Ethereum's GHOST protocol and its vulnerabilities to the Private and Balance Attack

Aidan Walsh
abwalsh@princeton.edu

Mohit-Pal Singh
mohitpal@princeton.edu

Tinotenda Chinamora
ttc2@princeton.edu

Abstract

With many blockchain protocols proposed and in use, an analysis on the security and characteristics of these protocols is essential to developing a robust and practical one. Examples of these protocols include GHOST (Greedy Heaviest Observed Sub-Tree), Longest-Chain, Fruit-Chains, Streamlet, and Prism, where Ethereum implements a modified version of GHOST and Bitcoin uses a Longest-Chain rule. This paper serves as a literature review and in-depth analysis of the GHOST protocol, as implemented by Ethereum. We begin by describing the GHOST protocol, its Ethereum version, then the nuances within Ethereum's version. We then describe the attacks that we attempt to perform on the GHOST protocol, and show our results and analysis. [1]

1 Background and Discussion

1.1 GHOST Protocol

The Bitcoin protocol's throughput can be increased by altering some key parameters, the mining difficulty (which determines the mining rate λ) and the size of blocks mined. Decreasing mining difficulty increases λ, and increasing the size of blocks mined increases Δ—both approaches increase the product $\lambda\Delta$ and this decreases the hash power that can be tolerated from adversaries, leading to more forking in the blockchain, undermining security. This is the primary shortcoming of the longest chain protocol. The Ghost (greedy heaviest subtree observer) protocol attempts to pick the chain miners mine on by measuring its quality using the weight of the subtree that supports a chain. More clearly: for each block, we calculate the weight of the subtree that is rooted at that block (which is the number of blocks included in the subtree, including the block itself). Now the chain to mine on can be determined by starting at the genesis block and picking the heavier subtree when we encounter forks.

In the context of the private attack, this protocol helps with safety. Network delay means that honest miners are mining on stale blocks in the longest chain protocol, so this allows the adversaries to pull ahead mining privately, and as $\lambda\Delta$ grows, the power needed for adversaries to successfully launch a private attack reduces (since there is a greater chance that honest miners are wasting hash power mining on stale blocks that have already been extended, creating a natural, non-adversarial forking instead of using all their power to extend blocks on the longest chain). For the GHOST protocol though, the weight of the honest subtree is proportional to the honest mining rate, as is the weight of the adversarial chain/subtree proportional to the adversarial mining rate, and this provides safety against a private attack if a majority of the hash power is honest and follows the ghost protocol (Sompolinsky and Zohar, 2013).

1.2 Balance Attack

The balance attack is also another attack that violates the safety property. An adversary can mine and release blocks to balance the heights of the two chains (each with mixed honest and adversarial blocks), switching honest miners back and forth on both chains and never letting a single chain stabilize as the longest chain miners are mining on. The threat to safety is that the ledger entries keep changing as the longest chain changes with the adversary's efforts to balance the two chains and don't stabilize, and this balancing need only be done for longer than confirmation depth k to break the safety of a protocol. As miners are observing this switching in chains, controlled by an adversary, their honest mining power could be split across both, due to the adversary's partitioning ability. For example, the adversary could conduct a partitioning attack (Saad et al., 2019) on the blockchain network such

[1] Our simulation can be found at https://github.com/Aidan-Walsh/GHOST-Protocol-and-Vulnerabilities

that 20 percent of the miners are only able to communicate amongst each other, leaving the other 80 percent mining on their own chain. This reduces the rate of growth of both chains, and a private attack could be launched in concert on the blockchain to outrun the stunted-growth chains attacked by the balance attack(we do not simulate this combination of attacks). This enables a version of the safety attack on the *k*-deep confirmation rule described in the regular private attack in Appendix A. Finally, GHOST is not secure when $\lambda\Delta$ is large as the adversary can mine the required blocks at the required rate within Δ, so its throughput is limited by security and safety concerns for this attack (Natoli and Gramoli, 2016).

