DC-PoET: Proof-of-Elapsed-Time Consensus with Distributed Coordination for Blockchain Networks

Amitangshu Pal and Krishna Kant
Computer and Information Sciences, Temple University, Philadelphia, PA 19122

***Abstract*—Blockchain technology has gained a significant amount of interest in recent years due to its decentralized control, immutability, transparency and robustness. In this paper we propose an enhancement to BlockChain built using *proof-of-elapsed-time (PoET)* consensus protocol to further increase its efficiency and transaction throughput. The proposed scheme, called DC-PoET, exploits distributed coordination (DC) among the nodes to avoid unnecessary transmission of conflicting blocks inspired by a similar mechanism in WiFi networks. We show that DC-PoET can support around 465 transactions per seconds with 30 MB block size, and even higher for larger blocks. We have also developed detailed analytical modeling for the performance of DC-PoET scheme using a two-dimensional Markov Chain, along with the validation of such modeling using Matlab simulations. The security analysis of our proposed scheme is also discussed.**

I. Introduction

The Blockchain (BC) technology was originated with the goal of developing a cryptocurrency (bitcoin) not controlled by any central authority. The technology has since attracted a lot of interest in building decentralized systems including financial, manufacturing, transportation, agriculture, supply chain etc. Thus, the performance of BC is becoming more crucial, which is a topic that we address in this paper.

Related works and their limitations: A BC network can be either permissionless or permissioned; in a permissionless network such as those for cyptocurrencies, anyone can join the network and issue transactions. Traditionally, such networks have used a "proof-of-work" (PoW) mechanism (such as solving a mathematical puzzle) to decide who gets to create the next block [1]. In a permissioned network, the identity of each participant is known and can be authenticated cryptographically which in turn can be used to control who can (a) read and append to ledger data, (b) issue transactions, (c) administer participation in the BC network.

As more applications are built on top of blockchain based systems, performance becomes a major bottleneck both in terms of throughput and latency of individual transactions. A PoW based consensus mechanism for such applications is not only unconscionably wasteful of computing power and energy but also unnecessary. Recently many other mechanisms have been offered as more efficient alternatives to PoW, such as proof-of-stake (PoS), proof-of-burn, "useful" proof of work, combined PoS and PoW, proof-of-space etc. However, the throughout limitations and higher energy consumption remains a key challenge of PoW consensus and its variants.

Another approach is to follow some variant of Byzantine fault tolerance (BFT) [2], usually the Practical BFT (PBFT) [3] where by disallowing equivocation, t faults can be tolerated with only $2t + 1$ nodes. Unlike PoW based protocols, BFT protocols assume that the number of participants are known, and so are applicable for permissioned blockchains. BFT also involves 3 rounds of communications and thus does not scale well in number of participants [4].

Another approach, proposed by Intel to securely generate a random waiting time and then choose a node with the smallest waiting. The resulting proof-of-elapsed-time (PoET) consensus mechanism requires a trusted execution engine (TEE) [5] to execute critical parts of the protocol to ensure fairness and block all potential pathways of cheating. Such a mechanism is well suited for permissioned chains where "bad actors" who might go to extreme lengths to break the security (e.g., by modifying hardware) can be tracked. While PoET resolves the energy consumption problems of PoW based mechanisms, the throughput is still limited by heavy duty processing and conflict resolution requirements.

DC-PoET and our contributions: In this paper, we enhance the PoET scheme using mechanisms similar to those for distributed arbitration in wireless networks such as WiFi and show that the improved system significantly reduces the probability of orphan blocks and also improves the throughput. In this context our main contributions are as follows. **First**, we develop a *proof-of-elapsed-time with distributed coordination (DC-PoET)* based consensus mechanism that can achieve (a) high throughput that is limited by the validation speed of the network, with (b) low orphan risk that is limited by the network latency, along with (c) low energy usage and (d) high scalability. **Second**, we developed a detailed analytical model of the DC-PoET scheme using a two-dimensional Markov chain model and validated it with Matlab simulations. Through simulations we have shown that DC-PoET can achieve a transaction throughput of $\sim$465 transactions per seconds with a block size of 30 MB and even more with larger blocks. The orphan risk of DC-PoET also remains low and independent of block size, which is a major advantage of DC-PoET as opposed to other consensus mechanisms. **Third**, we have conducted detailed security analysis of DC-PoET both in case of uncompromised TEEs, along with the scenarios where some TEEs are compromised. To the best of our knowledge this is the first proposal that provides comprehensive blockchain consensus protocol based on distributed coordination mechanism inspired by WiFi along with its performance/security analysis.

Paper organization: The outline of the paper is as follows. Section II describes some details of PoET and the SGX security enclave. Section III briefly describes the distributed

ISBN 978-3-903176-39-3

2021 IFIP Networking Conference (IFIP Networking) | 978-3-9031-7639-3/21/$31.00 ©2021110.23919/IFIPNETWORKING52078.2021.9472787

coordination mechanism and its application to improve PoET. Section IV presents a mathematical modeling of the proposed mechanism. Section V describes the performance evaluations of DC-PoET, along with its security and robustness. Finally, section VI concludes the paper.

II. Trusted Computing and PoET Consensus

A. Trusted Execution Environments

Given the vulnerability of the Operating System (OS) to attacks due to weak security model and/or bugs in the OS code, it is usually not possible to ensure the integrity or trustworthiness of the OS or the applications supported by it. The trusted execution environment (TEE) address this issue by creating a secure container or "enclave" to both hold secret data such as keys and code/data used in trusted execution. The enclave is carved out of a memory region at boot time and is isolated from direct access to or by the OS. Both the ARM and x86 ecosystems provide provide such features, namely, ARM Trustzone, AMD SEV (secure encrypted virtualization), and Intel SGX (Software Guard Extension) [6].

TEEs are designed for secure remote computation where the owner of code (and possibly data) wants to execute the code on a remote machine owned by an untrusted party with some integrity and confidentiality guarantees. For example, a remote node with Intel SGX establishes a secure container, called *enclave*, into which the remote code/data is loaded and executed without perturbation from or disclosure to the machine owner. SGX relies on software attestation via cryptographic signature that certifies the hash of the secure container's contents. Also, all of the enclave memory contents remain encrypted and so do all data transfers over the busses, so that any snooping does not offer valuable data. Execution flow can only enter an enclave via special CPU instructions similar to those for switching from user mode to kernel mode. Enclave execution always happens in protected mode, in ring 3, and uses the address translation set up by the OS kernel and hypervisor. SGX also uses cryptographic protections to assure confidentiality, integrity and freshness of the enclave pages while they are stored in untrusted memory. All IO and communications from and to the enclave is authenticated, encrypted and can be further protected by using trusted IO drivers.

