Brief Announcement Blockguard: Adaptive Blockchain Security

Shishir Rai, Kendric Hood, Mikhail Nesterenko(✉), and Gokarna Sharma

Department of Computer Science, Kent State University, Kent, OH 44242, USA
{srai,khood5}@kent.edu, {mikhail,sharma}@cs.kent.edu

Abstract. We change the security of blockchain transactions by varying the size of consensus committees. To improve performance, such committees operate concurrently. We present two algorithms that allow adaptive security by forming concurrent variable size consensus committees on demand. One is based on a single joint blockchain, the other is based on separate sharded blockchains. For in-committee consensus, we implement synchronous Byzantine fault tolerance algorithm (BFT), asynchronous BFT and proof-of-work consensus. We evaluate the performance of our adaptive security algorithms.

1 Definitions and Committee Consensus Algorithms

A set of n *peer processes* (or *peers*) forms a network to maintain the blockchain. The *blockchain* is a sequence of blocks or transactions. We use the terms interchangeably, i.e. we assume that a block contains a single transaction. A *transaction* is a unit of blockchain recording. Each subsequent transaction is cryptographically linked to the previous one. The first transaction in the blockchain is the *genesis* transaction. Peers communicate through broadcasts. Message delivery is FIFO. There is no message loss. Messages cannot be forged. Peers are either *honest* or *Byzantine*. A set of peers that cooperate to approve a transaction despite actions of Byzantine peers is a *consensus committee*.

Sharding. A *(recording) group* is a set of processes that maintain a single blockchain. There are as many groups as there are separate blockchains. In case of sharding, a peer in the consensus committee that approves a certain transaction in a blockchain does not necessarily belong to the group that records it. However, a peer may belong to only one recording group and only one consensus committee at a time.

PBFT and SBFT. In PBFT [2] The committee of peers elect the *leader*. The leader runs consensus on every transaction. It initiates several message exchanges with other committee peers. A non-leader Byzantine peer may delay messages or send incorrect messages. A Byzantine leader may temporarily block the consensus by sending different messages to different peers or not sending

See [4] for full text of the paper.

M. Ghaffari et al. (Eds.): SSS 2019, LNCS 11914, pp. 296–300, 2019.
https://doi.org/10.1007/978-3-030-34992-9_23

messages altogether. In either case, the honest peers discover the Byzantine leader and replace it by forcing a *view change*. PBFT is guaranteed to withstand up to $f < n/3$ Byzantine peers regardless of the message propagation delay. The operation of SBFT [1] is similar to PBFT. This algorithm relies on at least one honest peer confirming the transaction. However, it assumes that there is a bound on communication delay between honest peers. If a message is not received after a certain delay, it is guaranteed never to arrive. Thus, the algorithm has to delay to ascertain this lack of message receipt. In practice this may make SBFT slower. However, it has higher resilience threshold. It can tolerate up to $f < n/2$ Byzantine peers.

PoW. We implement proof-of-work consensus similar to Nakamoto [3]. To attach a new transaction to the blockchain, a peer *mines* the transaction by solving a computationally intensive task that links the new and previous transaction. Several peers may mine transactions concurrently. This is a *fork* in the blockchain. A branch of a fork may be extended by the addition of mined transactions on top of the current block. The shorter branch is discarded. PoW consensus operates correctly provided that the computational power of honest peers exceeds that of Byzantine peers. If peers have the same computational power, PoW consensus tolerates up to $f < n/2$ Byzantine peers.

2 The Adaptive Security Problem and Solutions

The Problem. *The* Adaptive Security Problem *requires, as a solution, an adaptive security algorithm, to assign committees to the transactions such that each committee satisfies the transaction security level.* We consider an adaptive security algorithm that selects appropriate size committees and processes transactions with as much parallelism as possible. We present two such algorithms: *Composite Blockguard* and *Dynamic Blockguard*.

Composite Blockguard Adaptive Security Algorithm. In this algorithm, peers are divided into storage groups maintaining independent blockchains. The algorithm maintains a list of idle groups and pending transactions. Once a new transaction arrives or a consensus committee is done, Composite Blockguard finds appropriate number of available groups, forms a consensus committee to process the next pending transaction and dispatches the transaction. If not enough idle groups are available, the pending transactions wait.