1.2.1 Ethereum's GHOST & Casper Upgrade

Ethereum's modifications to the GHOST protocol allowed the network to achieve consensus by considering the work done on uncle (non-included) blocks. The protocol's inclusion of uncle blocks aims to improve security as it is more expensive for an adversary to outpace the honest chain. Ethereum's adaptation of GHOST aimed to reduce centralization risks by allowing more frequent block production without significantly increasing the risk of a fork, and by rewarding miners, it "creates an incentive for all crypto miners in the chain", which incentivizes participation, leading to higher network throughput and quicker validation times (Wu, 2023). This protocol laid the foundations for Ethereum's move toward a PoS protocol, in which the Casper upgrade has fundamental changes where miners are replaced by validators who are required to stake their Ether as a form of security deposit, and introduces slashing as a penalty to deter malicious behavior.

2 Methodology

All simulations were done in python where data structures for both the blocks and miners were instantiated. Tests were run locally and ranged in duration from 2 minutes to 2 days. To simulate the GHOST protocol, honest miners and adversarial miners would mine, starting on the Genesis, until 500 to 10,000 blocks (B = 500 and B = 1000) were created. Before running the simulations, certain parameters were tuned. These parameters are the following: number of blocks to be mined (B), probability that an honest miner mines on the tip of the heaviest chain as opposed to continuing their mining on their current block (P_1), probability that an adversarial miner mines on the tip of the heaviest chain as opposed to continuing their mining on their current block (P_2), proportion of adversarial hash power (β), and total number of mining nodes (N). These probabilities are used to simulate incentives and potential adversarial strategies. Refer to Appendix B for a more detailed explanation of our implementation and methodology.

2.1 Simulating Incentives

We use P_1 to simulate honest miner incentives to the best of our ability. For example, when $P_1 = 0.5$, this means that there is a 50 percent chance that a miner will move over to mine on the new heaviest chain, when the chain updates. Alternatively, they would keep mining where they are, because Ethereum rewards "uncle" blocks. With the Casper update, adversaries may be disincentivized to create malicious blocks when they add blocks to the Ethereum blockchain, but this is not modelled in the simulation - we assume that the adversary is willing to accept losses in order to achieve its goal: minimize throughput, maximize "tip switches", and minimize chain quality. Furthermore, Casper totally changes GHOST, therefore we do not simulate it.

3 Results and Analysis

3.1 Private Attack

Refer to Appendix C for the figures.

Figure 1 shows relatively noisy results (refer to figure 3 for cleaner results). Since $P_2 = 0$, the adversary is only mining on their own chain. We see that when the adversary hash power approaches 50 percent, the chain quality sharply reduces to 0. This makes sense because when the adversary has the majority of the hash power, their chain will end up being heavier and their chain is entirely dominated by themselves. Interestingly, when $P_1 = 0.2$, the decrease starts sooner and is less sharp, **thus showing that a greater P_1 indicates more resistance to a private attack. Since $P_2 = 0$, this is a pure private attack, thus GHOST is very resistant to this attack**.

In figure 2, like figure 1, we see that a higher P_1 leads to a greater chain quality, however, the change to chain quality is far more gradual. We adjusted P_2 to 0.5 from 0, which means that the adversary will not just mine on its own private chain, but will mine on the actual tip from time to time. This modification to the private attack shows

that it is far more effective - chain quality is only 0.5 when adversary hash power is 0.22 for $P_1 = 0.2$. This modification to P_2 only makes the adversary behave more normally, thus, more normal behavior will be more beneficial for the adversary. **Thus, if the adversary only mines on the tip, we can infer that this will be very detrimental to the chain quality, especially if GHOST incentivizes people to not mine on the tip of the heaviest chain**.

3.2 Balance Attack

Figure 4 indicates that as the adversary partition power increases to 50 percent, the number of tip conflicts and switches approaches 0.15. This is very dangerous since the tip is switching 15 times for every 100 blocks inserted. For $P_1 = 1.0$, the maximum approaches 0.15, but for $P_1 = 0.5$, it approaches 0.11. Thus, the more frequently that an honest miner mines on the tip of the heaviest chain, the more difficult it is for the adversary to cause tip conflicts. **GHOST most likely has a P_1 closer to 0.5, thus, GHOST reduces the number of tip conflicts in a partitioning attack, but is still vulnerable to switches 10 percent of the time**.