B. SGX Based Proof of Elapsed Time

SGX provides BlockChain support through the "Sawtooth Hyperledger" that implements PoET based consensus protocol, designed for permissioned blockchains. A Sawtooth participant first must *join* the network and is authenticated via the PKI mechanism (private/public key pair). The key aspect in PoET is to securely generate a "waiting-time" within SGX for which the node must wait before attempting to commit a block. As such, PoET1.0 is defined to generate waiting-time with shifted exponential distribution given by [7]:

$$\mathsf{waitTime} = \mathsf{MinimumWait} - \mathsf{LocalMean}.\log(d) \quad (1)$$

where MinimumWait is a constant system parameter that represents the shift, LocalMean is the mean of the exponential distribution, and $d \in (0, 1)$, the uniformly distributed random number. The LocalMean is kept roughly proportional to the network size so as to maintain a constant inter-block interval as the network grows.

Although PoET specification ties itself to the shifted exponential distribution in couple of places (e.g., generation of waiting time, estimation of population, etc.), it is easy to alter it to work for any distribution. For simplicity in the analysis, we assume a uniform distribution (with specified minimum and mean) in this paper. We believe that the resulting limit on waiting time and uniformity are helpful in permissioned networks. Like other consensus mechanisms, in PoET also the conflicting version of BCs (i.e. "forks) can be resolved using the longest chain rule.

C. Security of SGX and PoET

Since its launch in 2015, the security of SGX has been examined extensively and numerous side-channel attacks have been explored [8], [9]. Unfortunately, the standard security analysis (e.g., resistance against chosen plaintext attacks) provides no information about the side-channel attack possibilities that exist with any cyber system. For example, the timing analysis and the power trace of the device computing the cryptographic operation can leak the secret key [8], [9]. Also, by monitoring cache usage, it is possible to effectively "read the memory" of another process since the read-latency depends on the cache contents, and there is dependence on this between different processes [10]. Such an attack does not require physical access to the machine running the victim code but do require co-location of attacker and the victim.

In view of this, it is important to make a protocol like PoET tolerant of a few nodes. Reference [7] has attempted to quantify the number of SGX based PoET nodes that need to be compromised in order to compromise the entire distributed system. It banks on the idea that a statistical test (such as the z-test used by PoET) is necessarily limited in its ability to determine whether the samples come from the assumed waiting time distribution. It shows that by compromising a fraction $\theta\left(\log\log n/\log n\right)$ of the n PoET nodes it is possible for the bad nodes to put blocks as fast as the fastest node. It is hard to pin down the precise number, but it can be less than the 50% compromise needed with PoW based blockchain. Although the practicality of coordinated attack by many compromised nodes in a permissioned BC remains unclear, it is concluded that a lower variance of waiting time distribution would require a higher level of collusion though at the cost of more collisions [7].

D. Issues with PoET Based BlockChain Consensus

Because of the fully distributed operation, conflicting updates to local BCs by different nodes are inevitable in PoET. However, such a scheme wastes both processing and communication resources due to "orphan blocks", i.e., conflicting blocks that are eventually rejected (or replaced by other blocks). The primary reason for this the transit/processing delay of blocks, and increases with the increase in network size and the physical extent of the network. The signal propagation time is 5 μs

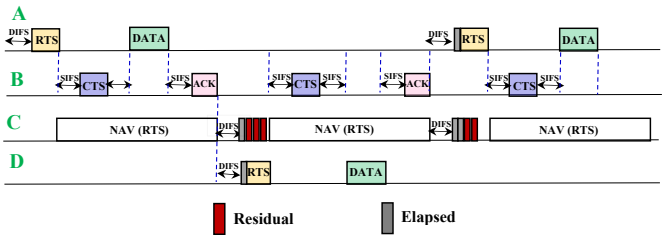


Fig. 1: IEEE802.11 DCF mechanism.

per Km in fibre cables, and the cables may take roundabout paths to distant places. So, a one way propagation delay within USA mainland could be 30 ms or more. Mathematically, we express the transmit time can be expressed as

$$\delta = \delta_0 + zB \tag{2}$$

where δ_0 is the network latency[1], whereas B is the number of bytes in the block and z is an empirical constant. Similar models are used in [11], [12]; the value of δ_0 is assumed to be 1 second and 10 seconds in [12] and [11] respectively, however, can be even lower for permissioned blockchains. The probability of orphan block generation is the probability that some other node in the network commits another block into the blockchain within the transit time, which is expressed by

$$P_{\text{orphan}} = \int_0^{\delta} \frac{1}{T_1} e^{\frac{t}{T_1}} dt = 1 - e^{-\frac{\delta}{T_1}} = 1 - e^{-\frac{\delta_0 + zB}{T_1}} \tag{3}$$

where T_1 is the average block interval (of the network).

Notice that P_{orphan} is a function of B, i.e. higher is the block size, higher will be the possibility of orphan risk. The primary goal of our consensus mechanism is to ensure that P_{orphan} is *independent of block size*.

III. Exploiting Distributed Coordination in Blockchain

Distributed coordination function (DCF) is the fundamental medium access control (MAC) technique of the IEEE 802.11-based WLAN standard (including WiFi). We describe this in the following and show how similar ideas can be exploited to make PoET more efficient.

A. IEEE 802.11 DCF

In IEEE 802.11 DCF when a station wants to transmit, it first checks whether a channel is idle for a Distributed Inter-Frame Space (DIFS) period. If the channel is not idle (either immediately or during the DIFS), it senses the channel until it becomes idle for a (DIFS) period. At this point the station goes into random backoff, and when the backoff timer expires, it transmits. The purpose of this random backoff is to avoid collision among multiple contending stations. The backoff timer freezes whenever the channel is sensed to be busy, and resumes after a DIFS period from the time when the current transmission finishes. If a packet is received successfully by the receiver, the receiving station sends back an ACK after a short inter-frame space (SIFS) period.

DCF adopts an exponential backoff scheme, where the backoff timer is randomly chosen between 0 to $w-1$ with w being the contention window size. In the beginning w is set to the minimum contention window size, i.e W_0; after every unsuccessful attempt, the window size is doubled up to a maximum value of W_m.

