Pay To Win: Cheap, Cross-chain Bribing Attacks on PoW Cryptocurrencies

Aljosha Judmayer[1,2], Nicholas Stifter[1,2], Alexei Zamyatin[3], Itay Tsabary[4], Ittay Eyal[4], Peter Gaži[5], Sarah Meiklejohn[6], and Edgar Weippl[2]

[1] SBA Research {ajudmayer,nstifter}@sba-research.org
[2] Uni Wien edgar.weippl@univie.ac.at
[3] Imperial College London a.zamyatin@imperial.ac.uk
[4] Technion and IC3 Ittay@technion.ac.il,itaytsabary@gmail.com
[5] IOHK peter.gazi@iohk.io
[6] University College London s.meiklejohn@ucl.ac.uk

Abstract. In this paper we extend the attack landscape of bribing attacks on cryptocurrencies by presenting a new method, which we call *Pay-To-Win* (P2W). To the best of our knowledge, it is the first approach capable of facilitating double-spend collusion across different blockchains. Moreover, our technique can also be used to specifically incentivize transaction exclusion or (re)ordering. For our construction we rely on smart contracts to render the payment and receipt of bribes trustless for the briber as well as the bribee. Attacks using our approach are operated and financed *out-of-band* i.e., on a funding cryptocurrency, while the consequences are induced in a different target cryptocurrency. Hereby, the main requirement is that smart contracts on the funding cryptocurrency are able to verify consensus rules of the target. For a concrete instantiation of our P2W method, we choose Bitcoin as a target and Ethereum as a funding cryptocurrency. Our P2W method is designed in a way that reimburses collaborators even in the case of an unsuccessful attack. Interestingly, this actually renders our approach approximately one order of magnitude cheaper than comparable bribing techniques (e.g., the whale attack). We demonstrate the technical feasibility of P2W attacks through publishing all relevant artifacts of this paper, ranging from calculations of success probabilities to a fully functional proof-of-concept implementation, consisting of an Ethereum smart contract and a Python client.

Keywords: Algorithmic Incentive Manipulation · Bribing · Smart Contracts · Ethereum · Bitcoin

1 Introduction

"*The system is secure as long as* ***honest*** *nodes collectively control more CPU power than any cooperating group of attacker nodes.*" Satoshi Nakamoto [19].

Despite an ever growing body of research in the field of cryptocurrencies, it is an open question if Bitcoin, and thus Nakamoto consensus, is incentive compatible under practical conditions, i.e., that the intended properties of the system emerge from the appropriate utility model [4,3]. *Bribing attacks*, in particular, target incentive compatibility and assume that at least some of the miners act ***rationally***, i.e., they accept bribes to maximize their profit. If the attacker, together with all bribable miners, can gain a sizable portion of the computational power, even for a short period of time, attacks are likely to succeed.

Since the first descriptions of bribing attacks [6,4], various attack approaches, which tamper with the incentives of protocol participants, have been presented for different scenarios and models. As bribing [22,15,17,24] , front-running [12,8,7] Goldfinger [13,11,3] and other related attacks, all intend to manipulate the incentives of rational actors in the system, we jointly consider them under the general term *algorithmic incentive manipulation* (AIM). So far, most proposed AIM attack strategies focus on optimizing a player's (miner's) utility by accepting *in-band* bribes, i.e., payments in the respective cryptocurrency [4,15,17,24] Thus, a common argument against the practicality of such attacks is that miners have little incentive to participate, as they would put the economic value of their respective cryptocurrency at risk, harming their own income stream. Another common counter argument against in-band bribing attacks is that they are considered expensive for an adversary (e.g., costs of several hundred bitcoins for one successful attack [15]), or require substantial amounts of computing power by the attacker.

In this paper, we present an AIM attack method called *Pay-To-Win* (P2W), which generalizes the construction of different AIM attacks on PoW Cryptocurrencies by leveraging smart contract platforms. Our attack requires no attacker hashrate, and an order of magnitude less funds than comparable attacks (i.e., the whale attack). To highlight the technical and economical feasibility of our approach, we provide a concrete instantiations of our P2W design, representing a new bribing attack. It uses a smart contract capable funding cryptocurrency (Ethereum) to finance and operate an attack on a (different) target cryptocurrency (Bitcoin). All bribes are paid in the funding cryptocurrency, i.e., out-of-band. Prior to our attacks, out-of-band payments have only been used in the context of Goldfinger-attacks, where the goal of an attacker is to destroy a competing cryptocurrency to gain some undefined external utility [13]. The attacks we present in this paper can be performed based on either strategy, using in-band profit, or as out-of-band Goldfinger-style attacks to destroy the value of the targeted cryptocurrency. In a multi-cryptocurrency world, P2W attacks demonstrate that utilizing out-of-band payments can pose an even greater threat to cryptocurrencies, as the argument that miners won't harm their own income stream must be critically examined in this context. Consider as an example two PoW cryptocurrencies that share the same PoW algorithm and have competing interests, for example Bitcoin and Bitcoin Cash. If rational Bitcoin miners face the opportunity of earning Ether for performing attacks on Bitcoin Cash, they may be willing to redirect their hashrate for this purpose, especially if they are guaranteed to receive the promised out-of-band rewards/bribes.

