Pitchforks in Cryptocurrencies:

Enforcing rule changes through offensive forking- and consensus techniques (Short Paper)

Aljosha Judmayer[1] ✉, Nicholas Stifter[1,2], Philipp Schindler[1], Edgar Weippl[1,2]

[1]SBA Research, Vienna, Austria
[2]Christian Doppler Laboratory for Security and Quality Improvement in the Production System Lifecycle (CDL-SQI), Institute of Information Systems Engineering, TU Wien
Email: (firstletterfirstname)(lastname)@sba-research.org

Abstract. The increasing number of cryptocurrencies, as well as the rising number of actors within each single cryptocurrency, inevitably leads to tensions between the respective communities. As with open source projects, (protocol) forks are often the result of broad disagreement. Usually, after a permanent fork both communities "mine" their own business and the conflict is resolved. But what if this is not the case? In this paper, we outline the possibility of malicious forking and consensus techniques that aim at destroying the other branch of a protocol fork. Thereby, we illustrate how merged mining can be used as an attack method against a permissionless PoW cryptocurrency, which itself involuntarily serves as the parent chain for an attacking merge mined branch of a hard fork.

1 Introduction

Merged mining is already known for posing a potential issue to the child cryptocurrencies, as for example demonstrated in the case of CoiledCoin[1], however so far no concrete example that merged mining can also pose a risk to the parent chain has been given. Since, (parent) cryptocurrencies can not easily prevent being merge mined[2], an attack strategy using this approach would be applicable against a variety of permissionless PoW cryptocurrencies. In this paper, we describe a scenario where merged mining is used as a form of attack against a parent chain in the context of a hostile protocol fork.

1.1 System Model and Attack Goals

For our attack scenario, we assume a permissionless PoW based cryptocurrency B, whose miners cannot agree whether or not to change the consensus rules. Some of the miners want to adapt the consensus rules in a way such that newly mined blocks may not be valid under the old rules, i.e., perform a hard fork. Thereby, we differentiate between the following actors:

- **Backward compatible miners ($\mathcal{B}$):** The fraction of miners (with hash rate β) in a currently active cryptocurrency B which does not want to change the consensus rules of B.

[1] cf. https://bitcointalk.org/index.php?topic=56675.msg678006#msg678006
[2] The inclusion of a hash value within a block to provably attributed it to the creator of the proof-of-work (PoW) is enough to support merged mining [7]

- **Change enforcing miners ($\mathcal{C}$):** The fraction of miners (with hash rate α) in a currently active cryptocurrency B which wants to change the consensus rules, i.e., perform a hard fork. Moreover, they want to make sure, that only their branch of the fork survives.
- **Neutral miners ($\mathcal{N}$):** The set of miners (with hash rate ω) that has no hard opinion on whether or not to change the consensus rules. They want to maximize their profits and act rationally to achieve this goal, with the limitation that they want to avoid changes as far as possible. If there is no immanent need which justifies the implementation costs for adapting to changes, they will not react[3].

For our example, we assume that $\mathcal{C}$ wants to increase the block size, while $\mathcal{B}$ does not want to implement any rule change. The goal of the attackers in $\mathcal{C}$ is twofold: 1) *Enforce* a change of the consensus rules in the respective cryptocurrency. 2) *Destroy* the other branch of a fork which uses the same PoW and does not follow the new consensus rules.

1.2 Background

For this paper, we are only interested in forking scenarios that are *not bilateral*. In a bilateral fork, conflicting changes are intentionally introduced to ensure that two separate cryptocurrencies emerge [16]. An example for such a scenario would be the changed chain ID between Ethereum and Ethereum Classic. It is commonly believed that in a non-bilateral forking event, the only reliable possibility to enforce a change requires that the majority of the mining power supports the change. Thereby, two main cases can be distinguished according to [16]:

If the introduced change *reduces* the number of blocks that are considered valid under the new consensus rules, all new blocks are still considered valid under the old rules, but some old blocks are no longer considered valid under the new rules. An example for such a scenario would be a *block size decrease*. In this case the first goal (*enforce*) of our attack is easy to achieve if $\alpha > \beta+\omega$ holds, since any fork introduced by α will eventually become the longest chain and be adopted by β and ω because of the heaviest chain rule. If $\mathcal{B}$ decides to continue a minority branch, they have to declare themselves as a new currency B' and change their consensus rules to permanently fork off the main chain in B such that larger blocks are again possible. Therefore, the goal to *enforce* is clearly reached in such a case. However, the *destroy* goal cannot be reached directly if $\mathcal{B}$ forks to a new cryptocurrency B'.