In addition the above-mentioned *basic* backoff based mechanism, DCF also provides an optional RTS-CTS based handshake, which is depicted in Fig. 1. In RTS-CTS based handshake mechanism, if a station wants to transmit, it waits until the station is sensed idle for a DIFS, goes to random backoff and when the backoff timer expires, transmits a short frame called request to send (RTS). After receiving the RTS, the receiving station waits for a SIFS period, and then sends a clear to send (CTS) frame. After receiving the CTS, the transmitting station waits for a SIFS before sending the actual data packet. Both RTS and CTS carry the length of the packet to be transmitted, thus the listening stations can update their network allocation vector (NAV) which contains the time the medium will remain busy. For example in Fig. 1, station C starts its NAV after receiving RTS from stations A and D.

In the following, we use some of these ideas to generate two variants of PoET, collectively called DC-PoET, or PoET with distributed coordination. These two variants differ in the way they resolve conflicts, we describe them as DC-PoET[1] and DC-PoET[2] in sections III-B and III-C respectively.

B. DC-PoET[1] with exponential backoff

The key proposed change to PoET protocol is the use of equivalent of RTS before the block transmission. In particular, whenever the wait-timer expires, a node sends an RTS to all other nodes. The reception of RTS by a node is a cue to other nodes not to send a conflicting block and thus reduce chances of orphan block transmission and processing. The RTS originating node sends the block shortly thereafter without any CTS-like exchange. As discussed in section III-D, we deliberately keep the RTS processing lightweight, so that its overall transit delay δ_{RTS} is close to the (rather large) assumed network delay δ_0. In the following, we draw further parallels between wait-timer in PoET and DCF, and characterize the DIFS, SIFS and NAV periods for DC-PoET.

In DC-PoET[1] the MinimumWait is equivalent to the DIFS period. Thus, generating the waitTime from the enclave is equivalent to waiting for a DIFS period along with a random backoff, i.e. in DC-PoET[1] the expression of wait-time is:

$$\text{waitTime} = \underbrace{\text{MinimumWait}}_{\text{DIFS period}} + \underbrace{\text{randomBackoff}}_{\text{Backoff timer}} \tag{4}$$

Before committing a block into the chain, each node first waits for a DIFS period, if it does not receive anything during this period, it waits for a random backoff randomBackoff. When the backoff timer expires, it transmits the RTS that contains the NAV. In DC-PoET[1] the NAV is set to the maximum block transit time. The nodes that receive the RTS freeze the timer

[1]The network delay is the time taken by a (short) message to travel from an originator to any other active nodes in the overlay network through flooding.

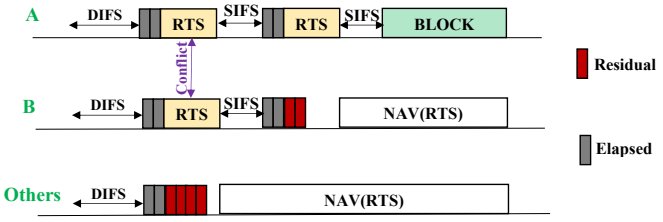


Fig. 2: An illustration of DC-PoET[1] mechanism.

and stay silent during the entire NAV. Unlike 802.11 DCF, there is no CTS and ACK exchanges in DC-PoET[1].

If the committed node does not receive RTS from any other nodes within the SIFS period (which is equal to the upper bound of δ_{RTS}), then it commits the block into the blockchain[2]. If it receives one or more RTS during the SIFS period, it leads to a conflict (which is equivalent to collision in 802.11 DCF). Notice that the probability of conflict in the first attempt is much less than that of PoET or other consensus mechanisms as $\delta_{\text{RTS}} << \delta$. In case of a conflict the node goes in random backoff with a contention window equal to the double of the current contention window (until the window size equals to W_m), and after that sends another RTS message. In this way DC-PoET[1] tries to resolve the conflicts among the nodes; after N attempts if the conflict is not resolved, then the conflicting nodes commits their blocks into the blockchain, which results in forks; however, such cases arises with very little probability. Also unlike 802.11 DCF, after a conflict the conflicting nodes do not start another DIFS, rather directly start the backoff timer.

The entire scheme is depicted in Fig 2. In this figure the RTS from nodes A and B results in a conflict in the first iteration, however the conflict is resolved in the next iteration. Other stations that receive RTS message from A and B remain silent for the entire NAV during the course of the block transmission. Also notice that in DC-PoET[1] if the RTS transmission is a success, then the following committed block transmission will be conflict-free, which is not the case of PoET. Thus the effect of block size does not affect the possibility of conflict in DC-PoET[1].

C. DC-PoET[2] with TEE timestamp

To to reduce the orphan risk further, we propose another variant of DC-PoET, called DC-PoET[2], that works as follows. The key issue is the expiry of wait-timers of two nodes within the δ_{RTS} time, which will result in a conflict. Since the RTS carries the wait-certificate, the intermediate nodes can use it as follows: if (a) this is the first RTS in that round, or (b) its recorded waiting time is less than the previously forwarded RTS of that round, then forward the RTS. Thus, at the end of the RTS propagation, the RTS with minimum waiting time will reserve the NAV time for its sender. Note that even if multiple nodes send their RTS within δ_{RTS}, the one with minimum waiting time will win. In this case the conflict happens when the timer of two nodes will generate the exact time, the possibility of which is almost zero.

[2]Notice that due to network uncertainty, the RTS transmitted from an originator may not be received by some nodes, which may result in a fork; however, such cases will be limited in normal situation.

D. Details of Changes to PoET Protocol

To avoid the misuse of RTS or its use as an attack vehicle, it too must be properly authenticated and protected by SGX. In particular, the generation of RTS follows the following protocol:

1) Enter PoET SGX enclave, assemble a proposed block (outside the enclave).
2) Generate a wait certificate containing the mean wait time ("LocalMean"), the generated (random) wait time, BlockID, BlockNumber, and ValidatorID.
3) Sign wait certificate with the private key of originating SG enclave (PSK) and wait for the generated random time (outside the enclave).
4) When the timer expires, broadcast ⟨waitCertificate, RTS, PPK⟩ over the P2P overlay network. Here PPK is the public key corresponding to the private key PSK.
5) Exit PoET SGX enclave.

In the above we still prepare the block as in the original protocol, but do not compute its signature for now. We will do that while waiting for SIFS period after sending RTS.

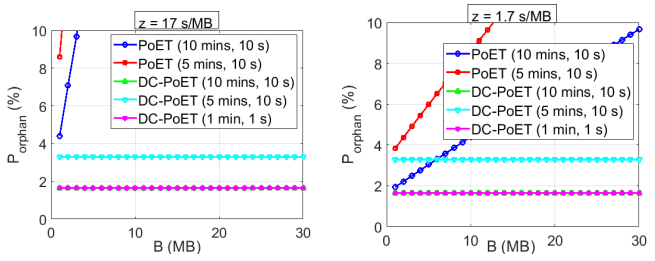
The block is assigned a BlockID and a BlockNumber. Although the latter is chosen locally by each node, it is intended to be eventually consistent with the global BlockNumber, or the Blockchain head. Both are sent out with RTS so as to unequivocally tie RTS and the block transaction that will come later. As in PoET, the enclave stores the wait certificate and the BlockNumber in a table so as to tie the two together. The certificate will be needed for block transmission.