We show that such sophisticated trustless *out-of-band* attacks on Bitcoin-like protocols can readily be constructed, given any state-of-the-art smart contract platform capable for verifying the consensus rules of the target for the duration of the attack. Moreover, we show that the cost for an attacker can be considerably reduced by guaranteeing that participating bribees are reimbursed. Furthermore, cross-chain transaction ordering attacks can also be executed as targeted bribing attacks using our method. This possibility for rational miners to (trustlessly) auction the contents of their block proposals (i.e., votes) to the highest bidder raises fundamental questions on the security and purported guarantees of most permissionless blockchains.

Contribution: We propose a new design pattern, called *Pay-To-Win* (P2W), for out-of-band algorithmic incentive manipulation (AIM) attacks. To highlight the concept behind our design approach, we provide a *new out-of-band AIM attack* to incentivize double-spend collusion (Section 3.1).[7] On the technical level, we introduce *ephemeral mining relays*, as an underlying construction which is required to execute our trustless, time-bounded, cross-chain attack method. Morover we describe *guaranteed payment* of bribed miners even if the attack fails, which actually reduces the costs of such attacks. All artifacts reaching from calculations,simulations,a PoC and scripts used to derive the operational costs are available online.[8]

2 Model

We focus on *permissionless* proof-of-work (PoW) cryptocurrencies, as the majority of related bribing attacks target Bitcoin, Ethereum, and systems with a similar design. That is, we assume protocols adhering to the design principles of Bitcoin [19], generally referred to as Nakamoto consensus, or Bitcoin backbone protocol [10,20]. Within the attacked cryptocurrency we differentiate between *miners*, who participate in the consensus protocol and attempt to solve PoW-puzzles, and *clients*, who do not engage in such activities. Following the models of related work [15,22,17,4], we assume the set of miners to be fixed, and their respective computational power within the network to remain constant.

To abstract from currency details, we use the term *value* as a universal denomination for the purchasing power of cryptocurrency units, or any other out-of-band funds such as fiat currency. Miners and clients may own cryptocurrency units and are able to transfer them (i.e., their value) by creating and broadcasting valid transactions within the network. Moreover, as in prior work [15,17], we likewise make the simplifying assumption that exchange rates are constant over the duration of the attack.

In this work we follow the established *BAR-model* [14] and split participating miners into three groups and their roles remain static for the duration of the attack. Additionally, we define the *victim(s)* as another group or individual without hashrate.

Byzantine miners or attacker(s) (**B**lofeld): The attacker B wants to execute an incentive attack on a *target cryptocurrency*. B is in control of bribing

[7] Three other new attacks which we also described, as well as an in-depth analysis of the herein proposed attack, can be found in the extended version of the paper.

[8] Link to repository blinded for review

funds $f_{\mathcal{B}} > 0$ and has some, or no hashrate ($p_{\mathcal{B}} \geq 0$) in the target cryptocurrency. B may deviate arbitrarily from the protocol rules.

Altruistic or honest miner(s) (**A**lice): Altruistic miners A are honest and always follow the protocol rules, hence they will not accept bribes to mine on a different chain-state or deviate from the rules, even if it would offer larger profit. Miners A control some or no hashrate $p_{\mathcal{A}} \geq 0$ in the target cryptocurrency.

Rational or bribable miner(s) (**R**achel): Rational miners R controlling hashrate $p_{\mathcal{R}} > 0$ in the target cryptocurrency They aim to maximize their short term profits in terms of *value*. We consider such miners “bribable”, i.e., they follow strategies that deviate from the protocol rules as long as they are expected to yield higher profits than being honest. For our analyses we assume rational miners do not concurrently engage in other rational strategies.

Victim(s) (**V**incent): The set of victims, or a single victim, which loses value if the bribing attack is to be successful. The victims control zero hashrate, and therefore can be viewed as a client.

It holds that $p_{\mathcal{B}} + p_{\mathcal{A}} + p_{\mathcal{R}} = 1$. The assumption that the victim of an AIM attack has no hashrate is plausible, as the majority of transactions in Bitcoin or Ethereum are made by clients which do not have any hashrate in the system they are using.

Whenever we refer to an attack as *trustless*, we imply that no trusted third party is needed between briber and bribee to ensure correct payments are performed for the desired actions. Thus the goal is to design AIM in a way that the attacker(s), as well as the collaborating miners, have no incentive to betray each other if they are economically rational.

Communication and Timing: Participants communicate through message passing over a peer-to-peer gossip network, which we assume implements a reliable broadcast functionality. As previous bribing attacks, we further assume that all miners in the target cryptocurrency have *perfect knowledge* about the attack once it has started. Analogous to [10], we model the adversary Blofeld as *rushing*, meaning that he gets to see all other players messages before he decides his strategy, e.g., executes his attack. While the attack is performed on a *target cryptocurrency*, the distinct *funding cryptocurrency* is used to orchestrate and fund it. We also assume that the difficulty and the mean block interval of the funding chain is fixed for the duration of the attack, and that no additional attacks are concurrently being launched against either cryptocurrency.