In figure 5, the adversary also has 20 percent mining power, so has a greater ability to maximize the number of tip conflicts and switches, since it can mine on any branch of its choosing. So, we see the number of tip switches massively jump up to 0.35 for $P_1 = 1.0$ and 0.30 for $P_1 = 0.5$, and the graph for $P_1 = 0.5$ is far wider, thus the adversary has the ability to cause tip conflicts even with less partitioning power. For example, with a partitioning power of 25 percent, in this case, the adversary achieves tip conflicts/switches 10 percent of the time. **We conclude: even with relatively smaller partitioning power and hashing power, the adversary can achieve significant tip switches above 10 percent**.

We see the exact same trend from figure 4 to 5, as from figure 5 to 6. In figure 6, the tip conflicts approach 0.6, and the standard deviation is far greater, because the adversary has more hash power: 0.4. When $P_1 = 0.5$, $\beta = 0.4$, and partitioning power is only 10 percent, the adversary is able to achieve tip conflicts 30 percent of the time! Thus, **with GHOSTS's incentives, as adversary hash power increases, the number of tip conflicts increases to dangerous values, and the necessity for more partitioning power significantly reduces.**

For figure 7, we use our results from figure 4, but use them to calculate throughput. **We see that the throughput is massively larger for $P_1 = 1.0$, where its minimum value is 0.5, but becomes 0.25 for $P_1 = 0.5$.** Here, the adversary has no mining power, and is able to halve the throughput with only maximized partitioning power. Intuitively, this makes sense since the adversary can partition half of the mining nodes, which cuts the throughput in half. Furthermore, the throughput will be larger if the honest miner is more likely to mine on the heaviest chain.

In figure 8, the adversary has 20 percent hashing power and we see that it is able to maintain the minimum with smaller partitioning power. Now the minimums are 0.35 and 0.15 for $P_1 = 1.0, 0.5$, respectively, and is able to reach these minimums at approximately 40 percent partitioning power - the suboptimal partitioning power. We also notice that the maximums, compared to figure 7, have decreased by at least 0.1 for both P_1. Thus, **with extra hash power, the adversary is able to achieve minimal throughput with decently less partitioning power and overall throughput is reduced by at least 0.1**.

In figure 9, the throughput has decreased far more: to 0.25 and 0.1 for $P = 1.0, 0.5$ respectively. Furthermore, compared to figure 7, the maximums have been reduced by at least 0.2 for both P_1. We also see that the curves are less steep compared to figures 8 and 7, **indicating that a larger adversary mining power reduces the necessity for greater partitioning power reduces. Furthermore, the throughput of GHOST can be reduced to as small as 0.1 in the optimal scenario for the adversary.**

4 Conclusion

For Ethereum's proof-of-work consensus, a modified GHOST protocol was used to improve security by disincentivizing adversarial mining by instead incentivizing participation through rewarding uncle block mining. With Ethereum's Casper upgrade, adversarial behavior from validators is disincentivized through slashing. Both mechanisms, supposedly, minimizes the chances of a private attack. We provide an in-depth discussion and literature review of GHOST and Ethereum, and perform the Private and Balance Attack on GHOST. We find that GHOST is very resistant to the Private Attack, where the adversary is better off mining on the tip of the heaviest chain. For the Balance Attack, we

also find that as adversary mining power slowly increases, its necessity for partitioning power massively reduces to achieve its ultimate goals: maximize tip conflicts (up to as high as 0.6) and minimize throughput of GHOST (as low as 0.1).

References

Sarah Azouvi and Daniele Cappalleti. 2021. Private attacks in longest chain proof-of-stake protocols with single secret leader elections. *Association for Computing Machinery*.

Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system.

Christopher Natoli and Vincent Gramoli. 2016. The balance attack against proof-of-work blockchains: The r3 testbed as an example. *CoRR*.

Muhammad Saad, Victor Cook, Lan Nguyen, My T. Thai, and Aziz Mohaisen. 2019. Partitioning attacks on bitcoin: Colliding space, time, and logic. *IEEE*.