When another node in the network (the "validator") receives the RTS message, it checks conducts the following checks before propagating it to its neighbors.

1) Verify the PPK belongs to a registered miner by checking the EndPoint registry.
2) Verify that the BlockNumber corresponds to the highest block number that the validator has. This ensures that a node does not accept bogus RTS messages.
3) Verify that the "LocalMean" in the wait certificate is correct by comparing against LocalMean computed locally.
4) Verify the sender has been winning elections according to the expected distribution using the z-test. This test thwarts early sending of RTS.
5) Exit PoET SGX enclave.

After sending the RTS, the originator does the following:

1) Wait for SIFS period (outside SGX) and then enter PoET SGX enclave.
2) Generate blockDigest of the block by hashing the block with SHA256 and then encrypting it with OSK. This is originator's private key (different from that of the SGX hosting it).
3) If an RTS is received by the originator during the SIFS period, restart the wait timer with doubled time period (exponential backoff). If RTS is received by another node that is waiting for its timer to expire, its timer is frozen.



(a) (b)

Fig. 3: Comparison of P_{orphan} for PoET and DC-PoET with (a) z = 17 s/MB and (b) z = 1.7 s/MB respectively. The values (x, y) within the brackets are T and δ_0 respectively. Notice that the *green* line is not visible, as the cases (10 mins, 10 s) and (1 min, 1 s) become identical because of same $\frac{\delta_0}{T}$.

4) Retrieve the wait certificate for the last BlockID and add to it transaction hash and the blockDigest.
5) Sign wait certificate with the private key of originating SG enclave (PSK).
6) When SIFS timer expires, broadcast ⟨waitCertificate, signature, block, OPK, PPK⟩ over the P2P overlay network. Here PPK and OPK are the public keys of the originating SGX enclave and and miner respectively.
7) Exit PoET SGX enclave.

When a validator receives the block, the actions are the same as in normal PoET except for two differences: (a) the check on LocalMean is skipped since it is already done in step 3 of RTS processing, and (b) if no prior RTS was received, the block is rejected. One of the steps in block handling procedure is to check if a block has arrived too early, and if so, hold it for some time.

E. Orphan Block Probability in DC-PoET

Given the small size and low processing cost of RTS, the orphan risk of the DC-PoET scheme is given by

$$P^{M}_{\text{orphan}} = 1 - e^{-\frac{\delta_0}{T_2}} \tag{5}$$

where T_2 is the average block interval. Notice that P^{M}_{orphan} is independent of B, and thus is only limited by the network latency.

We now compare the orphan risk of PoET and DC-PoET mechanisms using equation (3)-(5). Assume that T_1 and T_2 are the block intervals of PoET and DC-PoET respectively. With the same level of orphaning risk, $T_1 = T_2\left(1 + \frac{zB}{\delta_0}\right)$; thus, DC-PoET scheme reduces the average block time by a factor of $\left(1 + \frac{zB}{\delta_0}\right)$, i.e. the transaction rate is also improved by that amount. Assuming a block size B = 35 MB, z = 17 s/MB and δ_0 = 10 s [11], the transaction rate of DC-PoET scheme is increased by ∼60 times. This improvement will be even more with lower δ_0 and larger B.

Fig. 3(a) shows the comparison between PoET and DC-PoET where z is assumed to be 17 s/MB. The maximum value of block interval T is kept as 10 mins, which is the average block interval of Bitcoin network. The average transaction size is assumed to be 571 bytes [13]. From this figure we can observe that in PoET the P_{orphan} goes beyond 10% if the block size exceeds 3 MBs. On the other hand, in case of DC-PoET the P_{orphan} is independent of block size and remains at 1.6% when T and δ_0 are 10 mins and 10 secs respectively. If δ_0 is reduced from 10 secs to 1 sec, the the average block time can be reduced to even 1 min without compromising the orphan risk. On the other hand, reducing T to 5 mins in DC-PoET increases the orphan risk to ∼3.5%, but still remains much lower than PoET.

Fig. 3(b) shows the performance with z = 1.7 s/MB, which is much faster block validation rate as compared to the Bitcoin network, and is more applicable for permissioned blockchain. Notice that with z = 1.7 s/MB the orphan risk of PoET is lower as compared to Fig. 3(a), due to lower transit time of the block propagation but that of DC-PoET remains constant as evident from eqn (5). Therefore,

***Remark** 1: The orphan risk of DC-PoET is independent of the block size, and is limited by the overall network delay of small packets.*

IV. Analytical Modeling of DC-PoET Mechanism

In this section we present the analytical modeling of the DC-PoET considering the fact that there is a fixed number of nodes, all in *saturated* conditions, i.e. they all have blocks in their queue that they want to commit into the blockchain.

A. Modeling of DC-PoET[1]

In case of DC-PoET[1] we study the effect of the backoff process of a node using a two-dimensional Markov chain model, which is influenced by the pioneering work of Bianchi [14] for modeling IEEE 802.11 DCF.

1) Modeling the probability of conflict: Assume that there are n contending nodes each always having some blocks to commit. We also assume a discrete and integer timescale, i.e. the backoff timer decrements at the beginning of each time *slot*, where a slot can be either empty, contains a successful commit, or a conflict. According to the discussion in section III-E, the backoff window in the i-th step, denoted W_i, increases exponentially from a minimum size W_0 as $W_i = 2^i W_0$ until it reaches the maximum size W_m, after which it stays constant. We model this backoff process using Bianchi's Markov model [14] where each node is modeled by a pair of integers (i, k). The back-off stage i starts at 0 at the first attempt to transmit a RTS and is increased by 1 every time a transmission attempt results in a conflict, up to a maximum value of m. It is reset after a successful transmission. The counter k is initially chosen uniformly between $[0, W_i - 1]$, where $W_i = 2^i W_0$ is the range of the counter. The counter is decremented when there is no RTS/NAV reservation. The station transmits when k = 0.