3 P2W Attack Method

In this section, we introduce a new approach for algorithmic incentive manipulation attacks, which we call *Pay-To-Win* (P2W). Our approach relies on smart contracts and the specification of *block templates* by the attacker. These templates define the desired block structure for which Blofeld is willing to provide rewards in form of bribes. We consider *out-of-band* attacks to be technically more challenging, as well as more powerful regarding their capabilities (see be-low), therefore we focus on out-of-band attacks in this paper.[9] As the payment is performed *out-of-band*, we differentiate between a *target cryptocurrency*, where the attack is to be executed, and a *funding cryptocurrency*, where the attack is coordinated and funded. While the funding cryptocurrency must support sufficiently expressive smart contracts, there are no such requirements for the target cryptocurrency. For presentation purposes, we choose Bitcoin as target and Ethereum as the funding cryptocurrency to instantiate and describe our attacks. Theoretically, the attack can be funded on *any* smart contract-capable funding cryptocurrency, which is able to verify the PoW of the target. This advantage of being fund- and operable on any appropriate smart contract capable cryptocurrency renders these P2W attacks arguably more difficult to detect and protect against, as the victim(s) would have to monitor multiple, if not all, possible funding blockchains. Moreover, our attacks can also use additional privacy preserving techniques available on the funding cryptocurrency (e.g.,[18]) to hinder the traceability of funds and transactions of involved parties. Another advantage of out-of-band payments is, that they are not bound to the exchange value of the targeted cryptocurrency and thus can also be used for Goldfinger style attacks [13,3], as the assumption that miners of the target cryptocurrency would not harm their own revenue channel does not necessarily hold true anymore. This is an even more compelling argument in a world where multiple cryptocurrencies either share the same PoW algorithm, or hardware can be effectively used for mining other forms of PoW.

Our construction requires a combination of a smart contract based mining pool [23,16] and a temporary chain relay.[10] We call this underlying construction an *ephemeral mining relay* (EMR).[11] Chain relays are smart contracts which allow to verify the state of other blockchains, however, a naive chain relay implementation only allows to verify that a certain block (or transaction) was included in a chain with the most accumulated proof-of-work (i.e., heaviest chain). It does not allow to verify whether the blocks and transactions included in this heaviest chain are indeed *valid*, i.e., adhere to the consensus rules of the corresponding blockchain. In contrast to previous proposals, our EMR needs to be capable of validating if blocks adhere to the consensus rules of the target cryptocurrency. This is achieved by sufficiently restricting the allowed block structure. In our case the set of transactions within blocks generated by collaborating miners is specified by the block template provided by the adversary. As Blofeld wants to submit collected PoW solutions to Bitcoin, it is in his best interest to provide only templates including valid transactions. Conversely, collaborating rational miners do not care if the block template they mine on is actually valid in Bit-

[9] We also describe and evaluate three new in-band attacks targeting transaction ordering and transaction exclusion in the extended version of this paper. A in-band transaction exclusion attack was also described and analyzed in concurrent work by Winzer et al. [24], but no concrete instantiation was given.

[10] cf. `https://github.com/ethereum/btcrelay`

[11] We use the term "ephemeral" as the mining relay is instantiated only temporarily and does not require verification of the entire blockchain, but only the few blocks relevant for the attack.

coin, since the rewards they receive for solutions are guaranteed to be paid out by the smart contract in Ethereum.

Summarizing, our EMR takes care that the promised rewards are only paid to complacent bribees which have actively contributed to the attack. Therefore, the introduced attack can be considered *trustless*, both for the attacker as well as the collaborating bribed miners. Moreover, the attack does not require the adversary to control any hashrate, i.e., we assume $p_{\mathcal{B}} = 0$. To demonstrate the feasibility of our approach and the described attack, we implemented a fully functional prototype of our attack and evaluated its costs in Ethereum. The source code and all other artifacts of the evaluation are available on Github. [12]

3.1 Transaction Revision, Exclusion and Ordering Attack

To illustrate all underlying concepts, we present them within the context of a concrete attack. While we focus on transaction revision in our description, the presented attack also bears the possibility for arbitrary transaction exclusion and ordering. To execute our attack, Blofeld must construct a smart contract which temporarily rewards the creation of attacker-defined blocks on the target cryptocurrency. After its initialization, the smart contract can be used by him as well as by other collaborating miners/attackers/bribees to coordinate the attack and manage the investment and payout of funds.
Initialization phase (deploy,init): First the attacker (Blofeld) creates the uninitialized attack contract and publishes it on the Ethereum blockchain. This is done with a *deploy* transaction included in some Ethereum block e_0 from an Ethereum account controlled by the attacker.[13] Then, Blofeld creates a conflicting pair of Bitcoin transactions. The spending transaction tx_B is published on the main chain in Bitcoin immediately, and the double-spending transaction tx'_B is kept secret. After the confirmation period of k_V blocks (defined by the victim V) has passed on the Bitcoin main chain, Blofeld releases an initialization transaction, which defines the conditions of the attack in the smart contract on the Ethereum chain. The block e_1 represents the first block on the Ethereum chain after the Bitcoin block b_{k_V} has been published.