If the introduced change *expands* the set of blocks that are considered valid under the new consensus rules, then some blocks following the new rules will not be considered valid under the old rules. Therefore, any mined block that is only valid under the new rules will cause a fork. An example for such a scenario would be a *block size increase*[4]. In this case a permanent hard fork will only

[3] This should capture the observation that not all miners immediately perform merged mining if it is possible, even though it would be rational to do so [7].

[4] Our example, in which $\mathcal{C}$ wants to increase the block size and $\mathcal{B}$ does not want to implement any rule change, would resemble such an *expanding* protocol change.

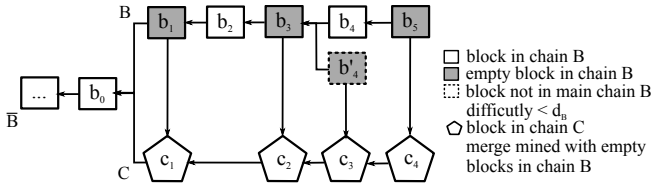

Fig. 1. Example of blocks mined in the two chains B and C after the forking event.

occur if the chain containing blocks following the new rules grows faster, i.e., $\alpha > \beta + \omega$ holds. The result would be that the forking event creates two different currencies: Cryptocurrency C, which includes big blocks, and cryptocurrency B, which forked from the main chain after the first big block. Therefore, again the *destroy* goal cannot be reached directly. To reach the goal *destroy*, some miners in $\mathcal{C}$ could be required to switch to the new currency B and disrupt its regular operation, e.g., by mining empty blocks. This of course has the drawback that the respective attacking miners that switched from C to B do not gain any profits in C, and their rewards in B will be worthless if they succeed in destroying the B fork.

The *pitchfork* attack method proposed in this paper aims to achieve both attack goals simultaneously, even in cases where $\alpha < \beta + \omega$ holds.

2 Pitchfork Attack Description

The basic idea of a pitchfork attack is to use merged mining as a form of attack against the other branch of a fork, in a permissionless PoW Cryptocurreny, that is the result of a disputed consensus rule change. The pitchfork should reduce the utility of the attacked branch to such an extent, that the miners abandon the attacked branch and switch to the branch of the fork which performs merged mining and follows the new consensus rules. We call the cryptocurrency up to the point of the fork *ancestor* cryptocurrency $\bar{B}$. After the forking event, the *backward compatible* cryptocurrency, which still follows the same rules, is denoted as B, whereas the *change enforcing* cryptocurrency branch that uses merged mining and the new consensus rules is denoted as C.

To execute the attack, the new merge mined branch C accepts valid *empty blocks* of B as a PoW for C. In the nomenclature of merged mining the chain B, which should be attacked, is called the *parent chain* and chain C is called the *child chain*. For a valid parent block b of B, the following additional requirements need to be satisfied: i) The block b has to be empty. Therefore, the contained Merkle tree root in the header of the respective block must only include the hash of the (mandatory) coinbase transaction. Given the corresponding coinbase transaction, it can then be verified that b is indeed empty. ii) The coinbase transaction of b must include the hash of a valid block c for C. The header of block c contains a Merkle tree root with the actual transactions performed in C.

