responsible for storing and retrieving the data on the blockchain. Note that this smart contract can only be called by SmartContract_1 and no one else.
- **Server/Miner:** These indicate the constituents of the blockchain network infrastructure who are responsible for both data storage and providing the required computation for the blockchain network to operate smoothly.

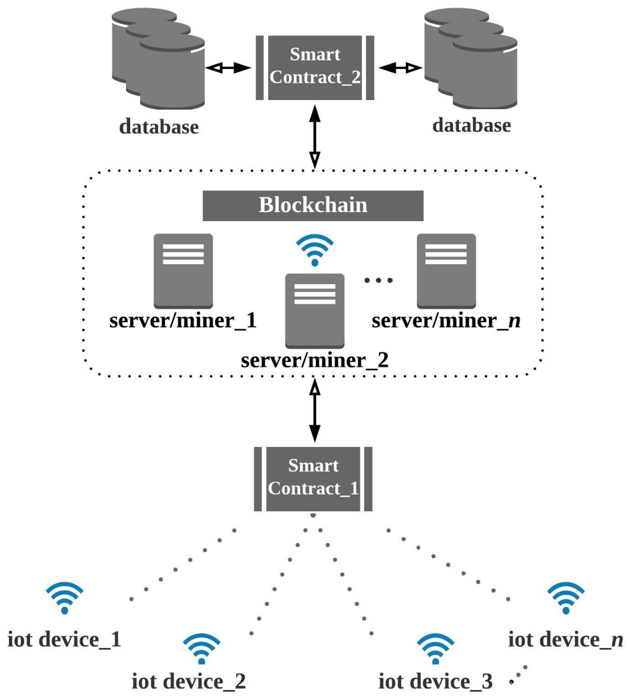


Figure 2: The BlockPro network model.

3.2 Assumptions

i. IoT devices are resource constrained.
ii. The PUF and the IoT device are assumed to be a system-on-chip (SoC) and any attempt to tamper with and/or remove the PUF will render the device useless.
iii. The blockchain network and its constituents, i.e., servers/miners, are not resource constrained. This results in a blockchain that can scale effectively relative to the number of IoT devices.
iv. The verifiers (miners) of the blockchain network are secure from physical as well as non-physical attacks.

3.3 Threat model

The objectives of an attacker for the proposed system design are as follows:

(1) Impersonate an IoT device and transfer maliciously modified data to the server.
(2) Tamper or modify the data sent by legitimate IoT devices.

4 PROPOSED SYSTEM DESIGN

In this section we describe the proposed system design for providing data provenance and data integrity.

4.1 BlockPro

The variant of blockchain used for BlockPro is Ethereum. Ethereum is selected because of the following reasons: firstly, it allows the operation of smart contracts, and secondly, it allows for faster transaction times, i.e. it can handle requests faster than Bitcoin. Every node of Ethereum has a 16-bit address and an account. Accounts in Ethereum can be of two types described as follows:

i. Externally owned accounts (EOAs): This type of account is used by a user who controls it with a private key. The user can interact with the Ethereum network by sending and/or receiving messages.
ii. Contract: This type of account is used by a computer program/code to control its operation.

IoT devices need to first register themselves through smart contracts to become part of the BlockPro network.

4.1.1 Smart contract design. The smart contract in BlockPro is designed to enable safe and secure communication among IoT devices and the blockchain constituents. Figure 3 holistically illustrates the information flow layout of IoT devices and the blockchain network. The IoT devices interact with the smart contract in the blockchain which in turn interacts with the distributed servers of the BlockPro network. Data can be transferred by the IoT devices only after they are registered. The functions $reg.iot\_device(addr)$ and $del.iot\_device(addr)$ are responsible for registering and deleting IoT nodes with respect to their Ethereum addresses respectively. Moreover, the $list.reg\_iot\_devices$ maintains a trusted list of all the IoT devices registered with the network and $CR.iot\_dev\_PUF$ maintains a list of PUF challenge response pairs for the registered IoT devices in the network.

