A severe Goldfinger attack vector on Proof-of-Work blockchains

Mads J. Damgaard*

July 14, 2024

Abstract

The Goldfinger attack is a 51% attack that aims to destroy the value of a cryptocurrency in order for the attackers to achieve some utility outside of the given blockchain. For example, it has been speculated in the past that attackers might take large short positions in the targeted cryptocurrency leading up to the attack. In this paper, however, we call attention to a more likely scenario where attackers achieve investment gain from a 51% attack, not by short-selling the cryptocurrency, but by investing in a rival blockchain. The described attack vector specifically targets blockchains with a Proof-of-Work consensus mechanism, such as Bitcoin, whereas blockchains relying on a Proof-of-Stake mechanism, such as Ethereum, are naturally more resistant to Goldfinger attacks.

1 Introduction

The Goldfinger attack, as defined by Kroll, Davey, and Felten [1], is a 51% attack[1] on a blockchain where the attackers deliberately seek to damage the value of the related cryptocurrency in order to achieve some political or monetary goal outside of the economy of the targeted blockchain.

One possible way of achieving monetary gains from such an attack is if the attackers manage to take large short positions in the cryptocurrency before the attack. But as Kroll et al. [1] point out, it would be hard for the attackers to acquire large enough short positions to make such an attack affordable, at least on a well-established blockchain like Bitcoin[8, 9], which leads them to dismiss the possibility of a Goldfinger attack motivated by investment gains.

However, short-selling a cryptocurrency is not the only way that attackers might gain monetary value from a crash. In this paper, we call attention to the fact that the attackers might also be stakeholders in a rival blockchain, who seek to remove their competitor in order to get a larger share of the market.

This scenario is particularly relevant now that Ethereum[12–15], which is the second-largest blockchain on the market at the time of writing,[7] has switched to a Proof-of-Stake (PoS) consensus mechanism as of 2022.[14] This makes Ethereum less prone to

*B.Sc. at the Department of Computer Science, University of Copenhagen. B.Sc. at the Niels Bohr Institute, University of Copenhagen. E-mail: fxn318@alumni.ku.dk.

a Goldfinger attack compared to Bitcoin, which still relies on a Proof-of-Work (PoW) consensus mechanism.[9] The stakeholders of Ethereum therefore do not have to fear a similar retaliation if they were to mount a successful Goldfinger attack on Bitcoin. Thus, if only the Ethereum stakeholders can afford such an attack, they might pursue this opportunity for knocking down their rival and conquering the number one spot on the market.

In the following section, we argue that the Ethereum stakeholders can indeed afford such an attack, if they pool their resources. And in Sections 3–4, we discuss how they can even bribe existing Bitcoin miners through smart contracts in order to make the attack more feasible. Lastly, in Section 5, we discuss how such attacks can be mitigated if only the Bitcoin community is willing to follow in the footsteps of Ethereum and switch to a PoS consensus mechanism in time.

2 A Goldfinger attack from a rival blockchain

The market capitalization of Ethereum is currently around $450 billion at the time of writing, whereas Bitcoin's market capitalization is around $1.2 trillion.[6] This is a combined value of around $1.65 trillion. If Ethereum were to gain the same level of popularity as Bitcoin, or more, it could therefore potentially grow to more than $825 billion in market value. This would thus be a growth of more than $375 billion.

In comparison, the daily rewards for the Bitcoin miners are around $27 million.[6] This means that $375 billion could in principle pay for a 51% attack lasting as long as $\$375 \times 10^9/(\$27 \times 10^6/\text{day}) \approx 14{,}000$ days, which is around 38 years.[1]

It goes without saying that a 38-year-long Denial-of-Service (DoS) attack on the Bitcoin blockchain is more than enough to destroy the value of Bitcoin almost completely. And even if the Ethereum stakeholders pay only 1% of this, they could still in principle pay for a 51% attack lasting for about 140 days $\approx$ 4.6 months. A Goldfinger attack from a rival blockchain like Ethereum could thus potentially cause serious harm to the Bitcoin economy.

