Mitigating IoT Device based DDoS Attacks using Blockchain

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

Ang Kiang Siang
National University of Singapore
Singapore
a0125528@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

Many IoT devices lack memory and computational complexities of modern computing devices, making them vulnerable to a wide range of cyber attacks. Among these, DDoS attacks are a growing concern in IoT. Such attacks are executed through the introduction of rogue devices and then using them and/or other compromised devices to facilitate DDoS attacks by generating relentless traffic. This paper aims to address DDoS security issues in IoT by proposing an integration of IoT devices with blockchain. This paper uses Ethereum, a blockchain variant, with smart contracts to replace the traditional centralized IoT infrastructure with a decentralized one. IoT devices are then required to access the network using smart contracts. The integration of IoT with Ethereum not only prevents rogue devices from gaining access to the server but also addresses DDoS attacks by using static resource allocation for devices.

CCS CONCEPTS

• **Networks** → **Denial-of-service attacks**; • **Computer systems organization** → *Embedded and cyber-physical systems*;

KEYWORDS

Internet of Things (IoT), Distributed Denial of Service (DDoS), Smart Contract, and Ethereum.

ACM Reference Format:
Uzair Javaid, Ang Kiang Siang, Muhammad Naveed Aman, and Biplab Sikdar. 2018. Mitigating IoT Device based DDoS Attacks using Blockchain. In *CryBlock'18: 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems , June 15, 2018, Munich, Germany*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3211933.3211946

1 INTRODUCTION

The Internet of Things (IoT) broadly refers to the integration of physical devices that can operate, actuate, and communicate autonomously to optimize and enable new services in a wide range of areas. With recent developments in device fabrication and communication technologies, the IoT is expected to facilitate pervasive sensing and efficient resource management in applications such as smart power grids, intelligent spaces, smart cities, industry automation, health care, etc. to name a few [13, 15]. A high-level overview of IoT based systems is shown in Figure 1 where different IoT devices (e.g. sensors and actuators) communicate with a centralized server through a communication network. It is estimated that by 2020, between 50-100 billion things (objects) will be connected to the Internet [9]. Due to the large size of the network and the sensitive nature of the data these devices produce, security is a serious concern for a number of IoT applications. Many IoT devices are resource constrained in terms of energy, memory and computational resources which exacerbate the security and architectural challenges [13, 14, 16]. For example, the computational complexity of classical security techniques makes them unsuited for IoT devices. Thus, in order to enable secure communication between a large number of devices, new security techniques and protocols are required that can cater to the specifications of conventional as well as new IoT devices.

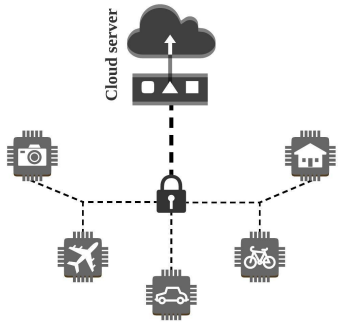

Figure 1: A conventional IoT architecture.

One of the major security concerns that needs to be addressed in IoT systems is distributed denial of service (DDoS) attacks. Such attacks are usually targeted at Internet-based services, financial institutions, and broadcasting networks. In DDoS attacks involving IoT devices, the devices may be used to overload the resources of a network or targeted computing device. Recently, DDoS attacks that exploit the simple service discovery protocol (SSDP) have increased

CryBlock'18, June 15, 2018, Munich, Germany

ACM ISBN 978-1-4503-5838-5/18/06…$15.00
https://doi.org/10.1145/3211933.3211946

because of the vulnerabilities present in it [12]. Many IoT devices (e.g. closed-circuit television (CCTV) cameras, wireless routers, and IP based devices) use this protocol. This puts them at high risk of being used for such attacks. IoT devices were widely used in the DDoS attack on the Dyn domain name server (DNS) servers in October 2016 [10].

