Security Implications of Blockchain Cloud with Analysis of Block Withholding Attack

Deepak K. Tosh[1], Sachin Shetty[2], Xueping Liang[3], Charles A. Kamhoua[4], Kevin A. Kwiat[4], Laurent Njilla[4]
[1]Department of Computer Science, Norfolk State University, Norfolk, VA
[2]Virginia Modeling Analysis and Simulation Center, Old Dominion University, Norfolk, VA
[3]College of Engineering, Tennessee State University, Nashville, TN
[4]Cyber Assurance Branch, Air Force Research Laboratory, Rome, NY
dktosh@nsu.edu, sshetty@odu.edu, xliang@tnstate.edu, {charles.kamhoua.1, kevin.kwiat, laurent.njilla}@us.af.mil

***Abstract*—The blockchain technology has emerged as an attractive solution to address performance and security issues in distributed systems. Blockchain's public and distributed peer-to-peer ledger capability benefits cloud computing services which require functions such as, assured data provenance, auditing, management of digital assets, and distributed consensus. Blockchain's underlying consensus mechanism allows to build a tamper-proof environment, where transactions on any digital assets are verified by set of authentic participants or miners. With use of strong cryptographic methods, blocks of transactions are chained together to enable immutability on the records. However, achieving consensus demands computational power from the miners in exchange of handsome reward. Therefore, greedy miners always try to exploit the system by augmenting their mining power. In this paper, we first discuss blockchain's capability in providing assured data provenance in cloud and present vulnerabilities in blockchain cloud. We model the block withholding (BWH) attack in a blockchain cloud considering distinct pool reward mechanisms. BWH attack provides rogue miner ample resources in the blockchain cloud for disrupting honest miners' mining efforts, which was verified through simulations.**

***Index Terms*—Blockchain, cloud computing, block mining, data provenance, proof-of-work, block withholding, distributed ledger, pool mining, blockchain security and vulnerability**

I. Introduction

Blockchain technology has attracted tremendous interest from wide range of stakeholders including finance, healthcare, utilities, real estate and government agencies. Blockchain networks utilize a shared, distributed and fault tolerant ledger platform that every participant in the network can share but no entity can control. Blockchains assume the presence of adversaries in the network and nullify the adversarial strategies by harnessing the computational capabilities of the honest nodes and information exchanged is resilient to manipulation and destruction. The blockchain technology will be beneficial to cloud services which have a strong desire for assured data provenance and support cloud auditing. To enable data integrity over the public ledger in a blockchain cloud, cryptographically enforced blocks join in the blockchain after a consensus is reached in the decentralized network, where transactions in the blocks are authenticated by peers of the network. This shared ledger could potentially contain history of every transaction related to any sort of asset irrespective of its type: financial, physical, or digital, that can be verified, monitored, and cleared without involvement of cloud administrator. The combination of cryptographic mechanism and decentralized public ledger allows to build any kind of application on top of the blockchain without worrying about trust components of users and maliciousness in the blockchain enabled cloud system.

† Approved for Public Release; Distribution Unlimited: 88ABW-2016-6384; Dated 09 Dec 2016.

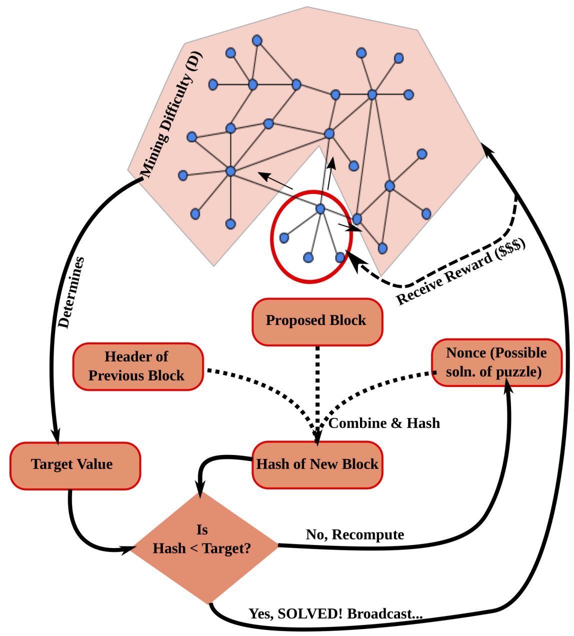

Fig. 1: Block Mining Process

For successfully adding a block in the blockchain, the miners need to solve a crypto-puzzle that is computationally hard. The process is depicted in Figure 1. The solution is hard to find but easy to verify, and the difficulty is dynamically set by the network. Therefore, solving such crypto-puzzle comes at a price in terms of hashing power, electricity, hardware etc., but succeeding in the competition rewards well too. Since mining alone is costly and receiving reward is so infrequent, honest miners prefer to work in pools. However, irrational malicious parties can come up with their own mining power to disrupt the other honest miners' operations. Block withholding attack [1] is one of the well known schemes, which adversaries may adopt to make the pool lose the block finding competition. In block withholding attacks, malicious pool members, who have joined for truthfully mining blocks, would actually never publish the successfully mined blocks, whereas they only submit their regular *shares* that are not solutions. Hence, the attacker decreases the expected revenue of the pool by withholding the valid blocks but increases its own reward by submitting as many *shares* as possible to the pool manager.

In order to realize a blockchain cloud, it is important to understand the vulnerabilities of the technology. In this paper, we focus on investigating the need of blockchain in developing secure cloud system and perform in-depth analysis on various possible vulnerabilities in the blockchain cloud. The vulnerabilities in a blockchain cloud will primarily arise due to the requirement of computational power to achieve proof-of-work (PoW) based consensus. There exists consensus mechanisms, such as, Proof-of-Stake (PoS) and Perfect Byzantine Fault Tolerance (PBFT) [2], which do not necessarily require high computational abilities to mine blocks. However, current public blockchain implementations rely on PoW to achieve consensus. So, the miners need to be computationally efficient to produce valid blocks as quickly as possible to get rewarded. At the same time, rogue miners may stand against to disrupt the block mining process by empowering their hash powers. Most importantly, we model a critical issue named block withholding attack, which may occur in blockchain cloud during pool mining, to identify the constraints on attacker's hashing power in order to defeat the purpose of pool mining. The scenario is analyzed individually when pool operator employs different rewarding schemes, such as proportional reward and Pay-per-Last-N-Shares (PPLNS) reward.

The paper is organized as follows. In Section II, we extensively discuss the blockchain preliminaries and its importance in provisioning cloud security. We also discuss several critical vulnerabilities related to blockchain in this section. Section III presents the system model that we consider to analyze block withholding attack in pool mining. Section IV expresses various possibilities that an attacker could increase its hashing limits. In Section V, we model the block withholding attack by considering two different pools with distinct reward mechanisms. The simulation results are discussed in Section VI. Section VII concludes the paper.