On the other hand, if Bitcoin stakeholders were to attempt a Goldfinger attack against Ethereum, they would have to become stakeholders in Ethereum themselves, due to its PoS consensus mechanism. As Lee and Kim [2] point out, however, a Goldfinger attack (or *short selling attack* as they call it) is still possible in theory, as long as only a fraction of the combined Ether[2] is staked in the consensus mechanism. But this vulnerability is rather easy to mitigate, e.g. by encouraging more stakeholders to stake their Ether. And even if such an attack were to succeed, the Ethereum network would most likely vote to revert the attack, which is not a difficult matter in a PoS network, where the voting power is distributed based on stake, and not on computational power.

This means that the stakeholders of Ethereum, despite the blockchain's smaller size in terms of market capitalization, are in principle able to destroy the economy of its larger competitor, unless the Bitcoin community decides to covert to a PoS consensus mechanism as well.

[1] This calculation does not even account for the fact that Bitcoin halves its mining rewards about every four years.[11]

[2] Ether is the cryptocurrency of Ethereum.[13]

3 Rewarding the miners through smart contracts

In this section, we shall see how Ethereum stakeholders can set up automatic rewards for Bitcoin miners who participate in a 51% attack through what is known as smart contracts.

A smart contract[16–18] is an automated transaction governed by a computer program, whose outcome might be determined by subsequent events on the blockchain, and in particular how clients interact with the contract. The smart contracts of Ethereum can be written as scripts in the high-level programming language Solidity, which are then compiled into the machine language of the Ethereum Virtual Machine.[17, 18] Solidity implements contracts as collections of functions and variables, much like objects in an object-oriented programming language. Clients can then call the functions of a contract from subsequent transactions on the blockchain, which allows them to either send Ether to the contract or withdraw Ether from it, given that the requirements of the called function are fulfilled. It also allows them to make changes to the memory state of the contract in general.

Through such smart contracts, the Ethereum stakeholders are in principle able to set up automatic rewards for successful 51% attacks. Here we will discuss a particular kind of 51% attack where the participating miners create a fork of the blockchain from an interior block and try to make the newly created chain, call it the *attack chain*, overtake the *original chain*, causing a reorganization of a significant part of the Bitcoin ledger.

To put this in more precise terms, let $(b_0, \ldots, b_n, \ldots, b_m)$ be the original chain at the time when the attack is initiated, where b_0 is the genesis block, b_n is the chosen fork point, and b_m is the most recent block. The participating miners then have to build a chain $(a_1, \ldots, a_l)$, such that $(b_0, \ldots, b_n, a_1, \ldots, a_l)$ becomes the longest well-formed blockchain in existence at the end of the attack, in terms of the amount of work[3] it contains.

According to Bitcoin's PoW-based protocol, this will make the attack chain the new valid version of the blockchain, meaning that the transactions contained in $(b_n, \ldots, b_m)$ will have been overwritten by the transactions contained in $(b_n, a_1, \ldots, a_l)$.

In order to be able to prove that the attack caused a significant reorganization of the targeted blockchain, the miners can make sure that a_1 contains a hash of b_m, which will thus prove that $(b_n, \ldots, b_m)$ already existed at the beginning of the attack.

In the following points, we sketch an example of how a smart contract that rewards this particular kind of 51% attack could function.

1. **Funding the contract.** Ethereum stakeholders who want to help fund the attack, call them the *funders*, can send Ether to the contract, where it is temporarily locked. This is the reward money in case of a successful 51% attack.

2. **Withdrawal of funds.** If a funder wants to withdraw their deposit again, they first have to put in a request and then wait a withdrawal period, t_{w}, before they are able to withdraw their money. If an attack succeeds in the meantime, the money will instead be included as part of the rewards. This stops funders from withdrawing their funds right when an attack seems to be underway.

[3] The amount of work contained in a blockchain can be calculated as the sum of the so-called difficulty of each block, which is inversely proportional to the block's target.[9, 10]

3. **Submitting proof of an attack.** If the participating miners manage to complete a 51% attack and want to claim their rewards, they can submit a proof of their attempt to a function of the contract for an initial verification. This proof has to contain all the blocks $\{a_1, \ldots, a_l\}$ and $\{b_0, \ldots, b_n, \ldots, b_m\}$, as well as a declaration of the fork point, $n < m$.