The smart contract in BlockPro was coded using Solidity in Remix IDE of Ethereum. It is a contract-oriented, high-level language for the Ethereum virtual machine (EVM) environment. The operation of the smart contract consists of two phases:

(1) **Initialization:** For initialization of the contract, a *server* node (e.g., operated by the owner of the IoT devices or a cloud based service provider) deploys the smart contract. This will then be recognized and known as the *server* variable by the smart contract and the contract will recognize this variable as the trusted host. This paper assumes that all the decentralized servers in BlockPro are trusted hosts. After the initialization by the trusted host, the address of the contract will be broadcasted in the BlockPro network so that accounts (IoT nodes) can interact with it.
(2) **Deployment:** In this phase, IoT devices interact with the server to get registered. The registration of IoT devices is facilitated by the Smart Contract. In order to get registered, IoT devices first need to register their PUF CRP to the BlockPro network. This CRP is stored by the smart contract and is used for assuring data provenance. Moreover, along with their PUF CRP, the address of IoT devices is also stored by the smart contract to validate their requests. When a device transmits (Tx) some data, the smart contract checks if it is in the registered list of trusted devices. If it is not in the registered list, the request is terminated. Otherwise, a challenge is sent to the IoT device and if it generates a positive response to its PUF challenge, the data is stored on the blockchain.

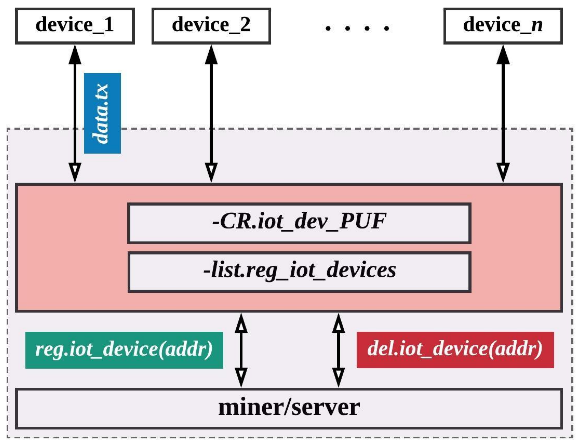


Figure 3: The information flow layout of BlockPro.

Algorithm 1: PUF challenge-response validation algorithm

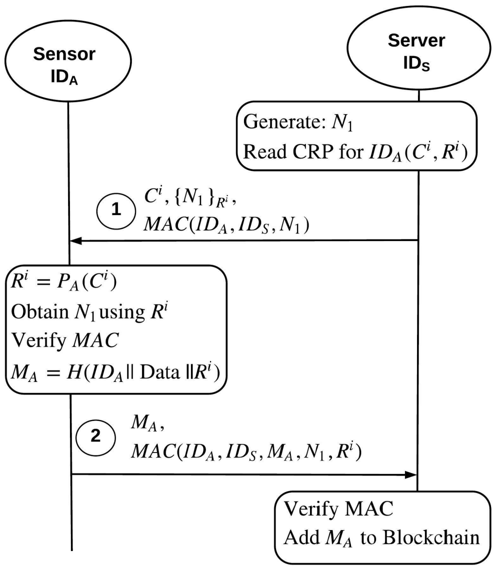
```
1  function: PUF_CR(d_i)
   Input  : tx(d_i)
   Output : pass, fail
2  if (tx(d_i) is uploaded and tx(d_i) is valid) then
       // Check d_i is registered/unregistered
3      if (d_i is registered in the trusted list) then
           // Check d_i has positive PUF_{d_i}
           // invoke PUF challenge-response protocol
4          if (PUF_{d_i} response = positive) then
5              return pass
6          else
7              return fail
8          end
9      else
10         return fail
11     end
12 else
13     return fail
14 end
15 end function
```

4.2 System operation

The operation of BlockPro is carried out by first registering the IoT devices and then storing their data for use by applications. The former ensures data provenance for the IoT devices and the latter enforces data integrity with its immutable chain of records.