This paper explores the use of blockchains for providing security solutions for the IoT, with particular emphasis on DDoS attacks. A blockchain is an online distributed ledger consisting of a list of blocks. Each block is an ordered record of a timestamp and a hash of the previous block, making blockchains highly resistant to data modification. Secondly, blockchains use online distributed ledgers to allow transparency in every transaction. Finally, blockchains use a decentralized approach for storing data all across the network, resulting in increased robustness. Therefore, many applications have adopted blockchains to provide decentralized and trust-free solutions. Moreover, blockchains are most widely used in cryptocurrencies such as Bitcoin [8]. Another use for blockchains is for *smart contracts* which are self-executable computer programs such as Ethereum [4]. Blockchains are increasingly being used for other applications as well. For example, [20] proposed a blockchain based IoT E-business model and in the recent past, more than 200 variants of blockchains have been proposed [6]. One of the features of blockchains that makes it an attractive choice for IoT systems is that it allows devices to freely transact without relying on third parties [8]. However, Blockchains are criticized for their scalability problems [5].

A typical IoT device-to-server communication infrastructure typically involves varying levels of security features and security capabilities on the IoT devices, the network, and the cloud server(s). This paper takes on a new perspective, to introduce blockchains as a defense mechanism against rogue physical devices and DDoS attacks they may launch by integrating the same IoT device-to-server communication architecture with the Ethereum blockchain. The solution proposed in this paper provides the following security and architectural properties:

(i) A blockchain based framework for detecting and preventing DDoS attacks in IoT devices.
(ii) A distributed framework for control and enabling trust-free operation of IoT devices.
(iii) Ability to integrate legacy IoT devices with very low computational capabilities.

This paper is organized as follows: Section 2 describes the network and threat models. Section 3 presents the proposed system design and Section 4 presents the security and performance analysis. Section 5 describes the implementation and evaluation results while Section 6 and Section 7 present the existing related work and conclusion to this paper, respectively.

2 NETWORK MODEL, ASSUMPTIONS, AND THREAT MODEL

2.1 Network model

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

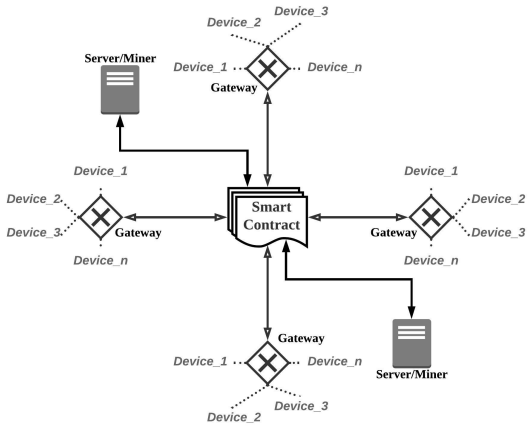

Figure 2: The IoT-Blockchain system model.

- **Device:** Devices represent "things" (IoT nodes) which are responsible for sensing, processing, and communicating data to the server through a gateway. A common gateway can be used to connect multiple, say n, devices. Each device has a *gas limit* (see Section 3.1 for a definition) which is equal to or lesser than the total load the IoT-Ethereum network can handle.
- **Gateway:** This represents a common gateway that can be used by a cluster of devices for communication purposes. The gateway provides network connectivity to the devices in its neighborhood and may provide additional functionalities such as data aggregation and security features. Different gateways may be used for different types of devices (for example, a gateway for temperature sensing devices), or a single gateways may be used for a range of devices.
- **Smart contract:** This is the regulatory body of the system which is responsible for authorizing the devices and ensuring that they do not operate beyond their gas limit. Its operation is discussed in detail in Section 3.
- **Server/Miner:** The IoT-Ethereum network includes multiple volunteers acting as the servers/miners. These entities are responsible for verifying the transactions and data exchange through smart contracts using high computational and processing capabilities.

2.2 Assumptions

We make the following assumptions regarding the network model and the proposed framework:

a. IoT devices are considered to be resource constrained with limited power, memory, and processing capabilities. They are also considered safe against physical attacks and spoofing.
b. The servers are not constrained in terms of resources.
c. Ethereum loss during a transaction is not possible.
d. DDoS attacks are not possible on Internet hosts and blockchain verifiers (miners).