Figure 1 shows the two cryptocurrencies after the fork. The last block in the ancestor cryptocurrency $\bar{B}$ before the forking event is b_0. The first empty block that is merge mined is b_1 in this example. This block (b_1) is valid under the old rules and fulfills the difficulty target in B. Moreover, the block b_1 was mined by

a miner in $\mathcal{C}$, which happens with probability α, and contains the hash of block c_1 in its coinbase. Therefore b_1 serves as a valid PoW for C as well. Block b_2 was not mined by a miner in $\mathcal{C}$, which happens with probability $1-\alpha$, and therefore it is not empty and does not contain a hash for a valid block for C in its coinbase. This shows that the two chains are not necessarily synchronized regarding their number of blocks. The block interval in C depends on the difficulty target of C. Since we assume that the attacker does not control the majority of the hash rate ($\alpha < \beta + \omega$), the difficulty d in C should be lower than in B at the beginning of the attack, i.e., $d_C < d_B$ holds. If the difficulty has been adjusted in C, then the overall number of blocks should be approximately the same for both chains. In such a case, there might be empty blocks such as b'_4, which do fulfill the difficulty target for C, but not for B. Still, if $d_C < d_B$ holds, then over time a fraction of all blocks in B, corresponding to α, will be mined by a miner in $\mathcal{C}$. If we assume that $\alpha \approx 0.34$ then approximately every third block in B should be empty.

Side note regarding difficulty: Theoretically it would be possible that chain C requires the same, or an even a higher difficulty than chain B. If $d_C \geq d_B$, then chain C would contain less blocks than chain B, this of course would have a negative effect on the latency in chain C, i.e., the time it takes till a transaction is confirmed. However any merge mined blocks that meet the difficulty requirement d_C will be considered valid in B. For example, when $d_C = d_B$, the number of blocks in C relative to B would only correspond to the fraction of the hash rate (α) that performs merged mining. Nevertheless, since chain C increased the block size, the throughput could theoretically remain the same or even be higher than in chain B (depending on the actual implementation). Some examples regarding an increased block size are discussed in [3,6]. Alternatively, Bitcoin-NG [4] could also be applicable. The latter approach would have the added benefit that the negative impact on latency and confirmation times is mitigated. To illustrate our attack, it is not of particular relevance which adaptation is used to increase the throughput in C.

2.1 Effects of the Attack

In the simplest case, if no counter measures are taken by the chain under attack, a pitchfork reduces the utility of the target chain B by the number of empty blocks, corresponding to the hash rate of the attackers (α). Considering the limited block size in B and past events in Bitcoin[5], where the number of unconfirmed transactions in the mempool peaked at around $175,000$ in December 2017, a hash rate of $\alpha \approx 0.34$ would likely have a non negligible impact on the duration of such periods, and hence transaction fees and confirmation times. This could sway both users and miners in $\mathcal{N}$ to switch to the attacking chain C, which further reinforces the attack. Two other advantages of the attack are, that it is *pseudonymous* and that the risk in terms of currency units in B is *parameterizable*.

Pseudonymous: Since the pitchfork attack is executed by miners through producing new blocks that are, in addition, merge mined with the attacking chain, it is in theory possible to hide the identities of the attackers because no unspent

[5] https://blockchain.info/charts/mempool-count?timespan=1year

transaction outputs need to be involved in the attack that could have a traceable history. However, additional care needs to be taken by these miners to ensure that their identity is not inadvertently revealed through their behavior [7].

Parameterizable: The attack is not an all-in-move and its costs, in terms of currency units in B, can be parameterized. The goal of the attack is to destroy the original chain B, but if this fails the attackers may not lose much. Due to merged mining the main costs of a failed attack result from the forgone profits from transaction fees that are not collected in chain B. Additional costs created by merged mining, i.e., running and additional full node for chain B, can be negligible compared to the overall costs related to mining [12]. Moreover, even a failed attack on B can still be profitable for the attacking miners, since the attackers in $\mathcal{C}$ are early adopters of C. If the value of the newly created cryptocurrency C increases enough, the additional income may not only compensate the reduced income from mining empty blocks in B, but could even create a surplus for the miners in $\mathcal{C}$. In addition, the attack can be made compatible with other available cryptocurrencies that can be merge mined with B. Therefore, additional revenue channels from existing merge mined cryptocurrencies are not affected by the pitchfork and can even help to subsidize the attack.