II. Blockchain Cloud and Associated Vulnerabilities

Among all the security issues exist in the cloud environment, blockchain will be very effective in addressing the challenges involved in implementation of assured data provenance [3]. We present the challenges associated with assured data provenance in the cloud and blockchain's capabilities to address them.

A. Blockchain and Cloud Security

Cloud computing allows users to remotely store their data into the cloud and provides on-demand applications and services from a shared pool of configurable computing resources. The security of the outsourced data in the cloud is dependent on the security of the cloud computing system and network. However, cloud's key characteristics, on-demand services, uninterrupted network access, resource pooling, and rapid elasticity are susceptible to vulnerabilities. In addition, the cloud computing's core technologies for virtualization, cryptography and web services have vulnerabilities, which are results of insecure implementation. At the same time, security controls, such as key management, in cloud computing environment have several challenges. For instance, to implement an effective key management system in cloud computing infrastructure requires management and storage of various kinds of keys. The difficulty in assigning standard key management stems from the fact that virtual machines usually have heterogeneous hardware/software, and the cloud based computing and storage are geographically distributed.

The protection of data exchanged within the cloud infrastructure currently relies on PKI based signatures. Specifically, there is a need for stronger attribution to detect unauthorized changes to the data and identify the responsible entity. Data provenance provides information on all changes performed on data exchanged between multiple entities. Researchers have proposed security solutions, such as PKI signatures, to ensure provenance. However, implementation of PKI signatures typically depends on a centralized authority, which is not effective in the cloud infrastructure.

Blockchain and Keyless signatures have been proposed as an alternative to PKI signatures, where the blockchain technology facilitates secure transfer of information through a sequence of cryptographically-secure keys across a distributed system. There is no need for a central authority because it is executed by a system of distributed ledgers, which records all actions performed on data and is shared among all the participating entities. The transactions in the public ledger are verified by a consensus of majority of participating entities. The blockchain contains a verifiable record of every transaction which cannot be changed. Keyless signature addresses the issue of "PKI key compromise" by decoupling the processes for identifying signer and integrity protection from the processes that are responsible for maintaining the secrecy of the keys [4]. The processes for identifying signer and integrity protection are handled by cryptographic tools which are chosen from the options such as, asymmetric cryptography and keyless cryptography [4]. Examples of keyless cryptography, include one-way collision-free hash functions. Keyless signature processes include, hashing, aggregation and publication. The realization of keyless signatures requires a Keyless Signature Infrastructure (KSI) which consists of hierarchy of co-operative aggregation servers which generate the global hash trees. The verification in KSI hinges on the security of hash functions and availability of a public ledger (blockchain). The ledger is publicly available and rules for updating, distributed consensus and mode of operation are well defined [5].

Guardtime has proposed changes to the traditional blockchain technology by integrating with KSI [5]. The KSI blockchain technology was developed to mitigate challenges associated with mainstream blockchain technologies. The challenges are lack of scalability, consensus time and lack of formal security proof. The KSI blockchain technique scales at $O(t)$ complexity as compared to $O(n)$ for traditional blockchain, where n refers to the number of transactions. In a blockchain cloud, it is likely that with the increasing granularity of measurements, the blockchain complexity will be concerning. The linear growth with time would lead to better scalability as there will be no dependence on the number of sensor measurements. There is a need to achieve quick consensus and ensure synchronous availability of all updates to the distributed ledger. Finally, the formal security proof will provide better assurance to the security of the blockchain. Recently, Ericsson and Guardtime have integrated the KSI blockchain technology in a cloud computing platform to enable organizations to assure provenance of data with forensically provable and complete attribution capabilities [6]. The capability will make real time governance of cloud operations and scalable data feasible.

1) Data Provenance in the Cloud: Assurance of data transfer within intra-cloud and inter-cloud environments is very crucial. Typical assurance of data focuses on ensuring the confidentiality, integrity and availability of the data contents. However, assurance of the ancestry of the data (where the data came from) is a challenge in cloud environments. Data provenance addresses such issue based on the detailed derivation of data objects. If true data provenance existed in the cloud for all data stored on cloud storage, distributed data computations, data exchanges and transactions, detecting insider attacks, reproducing research results, and identifying the exact source of system/network intrusions would be achievable. Unfortunately, the state-of-the art in data provenance in cloud does not provide such assurances.

Data provenance will be very critical for cloud computing system administrators to debug break-ins to the system or network. Cloud computing environments are typically characterized by data transfers between diverse system and network components. These data exchanges could take place within a data center or across federated data centers. The data does not usually follow the same path due to multiples copies of the data and diversity of paths taken to ensure resilience. This design adds degree of difficulty for administrators to accurately identify the origin of attack, what software and/or hardware components caused the attack, and the impacts of the attack. Security violations needed to be identified at a fine granularity and provenance can assist. Current provenance systems in the cloud support the above tasks through logging and auditing technologies. These technologies are not effective in cloud computing systems, which are complex in nature, due to several layers of inter-operating software and hardware components spanned across the geographical and organizational boundaries. To identify and resolve the malicious activities in cloud environment, it is required to analyze the forensics and logs from a diverse and disparate set of sources, which is an insurmountable task. Although cyber-threat information sharing [7][8] can be a viable option to achieve situational awareness about the cloud attack surface at a reduced investment [9], it faces the issue of information tampering. Maintaining provenance could expedite the gathering of information regarding the operations performed on every data object in the cloud and blockchain technology will come handy to ensure the content is unalterable.

2) Blockchain-based Data Provenance in Cloud: Cloud computing systems typically comprise of multiple of nodes (physical machines), which host one or more virtual machines(VMs). Each VM is owned by an owner, and includes components, such as software (application resources), data, etc. The execution of software in the VM and exchange of information with VM results in multiple artifacts, such as variables, intermediate data output, and final output artifacts. All of them are of interest and concern for the provenance. Blockchain technology provides such capability and resolves many needed functionalities as well as properties for effective provenance in cloud. In essence, blockchain is a peer-to-peer ledger system, where information that constitutes provenance for physical, virtual, and application resources can be stored publicly for transparent verifiability and audit. As such, both transparency and cost effectiveness are provided, while access control and privacy for individual users of the ledger are ensured through encryption techniques, where individuals can see only parts of the ledger that is related to them. Furthermore, blockchain technology provides the much needed functionalities, that happen to be part of the cloud, including assets transfer.

B. Blockchain Cloud Vulnerabilities

Blockchain cloud is realizable provided the majority of nodes in the network are honest and authentic. However, in the cloud environment, it is possible for rogue nodes to negatively impact the mining and/or consensus processes. In this section, we present the vulnerabilities in blockchain cloud.