The resultant Markov process is depicted in Fig. 4. Assuming the probability of conflict as p, the state-transition probabilities in Fig. 4 are given by:

$$\begin{aligned}
&P\{i,k|i,k+1\} = 1 && k \in (0, W_i - 2), i \in (0, m)\\
&P\{0,k|i,0\} = (1-p)/W_0 && k \in (0, W_0 - 1), i \in (0, m)\\
&P\{i,k|i-1,0\} = p/W_i && k \in (0, W_i - 1), i \in (1, m)\\
&P\{m,k|m,0\} = p/W_m && k \in (0, W_m - 1)
\end{aligned}$$

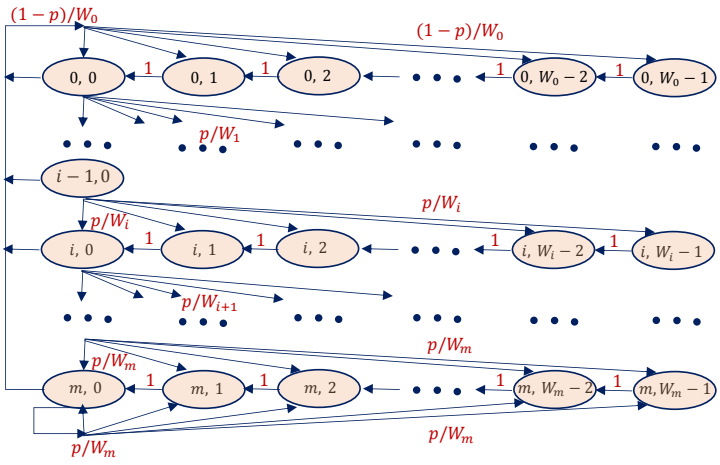


Fig. 4: Representation of DC-PoET[1] using Bianchi's model. The (i, k) pairs inside the circle represents the backoff stage, and the counter value in that stage respectively. W_0 is the minimum contention window size, which is doubled after every conflict will each reaches W_m.

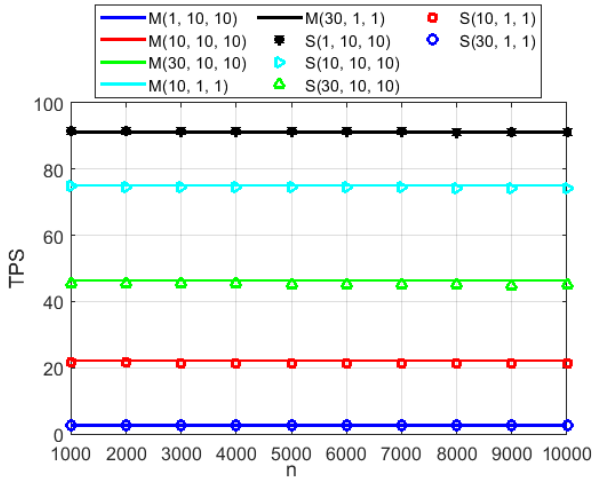


Fig. 5: Comparison of DC-PoET[1] with varying n; simulations vs modeling. In this figure M and S denote "modeling" and "simulation". The three parameters within braces are B (in MB), T (in minutes), and σ (in seconds).

Following the above derivations, the block commitment probability τ in a generic slot time can be written as [14]:

$$\tau = \frac{2(1-2p)}{(1-2p)(W_0+1)+pW_0\left(1-(2p)^m\right)} \tag{6}$$

Notice that τ depends on the values of p (assuming W_0 and m are predefined). The probability of conflict p is equal to the probability that at least one of the $n-1$ remaining nodes send RTS in that slot (assuming that an empty slot time $\approx \delta_{\text{RTS}}$). Thus

$$p = 1-(1-\tau)^{n-1} \tag{7}$$

The nonlinear equations(6)-(7) can be solved together for two unknowns τ and p.

As opposed to 802.11 DCF, in DC-PoET[1] W_0 is not a constant, but depends on the number of active miners; i.e. when the number of miners increase, W_0 will increase and vice versa. In practical scenario, W_0 can be adaptively updated similar to PoET [7]. Mathematically, if n miners choose a continuous random numbers $\{Y_1, Y_2, \ldots, Y_n \in U(0, M)\}$, then the block generation time $L_n = \min\{Y_1, Y_2, \ldots, Y_n \in U(0, M)\}$ follows an exponential distribution with mean $\frac{M}{n}$ (see [13] for proof). If T_{target} is the targeted block-generation time, i.e.

$$T_{\text{target}} \approx \frac{M}{2n} \Rightarrow M \approx 2nT_{\text{target}} \Rightarrow W_0 \approx \frac{M}{\sigma} = \frac{2nT_{\text{target}}}{\sigma} \tag{8}$$

2) Block Transaction Rate: Now we derive the expression of the transaction rate (i.e. the number of bits committed per second). Assume that the average slot time is T, which can either be empty, include a successful commit, or have a conflict. These can occur with probabilities $1-P_{\text{tr}}$, $P_{\text{tr}}P_s$ and $P_{\text{tr}}(1-P_s)$ respectively, where P_{tr} represents the probability that there is at least one commitment in a time slot and P_s denotes the probability of successful block commitment (i.e. no conflict). Hence,

$$T = (1-P_{tr})\sigma + P_{tr}P_sT_s + P_{tr}(1-P_s)T_c \tag{9}$$

where σ is the duration of an empty time slot and is $\approx \delta_{\text{RTS}}$. T_s is the average slot time because of a successful transmission, T_c is the average slot time because of a conflict. The expressions of T_s and T_c can be written as

$$\begin{aligned} T_s &= \mathsf{DIFS} + \zeta_{\text{RTS}} + \mathsf{SIFS} + \delta(B) \\ T_c &= \mathsf{SIFS} \end{aligned} \tag{10}$$

where $\delta(B)$ is the transit delay of a block with block size of B. Notice that the block transit delay $\delta(B) = \delta_0 + zB$ is a function of block size, i.e. transit delay increases with the increase in block size. Here ζ_{RTS} is the RTS preparation time by the enclave at the originator, expected to be quite small. Notice that unlike 802.11 DCF, the DIFS duration is not added with T_c, as every successful block commitment is preceded by a DIFS period (i.e. the MinimumWait time).

Also, P_{tr} can be written as:

$$P_{tr} = 1-(1-\tau)^n \tag{11}$$

The probability of success P_s is given by the probability that exactly one node commits, conditioned on the fact that there is at least one commit, i.e.,

$$P_s = \frac{\binom{n}{1}\tau(1-\tau)^{n-1}}{P_{tr}} = \frac{n\tau(1-\tau)^{n-1}}{1-(1-\tau)^n} \tag{12}$$

To calculate the transaction rate, we observe that during an average slot period T, a winner station commits a block with a probability of P_sP_{tr}. Hence, for an average block size of B (in bits), with a block transit delay of $\delta(B)$, the transaction rate (number of bits in unit time) is represented as

$$S_{\text{DC-PoET}^1} = \frac{P_sP_{tr}\delta(B)B}{T} \tag{13}$$

where δ, T_c, T_s and σ are in same units.