4. **Verification of the data.** The initial verification of the submitted data should first of all check that the chains $(b_0, \ldots, b_n, \ldots, b_m)$ and $(b_0, \ldots, b_n, a_1, \ldots, a_l)$ are both well-formed, i.e. follow rules of the Bitcoin protocol. It should also check that a_1 contains a hash of b_m, and that $W(b_n, a_1, \ldots, a_l) > W(b_n, \ldots, b_m)$, where W is a function of the amount of work contained in the given blockchain. Furthermore, it should be checked that $W(b_{n+1}, \ldots, b_m) \geq w_{\min}$, where $w_{\min}$ is some minimum amount of work that the attack chain has to be behind the original chain when a_1 is built. And lastly, it should be checked that $W(b_0, \ldots, b_n, a_1, \ldots, a_l) > w_{\text{rec}}$, where w_{rec} is a variable that is initialized as the greatest amount of work recorded for the Bitcoin blockchain at the time when the contract was created.

5. **Contest phase.** If the initial verification succeeds, all the funds of the contract are locked for a contest period, t_c, in which the funders, as well as anyone else, are able to contest that $(b_0, \ldots, b_n, a_1, \ldots, a_l)$ contains the greatest amount of work out of all the well-formed blockchains in existence. They can do so by submitting such chains of the form $(b_0, c_1, \ldots, c_k)$, $a_1 \notin \{c_1, \ldots, c_k\}$, to a contest function of the contract during this period. If one is submitted for which $W(b_0, c_1, \ldots, c_k) > W(b_0, \ldots, b_n, a_1, \ldots, a_l)$, the attack will have been successfully contested.

6. **A failed attack.** If the funders manage to contest the validity of the attack, the contest period simply ends early, and the participating miners are once again free to submit new attempts. However, the variable w_{rec} is now assigned a new value of $W(b_0, c_1, \ldots, c_k)$, where $(b_0, c_1, \ldots, c_k)$ is the winner of the contest, meaning that the miners now have to beat this new w_{rec} before being able to trigger another contest phase.

7. **A successful attack.** On the other hand, if the funders do not manage to contest the validity of the attack, the rewards are divided into l parts, which can subsequently be withdrawn by the owners of up to l different addresses on the Ethereum blockchain, determined by the miners of $(a_1, \ldots, a_l)$. Each miner who mined a block, a_i, can determine the ith address at the time of its creation, which is done by including a transaction of a special format in a_i that contains this address.

This example of a smart contract thus shows how the Ethereum stakeholders can bribe Bitcoin miners in an automated and anonymous way. It is not the most efficient solution possible,[4] but it serves to illustrate the point.

[4] In particular, it would be more efficient to use a more recent block than the genesis block in place of b_0, and simply define the initial balances at this block in the contract. However, one could also greatly reduce the computations needed by simply disregarding the validity of the transactions in $(b_0, \ldots, b_n, \ldots, b_m)$ and $(b_0, c_1, \ldots, c_k)$ in Points 4–5, and instead just assume that any blockchain of that size in existence must be a well-formed one for the most part. Additionally, the contract can state that no $a_i \in \{a_1, \ldots, a_l\}$ may contain any transactions other than ones of some special formats that are automatically valid. These transactions can then for instance be used to include the hash of b_m in a_1, or to define the Ethereum address of the miner in question, as described in Point 7.

4 Partial attacks

Ethereum stakeholders can also potentially reward what we might call *partial attacks*, where the attack chain does not need to overtake the original chain, but only needs to match some fraction of its pace. This might serve as a less expensive way to showcase the vulnerability, without having to fund a full 51% attack.

Such partial attacks can also be automatically rewarded via a similar kind of smart contract as the one described in the previous section, only where the requirement from Point 5 that $W(b_0, c_1, \ldots, c_k) > W(b_0, \ldots, b_n, a_1, \ldots, a_l)$ is replaced with a requirement that $W(b_0, c_1, \ldots, c_k) > W(b_0, \ldots, b_n, a_1, \ldots, a_l) + w_{\text{offset}}$ instead. Here w_{offset} is thus the maximum amount of work that a submitted attack chain is allowed to be behind its contestants at the end of the contest phase.

By choosing appropriate values for w_{offset} and $w_{\min}$, the Ethereum stakeholders are thus able to fund some initial, small-scale versions of the Goldfinger attack, which do not actually cause any reorganization of the Bitcoin ledger, but which nonetheless still showcase the potential for a true Goldfinger attack in the future.