In e_1 the contract is initialized with $k_V + 1$ new Bitcoin block templates, each carrying the transactions from the original chain to collect their fees, but instead of tx_B the conflicting transaction tx'_B is included. Collaborating miners are now free to mine these new block templates. For the first template they are only allowed to change the nonce and the coinbase field to find a valid PoW and include their payout Ethereum address in the coinbase. This prevents front

[12] Link to repository blinded for review

[13] It is also possible to deploy and initialize the attack contract at the same time (e_1), but publishing an uninitialized attack contract upfront ensures that potential collaborators can audit it and familiarize themselves with the procedure. In any case, it is important that the double-spend transaction tx'_B is disclosed after block b_{k_V} on the main chain, as otherwise Vincent may recognize the double-spending attack and refuse to release the goods.

running of solutions (see Section 3.1). Once a solution has been found, it has to be submitted by the respective miner to the attack contract, which verifies the correctness of the PoW and that only allowed fields (nonce and coinbase) have been changed. After the first block (b'_1) in the sequence, also the previous block hashes of subsequent blocks ($\{b'_2, \dots\}$) have to be adjusted by collaborating miners. If a submitted solution is valid, the contract knows which previous block hash it must use to verify the next solution and so forth.

As soon as Blofeld becomes aware that a valid solution was broadcasted in the Ethereum P2P network, he uses the PoW solution to complete the whole block and submits it to the Bitcoin P2P network. Blofeld and the collaborating miners have an incentive to submit solutions timely. The collaborating miners want to collect an additional bribe ϵ in case the attack succeeds, and the attacker wants his blocks included in the Bitcoin main chain to receive the Bitcoin block rewards to his Bitcoin address, and in the best case, perform a successful double-spend.
Attack phase (update): Bribed miners now proceed to mine $k_V + 1$ blocks on the attack chain. If additional blocks are found on the main chain, the attacker can update the attack contract with new block templates for blocks $k_V + 2$ to N, where N is the maximum number of attack blocks that can be funded by the adversary. Note that N is not necessarily known by Vincent, Rachel or any other observer.
Payout phase (pay): The payout phase starts as soon as the attack phase has ended. This happens when k_B blocks have been mined on top of the last block for which a block template has been provided to the smart contract. In the best case, this happens at block $T = k_V + 1 + k_B$, but in our example one `update` with an additional block template was required, leading to $T = k_V + 2 + k_B$. The delta of k_B is a security parameter defined by the attacker, which should ensure that every participant had enough time to submit information about the longest Bitcoin chain to the contract and that the sequence of blocks relevant for the attack has received sufficient confirmations.[14]

The attack terminates as soon as the first block of height T is committed to the contract. This can be a block of the main chain, or the attack chain. After the attack has terminated, the contract unlocks the payment of compensations and rewards for the miners of the associated blocks. Now all miners who joined the attack and contributed blocks can collect their compensations and/or bribes from the contract.

To accurately pay out funds, the contract on Ethereum has to determine which chain in Bitcoin has won the race and is now the longest chain. Thereby, the contract has to distinguish between two possible outcomes:

Attack failed (Main chain wins): In this case the contract must compensate the bribed miners for their contributed blocks to the attack chain, which are now stale. These are at most $\{b'_1, \dots, b'_N\}$, Every collaborating miner who mined and successfully submitted a block to the attack contract receives the original Bitcoin reward (in Ether) for that block, without an additional ϵ.

[14] Ideally k_B is specified as an acceptance policy logarithmic in the chain's length as described in [21].

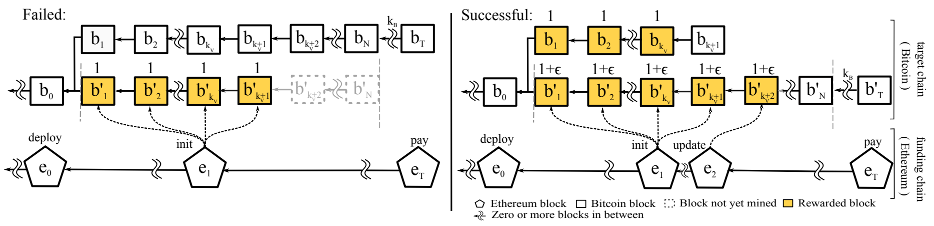

Fig. 1. Example blockchain structure and resulting payouts of a failed, and a successful attack. The colored blocks are rewarded by the attack contract, either with their original value (reward + fee normalized to 1) or with an additional ϵ if the attack was successful. The numbers above colored blocks indicate those normalized rewards.

Attack succeeded (Attack chain wins): If the attack chain wins, then the contract executes the following actions: 1) Fully compensate the miners of k_V main chain blocks starting from b_1, which are now stale. This is necessary to provide an incentive also for those miners to switch and contribute to the attack chain, as they otherwise would lose their rewards from blocks they contributed to the main chain if the attack is successful. 2) Pay the miner of every attack chain block, b'_1 to b'_{k_V+2} in our example (max. till b'_N), the full block reward plus an additional ϵ as a bribe in Ether.