1) Double Spending Attack: This vulnerability is particularly attributed in Bitcoin Technology, where adversaries look forward to using the same digital cryptocurrency for more than one transactions. Since reproducing of digital information is easy to achieve, the occurrence of double spending is possible in reality. To conduct such event, the attackers must have a strong hash power and must be able to generate a longer private chain compared to public blockchain. The attack is carried out in the following steps:

1) Starting from block N, privately mine to extend the blockchain as much as possible but do not publicize.
2) Broadcast the transaction to the organization of interest.
3) Wait patiently until enough confirmations are received and the transaction successfully gets recorded in the blockchain, so that the merchant dispatches the product.
4) Secretly mine for extending the private branch until it is longer than public branch. If succeeds in doing so, publicize the secret branch that will be eventually

accepted to be valid and the block containing payment to merchant will be discarded.

From the hash rate based analysis in [10], it is found that probability of succeeding in double spending attack (a_z) is governed by the following expression

$$a_z = \min\left(\frac{q}{p}, 1\right)^{\max(z+1,0)} = \begin{cases} 1 & \text{if } z < 0, q > p \\ (q/p)^{z+1} & \text{if } z \geq 0, q \leq p \end{cases}$$

where, p, q are hash rate proportion of honest nodes pool and attacker pool respectively and z is number of blocks by which the honest network has an advantage over the attacker. If z becomes negative (< 0), then attacker pool can publicize their blocks and overtake the honest users mined chain. Hence, the following remarks can be made on the success of such attack:

- The success of double spending depends on hash power of attacker and number of blocks (leading or lagging).
- If the attacker's hash power (q) is more than 50%, then it always succeeds.
- If $q < p$, then success probability decreases exponentially.

The above analysis was conducted when the number of confirmations on a transaction did not affect the merchant's decision. However, when the requirement of n confirmations for validating a transaction is enforced, the expression of success probability is the given in Eqn. 1.

$$a_z = \begin{cases} 1 & \text{if } q \geq p \\ 1 - \sum_{m=0}^{n} \binom{m+n-1}{m} p^n q^m - p^m q^n & \text{if } q < p \end{cases} \quad (1)$$

Thus, it is observed that double spending attack is successful for an attacker with any hash rate and the probability decreases exponentially as the number of confirmations increase. Hence, there is no relevancy of "6 confirmations requirement (often cited) as this value was chosen based on the assumption of an attacker may not have more than 10% hash rate compared to the rest of the network.

2) Selfish Mining Attack [11][12]: Since mining cryptocurrencies like Bitcoin is hard by a single miner due to the requirement of high computing power for solving the crypto-puzzle, a set of miners generally collude to form a pool among each other and share the received reward among themselves after successfully solving. This also helps the individual miners to generate a constant income instead of infrequent (random interval) payment, when they mine alone. It is argued that incentives to the pool of honest miners can be dominated if there exists a pool of selfish miners that intentionally aims to invalidate the work of honest miners by following a selfish mining strategy [11] and generate better revenue for themselves. Similar to double spending attacks, in selfish mining attack, the pool mines on their private chain and publish it strategically depending on the state of the pool. The states are defined based on the parameter $lead$ (difference in lengths of private chain and public chain) and branching (honest and selfish pool are working on different parent blocks). In brief, the selfish mining strategy can be stated as (assuming the honest pool always accepts the longest chain):

- If $lead = 2$ and honest pool mines the next block, then publish the entire private chain.
- If $lead = 0$, part of honest pool works on selfish pool's mined block, and selfish pool mines the next block, then publish the entire private chain.
- If $lead \geq 0$ and selfish pool mines the next block, then keep the mined block secret.

With the possibility of different states, Eyal et.al. calculated the following expected revenue (R_{pool}) of the selfish pool

$$R_{pool} = \frac{\alpha(1-\alpha)^2(4\alpha + \gamma(1-2\alpha)) - \alpha^3}{1 - \alpha(1 + (2-\alpha)\alpha)} \quad (2)$$

where, α is the hash power of the selfish pool and γ is the proportion of honest miners those choose to mine on the pool's block. Thus the selfish pool's revenue is governed by their hashing power and propagation factor (γ). In general, $0 \geq \alpha \geq 0.5$ must be satisfied in order to avoid 51% attack, however [11] proposed that pool of selfish miners can gain higher revenue if the following constraint on their hashing power (α) is satisfied:

$$\frac{1-\gamma}{3-2\gamma} < \alpha < 0.5 \quad (3)$$

It can be also observed that revenue of each selfish miner will increase if the pool size increases beyond the threshold. As a consequence, most of the honest miners would prefer to join the pool for generating higher incentives and eventually the pool becomes the major player that controls the blockchain. Thus, the decentralization would not hold any more.

To resolve this issue, the authors suggest that it is necessary to raise the threshold so that no pool can benefit by executing selfish mining. Rather, miners must propagate all the blocks when they learn about competing branches of same length in the blockchain and randomly choose one to mine. In that case, $\gamma = 0.5$, thus threshold gets raised to 0.25. But raising the threshold to 0.25 still keeps the range open to selfish mining attack, which may be successful if a pool can be formed with hash power of at least 25% of the total network.

An extension to selfish mining strategy has been proposed in [13] that states that revenue of a selfish miner can be even higher if it adopts their proposed stubborn strategies.

3) Eclipse Attack [14]: This type of attack is performed to take the advantage of peer-to-peer (P2P) network that is used to broadcast information among bitcoin nodes. In the bitcoin network, the nodes randomly select 8 other peers to maintain a long-lived outgoing connectivity for propagating and storing information about other peers. Additionally, nodes with public IP can accept up to 117 unsolicited incoming connections from any other IPs at max. Thus, the openness and decentralization of the P2P network attracts adversaries to join and perform eclipse attack, where "the attacker strategically takes the control of all the incoming and outgoing communications of a victim node", thereby stopping all connections from other legitimate nodes. The attack is performed by rapidly and repeatedly making unsolicited connection requests to the victim node from the attacker-controlled nodes by sending irrelevant information until the victim restarts. With such effort, there is a high chance that the victim will have the 8 outgoing connections to the attacker-controlled nodes.

In the core of Bitcoin's P2P network, the network information is propagated through DNS seeders (servers that resolve DNS queries with respective IP addresses) and ADDR messages (that are used to obtain network information from the peers and contain up to 1000 IP addresses). Each node also locally maintains two tables (`tried` and `new`) to keep the public IPs. `tried` table contains the addresses of the peers with whom the node has successfully established connection along with the timestamp information. Whereas, `new` table contains addresses of peers with whom connection is not yet initiated. When a node restarts or a connections is dropped, the next peer selection follows a probabilistic approach, where an address for $(\omega+1)^{th}$ connection is chosen from the `tried` table with following probability:

$$P[\text{Select from } \texttt{tried}] = \frac{\sqrt{\rho}(9-\omega)}{(\omega+1)+\sqrt{\rho}(9-\omega)} \quad (4)$$

where, $\rho =$ ratio of # of addresses in `tried` and `new` table.