As a further parameterization for the attack, it is also possible to execute it in stages. To test whether there is enough support for chain C, it is possible to first start with relatively low risk to the attackers by not requiring them to mine empty blocks and instead only demand the creation of smaller blocks which can still include high fee transactions. From there, the attackers can reduce the number of permissible transactions step by step. At a final stage, all coins earned through mining empty blocks in B can also be used to fund additional attacks, such as triggering additional spam transactions in B as soon as the 100 blocks cooldown period has passed. For instance, splitting the coinbase rewards into many individual outputs of a high enough value with different lock times and rendering the output scripts as *anyone can spend* can lead to a large influx of additional transactions, as users (and miners) compete to scoop up these free currency units. This is easy to verify as an additional rule in C, however more complex attack scenarios such as those outlined in [1,10,14] may also be included as additional consensus rules.

3 Countermeasures

In this section we outline some countermeasures that can be taken by $\mathcal{B}$.

Fork away empty blocks in B: The miners in $\mathcal{B}$ can decide to fork off empty blocks and just build on top of blocks containing transactions. This requires the coordinated action of all miners in $\mathcal{B}$. If $\beta > \alpha + \omega$ this approach will work in general. A possible counter reaction by the attackers in $\mathcal{C}$ would be to introduce dummy transactions to themselves in their blocks in B. Therefore, it has to be ensured that those transactions are indeed dummies. For example: All used output addresses of every transaction belong to the same entity, but this must not be correlated given just the block b_n in B. One way to achieve this, is to require that all output addresses in a block have been derived from the miner's public

key of the respective block, like in an Hierarchically Deterministic (HD) Wallet[6] construction. The *master public key property* of such a construction allows that future ECDSA public keys can be derived from current ones. This is done by adding a multiplication of the base point with a scalar value to the current public key. The corresponding secret key is derived in the same manner, but can only be computed by its owner. If it is not possible to perform a transaction to an address for which the miner does not have the corresponding private key, the utility of every transaction in the block is very limited. To check this condition on an arbitrary block b_n, the public key of the miner as well as the scalar value for the multiplication is required. These values can be added to the coinbase transaction of the corresponding block c_n in C.

If such dummy transactions are used, the miners of $\mathcal{B}$ would be required to monitor the chain C to deduce which block in B has been merge mined with C and includes only dummy transactions. If $\mathcal{B}$ finds such a block they then can still cause a fork in B to ignore it. Besides being more complex, this also poses a potential risk for all transactions in B. Since the block b_n could be released before c_n, there is no way to tell whether or not b_n was indeed merged mined and hence includes a hash to c_n before c_n has been published in C. With this knowledge, miners in $\mathcal{C}$ can intentionally create forks in B when releasing c_n. By slightly relaxing the rules for dummy transaction and allowing, for example, one transaction output address that is not required to be derivable by the HD construction, double spends can be executed more easily in B. In this particular case miners of merged mined blocks can include a regular transaction that they want to double spend in their block, being assured that this block will get excluded in retrospect by all miners β in B if c_n is released. Therefore, more fine grained exclusion rules on transaction level would be necessary.

These examples illustrate, that it is non-trivial to change the consensus rules in B such that the effects of a pitchfork attack are mitigated. Every change of the defender leads to an arms race with the attacker. Moreover, excluding all merge mined blocks in B requires active monitoring of C to detect them. Therefore, at least the miners in $\mathcal{B}$ have to change their individual consensus rules – which they wanted to avoid in the first place.

Use mining power to launch a counter-attack on C: Miners in $\mathcal{B}$ can use their mining power to stall the attacking chain C. However, this has several limitations: Since every block in C requires an empty parent block in B as part of its PoW, miners cannot create empty merge mined blocks in C while at the same time creating full blocks in B. To stall chain C, at least a fraction of β, e.g., $\beta_a \leq \beta$ has to mine empty blocks in B to create empty merge mined blocks for C. Thereby, the counter-attackers would actually help the pitchfork attack. For our analysis, we assume that the difficulty target in C is indeed lower than in B, i.e., $d_C < d_B$ holds. To clearly overtake the pitchfork chain C, the counter-attacking miners need to have more than 50% of the hash rate in C. If not, the lost throughput, caused by empty blocks in C, might be compensated by the increased block size. This introduces the first constraint for the counter-attackers

[6] cf. BIP32 https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki

that the hash rate β_a they dedicate to the counter-attack must follow $\beta_a \geq \alpha$. However, the counter-attackers must also take care not to push the total hash rate dedicated towards attacking B to over 50% in B, otherwise more destructive attack rules than mining empty blocks, such as requiring non-empty blocks to be ignored, may be rendered effective. If the defenders are able to reliably identify all attackers' blocks they can try to fork them away in B. However the disadvantages of any (additional) attack rules, such as anyone-can-spend transactions or fork-away-non-empty blocks, still apply and can cause damage to B through their own blocks mined by β_a. The second constraint hence requires that for a counter attack, the bound $\alpha + \beta_a < 0.5$ for the share of blocks in the heaviest chain of B holds.