Figure 1 shows the different stages of the attack on the funding cryptocurrency, as well as two different outcomes (failed and successful attack) on the target cryptocurrency. The paid out compensations (block rewards normalized to 1) and bribes (ϵ) are given above the respective blocks. Upon being invoked with a miner's cash-out transaction, the contract checks if the attack has already finished, i.e., a valid chain up to block height T is known, and which chain has won the race. Then the contract pays out accordingly.

Incentives to Submit Blocks: Since collaborating miners are competing for mined attack chain blocks and want the attack to be successful to receive the additional bribes, they have an incentive to timely submit their attack chain blocks to the contract. In any case, for every (valid) submitted Bitcoin attack chain block, the full Bitcoin reward is paid in Ether by the contract. In case of a successful attack an additional ϵ is paid. Therefore, Blofeld who initialized the contract and provided the funds has an incentive to submit the relevant part of the main chain, if a conflicting longer chain ($\{b_1, \ldots, b_T\}$) exists, since he would pay an additional ϵ for every block otherwise. Moreover, Blofeld has an incentive to submit every completed Bitcoin block (with the PoW provided by the bribees) to the Bitcoin network, because he is the one who receives the full Bitcoin block rewards as specified in his Bitcoin block template. **Ethereum Payout Address Derivation:** To determine the correct Ethereum payout addresses of collaborating miners, the following approaches are feasible: As soon as bribed miners start participating in the attack, they directly provide their Ethereum address as additional data in the coinbase field of every submitted Bitcoin block on the attack chain. As miners of the blocks b_1 to b_{k_V}, may not have disclosed their Ethereum address in the coinbase field already, another technique has to be used. For blocks where miners were not yet aware of the attack, they must prove to the contract that they indeed mined the respective block(s). If *Pay-to-Pubkey* outputs have been used in the respective coinbase transactions, the Bitcoin address public key can be used to derive the corresponding Ethereum address, as described and implemented in the Goldfinger attack example in [17]. This can also be achieved by providing the ECDSA public keys corresponding to *Pay-to-PubKey-Hash* payouts from the respective coinbase outputs to the smart contract. Thereby, the contract can verify if the keys correspond to the respective Bitcoin addresses and also derive the corresponding Ethereum addresses, as they rely on the same signature scheme.

3.2 Evaluation with Solely Rational Miners ($p_{\mathcal{R}} = 1$)

As rational miners will participate in the attack as long as it is expected to yield more profit than honest mining, the remaining question is, what budget in Ether is required by Blofeld (f_B) for the attack to succeed. As the Bitcoin block rewards and bribes have to be payed out in Ether, we assume a fixed exchange rate between cryptocurrencies to derive our lower bound in terms of BTC required. Blofeld has to lock funds in the attack contract for each submitted block template, to ensure complacent miners can be certain to receive their rewards if they submit blocks and thus are incentivized to join the attack. Therefore, the duration of the attack is the main driver for the required budget. As the duration is dependent on the security parameter k_V chosen by Vincent, $N > k_V$ has to hold for an attack to be feasible.
Necessary Attack Budget: For Bitcoin, a common choice is $k_V = 6$ requiring N to be at least 7. The budget of the attack contract must cover all rewards which could potentially be paid out by the contract. For the most expensive case, which is a successful attack, this encompasses: The bribes (ϵ) as well as Bitcoin block rewards including fees[15] (r_b), which we previously normalized to 1 in Figure 1. Assuming the current block reward (6.25 BTC), average fees (≈ 2 BTC), operational costs ($c_{operational} = 0.5$ BTC), as well as a bribe of $\epsilon = 1$ BTC, this leads to to a budget of 114.75 BTC which has to be provided to the attack contract in Ether upfront s.t., $f_B = k_V \cdot r_b + N \cdot (r_b + \epsilon) + c_{operational}$. As Blofeld receives the Bitcoin block rewards in case of a successful attack, the actual costs of the attack are *much smaller* than the required budget Blofeld has to lock in the contract.
Costs and Profitability of a Successful Attack: If the attack is successful, then Blofeld earns the block rewards on the main chain in BTC which compensate his payouts to bribed miners in Ether. The costs for a successful attack are thus reduced by $N \cdot r_b$ main chain blocks, whereas rewards must be paid for $N \cdot (r_b + \epsilon)$ block templates. The remaining costs of a successful attack stem from the $k_V \cdot r_b$ main chain blocks that have to be compensated on the attack chain s.t., $c_{success} = k_V \cdot r_b + N \cdot \epsilon + c_{operational}$. The initial k_V compensations are necessary to provide the same incentive for *all* miners that have already produced

[15] In a concrete attack of course r_b is not constant, but given by the coinbase output values of every submitted block.

blocks on the main chain to switch to the attack chain. Since we assume rational miners, the attack in this scenario is always successful if $N > k_V$ and $\epsilon > 0$ hold. For Bitcoin, this means that the costs of a successful double spend with $k_V = 6$ and $r_b = 8.25$ and $\epsilon = 1$ are $c_{success} = 57$ BTC. For a successful attack to be profitable, the value of the double-spend (v_d) has to be greater than this value. In Bitcoin, transactions carrying more than 57 BTC are observed regularly.[16] For comparison, in its cheapest configuration, the whale attack costs approximately 770 BTC [15], but it was simulated for a previous Bitcoin reward epoch, where block rewards have been higher. Even if we assume $r_b = 12.5$ BTC, our attack would cost 94.5 BTC, which is considerably lower than the whale attack. The remaining difference to our approach is that the whale attack does not assume all miners to be rational. In Section 3.3 we also extend our evaluation to this model by introducing altruistic miners.