2.3 Threat model

The objective of the attacker is to overload and flood the servers with data traffic, in turn causing outages. We assume that the adversary is capable of compromising the IoT devices and make them send arbitrary amount to data to the target of the DDoS attack. In this paper, the attacker is assumed to use the IoT devices to launch DDoS attacks on the servers to which they send their data. The proposed solution may be extended to the scenario where the IoT devices are used to target any arbitrary server. In our model, we assume that the gateways are secure and cannot be compromised. In addition, we assume that the adversary cannot compromise the blockchain.

3 PROPOSED SYSTEM DESIGN

The attacks of interest in this paper are denial of service and distributed denial of service attacks, i.e., how IoT devices can contribute and facilitate in orchestrating DoS/DDoS attacks. Both attacks are assumed and a framework based on the Ethereum blockchain is proposed as a means of defense.

3.1 The IoT-Blockchain model

Ethereum is one of the largest online established software platforms that allows smart contracts and decentralized applications (DApps) to be built on blockchains along with their state. The state is composed of objects called accounts which have the following fields [4]:

(1) A 20-byte address.
(2) A nonce so each transaction is processed only once.
(3) A balance of Ether or the internal numbers used to pay fees.
(4) A contract code that may be empty.
(5) Storage, which may also be empty.

State in Ethereum refers to the data present in the blockchain and a state transition occurs when a transaction happens. Moreover, there are two types of accounts:

a. Externally owned accounts (EOAs): This is controlled by a private key and the user who owns it can send or receive messages from it.
b. Contract: This is a computer program and its corresponding account has its own code and is controlled by the same.

A gas limit exists for each transaction and process within Ethereum. Gas is an analogous word for "resource" in Ethereum terms, i.e., a certain amount of gas for a function means that the execution of the function has that much resource. Secondly, IoT devices have to use gas for their operation. It can be interpreted as a cost factor for the IoT devices but it also ensures security by causing adversaries to consume their resources instead if they attack the system.

The operation of the proposed model relies on Ethereum for generating addresses for the devices and server/miner nodes, and implementing our custom coded smart contract. The contract is coded to differentiate between trusted and untrusted devices. Devices are required to be first registered with the system and provided with a specific gas limit. The registration will generate an account with a unique address for each device and gas limit for a device is related to their specifications, i.e., it depends on the bandwidth and resource requirements of the device. Interactions between devices and servers are enabled by using the smart contract governed by the server node.

3.2 Overview of the smart contract

The smart contract in the system is responsible for facilitating secure communication between the IoT devices and the distributed servers as illustrated in Figure 3. It is developed using Solidity, which is a contract-oriented, high-level language for the Ethereum virtual machine environment. It has two phases of operation: Initialization and Deployment.

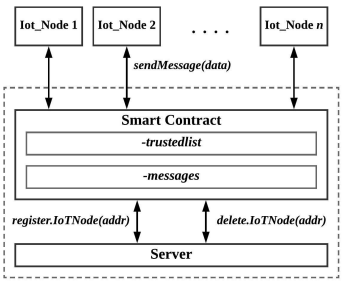

Figure 3: System components with information flow layout.

3.2.1 Initialization: In this phase, a server node deploys the smart contract, which will then be known as the *server* variable by the contract to recognize the trusted host. The paper assumes that servers are the trusted hosts. The smart contract address will then be broadcasted among the IoT nodes for them to engage with the contract instance. The gas limit for each transaction in the contract is set in this phase to defend against DDoS attacks.

3.2.2 Deployment: In this phase, IoT nodes will communicate and interact with the server node which deployed the smart contract so that they can get registered. Only the server can authorize devices to get registered or deleted. Upon receiving confirmation from the server, an IoT node address will be registered and kept in the smart contract list of trusted devices. If the server suspects a node to be malicious or if it is needed to be removed, it can call the delete function which will remove that particular node.

3.3 System operation

All IoT devices in the IoT-Ethereum system are able to call the smart contract to send a message. The message will be sent through and stored in the blockchain for retrieval only if the IoT node is granted access. The IoT node is checked with the list of authorized addresses by the smart contract and is passed as a trusted device if stated so in the contract. If the node fails to get access, the message(s) and interaction(s) will be dropped and rendered void. The algorithm used for the operation of smart contracts is detailed in Algorithm 1. In this algorithm, access function means if a device will be blocked or given access to the system while messages refer to data generated by the IoT devices in the system.