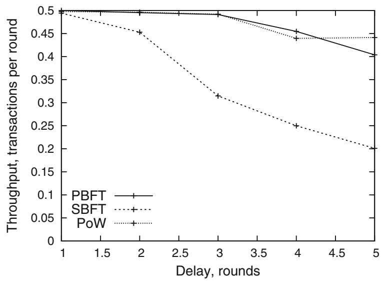
Dynamic Blockguard Adaptive Security Algorithm. This algorithm has a single blockchain and thus a single recording group. A consensus committee is selected out of this group of peers. Multiple consensus committees may operate concurrently if their members do not intersect. This means that the committees have to concurrently write to the same blockchain. To ensure the integrity of the blockchain, the computation proceeds by alternating two stages: consensus stage and recording stage. In the *consensus* stage, committees agree on blocks to be written to the blockchain. Every committee must reach consensus before any committee may proceed to the next stage. In the *recording* stage, each committee broadcasts the transaction to the group maintaining the blockchain. That is, they broadcast it to the whole network. Each written transaction is cryptographically linked to all the written transaction in the previous recording stage. This way, the resultant blockchain is a series-parallel graph. *Committee selection window* is the set of unique peers that published in the blockchain most recently. Committee peers are picked at random from the committee selection window.

3 Performance Evaluation

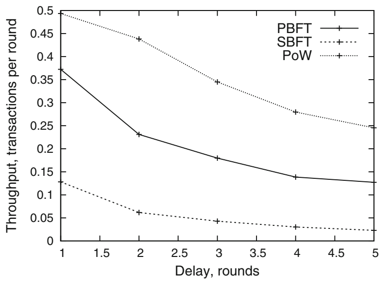
Setup. We evaluate the performance of Composite and Dynamic Blockguard using abstract simulation. The behavior of each algorithm is represented as a sequence of rounds. In every round, each peer may receive a single new message, do local computation and send messages to other peers.

Byzantine peers' goal is to successfully commit a fraudulent transaction to the blockchain, we model this as follows. A committee is *reliable* if the number of Byzantine peers in it does not exceed its tolerance threshold, *defeated* otherwise. The tolerance threshold is 1/3 for PBFT and 1/2 for SBFT and PoW. Defeated committees commit only fraudulent transactions to the blockchain, and reliable committees never commit fraudulent transactions. Byzantine leaders propose only fraudulent transactions. If a fraudulent transaction is proposed in a reliable committee then a view change occurs. This repeats until a non-byzantine leader is found. In PoW, if a Byzantine peer is the first to mine in a reliable committee then nothing is recorded and mining restarts.

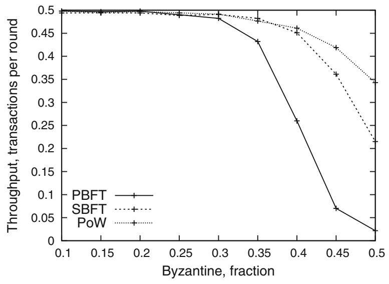
Experiment Parameters and Evaluation Metrics. Unless stated otherwise, in the below experiments, the parameters are set as follows. The fraction of Byzantine faults is $n/10$. The number of peers in the network is 1024. There are 1000 rounds in a computation. Each data point is the average of 10 computations. A new transaction is generated every two rounds. We have 5 security levels. The highest security level is the 5-th level which contains the whole network. Each lower level contains half of the peers of the higher level. We use geometric distribution to select the security level of newly generated transaction. In PoW, we use binomial distribution to determine the number of rounds it takes the peers to mine a transaction. The mode, i.e. most frequently occurring value, is 5 and variance 2.5. We vary maximum message delay and the fraction of Byzantine peers in the network. We consider a transaction approval as a consensus. We compute the following metrics. *Throughput* is the number of consensuses per round. Consensuses of defeated committees are not counted. *(Transaction) waiting time* is computed as follows. For coordinated consensus algorithms, i.e. PBFT and SBFT, it is the number of rounds from the moment the transaction is generated till the first peer determines that the transaction is committed. For PoW, it is the time for this transaction to be mined. The waiting time for transactions of defeated committees is counted.