Costs of a Failed Attack: Although the attack cannot fail in a model where all miners are rational and the attacker has enough budget, it is relevant for a scenario where $p_{\mathcal{R}} < 1$ to determine the worst case cost for an unsuccessful attack. In the worst case, the attack duration is N and not a single block produced by complacent miners (according to a published block template) made it into the main chain. Then the costs are determined by the duration N and the block rewards including fees (r_b) s.t., $c_{fail} = N \cdot r_b + c_{operational}$. Setting the same values for r_b and N amounts to approximately $c_{fail} = 58.25$ BTC in our example.

3.3 Evaluation with Altruistic Miners ($p_{\mathcal{A}} > 0 \ \land \ p_{\mathcal{R}} + p_{\mathcal{A}} = 1$)

We now discuss a more realistic scenario where not all miners switch to the attack chain immediately, i.e., some of them act altruistically. Altruistic miners follow the protocol rules and only switch to the attack chain if it becomes the longest chain in the network – but do not attempt to optimize their revenue, contrary to economically rational, i.e., bribable, miners.[17]

We derive the probability of the attack chain to win a race against altruistic miners, based on the budget of the attacker and the initial gap between those chains which has to be overcome k_{gap} where k_{gap} is initially set to $k_{gap} = k_V$. The difference between k_V and k_{gap} is that k_{gap} can increase when altruistic miners find a new block, while k_V is static. In other words, the attack chain must find $k_{gap}+1$ more blocks than the altruistic main chain – but must achieve this within the upper bound of N blocks (maximum funded attack duration). Each new block is appended to the main chain with probability $p_{\mathcal{A}}$, and to the attack chain with probability $p_{\mathcal{R}}$ respectively ($p_{\mathcal{A}} + p_{\mathcal{R}} = 1$). We therefore seek all possible series of blocks being appended to either chain, and calculate the sum of the probabilities of the series which lead to a successful attack. In a successful

[16] cf. `https://blockchair.com/bitcoin/outputs?s=value(desc),time(desc)&q=time(2020-10),value(6000000000..)#`

[17] Another explanation can be that some miners have imperfect information, which might be the case in practice.

series $i \in \mathbb{N}$ blocks are added to the main chain and $k_{gap}+i+1$ blocks are added to the attack chain. The probability for such a series is $p_{\mathcal{R}}^{k_{gap}+i+1} \cdot p_{\mathcal{A}}^{i}$.
For any prefix strictly shorter than the whole series, the number of appended blocks to the attack chain is smaller than $k_{gap}+1$, as otherwise the attack would have ended sooner. It follows that the last block in a successful series is always appended to the attack chain. The number of combinations for such a series is derived similarly to Bertrand's ballot theorem, with a difference of k_{gap} for the starting point. Assuming the attacker can only fund up to N blocks on the attack chain, the probability of a successful attack is hence given by:

$$\sum_{i=0}^{i \leq N-k_{gap}-1} \binom{k_{gap}+2i}{i} - \binom{k_{gap}+2i}{i-1} \cdot p_{\mathcal{R}}^{k_{gap}+i+1} \cdot p_{\mathcal{A}}^{i} \tag{1}$$

Using formula 1 we can calculate the success probability of the attack. Clearly, the attack requires $N > k_V$ to have a chance of being successful. As with the classical 51% attacks, the attack eventually succeeds once the bribable hash rate is above the 50% threshold and the number of payable blocks N grows. In other words, assuming more than $p_{\mathcal{R}} > 0.5$ rational hashrate, bribing attacks are eventually successful if they can be funded long enough. The relevant question is how expensive it is to sustain the attack for a long enough period s.t., the attack is expected to be successful.

Rational hashrate $p_{\mathcal{R}}$	Average whale attack costs epoch reward 12.5 c_{whale} in BTC	P2W epoch reward 12.5 $c_{expected}$ in BTC	P2W cost compared to whale	P2W N average	P2W epoch reward 6.25 $c_{expected}$ in BTC
0.532	293e+23	196.50	≈0.00%	109	159.00
0.670	999.79	108.50	10.85%	21	71.00
0.764	768.09	101.50	13.21%	14	64.00
0.828	1265.14	98.50	7.79%	11	61.00
0.887	1205.00	96.50	8.01%	9	59.00
0.931	1806.67	96.50	5.34%	9	59.00
0.968	2178.58	95.50	4.38%	8	58.00
0.999	2598.64	95.50	3.67%	8	58.00

Table 1: Comparison of attack costs for $k_V = 6$, all costs given in BTC. The costs for the whale attack are the average from 10^6 simulation results provided in [15]. For comparision different Bitcoin block reward epochs (12.5 and 6.25 BTC) are provided for our P2W attack, all with $c_{operational} =$ 0.5 BTC, and average fee per block of 2 BTC and a bribe $\epsilon = 1$ BTC.