The eclipse attack takes the advantage of above selection process for monopolizing all connections of a victim node.

i) Populates the `tried` table with attacker-controlled nodes' IP address by sending unsolicited messages
ii) Overwrite addresses of the `new` table with garbage addresses (not related to peers' IPs)
iii) Once the node restarts, with high probability all the connections are monopolized

4) Block Discarding Attack and Difficulty Raising Attack [15]: Block discarding attack is carried out by an attacker that has a good hold of network connections compared to a normal node. Since propagation of mined blocks is an important characteristic to add it into the mainstream blockchain, the attacker would preferably place multiple slave nodes to improve its network superiority. With this placement, the attacker could easily get informed of freshly mined blocks and instantly propagate the attacker's block faster than rest of the network. Thus, when any node publicizes a block, the attacker can immediately dispatch its own mined blocks, so as to discard others' blocks.

However, the difficulty raising attack takes the advantage of attacker's hashing power to manipulate the difficulty level of the crypto-puzzle. In this attack, it is claimed in [15] that probability of discarding a block at depth n (i.e. $(n-1)$ blocks have been mined after this) is p^n, where p is ratio of hash power of attacker and rest of the network. To succeed in doing so, the attacker must wait for enough time.

5) Block Withholding Attack [16]: In this type of attacks, some pool members, who have joined to help mining blocks, would actually never publish any block, thus decreasing the expected revenue of the pool. These attacks are also known as "Sabotage" attacks, where the rogue miner never gains anything rather makes everyone loose. However, an analysis with practical instantiation in [1] claims that a rogue miner could also gain profit from such attack.

6) Anonymity Issues in Blockchain Cloud: It has been acknowledged that the underlying blockchain technology of Bitcoin ecosystem is not completely anonymous in nature. The transactions are permanently recorded in the public ledger, hence everybody can see the balance, and transactions related to any Bitcoin address. The real identity and privacy of a user will not get exposed until the user reveals any information during purchase or any special circumstances. Therefore, Bitcoin is pseudo-anonymous, i.e. bitcoin addresses can be created by anyone but tracing back to the real person is not possible unless any information is found from another source. To maintain a higher privacy and better anonymize the transactions in Bitcoin environment, the users are encouraged to have multiple Bitcoin addresses. Since, the convenience of e-cash system and pseudo-anonymity attract darknet markets to make illegal transactions anonymously, it has been a topic of interest for government and security industries to track down such illicit activities from the publicly available blockchain.

The work presented in [17] focuses to deanonymize the owner of Bitcoin transactions through mapping the Bitcoin address with IP address of the actual owner. By collecting all the network level traffic data including IP information and performing offline processing, they found evidences of tracing back the IP address from bitcoin address. Besides pruning irrelevant transaction data, there are five crucial steps they follow to achieve the mapping: (1) hypothesizing an owner IP for each TX, (2) create granular pairings of (Bitcoin Address, IP), (3) define statistical metrics for the pairings, (4) identify potential pairings that signify actual ownership, (4) remove unwanted pairing based on a threshold. Targeting the Bitcoin peers behind NAT or firewall, [18] proposed a generic method to build a correlation between pseudonym of bitcoin users and their public IPs. The method utilizes the connected user set or entry nodes for identifying the origin of transactions. The outcome is a list, $\mathbf{I} = \{(IP, Id, PK)\}$, where Id is used to distinguish the clients using same IP, and PK is the pseudonames used in a transaction. Finding the entry nodes (at least 3) and mapping the transactions to these nodes are two important steps as mentioned in the paper to effectively deanonymize the clients.

After understanding the important security vulnerabilities related to the blockchain implementation, we undertake one important case called block withholding attack and analyze the situation rigorously to understand the strategies of a powerful attacker that lead to bring anomaly in the mining pool. The attacker joins in the pool with intentions to withhold the successful blocks and look for opportunities to specifically demotivate the honest members to mine in the pool. We theoretically analyze the attacker's strategy by considering different types of pools whose rewarding schemes are different.

III. System Model

In this work, we consider a pool $\mathcal{P}$, where n miners work continuously trying to solve the cryptographic puzzle using their hashing power. The pool is assigned with a pool operator, who is responsible for collecting transactions from the network, creating a block, keeping track of puzzle difficulty, recording the number of *shares* submitted by pool members, dispatching the successfully mined block to the P2P network, collecting and redistributing the reward among miners. The pool members are considered to be honest in nature, which means, they report their solutions or *shares* to the pool operator immediately as they find. Assuming the miner i has hashing power α_i of total mining power ($\mathcal{M}$), where $0 \leq \alpha_i \leq 1$, the overall mining power of the pool is $\beta = \sum_{i=1}^{n} \alpha_i$. For simplicity, we assume that $\beta < 1$, thus the pool is not the only computing entity but there exists other solo miners or pools whose computational power is fixed with respect to the pool $\mathcal{P}$. Additionally, the miners of pool $\mathcal{P}$ receive reward based on their contributions in a round, where a round is defined as the interval between two valid blocks found. The contribution of the miners is calculated based on the number of *shares* reported, and each *share* could be a full solution with probability $1/D$, where D is the overall difficulty of the crypto-puzzle which is assumed to be fixed. Finding a *share* is inevitably easier than finding the valid solution because the *share* is only meant to prove a miner has worked enough to find it. Thus, the difficulty for finding a *share* is determined by the pool operator, which is less that D. As a standard, a particular hash is a valid *share* with probability $\frac{1}{2^{\kappa}}$, where κ is specified by the private blockchain creator so that $0 \ll 2^{\kappa} < D$.

On the other hand, the attacker is considered to be a powerful miner who willingly participates in the pool $\mathcal{P}$ and aims to maximally damage the mining activity of the pool by withholding successfully mined blocks. The attacker joins the pool $\mathcal{P}$ with initial hashing rate of $\alpha_{\mathcal{A}}$ but we assume that the attacker has ability to augment its mining power by incorporating additional physical ASIC resources or leasing computing power from cloud vendors. Although, such efforts for increasing hashing power may be costly by itself, the attacker's irrationality makes it feasible by our assumption. In this paper, we aim to understand how much of extra mining power a block withholding attacker may need to completely sabotage a pool, thus leading to an situation where no single miner would prefer to honestly mine in the pool. This analysis would then help to perform a cost benefit analysis of leasing cloud/non-cloud computational resources for successful attack. By saying sabotaging a pool, we mean that the attacker comes up with such strategies that dominate the reward distribution in the pool and therefore leading to a situation where other members' reward variance out of pool mining is more than the variance out of solo mining.