Algorithm 1: Smart contract algorithm for validation

```
function access(device_i)
Input  : message(device_i)
Output : trusted, untrusted
if (message(device_i) exists and message(device_i) is valid) then
    // Check device_i is trusted/untrusted
    if (device_i is registered in the trusted list) then
        // Check device_i has good gas limit
        if (device_i.gas.used ≤ gas.limit) then
            return trusted
        else
            return untrusted
        end
    else
        return untrusted
    end
else
    return untrusted
end
end function
```

4 PERFORMANCE AND SECURITY ANALYSIS

The ability to create and deploy smart contracts in an IoT-Ethereum network removes the possibility of downtime and censorship by avoiding a single point-of-failure (attributed to a centralized server approach). Thus, the proposed IoT-Ethereum framework can be a potential solution to authentication, trust, and single point-of-failure problems that IoT based systems are currently facing. The advantages of such a system architecture can be reflected in the following ways.

4.1 Centralized versus distributed servers

Conventional network designs use a centralized server for the integration of IoT devices, commonly referred to as the cloud server as illustrated in Figure 1. The cloud server handles and is solely responsible for all the computation and decision making tasks. Note that while the cloud server may in reality be replicated for redundancy, for the purposes of authentication and decision making, the system can be considered as a single entity. This centralized approach leads to many security issues. For example, if an adversary can gain access into this server, the whole infrastructure of the system can be compromised.

The proposed IoT-Ethereum framework eliminates this problem by distributing control and trust, i.e., multiple nodes (participants) are responsible for the sustainability, reliability, and proper functioning of the network resulting in high security fidelity. The proposed framework can be considered decentralized in two ways. First, the computational requirements for running the blockchain are distributed among all the operating nodes. Second, instead of trust established by third parties, a consensus mechanism, proof-of-work (PoW), is used among the nodes in the network [4]. Thus, if one or more node(s) fail, the system is not compromised as it is an adversary versus everyone else in the system.

4.2 Blocking rogue devices

Using a list of authorized devices maintained on the smart contract of the IoT-Ethereum model, the contract will authorize each device. When a device calls for a function, it is checked against the list of authorized device addresses by the contract and granted access only if it is registered in the list. If the device fails to get access or is not in the list, all messages (data) and interactions with it will be dropped and rendered void.

4.3 Defense against DDoS attacks

The DDoS problem addresses in this paper is that one or more devices are sending extremely large quantities of data to overload a server and consume its resources. In our system model in Figure 2, any of the n devices may all start continuously uploading data to the server and effect a DDoS attack. Such attacks can be prevented with the gas limit attribute of Ethereum as it ensures no further resources can be consumed once the limit is exceeded. In the proposed smart contract, a gas limit is set for each transaction processed through it which acts as a mechanism to prevent the system from overloading. Let us consider n IoT devices, each having a gas limit of g_i. If the maximum bandwidth available at the server is $\boldsymbol{B}$, then we have

$$\sum_{i=1}^{n} g_i \leq \boldsymbol{B}. \tag{1}$$

Equation (1) indicates that even if all the devices start sending data at their gas limit at the same time, the server bandwidth still cannot be exhausted. Moreover, any DoS/DDoS attack that intends to consume the server resources will first require the malicious device to consume its own resources (gas) first until they are exhausted. This will terminate the malicious device's operation (in terms of its packets being forwarded to the server) once the gas limit is reached and prevent the server from overloading. The gas limit can be set as desired and each device has its limit defined when it registers in the IoT-Ethereum network.