Table 1 shows a comparison between the expected costs of a successful P2W attack, against the average costs of 10^6 simulations of the whale attack as presented in [15]. At a first glance, given that the attacker must pay collaborating miners regardless of the outcome of the attack, one may assume that the costs faced by the attacker are high compared to other bribing schemes. However, this is not the case. In our attack miners face *no risk* from participation – requiring only a *low bribe value* to incentivize sufficient participation for a successful attack, contrary to existing bribing attacks like the whale attack.

It can be observed that, in contrast to the whale attack, our attack becomes cheaper when $p_{\mathcal{R}}$ grows large since the race is won faster and therefore fewer bribes have to be paid. Moreover, the whale attack has to pay substantially more funds to account for the risk rational miners face if the attack fails. Our approach is hence approximately $\approx 87\%$ to $\approx 96\%$ cheaper than the whale attack. For $p_{\mathcal{R}} = 0.532$ the difference is so large, that the costs of our P2W attack are insignificant compared to the whale attack. The switch to a new Bitcoin block reward epoch has further reduced the costs of the attack s.t., the costs of a successful double-spending attack ($k_V = 6$) using our technique are around 60 BTC. In October 2020 alone, there where around 60 thousand Bitcoin transactions with outputs greater than 60 BTC.[18]

3.4 Evalution of the operational costs

We implement a fully functional attack contract including the EMR on Ethereum, which is capable of verifying the state of the Bitcoin blockchain .[19] We use Solidity v0.6.2 and a local Ganache instance for cost analysis, with a current gas price of 45 Gwei and an exchange rate 500 USD/ETH. Submitting a block template for a Bitcoin block amounts to 302,228 Gas ($ 6.80 USD). The costs for submitting and verifying a new Bitcoin block are 468,273 Gas ($ 10.54 USD) in the worst case. In total the costs of an example attack on Bitcoin with $k_V = 6$ and $k_B = 6$ are about $ 355.24 USD. This confirms that the costs for maintaining an attack contract including an EMR are marginal when compared to the potential scale of incentive attacks described in this paper. For comparison, the reward for a single Bitcoin block (*excluding* transaction fees) at the time of writing is approximately $ 120 000 USD.

4 Discussion and Mitigations

Our AIM attack highlights the security dependency between transaction value and confirmation time k_V, as also stated in [21]. As with the negative-fee mining pools presented by Bonneau in [4], there exists an interesting analogy between such an incentive manipulation attack and a mining pool. At an abstract level, the presented attack relies on a construction comparable to a mining pool, where the pool owner/attack operator defines specific rules for block creation for the targeted cryptocurrency within a smart contract. Moreover, every participant must be able to claim their promised rewards in a trustless fashion, based on the submitted blocks and state of the targeted cryptocurrency. The construction of an *ephemeral mining relay*, presented within this paper, provides exactly this functionality. Luu et al. [16] also proposes a mining pool (Smart pool) which itself is governed by a smart contract. However, its design and intended application scenarios did not consider use-cases with malicious intent. Smart pool does not enforce any properties regarding the content and validity of submitted blocks beyond a valid PoW, as an intrinsic incentive among participants is assumed to

[18] c.f. `https://blockchair.com/bitcoin/outputs?s=value(desc),time(desc)&q=time(2020-10),value(6000000000..)#`

[19] Blinded for review

earn mining rewards in the target cryptocurrency, which is only possible if valid blocks have been created.

Practical possibility: The focus of this paper is to improve upon existing attacks and demonstrate the technical feasibility of advanced bribing attacks, as well as to evaluate the associated costs. Hereby, the long term interests of miners of course also play an important role. There may be scenarios where miners are capable of providing PoW for a target blockchain, but at the same time do not have any long-term interest in the well-being of the target. Consider the real-world example of Bitcoin and Bitcoin Cash which utilize the same form of PoW and can be considered competitors. Thus, the question if the proposed attacks are possible in practice is difficult to answer scientifically. There is already empirical evidence from previous large scale attacks by miners, e.g., recent 51% attacks on Ethereum Classic and Bitcoin Gold, as well as incentive manipulation attacks and front-running [7]. To the best of our knowledge, none of the observed attacks has been as sophisticated as the new technique proposed in this paper, but of course, they can get better over time. Nevertheless, these cases demonstrate that large scale attacks happen, and that the topic of incentives in cryptocurrencies is an area which deserves further study. We see our paper as another important contribution in this direction.

Counter attacks: Counter bribing refers to the technique of countering bribing attacks with other bribing attacks [4,3]. For the victim(s), counter bribing is a viable strategy against AIM. The difficulty of successfully executing counter bribing highly depends on the respective scenario. In the end, counter bribing can also be countered by counter-counter bribing and so forth. Therefore, as soon as this route is taken, the result becomes a bidding game. If defenders have imperfect information, they may not be able to immediately respond with counter bribes. This illustrates an important aspect of AIM, namely their visibility. On the one hand, sufficiently many rational miners of the target cryptocurrency have to recognize that an attack is occurring, otherwise they won't join in and the attack is likely to fail. On the other hand, if the victims of the attack recognize its existence, they can initiate and coordinate a counter bribing attack. So the optimal conditions for AIM arise if all rational miners have been informed directly about the attack, while all victims/merchants do not monitor the chain to check if an attack is going on and are not miners themselves.