IV. Augmenting with Extra Hashing Power

For increasing the hashing ability of a miner, there are different possible options available to undertake. The first option is mining hardware. Mining has progressed from the era of CPU to GPU and finally to Application Specific Integrated Circuit (ASIC) era. Currently, miners opt ASIC chips to mine blocks since these give a great amount of hashing power at a minimum cost. ASIC chips are designed purposely for block mining, hence cannot be used for any other task. The ASIC mining hardware offers 50x to 100x increase in hashing power while reducing the power usage by 7x compared to previous technologies. The company, Avalon [19], manufactures ASIC mining chips for the bitcoin miners' market where each server can process 3.65 TH/s at a power efficiency of 0.29 W/GH.

The second option is renting mining services from cloud providers, which is called cloud mining or cloud hashing. There are three kinds of possibilities to perform remote mining: (1) Hosted mining, (2) Virtual hosted mining, (3) Leased hashing power. In case of hosted mining, the user leases a machine that is capable of mining and hosted by the provider on cloud. In virtual hosted mining, the user creates is mining environment from scratch on the virtual private server. In case of leased hashing power technique, which is mostly used in current scenario, the user leases required amount of hashing power from the provider without any hassle of managing the infrastructure. This allows users to purchase hashing capacity from the cloud providers' mining hardware that are already installed in their data centers. This enables another viable way to get rid of the issues related to installing mining hardware, managing electricity consumption, or network connectivity and bandwidth requirements. Such services exist in practice and providers, such as Hashflare, Genesis Mining, Hashnest, Eobot, and MineOnCloud offer competitive prices in exchange of hashing capabilities.

V. Disruptive Attack Strategy Analysis

Considering the attacker is going to withhold the valid blocks only, it may publish the *shares* that are not exactly solutions of the crypt-puzzle. Thus, the only way an attacker can do damage to the pool is to take away as much reward as it can by submitting sufficient number of *shares*. Since, mining pools are characterized by the schemes they adopt to reward the participating miners, the attacker's disruption strategy may vary accordingly, which we discuss in the following.

A. *Proportional Reward:*

This is a very naive scheme [16], where the total reward is divided proportionately according to the number of *shares* each miner contributed in that round of competition. The round in this context means the interval between finding two successfully mined valid blocks. As a pool operator receives a challenge from the blockchain network, the competition round starts, where it assigns the task to participating miners in the pool. The members utilize their hashing ability to solve the crypto-puzzle. Upon finding a valid block, the honest members usually forward it to the pool operator which then broadcasts in the blockchain network. The pool operator receives a fixed reward R if the network of miners reach to consensus on its mined block and from this point the next round starts. The pool operator may keep a fixed percentage of the reward and the remaining is then distributed among the members in proportion to the number of *shares* each miner has contributed with respect to total number of *shares* received in that round.

Assuming that δ portion of reward is reserved for the pool operator, the pool members share the total reward of $(1-\delta)R$. Now, to estimate the expected number of *shares* in a particular round of competition, we assume that the time taken to find a *share* by miner i is exponentially distributed with parameter α_i. Thus, the expected time of finding a *share* is $\frac{1}{\alpha_i}$, which can be a full solution with probability $\frac{1}{D}$. Considering the round lasts for T units of time, the miner i could produce $\alpha_i \mathcal{M} T$ number of hashes on average in that round. However, the total number of *shares*, the miner produces is modeled as the function, $h(\alpha_i \mathcal{M} T)$, where $h(.)$ is monotonically increasing function with respect to i's mining power, i.e. $\frac{\partial h}{\partial \alpha_i} > 0$. Thus, the total number of *shares* submitted to the pool operator can be represented as:

$$H = \sum_{i \in \mathcal{P}} h(\alpha_i \mathcal{M} T) \tag{5}$$

So, the expected reward received by miner i out of the pool that adopts the proportional reward scheme is:

$$U_i = \frac{(1-\delta) R h(\alpha_i \mathcal{M} T)}{H} \tag{6}$$

Block withholding attack in the proportional reward pool: When a malicious miner with hashing power $\alpha_{\mathcal{A}}$ joins the pool, then the inherent power of the pool increases to $\sum_{i=1}^{n} \alpha_i + \alpha_{\mathcal{A}}$. Thus, the number of *shares* submitted in a round increases from H to H', where $H' = \sum_{i \in \mathcal{P}} h(\alpha_i \mathcal{M} T) + h(\alpha_{\mathcal{A}} \mathcal{M} T)$. As the hashing power of attacker increases, its *share* contribution in the pool also increases proportionately. Since the attacker never submits valid blocks (solutions) rather withholds them, the length of the round does not depend on its mining power. The goal of the attacker is to impose a maximum disruption to the pool members so that pool mining becomes no longer profitable for them, and forces them to leave the pool eventually. There are two different ways an attacker can affect the block mining of pool members: (1) eclipsing the blockchain network, where attacker could control the network connections and hence manipulate the victims mining activity directly, (2) increasing hashing power, which could produce more number of *shares* and hence overall reward received may dominate over rewards to all other members. We have chosen to analyze the second scenario, where the attacker could get extra mining power to dominate in the pool and increase the reward variance of pool members.

To start absorbing rewards, the attacker needs to know the amount of extra hashing power on top of $\alpha_{\mathcal{A}} \mathcal{M}$ it requires to generate x more *shares* because the only way to decrease the payout of pool members is to submit more *shares* to the pool operator. Now, if the attacker generates extra x *shares*, the rewards to the pool member i can be represented as:

$$U_i = \frac{(1-\delta) R h(\alpha_i \mathcal{M} T)}{H' + h(x)} \tag{7}$$

This means, payoff of miner i is inversely proportional to the mining power of the attacker and therefore its net reward goes down if the attacker generates more *shares*. However, it is not possible to make the utility of honest miners very close to zero because it will need an enormous amount of *shares* to generate. Pursuing that would equivalently require large computing ability which may not be possible for an attacker. Instead, we plan to find the lower bound on x that will still paralyze the miners to be better off with solo mining. Thus, the attacker's optimization function can be characterized as:

$$x^* = \underset{x}{\text{minimize}} \left[\sum_{i=1}^{n} \frac{(1-\delta) R h(\alpha_i \mathcal{M} T)}{H' + h(x)} \right] \tag{8}$$

Subject to,

$$prop\_var(U_i) \geq var(U^i_{solo}) + \epsilon, \forall i \in [1, \cdots, n]$$

Where, $0 < \epsilon \ll 1$ is a very small amount that signifies the point at which the payout variance out of pool mining crosses the variance of solo mining. $prop\_var(U_i)$ is the payout variance of miner i per each *share* while mining in the pool and $var(U^i_{solo})$ is the payout variance per *share* while mining solo. The constraint signifies that if $prop\_var(U_i)$ is higher than the that of solo mining, then the attacker successfully demotivates them to not participate in the pool. If the variance is high, then the reward will not be consistent, which is what the attacker wants to achieve.