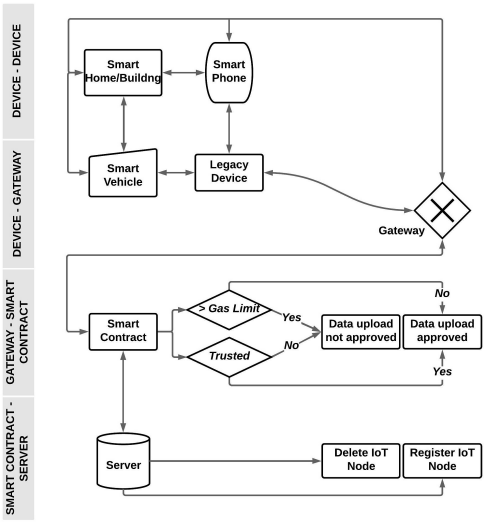

Figure 4: Flow diagram of the IoT-Blockchain network.

4.4 Trust-free system operation

Conventional IoT system designs rely heavily on trust since it is one of the key security requirement. Trust is typically established with the help of third parties which work as intermediaries between two devices. These third parties have their own fee and associated costs in terms of latency and labor costs. IoT-Ethereum eliminates this need since it does not need any third party to guarantee its operation [3]. A consensus protocol is used instead where different devices come to terms with each other directly and thus, a trust-free system operation is realized.

5 IMPLEMENTATION AND EVALUATION

For evaluation and proof-of-concept purposes, a smart contract of Ethereum was custom coded to create the IoT-Ethereum framework. Subsequently, simulations were conducted to validate the interactions between an IoT device and a server node. The server node is also used for registering/deleting devices. The code for the smart contract can be found at [1].

5.1 Setup

Linux (Ubuntu 16.04) was used as the operating system for the simulation environment. The Ethereum development package was used to setup the nodes using the Ethereum Go client (geth). Separate accounts were used to simulate the server and IoT devices communicating with each other using the smart contract.

5.2 Setting up the Nodes

The Linux platform that Ubuntu OS offers through its **Terminal** was used. The nodes were simulated according to Algorithm 2.

5.3 Operational Flow of the Smart Contract

With the two types of nodes set up (server and IoT), the smart contract is deployed on the server node such that the contract will register the address of this node as the server for this IoT system. First, the smart contract has to be compiled before deployment. The flow of operation of the smart contract is as follows.

5.3.1 Compilation. For compiling the smart contract, the online browser Solidity compiler was used. The output of the compiler is shown in Figure 5. The variables given in the Web3 deploy are executed in the geth terminal of the server node.

5.3.2 Deployment. With the contract compiled, it is then deployed on the server node and proceeded with mining for an address to be used for interacting with the contract instance. The contract address is then broadcasted to the IoT devices to allow them to interact with the contract. The contract application binary interface (ABI) will be required and can be obtained from the solidity compiler. A monitoring function in the contract allows to watch over all logs at specific events of the contract such that necessary information will be outputted accordingly.

5.3.3 Operation. With the contract deployed and set up on both the server and the IoT devices, interaction is possible with the contract from the devices to simulate IoT-Server interactions. Only a server node can successfully execute the function for registering and removing IoT devices as the contract will authenticate the transaction sent for that function with the credentials of the server. These are the functions used to maintain a list of trusted IoT devices in the contract to perform authentication. The flow diagram of IoT-Ethereum network interactions is presented in Figure 4.

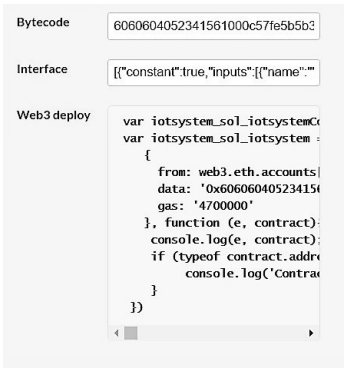

Figure 5: Output console of browser Solidity.