Depending on the exact implementation of merged mining in C, the counter-attackers have some options to avoid that their empty blocks in B, which they are required to provide as PoW, further reduce the utility of B. For example, in a naïve approach they could only submit PoW solutions to C that fulfill the difficulty target for C but not for B. This has the marked disadvantage that any blocks meeting the difficulty target of B also cannot be submitted as solutions in C, effectively reducing the counter-attackers' hash rate β_a in C by a factor dependent on the particular difference in difficulty between C and B. A better counter-attack can be achieved if the defenders intentionally construct blocks for the parent chain B that are unlikely to end up in the main chain, yet are still accepted as a valid proof-of-work in C. For instance, stale branches in B could be created and extended, however this is only effective if the freshness requirements for parent blocks in C are not too tight. In both cases, since β_a is no longer contributing toward the effective hash rate of B, its remaining honest miners $\omega + \beta - \beta_a$ must still retain a hash rate that exceeds that of the adversary to ensure that honest blocks constitute a majority of the heaviest chain. Therefore, the original attacker gains an advantage from merged mining since he can use his full hash rate in both chains at the same time. Moreover, the counter-attacking fraction of the miners would forgo their rewards in B for the duration of the counter-attack.

Figure 2 shows the hash rates achievable by the defender/counter-attacker on the respective chains B and C for different values for the hash rate α of the pitchfork attacker. In this figure the simplified assumption is made that the total hash rate of neutral miners ω is zero and hence the total hash rate of the defender/counter-attacker ($\beta = 1 - \alpha$) can be split between the two chains B and C arbitrarily. In this case, an attacker with $\alpha > \frac{1}{3}$ total hash rate cannot be countered on both chains simultaneously without losing the majority $\beta < 0.5$ on one of the chains.

4 Related work

In [7] it is argued that merged mining could also be used as an attack vector against the parent chain, however no concrete examples are given. Different forking techniques in the context of cryptocurrencies are described in [16]. The focus is placed on a non-malicious forking technique called *velvet fork*, initially proposed in [8]. Different methods that can be used in hostile blockchain takeovers

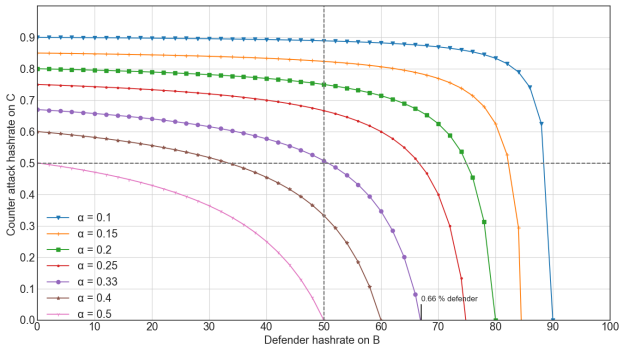

Fig. 2. Calculation for the hash rates of the defender/counter-attacker in the respective chains B and C for different values of pitchfork attacker hash rate α

are described in [2], placing the focus on attacks where the attacker has an extrinsic motivation to disrupt the consensus process, i.e., Goldfinger attacks [9]. The example given in the paper at hand is a concrete instance of such a situation. Therefore, most of the described methods can be used in conjunction with our proposed attack. The same holds true for the large body of work on bribing [1,11] and incentive attacks that distract the hash rate of participants [14,15]. Furthermore, selfish mining and its variants [5, 13] may be used in combination with pitchfork attacks.