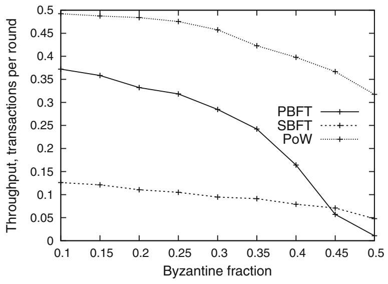
(a) Throughput, Composite Blockguard, varying delay



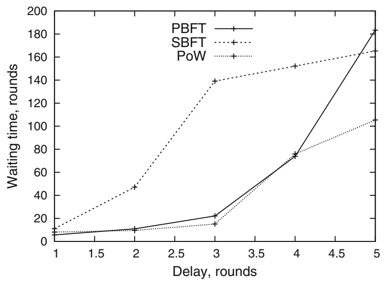
(b) Throughput, Dynamic Blockguard, varying delay



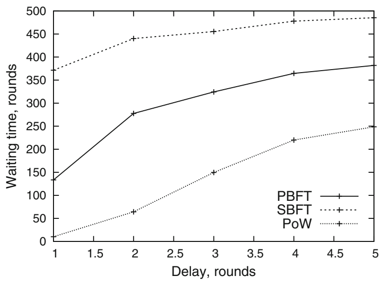
(c) Throughput, Composite Blockguard, varying Byzantine fraction



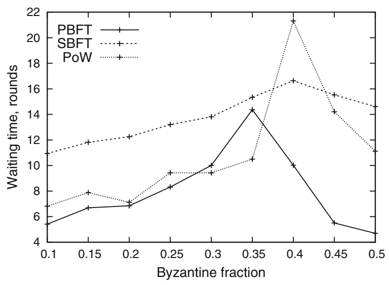
(d) Throughput, Dynamic Blockguard, varying Byzantine fraction



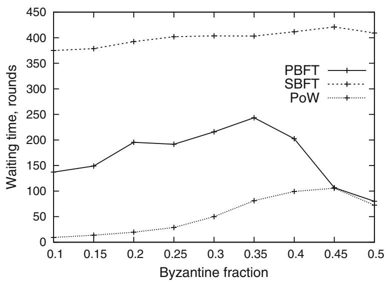
(e) Waiting time, Composite Blockguard, varying delay



(f) Waiting time, Dynamic Blockguard, varying delay



(g) Waiting time, Composite Blockguard, varying Byzantine fraction



(h) Waiting time, Dynamic Blockguard, varying Byzantine fraction

Fig. 1. Performance of Blockguard adaptive security algorithms.

Algorithm Performance Experiments. The results of the performance evaluation of the adaptive security algorithms are shown in Fig. 1. Figures 1a and b demonstrate how throughput depends on the network delay for Composite and Dynamic Blockguard respectively. As network delay increases, the throughput declines. However, different consensus committees react to this increase differently. PBFT has the best performance and lowest decline since the committees just wait for the actual messages to arrive. SBFT exhibits the most sensitivity to the network delay. The reason is that SBFT has to wait for the maximum delay to determine that the message is not coming. Let us discuss Figs. 1c and d. It shows that the performance of Composite and Dynamic Blockguard decreases as the fraction of Byzantine peers in the network increase. This is due to Byzantine peers slowing down the consensus algorithms. PBFT suffers the most since its tolerance threshold is only a third of the peers.

Figures 1e and f show the dependency of transaction waiting time on network delay. As expected, the waiting time increases with delay. SBFT is the most vulnerable to this increase since it has to wait for maximum delay time. Figures 1g and h show how waiting time varies with the fraction of Byzantine peers. Let us explain the trends in the data. As the consensus committee approaches its resiliency threshold, the number of view changes or repeated transaction mining increases which increases the transaction waiting time. If the fraction is away from this threshold, the committees are either reliable or defeated. In either case the waiting time is relatively low. Thus, there is a peak near $n/3$ for PBFT and near $n/5$ for SBFT and PoW. This trend is less pronounced in Dynamic Blockguard since it is masked by synchronization across consensus committees in the same stage.

The results of our experiments indicate that both Composite and Dynamic blackguard algorithm provide adaptive security with a trade-off between performance and security parameters.

References

1. Abraham, I., Devadas, S., Nayak, K., Ren, L.: Brief announcement: practical synchronous Byzantine consensus. In: DISC, pp. 41:1–41:4 (2017)
2. Castro, M., Liskov, B.: Practical Byzantine fault tolerance and proactive recovery. ACM Trans. Comput. Syst. **20**(4), 398–461 (2002)
3. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008). http://bitcoin.org/bitcoin.pdf
4. Rai, S., Hood, K., Nesterenko, M., Sharma, G.: Blockguard: adaptive blockchain security. arXiv e-prints arXiv:1907.13232, July 2019