Algorithm 2: Instantiation of IoT devices and servers/miners

```
1:  procedure SETUP(device_i, server_j)
2:      DEVICE INITIALIZATION // users
3:      genesisfile(xxx.json) ← DEFINE // gas limit
4:      device_i ← CREATE NODE // where i ≥ 1
5:      device_i ← MAKE ACCOUNT // outputs address
6:      device_i.account ← SIGN // with private key
7:      device_i.account ← ALLOCATE SOME ETHER
8:      repeat DEVICE INITIALIZATION // for i devices
9:      SERVER INITIALIZATION // miners
10:     server_j ← CREATE NODE // where j ≥ 1
11:     server_j ← MAKE ACCOUNT // outputs address
12:     server_j.account ← SIGN
13:     repeat SERVER INITIALIZATION // for j servers
14:     device_i and server_j ← RUN
15:     server_j ← SMART CONTRACT // deploy
16:     device_i with server_j ← INTERACT // via contract
17: end procedure
```

Authorization is done by the contract when IoT devices call the *sendMessage*() function that allows them to send in data to the server as shown in Figure 3. The data will only go through if an IoT node is authenticated successfully. Otherwise, it will be dropped preventing unauthorized devices from accessing the server.

In the case where an authorized device becomes compromised and exhibits malicious behavior by trying to send a large amount of data to the server to attempt a DoS attack, its requests will be terminated due to exhaustion of the gas limit in the Ethereum framework. This will protect the server from exhausting its resources to handle such malicious activity as the contract executed for this large data transaction will be consuming the gas from the malicious node itself. Moreover, mining of blocks for the transactions to be stored in the blockchain is performed on the server/miner nodes, allowing distributed computation which effectively reduces load on all

servers. This is also a good mitigation strategy to reduce malicious activities since the consumption of the resources is on the attacker's end instead. Besides, all transactions will have a transaction receipt tagged with the sender's address and stored in the blockchain, allowing for future references while ensuring non-repudiation of activities performed. This helps in identifying anomalies in IoT devices and single out possible compromised nodes.

6 RELATED WORK

Research in various areas related to IoT has received considerable attention recently. The author of [13] highlights various challenges in IoT based systems and identifies the following areas for research: scaling, architecture and dependencies, creating knowledge and big data, robustness, openness, security, privacy, and human-in-the-loop. IoT has two major requirements in terms of security: *trust* and *control* which are difficult to achieve given the large size of the network. Although public key infrastructure (PKI) techniques have proven themselves for large-scale systems (e.g. global payments system) as a security solution, key management in an IoT environment may not be feasible due to resource constraints.

One of the fundamental challenges in the Internet today is that of security [11, 17], which also extends to the IoT. Among the various security threats faced by IoT devices and systems, DDoS attacks pose a significant threat to the IoT infrastructure. Such attacks are typically executed using malicious software (malware), such as Mirai, which exploit vulnerabilities of networked devices and turns them into remotely controlled devices, *bots*. Bots are then used to create a *botnet* which can be used for cyber attacks including DDoS attacks. A large body of work for defending against DDoS attacks in the Internet is mainly based on the use of firewalls, filters, and hardware devices that monitor network activity [18]. Specific to IoT, the authors in [7] present *AntibIoTic* for protection of IoT devices from DDoS attacks. Their solution is limited since it is centralized and it requires patching vulnerable IoT devices first which incurs additional computation and time costs. The authors in [19] propose an algorithm which can help avoid DDoS attacks. Their algorithm provides distinction between malicious and legitimate requests, analogous to a firewall, and processes them differently. Although the algorithm provides useful distinction, it requires considerable processing which makes it less efficient for large-scale systems. Another DDoS detection technique is proposed in [2] but it is only limited to detection and no preventive measures are provided.

The proposed IoT-Ethereum framework removes the need for PKI in its design since it is a trust-free system. Moreover, it does not require the IoT devices to get a hardware upgrade. Rather, it can be implemented on top of a conventional system as an overlay network. Finally, the proposed system provides an integrated solution for detection and prevention of DDoS attacks.

7 CONCLUSION

This paper presented an IoT device and server communication framework on Ethereum using a customized smart contract which enables a better defense mechanism against DDoS and rogue device attacks. The proposed system is able to provide distinction between trusted and untrusted devices and allocates a static resource limit to each device above which it cannot operate.

ACKNOWLEDGMENTS

This work is supported in part by the National University of Singapore under Grant No.: R-263-000-C50-133.