Now, we find the variance of payout for miner i when mining solo with constant hash rate of α_i. In case of solo mining, the reward is given if it finds an actual block. Thus, the expected reward out of solo mining is $Pr(\text{finding a block})R = \frac{\alpha_i R}{2^{\kappa} D}$. Block finding as a solo miner can be characterized as a Poisson process with rate parameter $\lambda = \frac{\alpha_i}{2^{\kappa} D}$. Hence total expected reward that solo miner i can receive by mining for T amount of time is the following.

$$U^i_{solo} = \frac{\alpha_i T R}{2^{\kappa} D} \tag{9}$$

From the property of Poisson distribution, we know that variance (σ^2) is same as the rate parameter (λ). Hence, the variance of payout from solo mining can be

$$var(U^i_{solo}) = \frac{\alpha_i T R^2}{2^{\kappa} D} \tag{10}$$

To model the payout variance per *share* in pool mining, we first need to consider an appropriate distribution for total number of *shares* produced in the pool. Now, considering the $N = H' + h(x)$ as the random variable representing the total number of *shares* reported in the pool during a round, $Pr(N)$ represents the corresponding probability distribution function (PDF). The average value of reward per *share* to miner i in the round can be

$$\mathbb{E}[U_i] = \sum_{N=1}^{\infty} \frac{(1-\delta)R}{N} Pr(H' + h(x) = N) \tag{11}$$

where, $\frac{(1-\delta)R}{N}$ is the per *share* reward as we found earlier. Now, the expected squared payout can be defined as:

$$\mathbb{E}[U_i^2] = \sum_{N=1}^{\infty} \frac{(1-\delta)^2 R^2}{N^2} Pr(H' + h(x) = N) \tag{12}$$

Hence, the payout variance can be expressed as:

$$prop\_var(U_i) = (1-\delta)^2 R^2 \sum_{N=1}^{\infty} \frac{\left[Pr(N) - (Pr(N))^2\right]}{N^2}$$

Now, to find the concrete value of the payout variance, we can model the total number of *shares* using different standard probability distribution functions such as geometric and negative binomial. These PDFs provide near natural model to estimate the total number of *shares* in a round [16].

B. Pay-Per-Last N-Shares (PPLNS) Reward

Since proportional reward scheme is very naive in nature, it suffers from pool-hopping attack, where malicious miners could strategically choose different pools to mine in a round to obtain maximum reward. Therefore, several advanced schemes are proposed to avoid such scenario and PPLNS reward scheme one of them that supposedly resists the pool-hopping nature of greedy miners. Unlike proportional reward scheme, which is a round based, PPLNS considers temporal *share* submission activities to reward the pool members irrespective of rounds in the mining process. In this scheme, the reward is distributed among the miners who recently submitted their *shares* no matter how many actual blocks are found in the considered interval.

The simple variant of PPLNS is to set a threshold for total recent *shares* to N and the total reward is distributed among based on the last N *shares* submitted. Now taking the same notations into account and considering total B blocks are found during the last N *shares*, the payout per *share* (U_{pplns}) can be represented as:

$$U_{pplns} = \frac{(1-\delta)RB}{N} \tag{13}$$

where, B follows the Poisson distribution with mean $\frac{N}{2^\kappa D}$. In this variant, it is assumed that the difficulty D and reward R are kept constant. Altogether, RB defines the total expected reward received by the pool operator during the last N *shares*.

The pool employing PPLNS reward function typically works in the following manner. The operator keeps track of the history of at least last N submitted *shares* irrespective of the round they have been reported. Therefore, the rounds in this case are interdependent on each other. The ordering of *shares* is maintained using a sliding-window of length N, which later is used to divide the total reward proportionately among the contributors. Considering the sliding window at k^{th} *share* as $\mathbf{s_k} = \{s_{k-N}, s_{s-N+1}, \cdots, s_k\}$, the reward for miner i can be expressed as:

$$U^i_{pplns} = (1-\delta) \times \frac{\#\{s_j : s_j \in \mathrm{s} \text{ and } s_j = i\}}{N} \times RB \tag{14}$$

In the above expression, the total reward received in the window of last N *shares* is divided proportionately among each pool member $i \in \mathcal{P}$, who submitted their *shares* in the considered window.

Block withholding attack on PPLNS reward pool:

Similar to the previous reward scheme, one malicious miner is introduced in the PPLNS pool who aims to disrupt the normal operation of pool mining via accumulating as much reward to itself. Unlike the strategy adopted in proportional reward pool, the attacker would have to opt for a different strategy in the PPLNS pool because submitting as many *shares* may not be the optimal case anymore due to the fact that only last N *shares* will be considered for the reward distribution irrespective of the rounds in which they are submitted. If we consider that there are N slots to place N *shares*, the attacker can opt for the following strategies to disturb the fairness of pool reward system. First the attacker with mining power $\alpha_{\mathcal{A}}$ must mine at a faster rate compared to other pool members, where successfully mined blocks are withheld. Next, the *shares* are published in such a way that they reside in the window of last N *shares*, so that the majority of reward is returned to the attacker instead of the pool members.

The attacker's reward can equivalently defined as per Eqn. (14), where number of *shares* it contributes in the last N slots is dependent on its hashing power.

$$U^{\mathcal{A}}_{pplns} = (1-\delta)RB\left[\frac{\Phi(N, \alpha_{\mathcal{A}})}{N}\right] \tag{15}$$

where, $\Phi(.)$ is a function that models the average number of *shares* the attacker can contribute in the window of N slots. The function Φ depends on the hashing power of attacker and follows a Poisson distribution with mean parameter of $\lambda = \frac{\alpha_{\mathcal{A}}\mathcal{M}N}{2^\kappa}$. At the same time, the random variable B also follows a Poisson distribution with mean $\frac{\mathcal{M}N}{2^\kappa D}$. However, due to the fact that the attacker withholds the successfully mined blocks, the mean value of B will exclude the hashing capability of the attacker. So, the mean of Poisson distributed random variable B will be $\frac{(1-\alpha_{\mathcal{A}})\mathcal{M}N}{2^\kappa D}$. Hence, the expected reward for the attacker can be represented as

$$\mathbb{E}[U^{\mathcal{A}}_{pplns}] = (1-\delta)R\frac{\alpha_{\mathcal{A}}(1-\alpha_{\mathcal{A}})\mathcal{M}^2 N}{2^{2\kappa}D} \tag{16}$$

Understanding the overall utility of the attacker, its goal would be to maximize this quantity with the help of additional hashing power from external sources. Compared to the optimization goal defined in the proportional reward pool, where the goal was to minimize the other pool members' reward by finding an optimal amount of computational power, here the objective is to maximize its own reward in the window of N-slots by submitting as many *shares* so that other members' net reward will automatically decline. Now, considering the attacker comes up with additional $y\mathcal{M}$ amount of computational power into the pool, then the net objective function of the attacker can be defined as follows:

$$y^* = \underset{y}{\text{maximize}} \left[(1-\delta)R\frac{(\alpha_{\mathcal{A}}+y)(1-\alpha_{\mathcal{A}})\mathcal{M}^2 N}{2^{2\kappa}D}\right] \tag{17}$$