This paper assumes that each IoT device comes equipped with a PUF. Moreover, the response to a specific challenge can be obtained using only two ways, viz. either by the IoT device using its PUF or by the operator from a saved copy in its memory. When a certain device d_i is to be registered in BlockPro, a CRP for its PUF is already recorded by the operator in the network by interacting with the smart contract. This way, each device has its own unique ID along with a unique response. BlockPro achieves data provenance using PUFs. After data is transmitted by a device d_i, the smart contract checks its validation using the algorithm detailed in Algorithm 1. In this algorithm, function $\mathrm{PUF}_{CR}(d_i)$ is used to check data provenance of device d_i. When d_i transmits data, the algorithm first checks if the data is coming from a trusted list of registered IoT devices. If it is, the algorithm then checks whether its PUF challenge-response is correct or not. It does so by invoking the PUF challenge-response protocol shown in Figure 4. The steps for this protocol are as follows:

i. The server in the BlockPro network with identity ID_S reads the CRP (C^i, R^i) for device ID_A and generates a nonce N_1 for it.
ii. The server ID_S then sends the nonce N_1 which is encrypted using R^i, i.e., $\{\mathrm{N}_1\}_{R^i}$ and the challenge C^i to the IoT device ID_A in message 1.
iii. Upon obtaining the nonce from the server ID_S, IoT device ID_A then obtains the corresponding response R^i for the challenge C^i with the help of its PUF.
iv. After obtaining the response R^i, ID_A performs the following steps:
   a. Obtain N_1 using R^i as the secret key.
   b. Using the parameters in its memory, verify and validate the MAC.
   c. Once it verifies the MAC, its produces a hash: h(ID_A, $data$, R^i) and sends it to the server in message 2.
v. Once the server ID_S receives message 2 from the IoT device ID_A, it checks and verifies the MAC and the hash. If both are valid, the request to transmit data is entertained. Otherwise, the request is dropped.



Sensor ID_A
Server ID_S
Generate: N_1
Read CRP for ID_A(C^i, R^i)
1
C^i, {N_1}_{R^i}, MAC(ID_A, ID_S, N_1)
R^i = P_A(C^i)
Obtain N_1 using R^i
Verify MAC
M_A = H(ID_A || Data || R^i)
2
M_A, MAC(ID_A, ID_S, M_A, N_1, R^i)
Verify MAC
Add M_A to Blockchain

Figure 4: The PUF challenge-response protocol.

5 PERFORMANCE AND SECURITY ANALYSIS

With BlockPro, a decentralized and trust-free operation of IoT devices is obtained. It is able to provide defense against impersonation and data tampering attacks. Instead of the conventional centralized IoT architecture as illustrated in Figure 1, in which there is a single server, BlockPro has decentralized architecture. Moreover, the ability to deploy smart contracts ensures that the devices operating are registered and trusted ones.

The advantages of the BlockPro architecture can be reflected in the following ways:

5.1 Centralized v/s decentralized architecture

Figure 1 shows a traditional, centralized IoT architecture. A centralized architecture design is prone to single-point-of-failure problems which can possibly bring down the whole system. Moreover, computations are not distributed but mainly concentrated in a centralized fashion in the network. BlockPro addresses these issues by providing a decentralized platform. By using a blockchain as its platform, the computation is distributed among the constituents of the network. This eliminates single-point-of-failure problems, empowers the system to withstand them and to continue operating even if the constituent(s) fall down.

5.2 Data provenance and data integrity

The proposed framework uses PUFs to establish the source of the data. As each PUF produces a unique response, therefore, data provenance is established through the use of PUFs with each IoT device. In addition, the use of the blockchain platform enforces data integrity. Blockchain provides an immutable chain of records i.e. starting from the first block, the subsequent ones are added in a chronological order. To change one block, one must trace it back to the first one which is practically infeasible. The proposed framework has the following salient features:

(1) Each IoT device has a unique ID relative to its PUF. This provides immunity from impersonation attacks, thereby providing an effective way for establishing data provenance.
(2) The smart contract coded for BlockPro provides a safe and secure mechanism for the transmission, authentication and storage of requests and data, respectively.
(3) With an ever-growing chain of records, all the data is validated first and then stored on the blockchain permanently which cannot be tampered with afterwards, thereby providing and preserving data integrity.