Yonatan Sompolinsky and Aviv Zohar. 2013. Secure high-rate transaction processing in bitcoin.

Baixi Wu. 2023. Analysis of ethereum ghost protocol under blockchain framework. *Highlights in Science, Engineering and Technology*, 60:121–127.

5 Appendix A

5.1 Longest Chain Rule

Miners in the Bitcoin protocol compete to solve a random hash puzzle $H(nonce, data) < T$, and obtain proof of work when they find a nonce that satisfies the puzzle for the data (from which a block will be created/mined). When blocks in Bitcoin are mined, ideally this should lead to a similar rate of growth for the transaction ledger (i.e all blocks mined are included in the ledger), but this is not always possible for a variety of benign or malicious reasons. Two blocks may be mined with the same parent block at the same time by two miners (but only one should be included in the ledger) due to network delay Δ, or a miner may not publish a mined block (stunting the growth of the ledger). This sharing of a parent creates forking, and a tree in the set of blocks mine rather than a chain. When other users attempt to mine a block, the tip of the blockchain they should consider is that at the end of the longest chain from the genesis block. This also means that the set of blocks considered as the ledger is those blocks that are on this chain from the genesis block to the tip. If there are ties in longest chain length, the protocol favors the branch of the branch of the forking in these longest chains where the block hashes were smallest. If a miner or coalition of miners control a majority of the hashing power, they could control the longest chain, and thus allowing them to exclude or modify transactions that belong to the ledger (Nakamoto, 2008).

5.2 Private Attack

For adversarial miners, a coalition of them could pool mining resources together and when they mine blocks, keep them private and not broadcast them to the network, creating fork in the blockchain, private to them. Honest users who never see these blocks will mine on the legitimate tip of the chain. If adversarial miners either get lucky (due to randomness of Hash puzzle), or control most of the hash power of all miners and mine more blocks after creating the fork than the other 'coalition' of honest miners (say 10 to 7), then release it, the honest miners will have to start mining on the adversarial chain as that is now the new longest chain. They then give up their rewards, the transactions in their blocks are erased from the ledger, all allowing these adversaries to monopolize mining rewards in addition to determining the state of the ledger, even being able to double spend tokens. In theory,

this means that blocks are always vulnerable to being dislodged from the blockchain, which necessitates the definition of a rule for when a block in the chain can be considered confirmed (intuitively, a block is more confirmed the more blocks it has following it, as it's less likely to be dislodged). The k-deep confirmation rule, then, means a block is considered confirmed if it's buried under k blocks, as changes in the longest chain will happen towards the tip of that chain. The safety property in blockchains means that a transaction confirmed by one user is soon confirmed by all other users and remains confirmed forever. Adversarial miners who want to violate safety of the blockchain should then wait for a block to be confirmed before revealing their private chain, potentially unconfirming/modifying previously confirmed transactions (Azouvi and Cappalleti, 2021).

6 Appendix B

6.1 Simulating the Private Attack

To simulate the private attack, we used N = 100, and B = 1000 (10,000 for one experiment) where each node has equal hash power. β was our variable on the x-axis. So, β tells us how many adversarial nodes (α) and honest nodes (h) exist: $\alpha = 100\beta$ and $h = 100 - 100\beta$. We ran two tests for the private attack: the first has $P_2 = 0$, and the second has $P_2 = 0.5$ to test different adversarial strategies. Within each one of these simulations, we also test $P_1 = \{0.2, 0.6, 1.0\}$ to see how honest miner actions influence the outcome. We calculate chain quality as a result of the simulations.