Subject to,

$$pplns\_var(U_i) \geq var(U^i_{solo}) + \epsilon, \text{ for an } i \in \{1, \cdots, n\}$$

The constraint posed in the above optimization problem is similar to the proportional pool but here we are looking for one such miner whose reward variance from pool mining is more than solo mining. If that case happens, then miner i

would prefer to leave the pool and mine alone. Therefore, this process can be iterated to eliminate miners one by one and thereby harming the pool mining process. Now, to define the variance of a miner i from PPLNS pool when the attacker has incorporated extra $y\mathcal{M}$ amount of hashing power, the expected reward for miner i needs to be formulated, which is $\frac{(1-\delta)R\alpha_i\mathcal{M}'N}{2^\kappa DN} = \frac{(1-\delta)R\alpha_i\mathcal{M}'}{2^\kappa D}$. Due to the presence of block withholding attacker, the total mining power of the pool is modified to $\mathcal{M}'$, which is defined as $(1-\alpha_\mathcal{A}-y)\mathcal{M}$. Hence the corresponding reward variance can be presented as:

$$pplns\_var(U_i) = \frac{(1-\delta)^2R^2\alpha_i(1-\alpha_\mathcal{A}-y)\mathcal{M}}{2^\kappa DN} \tag{18}$$

The reward variance out of solo mining will be same as defined in the previous subsection, where the period T will be replaced with N.

VI. Simulation Results and Discussion

In this section, we evaluate the block withholding attack instance by considering a private proof-of-work based blockchain that may be operable in small scale enterprises. It is assumed that all nodes mine on a single pool, and one of them is malicious in nature and looks forward to disrupt the honest mining performed by rest of the nodes. We consider the initial total mining power of the pool ($\mathcal{M}$) is 100 GH/s and the difficulty of crypto-puzzle (D) to be 2096, which is kept fixed throughout. Considering a homogeneous environment, where all the miners have equal computational power, we have experimented by provisioning extra hashing to the attacker and observing the average reward variation. All the miners including the attacker are assumed to have 20% of the hashing power in the beginning. For computing number of *shares* in a round, we assume the value of κ as 10 which is typically half number of bits than the actual difficulty (D). The total *shares* submitted by a miner with power α_i, denoted by $h(\alpha_i)$, is sampled from a Poisson distribution of mean $\frac{\alpha_i}{2^\kappa}$.

Since the attacker has possible means of increasing its hashing capability, we first analyze the impact of such action on the overall reward of the attacker as well as the honest miners. Considering a maximum reward (R) of 10 for successfully mining a block, we can observe from Figure 2 that increasing hash power of an attacker in the pool mining has an adverse effect on the honest miner group. Therefore, the honest miners' reward decreases gradually as depicted in the red dotted line. Although the attacker could potentially increase its hashing ability to a higher amount, such costly attempt is not necessary for demotivating honest miners to leave the pool. Rather the attacker would prefer to have a threshold amount of computational power so that the total reward of honest miners just goes below the attacker's reward, which happens to occur at $x = 0.6$ in our considered case. When the submitted *shares* are considered with respect to increasing hashing power of attacker, we could observe from Figure 3 that the total number of *share* submissions from attacker increases exponentially, whereas the honest pool's submissions are fixed on an overage. This situation is created due to the fact that attacker has growing computational power, while the honest miners' hashing limits are fixed.

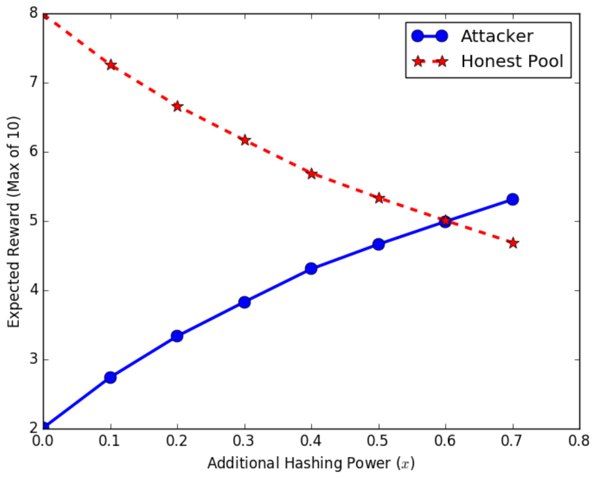

Fig. 2: Effect on Average Reward in Proportional Pool

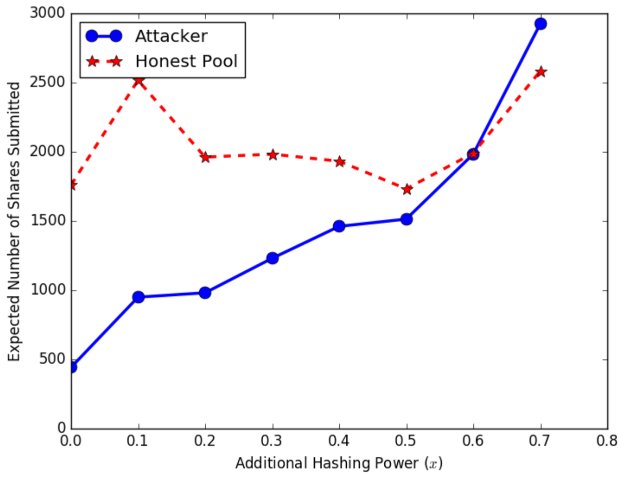

Fig. 3: Effect on Average # of *Shares* in Proportional Pool

In Figure 4, the reward variation with respect to attacker's increasing hashing ability is depicted, when the pool employs PPLNS reward mechanism. Here, we observe that the overall reward a pool could obtain can be higher than proportional pool. This is because of the fact that more than one successful blocks could be found during the submission of last N *shares*. However, when the attacker gathers extra computational power, it can take away the honest miners' incentives of mining in a pool. We could observe that with additional power of $y = 0.6 + \epsilon, \epsilon > 0$, the attacker dominates over the honest pool members. Another interesting observation can be made from Figure 5 that with increasing the window size in multiples of difficulty (D), the total reward to the attacker does not increase at growing rate, whereas the honest miners are rewarded higher than attacker. This happens due to the fact that more number of *shares* come from the honest pool while the attacker with a constant hash rate could not make profit unless extra computational power is added. This gives an idea