5.3 Distributed consensus

Traditional IoT system design relies profusely on trust because it is one of the enabling factors of system operation. Typically, due to centralized structures, there is a third party involved between an IoT device and a server. This third party may be a storage solution, an entity providing computational power or other forms of service.

Although the notion of service is lucrative, it does not come for free. The inclusion of third parties involves extra time and labor along with an associated monetary cost.

BlockPro eliminates the need for third parties by distributing computation and consensus among the participants of the network. Not only are they responsible for providing the necessary computational power for the network to operate, but they also provide a trust free environment using distributed consensus protocols. The distributed consensus protocol used by BlockPro is proof-of-work (PoW). This cuts down the extra cost and time labor associated with third parties and puts the control back into the hands of the network constituents.

5.4 Defense against botnets and bogus requests

Compromised IoT devices may operate as botnets and/or rogue devices. These devices are usually infected with malicious software (malware). Botnets can in turn be used to orchestrate a range of cyber attacks such as Denial of Service (DoS), and Distributed Denial of Service (DDoS) etc. Attacks can also come in forms of bogus requests and/or other forms of requests. The objective of the attack is to exhaust the system of its resources. It is usually hard to prevent such attacks once they have launched, therefore proactive measures are better than reactive ones. BlockPro addresses this issue by registering each IoT device in the network first with its unique PUF CRP and by maintaining a list of trusted devices. This way botnets and rogue devices are blocked out from the system since they are not registered and their requests will not be entertained.

6 IMPLEMENTATION AND EVALUATION

For evaluation and proof-of-concept purposes, two smart contracts of Ethereum were custom coded to create the BlockPro framework. The IoT devices are assumed to be embedded with PUFs and their respective CRPs are stored in the smart contract. Furthermore, to validate and evaluate BlockPro, simulations were conducted with IoT nodes and a server node on Ethereum.

6.1 Setup

Ubuntu 17.04 (Linux) was used as the operating system for the BlockPro simulation environment. Ethereum Go client *geth* was used for initializing two Ethereum IoT nodes ID_A, ID_B and a server node ID_S. Separate Ethereum accounts were also created for the nodes so that they can interact with each other through the smart contracts.

6.2 Initializing nodes

Terminal offered by Ubuntu OS as a Linux working environment, was used for simulating the Ethereum nodes. The nodes were simulated according to Algorithm 2. The server ID_S includes the genesis (first) block definition of BlockPro and by interacting with the IoT devices ID_A and ID_B, it grows with succeeding blocks added together chronologically with the genesis block.

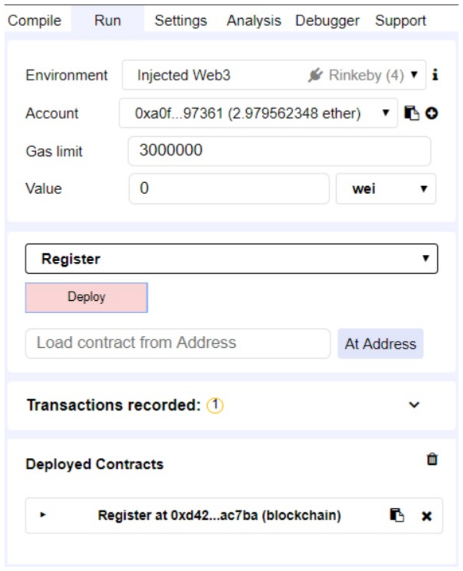
6.3 Smart contracts execution flow

After the nodes are initialized, the smart contracts need to be deployed. Both smart contracts 1 and 2 are deployed on the server node ID_S which will register the address of ID_S as the server for the BlockPro simulation purposes. Before the deployment of contracts, it is essential that they be compiled first.

The operational flow of the smart contracts is as follows:

6.3.1 Compilation. The contracts were compiled using the online Solidity IDE, ***Remix***. The output of Remix can be seen in Figure 5. After the contracts are compiled with the output variables, it can be deployed on ID_S.