3) Validation of Analytic Model: To validate the modeling of DC-PoET^1, we develop a Matlab simulator imitating the proposed mechanism. Fig. 5 shows the validation of achievable transactions per second (TPS) with different number of miners. We assume the mean transaction size of 571 Bytes, which is representative of bitcoin network [13]. From this figure we can observe that because of the adaptation of W_0 in equation (8), the TPS stays almost identical with the variation of n. With the block size of 1 MB, T_{target} of 10 minutes, and σ = 10 secs, the TPS varies close to 5 TPS which is close to what the Bitcoin network achieve. However, with the increase in block size to 30 MB, and by decreasing T_{target} and σ to 1 minute and 1 sec respectively, the TPS can be increased to more than 95/sec.

B. Modeling DC-PoET^2

Let us now calculate the TPS of DC-PoET^2 mechanism. Assume that the nodes choose the random wait time within (0, M). As the waiting time is a chosen from a continuous distribution, ideally the conflict probability is 0. However, a small M may result in more RTS message transmissions; thus we keep M equal to $2nT_{\text{target}}$, which results in one committed block after $\sim T_{\text{target}}$ time instance. The maximum block interval in DC-PoET^2 is then given by

$$T = \mathsf{DIFS} + M/2 + \zeta_{\text{RTS}} + \mathsf{SIFS} + \delta(B) \tag{14}$$

and thus the throughput is given by $S_{\text{DC-PoET}^2} = \frac{B}{T}$.

***Remark** 2: With large block size B and a low probability of conflict, the maximum TPS of DC-PoET is limited by the block validating power z of the blockchain network.*

Proof 1: Notice that with low probability of conflict, the throughput of $S_{\text{DC-PoET}^1}$ converges to that of $S_{\text{DC-PoET}^2}$. In that case the throughput becomes

$$\begin{aligned} S_{\text{DC PoET}} &= \frac{B}{T} = \frac{B}{\mathsf{DIFS} + M/2 + \zeta_{\text{RTS}} + \mathsf{SIFS} + \delta_0 + zB} \\ &= \frac{B}{\Gamma + zB} = \frac{1}{\Gamma/B + z} \to \frac{1}{z} \end{aligned} \tag{15}$$

as the block size B becomes large. In equation(15), $\Gamma = \mathsf{DIFS} + M/2 + \zeta_{\text{RTS}} + \mathsf{SIFS} + \delta_0$. Thus the throughput is limited to the block validating power z of the blockchain. If the average number of transaction per block is χ, the transaction rate is given by TPS = $\frac{1}{\chi z}$.

V. Performance evaluation and Security analysis

A. Transaction speed of DC-PoET

For our performance evaluation we vary the block size from 1 MB to 30 MB, which is in the similar range assumed in [11]. The maximum block size in Bitcoin network is 1 MB; one of the reasons of this is the increase in orphan risk beyond few MBs as observed in Fig. 3(a)-(b). However, in DC-PoET the orphan risk is independent of block size, thus we can increase the block size further to improve the transaction rate; however, very large block sizes may lead to unacceptably large commit delays. In fact few blockchains like Multichain proposed larger blocks size; it limits the maximum block size to 32 MB, with a possibility of further increase [15]. However, keeping a large block size allows an attacker to send rogue, flooding messages which will unnecessarily increase the validation overhead of the network nodes. The mean transaction size χ is kept to 571 Bytes. The values of DIFS and SIFS are kept same as δ_0, whereas m is assumed to be 6. ζ_{RTS} is quiet low as compared to the network latency and so is neglected.

Fig. 6(a)-(b) show the variation of TPS with different block sizes in case of DC-PoET^1 and DC-PoET^2, where the number of miners are kept as 10k. z is assumed to be 17 s/MB. From this figure we can observe that the TPS of both DC-PoET^1 and DC-PoET^2 are almost identical as far as T_{target} is more than 1 minute. This is because of the fact that with T_{target} equal to 1 minute or more, the probability of conflict is extremely small (below 2%) and so both schemes provide almost similar throughput. However, reducing T_{target} significantly results in more chance of conflict, which leads to more backoff, retransmission of RTS; thus results in reduced throughput in case of DC-PoET^1 as compared to DC-PoET^2, as observed from Fig. 6(b). However, reduced T_{target} indeed result in more TPS for both the schemes. Although there is (almost) no conflict in DC-PoET^2, T_{target} should not be reduced significantly as this will result in large number of RTS transmissions from multiple nodes.

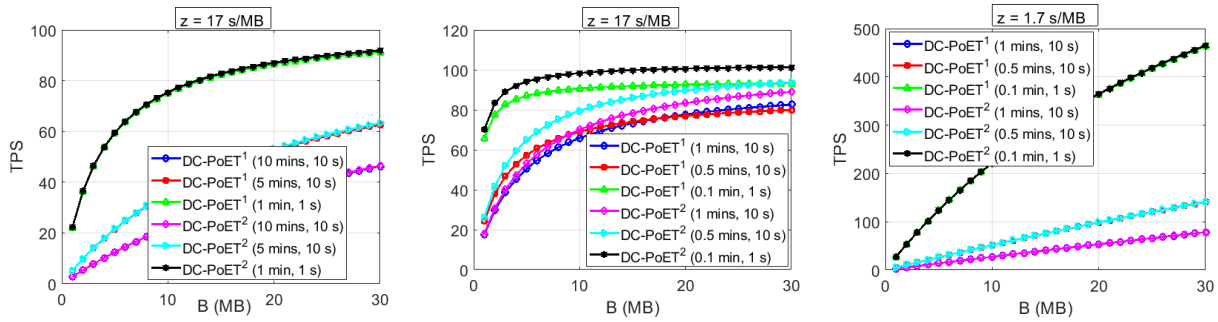
From Fig. 6(a) we can also observe that when when T_{target} and δ_0 are 10 mins and 10 secs respectively, the achievable throughput goes upto 46 TPS with a block size of 30; while reducing them by $\frac{1}{10}$-th increases the throughput almost 95 TPS. The throughput can be improved further by reducing T_{target} further, which is observed from From Fig. 6(b).