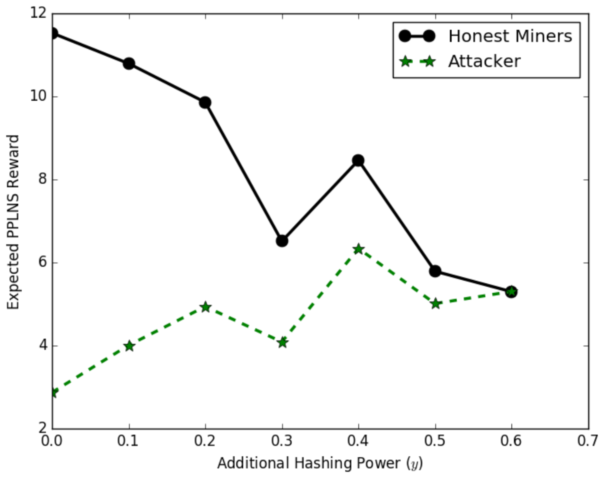

Fig. 4: Effect on Average Reward in PPLNS Pool

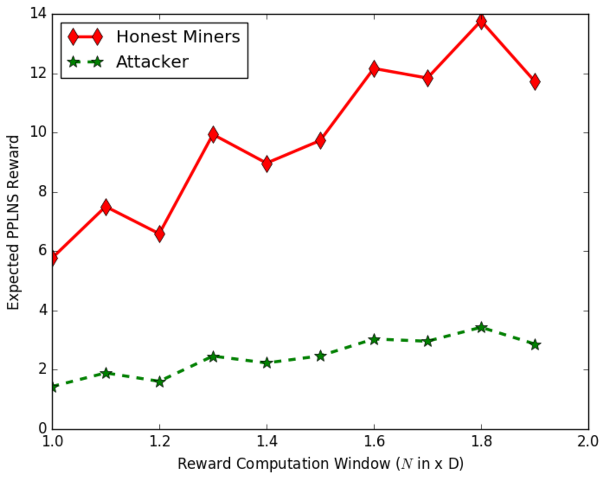

Fig. 5: PPLNS Reward Variation Vs. Window Length (N)

that how the window size (N) can be chosen to build a barrier for a block withholding attacker that aims to take away the honest miners' rewards.

VII. Conclusions and Future Directions

Since blockchain technology is one of the next generation technologies that employs cryptographically enforced distributed ledger system, its security evaluation is necessary to assure its usefulness in cloud computing domain. Therefore, we investigate the applicability and security implications of blockchain in realizing blockchain cloud. A number of security vulnerabilities are discussed that may have harmful impact while integrating blockchain on cloud system. We then particularly model the issue of block withholding attack that is prevalent in PoW based mining pools to understand the attacker's strategy toward taking over the pool members' rewards. Simulation results demonstrate that attacker's access to extra computational power could disrupt the honest mining operation in blockchain cloud. Attacker's strategy is analyzed based on two different pools where reward schemes are different, we found that pay per last N shares (PPLNS) scheme could be useful in keeping the attacker's impact lesser than proportional reward scheme. In future, we aim to extend our analysis on proof-of-stake based blockchain cloud and test the instances on real-time private blockchain platform.

VIII. Acknowledgement

This work was supported by Office of the Assistant Secretary of defense for Research and Engineering (OASD (R&E)) agreement FA8750-15-2-0120 and Air Force Material Command award FA8750-16-0301.

References

[1] N. T. Courtois, L. Bahack, On subversive miner strategies and block withholding attack in bitcoin digital currency, arXiv preprint arXiv:1402.1718.

[2] M. Castro, B. Liskov, Practical byzantine fault tolerance and proactive recovery, ACM Transactions on Computer Systems (TOCS) 20 (4) (2002) 398–461.

[3] X. Liang, S. Shetty, D. Tosh, C. Kamhoua, K. Kwiat, L. Njilla, Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability, in: International Symposium on Cluster, Cloud and Grid Computing, IEEE/ACM, 2017.

[4] A. Buldas, A. Kroonmaa, R. Laanoja, Keyless signatures infrastructure: How to build global distributed hash-trees, in: Nordic Conference on Secure IT Systems, Springer, 2013, pp. 313–320.

[5] KSI Blockchain Technology, https://guardtime.com/technology/ksi-technology.

[6] Ericsson and Guardtime create secure cloud and big data,https://www.ericsson.com/news/1853499.

[7] D. Tosh, S. Sengupta, C. A. Kamhoua, K. A. Kwiat, Establishing evolutionary game models for cyber security information exchange (cybex), Elsevier Journal of Computer and System Sciences. URL http://dx.doi.org/10.1016/j.jcss.2016.08.005

[8] C. Kamhoua, A. Martin, D. K. Tosh, K. Kwiat, C. Heitzenrater, S. Sengupta, Cyber-threats information sharing in cloud computing: A game theoretic approach, in: IEEE 2nd International Conference on Cyber Security and Cloud Computing (CSCloud), 2015, pp. 382–389.

[9] D. K. Tosh, M. Molloy, S. Sengupta, C. A. Kamhoua, K. A. Kwiat, Cyber-investment and cyber-information exchange decision modeling, in: IEEE 7th International Symposium on Cyberspace Safety and Security, 2015, pp. 1219–1224.

[10] M. Rosenfeld, Analysis of hashrate-based double spending, arXiv preprint arXiv:1402.2009.

[11] I. Eyal, E. G. Sirer, Majority is not enough: Bitcoin mining is vulnerable, in: International Conference on Financial Cryptography and Data Security, Springer, 2014, pp. 436–454.

[12] A. Sapirshtein, Y. Sompolinsky, A. Zohar, Optimal selfish mining strategies in bitcoin, arXiv preprint arXiv:1507.06183.

[13] K. Nayak, S. Kumar, A. Miller, E. Shi, Stubborn mining: Generalizing selfish mining and combining with an eclipse attack, in: 2016 IEEE European Symposium on Security and Privacy (EuroS&P), IEEE, 2016, pp. 305–320.

[14] E. Heilman, A. Kendler, A. Zohar, S. Goldberg, Eclipse attacks on bitcoins peer-to-peer network, in: 24th USENIX Security Symposium (USENIX Security 15), 2015, pp. 129–144.

[15] L. Bahack, Theoretical bitcoin attacks with less than half of the computational power (draft), arXiv preprint arXiv:1312.7013.

[16] M. Rosenfeld, Analysis of bitcoin pooled mining reward systems, arXiv preprint: http://arxiv.org/abs/1112.4980.

[17] P. Koshy, D. Koshy, P. McDaniel, An analysis of anonymity in bitcoin using p2p network traffic, in: International Conference on Financial Cryptography and Data Security, Springer, 2014, pp. 469–485.

[18] A. Biryukov, D. Khovratovich, I. Pustogarov, Deanonymisation of clients in bitcoin p2p network, in: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, ACM, 2014, pp. 15–29.

[19] https://bitcoinmagazine.com/articles/avalon-releases-new-asic-miner-begins-shipping-worldwide-through-blockc-partnership 1447268188.