6.3.2 Deployment. With both the contracts compiled, they are deployed on the server node ID_S. This enables ID_S to identify the contracts using their addresses. Subsequently, ID_S broadcasts the address of only SmartContract_1 to the whole BlockPro network to enable interactions and communication among its constituents (IoT devices and the distributed servers). It is noteworthy that the address of SmartContract_2 is not broadcasted because it contains a private and not a public function that can only be called and accessed by SmartContract_1.



Environment Injected Web3 Rinkeby (4)
Account 0xa0f...97361 (2.979562348 ether)
Gas limit 3000000
Value 0 wei
Register
Load contract from Address
Transactions recorded: 1
Deployed Contracts
Register at 0xd42...ac7ba (blockchain)

Figure 5: An output view of online Solidity IDE, *Remix*.

6.4 Evaluation

With the nodes initialized and the smart contracts deployed, interactions between IoT devices ID_A and ID_B and the server ID_S are now possible. The server ID_S is responsible for registering and deleting the IoT devices. For evaluation, both the device nodes ID_A and ID_B are registered with ID_S with their respective PUF CRPs. This way ID_S has two devices registered with it and their addresses are stored in the trusted list of devices in SmartContract_1.

To upload data, ID_A or ID_B has to call $data.tx()$ function that allows them to send data to the server as shown in Figure 3. The data will only go through if the following two conditions are met:

i. If the device is registered.
ii. If the device can successfully complete the PUF challenge-response protocol.

Algorithm 2: Initializing IoT device and server nodes

```
procedure INIT($ID_S$, $ID_A$, $ID_B$)
    $ID_S$ INITIALIZATION // server with first block
    BlockPro.json ← DEFINE // genesis block
    $ID_S$ ← CREATE NODE // where $S \geq 1$
    $ID_S$ ← MAKE ACCOUNT // outputs address
    $ID_S$.account ← SIGN // with private key
    $ID_S$.account ← ALLOCATE SOME ETHER
    repeat $ID_S$ INITIALIZATION // for $S$ servers
    $ID_A$ INITIALIZATION // IoT device: 1
    $ID_A$ ← CREATE NODE
    $ID_A$ ← MAKE ACCOUNT // outputs address
    $ID_A$.account ← SIGN
    $ID_B$ INITIALIZATION // IoT device: 2
    $ID_B$ ← CREATE NODE
    $ID_B$ ← MAKE ACCOUNT // outputs address
    $ID_B$.account ← SIGN
    $ID_S$, $ID_A$ and $ID_B$ ← RUN
    $ID_S$ ← SMART CONTRACT // deploy
    $ID_A$ AND $ID_B$ with $ID_S$ ← INTERACT // via smart
contracts
end procedure
```

If a device fails any check, the communication link is then terminated between it and the server. In contrast, if a device passes these checks, the SmartContract_1 calls the SmartContract_2 to entertain its request; be it storing or fetching the data.

7 RELATED WORK

IoT has two major requirements in terms of security: *trust* and *integrity* which are difficult to achieve given its low-level architecture design and relatively simpler specifications of its devices.

Some of the recent works on data provenance include the following: the authors in [3] propose a physical unclonable function (PUF) enabled solution with received signal strength indicator (RSSI) to establish secure data provenance. However, it does not address safe data storage/retrieval concerns. Provenance based trust management solution is proposed by the authors of [6] which helps in providing data provenance. However, this technique relies on a memory intensive model. Furthermore, [2] presents an attractive choice of authentication with low overhead for IoT devices but fails to provide a secure storage platform and requires a high message exchange rate for the authentication to work. Finally, [8] presents a privacy preserving data provenance solution for IoT but its design relies on the trust of the server itself. Moreover, the aforementioned system designs are centralized in nature and require trust to be established first and foremost.

The proposed framework, BlockPro, removes the need of trust in its design since by nature, it is decentralized and employs a trust-free operation. Furthermore, for it to operate in IoT environments, it does not require the IoT devices to get a substantial hardware upgrade because it only needs them to be registered in the network. Finally, the proposed system provides an integrated solution for providing data provenance along with data integrity for IoT infrastructures, thereby protecting both the devices and their data whilst safeguarding them from adversaries.