5 Discussion and future work

In this paper, we outline that merged mining can be used as an attack method against a PoW cryptocurrency in the context of a hostile protocol fork. The general idea of such an *offensive consensus attack* is, that the participants of the offensive system are required to provably attack a different system as part of the consensus rules. We show that such attacks are theoretically possible and can lead to an arms race in which defenders are forced to adapt their consensus rules. Still, the consequences as well as the economic and game theoretic incentives of such attacks have yet to be analyzed in greater detail to better understand if they are practicable, and if so, how to protect against them.

Acknowledgments

We thank Georg Merzdovnik and Alexei Zamyatin as well as the participants of Dagstuhl Seminar 18152 “Blockchains, Smart Contracts and Future Applications” for valuable discussions and insights. This research was funded by Bridge Early Stage 846573 A2Bit, Bridge 1 858561 SESC, Bridge 1 864738 PR4DLT (all FFG) , CDL-SQI at the Institute of Information Systems Engineering TU Wien, and the competence center SBA-K1 funded by COMET.

References

1. J. Bonneau. Why buy when you can rent? bribery attacks on bitcoin consensus. In *BITCOIN ’16: Proceedings of the 3rd Workshop on Bitcoin and Blockchain Research*, February 2016.

2. J. Bonneau. Hostile blockchain takeovers (short paper). In *5th Workshop on Bitcoin and Blockchain Research, Financial Cryptography and Data Security 18 (FC). Springer*, 2018.
3. K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, and E. Gün. On scaling decentralized blockchains. In *3rd Workshop on Bitcoin and Blockchain Research, Financial Cryptography 16*, 2016.
4. I. Eyal, A. E. Gencer, E. G. Sirer, and R. van Renesse. Bitcoin-ng: A scalable blockchain protocol. In *13th USENIX Security Symposium on Networked Systems Design and Implementation (NSDI'16)*. USENIX Association, Mar 2016.
5. I. Eyal and E. G. Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography and Data Security*, pages 436–454. Springer, 2014.
6. A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdo rf, and S. Capkun. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC*, pages 3–16. ACM, 2016.
7. A. Judmayer, A. Zamyatin, N. Stifter, A. G. Voyiatzis, and E. Weippl. Merged mining: Curse or cure? In *CBT'17: Proceedings of the International Workshop on Cryptocurrencies and Blockchain Technology*, Sep 2017.
8. A. Kiayias, A. Miller, and D. Zindros. Non-interactive proofs of proof-of-work. Cryptology ePrint Archive, Report 2017/963, 2017. Accessed:2017-10-03.
9. J. A. Kroll, I. C. Davey, and E. W. Felten. The economics of bitcoin mining, or bitcoin in the presence of adversaries. In *Proceedings of WEIS*, volume 2013, page 11, 2013.
10. K. Liao and J. Katz. Incentivizing blockchain forks via whale transactions. In *International Conference on Financial Cryptography and Data Security*, pages 264–279. Springer, 2017.
11. P. McCorry, A. Hicks, and S. Meiklejohn. Smart contracts for bribing miners. In *5th Workshop on Bitcoin and Blockchain Research, Financial Cryptography and Data Security 18 (FC). Springer*, 2018.
12. Narayanan, Arvind and Bonneau, Joseph and Felten, Edward and Miller, Andrew and Goldfeder, Steven. Bitcoin and cryptocurrency technologies. http://bitcoinbook.cs.princeton.edu/, 2016. Accessed: 2016-03-29.
13. K. Nayak, S. Kumar, A. Miller, and E. Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *1st IEEE European Symposium on Security and Privacy, 2016*. IEEE, 2016.
14. J. Teutsch, S. Jain, and P. Saxena. When cryptocurrencies mine their own business. In *Financial Cryptography and Data Security (FC 2016)*, Feb 2016.
15. Y. Velner, J. Teutsch, and L. Luu. Smart contracts make bitcoin mining pools vulnerable. In *International Conference on Financial Cryptography and Data Security*, pages 298–316. Springer, 2017.
16. A. Zamyatin, N. Stifter, A. Judmayer, P. Schindler, E. Weippl, and W. J. Knottebelt. (Short Paper) A Wild Velvet Fork Appears! Inclusive Blockchain Protocol Changes in Practice. In *5th Workshop on Bitcoin and Blockchain Research, Financial Cryptography and Data Security 18 (FC). Springer*, 2018.