6.2 Simulating the Balance Attack

To simulate the balance attack, we used N = 50, and B = 1000 where each node has equal hash power. Here, β was something that we changed from one experiment to another, because our x-axis variable was the adversary partitioning power. For example, if our adversary partitioning power was 1 percent, then this means that they could partition the honest miners into two groups: one group contains 1 percent of the total honest miners, and the other contains the other 99 percent. With this, we performed three experiments where we tuned β to 0, 0.2, and 0.4 for each one. Within each one of these, we used $P_1 = \{0.5, 1.0\}$ and instead of using P_2, our adversarial nodes always mined on the tip of the smaller of the two chains, to maximize the number of heaviest-chain-switches and minimize throughput. We measure the effectiveness of the Balance Attack by calculating the throughput of Ethereum (the number of honest blocks in the heaviest chain divided by the total number of blocks), and the proportion of chain switches (the number of chain switches divided by the total number of blocks). We use the former measurement because the greatest rewards are given to the blocks that are actually in the heaviest chain, and so if an adversary minimizes the proportional size of this chain, they minimize the rewards involved. They also minimize the number of verified blocks. We use the latter measurement because "tip switches" are dangerous, lead to spontaneity in the chain, inconsistencies, double spending, and wasted computation on identical transaction verification .

7 Appendix C

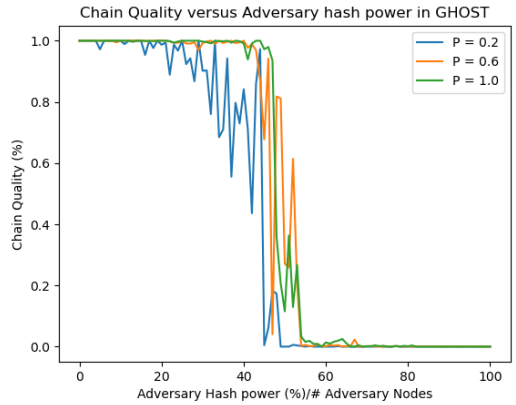

Figure 1: N=100, B = 1000, $P_2 = 0$, $\lambda = 1$. Chain Quality of heaviest chain versus adversary hash power.

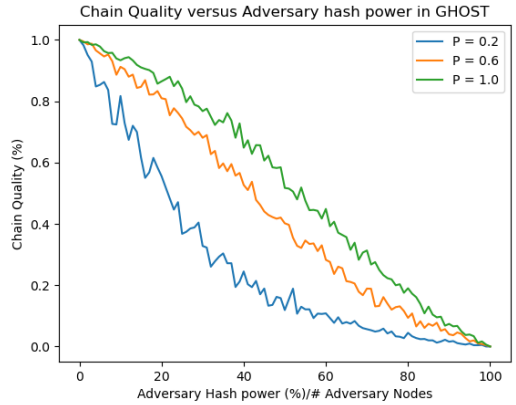

Figure 2: N=100, B = 1000, $P_2 = 0.5$, $\lambda = 1$. Chain Quality of heaviest chain versus adversary hash power.

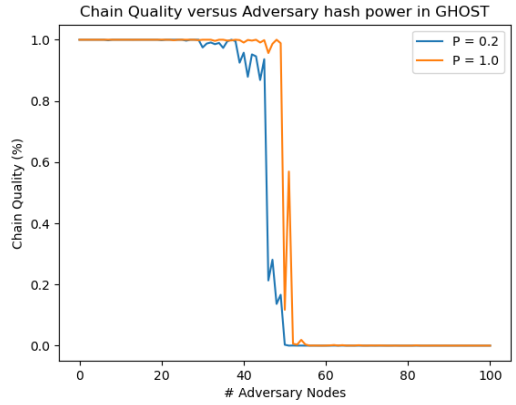

Figure 3: N=100, B = 10,000, $P_2 = 0$, $\lambda = 1$. Chain Quality of heaviest chain versus adversary hash power.