REFERENCES

[1] [n. d.]. Secure Contract Source Code, howpublished = http://https://www.ece.nus.edu.sg/stfpage/bsikdar/scripts/contracts/, note = Accessed: 04-02-2018. ([n. d.]).
[2] Talal Alharbi, Ahamed Aljuhani, Hang Liu, and Chunqiang Hu. 2017. Smart and Lightweight DDoS Detection Using NFV. In *Proceedings of the International Conference on Compute and Data Analysis (ICCDA '17)*. ACM, New York, NY, USA, 220–227. https://doi.org/10.1145/3093241.3093253
[3] Roman Beck, Jacob Stenum Czepluch, Nikolaj Lollike, and Simon Malone. 2016. Blockchain – The Gateway to Trust-Free Cryptographic Transactions. In *Twenty-Fourth European Conference on Information Systems (ECIS), Istanbul, Turkey, 2016*. Springer Publishing Company, United States, 1–14. ISBN to be added later.
[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), 106-125 pages.
[6] Tien Tuan Anh Dinh, Ji Wang, Gang Chen, Rui Liu, Beng Chin Ooi, and Kian-Lee Tan. 2017. BLOCKBENCH: A Framework for Analyzing Private Blockchains. In *Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD '17)*. ACM, New York, NY, USA, 1085–1100. https://doi.org/10.1145/3035918.3064033
[7] Michele De Donno, Nicola Dragoni, Alberto Giaretta, and Manuel Mazzara. 2017. AntibIoTic: Protecting IoT Devices Against DDoS Attacks. *CoRR* abs/1708.05050 (2017).
[8] Satoshi Nakamoto. 2009. Bitcoin: A peer-to-peer electronic cash system. (03 2009).
[9] Charith Perera, Rajiv Ranjan, Lizhe Wang, Samee Ullah Khan, and Albert Y. Zomaya. 2014. Privacy of Big Data in the Internet of Things Era. *CoRR* abs/1412.8339 (2014). arXiv:1412.8339 http://arxiv.org/abs/1412.8339
[10] Nicole Perlroth. 2016. Hackers use new weapons to disrupt major webites across US. *The New York Times* 21 (2016).
[11] S. Ravi, A. Raghunathan, and S. Chakradhar. 2004. Tamper resistance mechanisms for secure embedded systems. In *17th International Conference on VLSI Design. Proceedings.* 605–611. https://doi.org/10.1109/ICVD.2004.1260985
[12] Fabrice J. Ryba, Matthew Orlinski, Matthias Wählisch, Christian Rossow, and Thomas C. Schmidt. 2015. Amplification and DRDoS Attack Defense - A Survey and New Perspectives. *CoRR* abs/1505.07892 (2015).
[13] J. A. Stankovic. 2014. Research Directions for the Internet of Things. *IEEE Internet of Things Journal* 1, 1 (Feb 2014), 3–9. https://doi.org/10.1109/JIOT.2014.2312291
[14] 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. https://doi.org/10.1109/ICCSEE.2012.373
[15] 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. https://doi.org/10.1109/COMST.2017.2694469
[16] 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. https://doi.org/10.1109/ICCAD.2014.7001385
[17] Wenyuan Xu, Wade Trappe, Yanyong Zhang, and Timothy Wood. 2005. The Feasibility of Launching and Detecting Jamming Attacks in Wireless Networks. In *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '05)*. ACM, New York, NY, USA, 46–57. https://doi.org/10.1145/1062689.1062697
[18] Saman Taghavi Zargar, James Joshi, and David Tipper. 2013. A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks. *IEEE communications surveys & tutorials* 15, 4 (2013), 2046–2069.
[19] Congyingzi Zhang and Robert Green. 2015. Communication Security in Internet of Thing: Preventive Measure and Avoid DDoS Attack over IoT Network. In *Proceedings of the 18th Symposium on Communications & Networking (CNS '15)*. Society for Computer Simulation International, San Diego, CA, USA, 8–15. http://dl.acm.org.libproxy1.nus.edu.sg/citation.cfm?id=2872550.2872552
[20] Yu Zhang and Jiangtao Wen. 2017. The IoT electric business model: Using blockchain technology for the internet of things. *Peer-to-Peer Networking and Applications* 10, 4 (2017), 983–994.