Since such partial attacks will cost less money and require fewer Bitcoin miners, and since they do not actually cause any *direct* harm to the Bitcoin blockchain, they might therefore be better at attracting the necessary funding at an initial stage.

5 Mitigation

As long as Bitcoin keeps relying purely on PoW, which means that the voting power of the system is purely determined by the computational power of the miners, the blockchain will in theory remain vulnerable to a Goldfinger attack that attempts to bribe these miners. However, there is nothing in principle that stops Bitcoin from converting to a PoS consensus mechanism as well. The Ethereum blockchain has already shown that such a conversion is possible.

Of course, a switch to PoS might be seen as a concession by some of the Bitcoin investors, and especially by those who have previously regarded PoW as a superior consensus mechanism. It is therefore possible that the market value of Bitcoin would drop by some amount as a consequence of a switch to PoS. However, it is reasonable to speculate that if the Bitcoin community delays too much in switching to PoS, and waits until either a partial or a full Goldfinger attack *forces* them to do so, the general esteem of Bitcoin would only suffer all the more, resulting in a greater drop in its value. A switch to PoS as soon as possible therefore seems to be the best course of action for Bitcoin, as well as for other PoW-based blockchains.

References

[1] J. A. Kroll, I. C. Davey, and E. W. Felten, The economics of Bitcoin mining, or Bitcoin in the presence of adversaries, *The Twelfth Workshop on the Economics of Information Security (WEIS 2013)*, June 11–12, 2013.

[2] S. Lee and S. Kim, *Short selling attack: A self-destructive but profitable 51% attack on PoS blockchains*, Cryptology ePrint Archive, Paper 2020/019.

[3] K. Dwivedi, A. Agrawal, A. Bhatia, and K. Tiwari, *A novel classification of attacks on blockchain layers: Vulnerabilities, attacks, mitigations, and research directions*, arXiv:2404.18090 [cs.CR].

[4] D. Faust. (November 2017). Crypto options and Goldfinger attacks. SoundCrypto, https://soundcrypto.com/blog/2017/11/16/crypto-options-and-goldfinger-attacks.

[5] A. Hayes. (March 2024). Short selling: Pros, cons, and examples. Investopedia, https://www.investopedia.com/terms/s/shortselling.asp.

[6] BitInfoCharts. (July 2024). Cryptocurrency statistics. https://web.archive.org/web/20240702024855/https://bitinfocharts.com.

[7] BitInfoCharts. (June 2024). Cryptocurrency charts. https://web.archive.org/web/20240620223723/https://bitinfocharts.com/cryptocurrency-charts.html.

[8] S. Nakamoto, *Bitcoin: A peer-to-peer electronic cash system*, Available at: https://bitcoin.org/bitcoin.pdf, 2008.

[9] Bitcoin.org. (Visited July 2024). Block chain. https://developer.bitcoin.org/devguide/block_chain.html.

[10] Bitcoin Wiki. (Visited July 2024). Difficulty. https://en.bitcoin.it/wiki/Difficulty.

[11] L. Conway. (May 2024). What is Bitcoin Halving and why it matters for crypto investors. Investopedia, https://www.investopedia.com/bitcoin-halving-4843769.

[12] Ethereum.org. (March 2024). Ethereum whitepaper. https://ethereum.org/en/whitepaper

[13] Ethereum.org. (June 2024). Intro to Ethereum. https://ethereum.org/en/developers/docs/intro-to-ethereum.

[14] Ethereum.org. (July 2024). Proof-of-Stake (PoS). https://ethereum.org/en/developers/docs/consensus-mechanisms/pos.

[15] Ethereum.org. (June 2024). Transactions. https://ethereum.org/en/developers/docs/transactions.

[16] Ethereum.org. (April 2024). Introduction to smart contracts. https://ethereum.org/en/developers/docs/smart-contracts.

[17] SolidityLang.org. (Visited July 2024). Introduction to smart contracts. https://docs.soliditylang.org/en/v0.8.26/introduction-to-smart-contracts.html.

[18] SolidityLang.org. (Visited July 2024). Solidity by example. https://docs.soliditylang.org/en/v0.8.26/solidity-by-example.html.