Fig. 6(c) shows the transaction speed of two protocols with z = 1.7 s/MB. From this figure we can observe that with B = 30 MB, the TPS of DC-PoET can reach to $\sim$465 TPS and can go even higher with higher B. In fact as mentioned in ***Remark 2***, the TPS of DC-PoET with z = 1.7 s/MB can reach up to $\frac{1}{\chi z}$, i.e. $\sim$1030 transactions per second with larger blocks.

Notice that even if DC-PoET^2 performs almost similar or better than that of DC-PoET^1, the applicability of DC-PoET^1 is more generic as compared to that of DC-PoET^2. For example, the backoff mechanism of DC-PoET^1 can also be used in PoW and its variants in a permissionless blockchain, however, DC-PoET^2 strictly requires TEE.

B. Security of DC-PoET with uncompromised TEE

We first delve into the scenario where the TEEs are not compromised, but still some miners are able to launch successful attacks by colluding. Two most fundamental attacks on blockchain network are **double-spending attacks** and **selfish attacks** [16], which are applicable to DC-PoET as well. In a double-spending attack [1], the attacker purchases something from a seller using transaction $\mathbb{T}$. Assume that before the transaction, the last block of the main-chain was B_0. After the transaction, the honest miners add their blocks $H_1, H_2, \ldots$ after B_0, where the block H_1 carries transaction $\mathbb{T}$. On the other hand, the attackers can form a pool and privately mine an alternate fork after B_0, producing blocks $A_1, A_2, \ldots$. In the attackers private chain, suppose A_1 carries a transaction $\widehat{\mathbb{T}}$



(a) (b) (c)

Fig. 6: Comparison of DC-PoET^1 and DC-PoET^2 with varying B. The values (x, y) within the brackets are T_{target} and δ_0 respectively. Notice that in (a) and (c) the performance if DC-PoET^1 and DC-PoET^2 is almost identical, so the lines come on top of each others.

that conflicts with $\mathbb{T}$ in the main-chain; i.e. the attacker can avoid paying the seller through transaction $\widehat{\mathbb{T}}$. To thwart the attack, the seller waits for z blocks after B_0 before releasing the item. On the other hand, to make a successful double-spending attack, the attacker keeps $A_1, A_2, \ldots$ private and commits them into the main-chain when its length grows to atleast $z+1$ and is also longer than the honest miner's main chain. At this stage, the honest miners will adopt the blocks $A_1, A_2, \ldots$ that has the conflicting transaction $\widehat{\mathbb{T}}$, which leads to a successful double-spending attack. Notice that a larger value of z decreases the probability of a successful double-spending attack.

On the other hand, in a selfish mining attack an attacker tries to improve his relative proportion of blocks (or rewards) by wasting resources of the honest miners. In [17] the authors have shown that an attacker pool with minority population can still launch selfish mining attack in a blockchain. The crux of this attack lies in the following: a mining pool consisting of a bunch of selfish miners collude and keep their discovered blocks private. When the honest miners continue to mine on the public chain, the selfish mining pool mine but keep the blocks on their own private branch. If the pool discovers more blocks, it develops a longer lead on the public chain; when the public branch approaches the pool's private branch in length, the selfish miners commit their blocks from their private chain to the public. As the length of their private blocks is longer than the current public chain, all the blocks in the public chain are replaced. If this happens very often, then the honest miners (a) may stop mining as they spend their electricity for mining blocks that are replaced by the selfish miners, or (b) they start joining the selfish pool for profit, which monopolizes the process in favor of the selfish mines.

In both of these above mentioned attacks, a minority attackers pool cannot create a longer fork than the honest miners *in long-term*, however, the attacker can still get lucky to make some short-term gains. We argue that the success from such double-spending and selfish mining can be very limited in schemes like DC-PoET. First, in DC-PoET the TEE can put a timestamp at the time of the certificate generation with the attestation; the most recent block committed by a miner cannot be older than some threshold. Thus the selfish miner takes more risk by holding the most recent block.

Second, we show that with a minority pool, the chances that the attacker can produce more blocks than the majority honest miners decreases exponentially with time, as well as with the number of blocks. Suppose the number of miners in the honest and selfish pools are p_h and p_s respectively. In [13], the authors have shown that the inter-arrival process of the blocks in these two pools follows an exponential process with an arrival rate of $r_1 = \frac{p_h}{M}$ and $r_2 = \frac{p_s}{M}$ respectively (where the nodes waiting time is uniformly distributed in between 0 and M), i.e. the arrival process follows the Poisson process with the same arrival rates. We assume that $X = \text{Poisson}(r_1 t)$ and $Y = \text{Poisson}(r_2 t)$ are independent random variables with arrival rates r_1 and r_2, where $r_1 > r_2$ and t is a time interval. Then the probability that after an interval t, the selfish miners chain will remain longer than that of the honest miners is given by [18]:

$$\text{Prob}(Y - X \geq 0) \leq e^{\left(\sqrt{r_1 t} - \sqrt{r_2 t}\right)^2} = e^{(\sqrt{r_1} - \sqrt{r_2})^2 t} \quad (16)$$

which shows that the chances drops exponentially with t. In fact $Y - X$, which is the difference between two independent Poisson-distributed random processes follow a Skellam distribution having the probability mass function (PMF) as follows:

$$\text{Prob}\left(Y - X = k\right) = e^{-(r_1 t + r_2 t)} \left(\frac{r_2}{r_1}\right)^{k/2} I_k\left(2\sqrt{r_1 r_2} t\right) \quad (17)$$

where $I_k(z)$ is the modified Bessel function of the first kind [19]. Fig. 7 shows the variation of PMF with different k and different proportion of attacker and honest miners. In Fig. 7 t is assumed to be 5 hours, and the overall block generation rate is assumed to be 1/10 minutes. From this figure we can observe that the probability that a pool of minority attackers to produce a chain that is longer than the honest miners remains below 6% even with an attacker pool of 45% miners, and reduces almost exponentially with increasing difference in the number of blocks between the attackers chain and the main chain. In fact with 30% miners, the attackers chain can grow longer than the main chain with a probability of less than 1%.

Third, as opposed to bitcoin mining, in DC-PoET the miners do not spend energy/power for mining. Thus there is no point for the honest miners to stop mining even if they start losing.

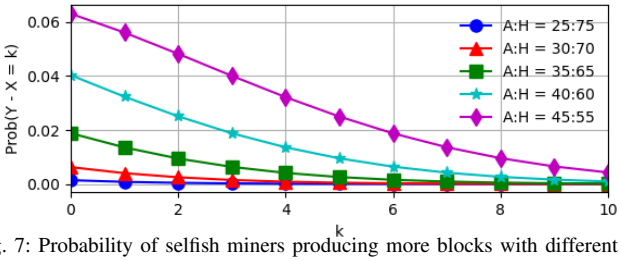


Fig. 7: Probability of selfish miners producing more blocks with different k, and different proportion of attacker (A) and honest (H) miners.