8 CONCLUSION

This paper presented a PUF and blockchain based solution called BlockPro for data provenance and data integrity for secure IoT environments. The use of PUFs gives each IoT device a unique hardware fingerprint which is exploited to establish data provenance. Moreover, the decentralized approach for data storage and retrieval using blockchain forms the basis for data integrity. Ethereum and two custom coded smart contracts were used to implement the proposed framework. BlockPro can be used as an effective solution to provide both data provenance and data integrity in IoT environments for their secure and reliable operation.

ACKNOWLEDGMENTS

This work is supported by the National University of Singapore under Grant No.: R-263-000-C50-133 and by the National Research Foundation, Prime Minister's Office, Singapore, under its Corporate Laboratory@University Scheme, National University of Singapore and Singapore Telecommunications Ltd.

REFERENCES

[1] Muhammad Naveed Aman, Kee Chaing Chua, and Biplab Sikdar. 2016. Physical Unclonable Functions for IoT Security. In *Proceedings of the 2nd ACM International Workshop on IoT Privacy, Trust, and Security (IoTPTS '16)*. ACM, New York, NY, USA, 10–13. DOI:http://dx.doi.org/10.1145/2899007.2899013

[2] M. N. Aman, K. C. Chua, and B. Sikdar. 2017. Mutual Authentication in IoT Systems Using Physical Unclonable Functions. *IEEE Internet of Things Journal* 4, 5 (Oct 2017), 1327–1340. DOI:http://dx.doi.org/10.1109/JIOT.2017.2703088

[3] Muhammad Naveed Aman, Kee Chaing Chua, and Biplab Sikdar. 2017. Secure Data Provenance for the Internet of Things. In *Proceedings of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security (IoTPTS '17)*. ACM, New York, NY, USA, 11–14. DOI:http://dx.doi.org/10.1145/3055245.3055255

[4] Vitalik Buterin. 2014. Ethereum: A next-generation smart contract and decentralized application platform. https://github.com/ethereum/wiki/wiki/White-Paper. (2014). https://github.com/ethereum/wiki/wiki/White-Paper Accessed: 2016-08-22.

[5] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, Dawn Song, and Roger Wattenhofer. 2016. On Scaling Decentralized Blockchains. (02 2016).

[6] M. Elkhodr, B. Alsinglawi, and M. Alshehri. 2018. Data Provenance in the Internet of Things. In *2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA)*. 727–731. DOI:http://dx.doi.org/10.1109/WAINA.2018.00175

[7] Satoshi Nakamoto. 2009. Bitcoin: A peer-to-peer electronic cash system. (03 2009).

[8] J. L. C. Sanchez, J. B. Bernabe, and A. F. Skarmeta. 2018. Towards privacy preserving data provenance for the Internet of Things. In *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*. 41–46. DOI:http://dx.doi.org/10.1109/WF-IoT.2018.8355229

[9] J. A. Stankovic. 2014. Research Directions for the Internet of Things. *IEEE Internet of Things Journal* 1, 1 (Feb 2014), 3–9. DOI:http://dx.doi.org/10.1109/JIOT.2014.2312291

[10] H. Suo, J. Wan, C. Zou, and J. Liu. 2012. Security in the Internet of Things: A Review. In *2012 International Conference on Computer Science and Electronics Engineering*, Vol. 3. 648–651. DOI:http://dx.doi.org/10.1109/ICCSEE.2012.373

[11] S. Verma, Y. Kawamoto, Z. M. Fadlullah, H. Nishiyama, and N. Kato. 2017. A Survey on Network Methodologies for Real-Time Analytics of Massive IoT Data and Open Research Issues. *IEEE Communications Surveys Tutorials* 19, 3 (thirdquarter 2017), 1457–1477. DOI:http://dx.doi.org/10.1109/COMST.2017.2694469

[12] T. Xu, J. B. Wendt, and M. Potkonjak. 2014. Security of IoT systems: Design challenges and opportunities. In *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 417–423. DOI:http://dx.doi.org/10.1109/ICCAD.2014.7001385