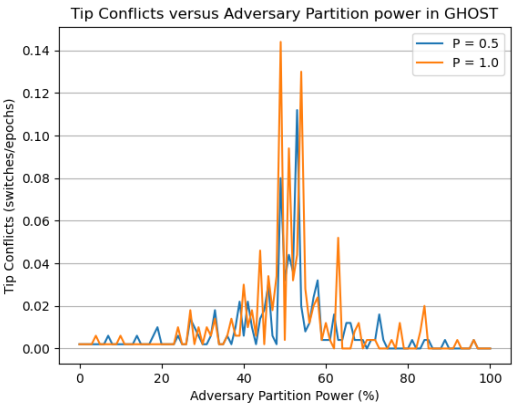

Figure 4: N=50, B = 1000, $\beta = 0$, $\lambda = 1$

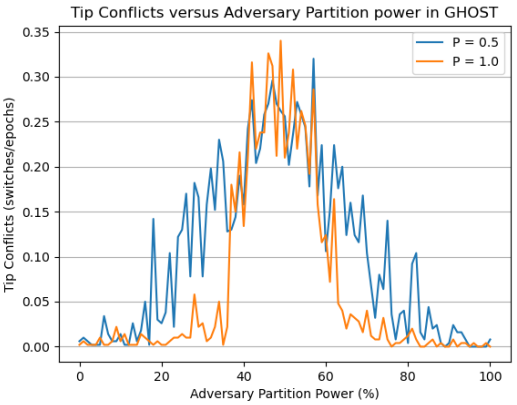

Figure 5: N=50, B = 1000, $\beta = 0.2$, $\lambda = 1$

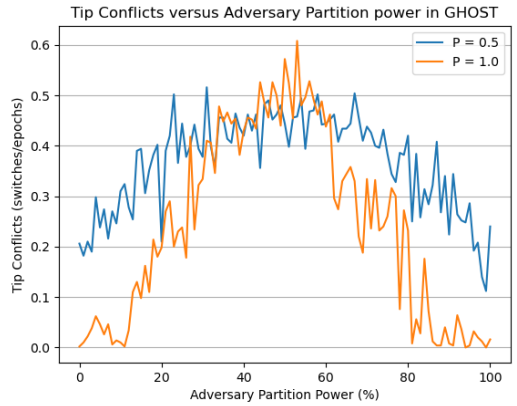

Figure 6: N=50, B = 1000, $\beta = 0.4$, $\lambda = 1$

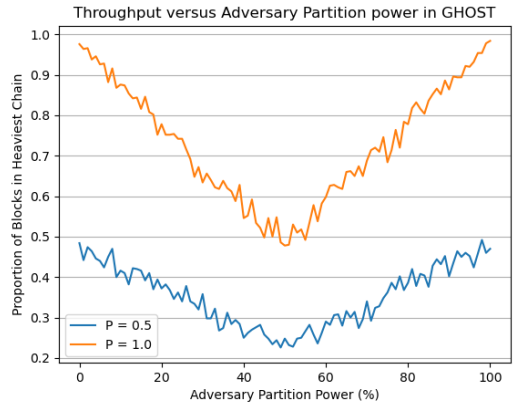

Figure 7: N=50, B = 1000, $\beta = 0$, $\lambda = 1$

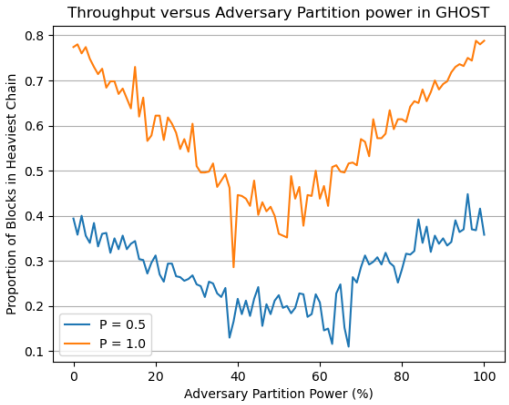

Figure 8: N=50, B = 1000, $\beta = 0.2$, $\lambda = 1$

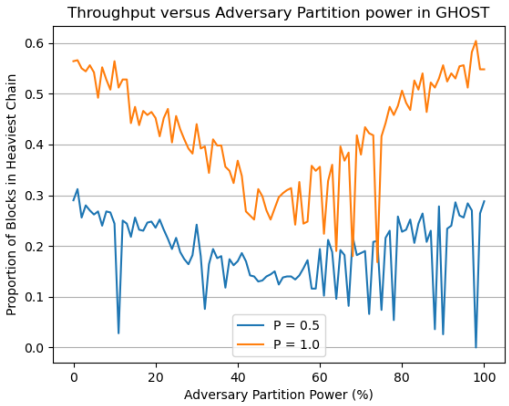

Figure 9: N=50, B = 1000, $\beta = 0.4$, $\lambda = 1$