So the chances of monopolize the chain in favor of selfish miners is much limited.

Like any other blockchain networks, PoET network with or without modifications is susceptible to **eclipse** attacks [20] where an adversary takes over all connections to and from a victim node. However, such an attack is unlikely in permissioned network, particularly in an organizational network.

C. Robustness of DC-PoET Under Compromised SGX

In this section, we briefly discuss how DC-PoET will behave in an environment where some SGX nodes may be compromised and deliberately try to exploit the protocol to their advantage. We do not consider Denial-of-Service (DoS) attacks here as they are possible in most situations and can be handled using standard mechanisms [21].

The main protection offered by PoET against compromised nodes includes the following: (a) Use of two PKIs with secret keys PSK (for the enclave) and OSK (for the originator), both of which need to be stolen, (b) agreement on BlockNumber to stay on the main chain w/o majority collusion, (c) z-test to ensure that the waiting times conform to the desired distribution, and (d) holding off of early arriving blocks.

With DC-PoET, the RTS phase checks the LocalMean and also does the z-test on the waiting time. Thus if a compromised node sends RTS too early, it will get through but subject to z-test, but an early block transmission will still be held back. The reception of an early RTS by an honest node (during its SIFS period) will cause it to backoff and double its waiting time. The compromised node could then safely increase its block transmission time so as to still pass tests (c) and (d). With only one compromised node, this attack rapidly becomes unlikely as the network size increases due to the z-test on RTS as well; also, the honest validator will disallow repeated winning by the same node. For large scale collusion, the result in [7], should still apply.

Another potential situation is where a compromised node exploits an arriving RTS as cue to send its own (likely conflicting) block immediately thereby making the originator's block stale; if no RTS was sent thus far, it is sent first to avoid reject of the block. This attack too becomes rapidly ineffective with increasing number of nodes due to checks (c) and (d). Also, the result in [7] applies for large scale collusion.
among all neighbors of a node. This is similar to the eclipse

A compromised intermediate node could forward or hold back RTS based on the identity of the originator (and in DC-PoET^2 based on the waiting time), thereby favoring or working against a given node. To succeed, this requires collusion attack, except it depends on compromise to selected overlay nodes rather than an indiscriminate blocking of IP addresses.

VI. Conclusion

We have proposed a modification to the proof of elapsed time (PoET) blockchain consensus protocol to substantially reduce the orphan blocks and thus improve the transaction rate. The proposed mechanism achieves ∼465 transaction per second with 30 MB block size and even more for larger blocks. It is also robust against the double spending and selfish-mining attacks. Its robustness against node compromises also remains unaltered. In the future, we will examine ways to make the protocol even more efficient by building the blockchain overlay network more intelligently and dynamically. We will also analyze the performance of DC-PoET with more general waiting-time distributions than uniform.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2009. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf

[2] S. Liu *et al.*, "XFT: practical fault tolerance beyond crashes," in *USENIX OSDI*, K. Keeton *et al.*, Eds., 2016, pp. 485–500.

[3] M. Castro *et al.*, "Practical byzantine fault tolerance and proactive recovery," *ACM Trans. Comput. Syst.*, vol. 20, no. 4, pp. 398–461, 2002.

[4] T. T. A. Dinh *et al.*, "BLOCKBENCH: A framework for analyzing private blockchains," in *ACM SIGMOD*, 2017, pp. 1085–1100.

[5] B. Curran, "What is proof of elapsed time consensus? (poet) complete beginner's guide," 2018. [Online]. Available: https://blockonomi.com/proof-of-elapsed-time-consensus/

[6] R. P. Pires, "Distributed systems and trusted execution environments: Trade-offs and challenges," *arXiv preprint arXiv:2001.09670*, 2020.

[7] L. Chen *et al.*, "On security analysis of proof-of-elapsed-time (poet)," in *SSS*, vol. 10616, 2017, pp. 282–297.

[8] Y. Lindell, "The security of intel sgx for key protection and data privacy applications," 2018. [Online]. Available: https://cdn2.hubspot.net/hubfs/1761386/security-of-intelsgx-key-protection-data-privacy-apps.pdf

[9] V. Costan *et al.*, "Intel sgx explained." *IACR Cryptol. ePrint Arch.*, vol. 2016, no. 86, pp. 1–118, 2016.

[10] E. Tromer *et al.*, "Efficient cache attacks on aes, and countermeasures," *Journal of Cryptology*, vol. 23, no. 1, pp. 37–71, 2010.

[11] P. R. Rizun, "Subchains: A technique to scale bitcoin and improve the user experience," *Ledger*, vol. 1, pp. 38–52, 2016.

[12] V. K. Bagaria *et al.*, "Prism: Deconstructing the blockchain to approach physical limits," in *ACM SIGSAC*, L. C. et al, Ed., 2018, pp. 585–602.

[13] S. Kasahara *et al.*, "Effect of bitcoin fee on transaction-confirmation process," *Journal of Industrial and Management Optimization*, vol. 15, pp. 365–386, 2019.

[14] G. Bianchi, "Performance analysis of the ieee 802.11 distributed coordination function," *IEEE Journal on Selected Areas in Communications*, vol. 18, pp. 535–547, 2000.

[15] "Maximum blocksize - post alpha26." [Online]. Available: https://www.multichain.com/qa/3780/maximum-blocksize-post-alpha26

[16] G. Bissias *et al.*, "Bobtail: Improved blockchain security with low-variance mining," in *NDSS*, 2020.

[17] I. Eyal *et al.*, "Majority is not enough: Bitcoin mining is vulnerable," in *FC*, vol. 8437, 2014, pp. 436–454.

[18] G. M. Kamath *et al.*, "Optimal haplotype assembly from high-throughput mate-pair reads," *CoRR*, vol. abs/1502.01975, 2015. [Online]. Available: http://arxiv.org/abs/1502.01975

[19] J. G. Skellam, "The frequency distribution of the difference between two poisson variates belonging to different populations," *Journal of the Royal Statistical Society*, vol. 109, no. 3, p. 296, 1946.

[20] E. Heilman *et al.*, "Eclipse attacks on bitcoin's peer-to-peer network," in *USENIX Security*, J. Jung *et al.*, Eds., 2015, pp. 129–144.

[21] T. Mahjabin *et al.*, "A survey of distributed denial-of-service attack, prevention, and mitigation techniques," *International Journal of Distributed Sensor Networks*, vol. 13, no. 12, pp. 1–33, 2017.