The great benefit of the herein described attacks is that bribes are paid out-of-band. Hereby, our attacks are rendered more stealthy to victims, who only monitor the target cryptocurrency. Of course their received rewards can be traced in the funding cryptocurrency, but available privacy techniques may be used to camouflage the real recipient of the funds e.g., [18]. It can hence be argued that counter attacks by victims are harder to execute as they are not immediately aware of the bribing value that is being bet against them on a different funding cryptocurrency. We also follow the argument in [4] that requiring clients to monitor the chain and actively engage in counter bribing is undesirable, and our out-of-band attacks further amplifies this problem as clients would have to concurrently monitor a variety of cryptocurrencies.

Cross-chain Verifiability: One crucial aspect of our attacks is that a smart contract within the funding cryptocurrency must be able to validate core protocol and consensus rules of the target chain, in particular it must be able to determine the validity of blocks. If this is not possible, the attack cannot be executed trustlessly. For example, it is currently not possible to execute an AIM against Litecoin using Ethereum as a funding cryptocurrency in a fully trustless manner, as it is economically unfeasible to verify the Scrypt hash function within a smart contract. However, it is generally beyond the reach of an individual cryptocurrency to dictate or enforce what other cryptocurrencies support in future versions of their smart contract languages. Thus, any such defensive decision of the target cryptocurrency may be mitigated by future changes in another cryptocurrency. Hence, such measures can not guarantee lasting protection.

5 Implications and Future Work

In this paper we introduced a new AIM attack method called Pay-To-Win (P2W) and showed that attacks utilizing the described techniques can readily be constructed given current smart contract platforms. The implications of our proposed method (and related AIM/bribing attacks) regarding the security guarantees of PoW cryptocurrencies are not yet conclusive and topic of future work. On the theoretical side, embedding and modeling incentive attacks in formalisms of Nakamoto style cryptocurrencies is non-trivial, as prevalent approaches do not consider rational participants [9,20,2], or explicitly exclude bribing [1]. Furthermore, no agreed upon game theoretic analysis technique for (PoW) cryptocurrencies currently exits, and it remains an open question if such an analysis could be rendered universally composable. The generalization and inclusion of AIM attacks and rational behavior in formal analysis frameworks for Nakamoto consensus based cryptocurrency designs, including approaches such as *Proof-of-Stake*, hence poses an interesting and important open research challenge. On the practical side, our new attack, as well as the existing body of research on AIM, demonstrates that it is not only the hashrate distribution among permissionless PoW based cryptocurrencies that plays a central role in defining their underlying security guarantees. The ratio of *rational* miners and available funds for performing AIM also form a key component, as rational miners can be incentivized to act as accomplices to an attacker. The possibility of trustless out-of-band attacks highlights that being able to cryptographically interlink cryptocurrencies increases this attack surface. Further, smart contract based AIM introduces the possibility to align the interests of multiple attackers who want to perform double-spends during the same time period, making low value double-spends theoretically feasible (as economically analyzed in [5]). Together with the topic of counter bribing, new research directions are opened up that raise fundamental questions on the incentive compatibility of Nakamoto consensus. Real world attacks targeting incentives, such as front-running [7], demonstrate that the existence of incentives cannot be ignored in PoW cryptocurrencies. To accurately reflect the security properties of permissionless PoW cryptocurrencies, some form of rationality has to be taken into account. The problem is, that as soon as rational players are considered, all previously proposed AIM/bribing methods, as well as the attack described in this paper, lead to interesting questions whether or not the incentive structures of prevalent cryptocurrencies actually encourage desirable outcomes. Even more so, in a world where multiple cryptocurrencies coexist it is likely not sufficient to model them individually as closed and independent systems.

6 Acknowledgements

We would like to thank the participants of the Dagstuhl Seminar 18152 (*Blockchains, Smart Contracts and Future Applications*), especially Samuel Christie and Sebastian Faust, as well as the participants of the Dagstuhl Seminar 18461 (*Blockchain Security at Scale*) for all the frutiful discussions.

This paper is based upon work partially supported by (1) the Christian-Doppler-Laboratory for Security and Quality Improvement in the Production System Lifecycle; The financial support by the Austrian Federal Ministry for Digital and Economic Affairs, the Nation Foundation for Research, Technology and Development and University of Vienna, Faculty of Computer Science, Security & Privacy Group is gratefully acknowledged; (2) SBA Research; the competence center SBA Research (SBA-K1) funded within the framework of COMET Competence Centers for Excellent Technologies by BMVIT, BMDW, and the federal state of Vienna, managed by the FFG; (3) the FFG Bridge 1 project 864738 PR4DLT. (4) the US-Israel Binational Science Foundation (BSF) (5) the Israel Cyber Bureau (6) the Technion Hiroshi Fujiwara cyber-security research center

References

1. Badertscher, C., Garay, J.A., Maurer, U., Tschudi, D., Zikas, V.: But why does it work? A rational protocol design treatment of bitcoin. In: EUROCRYPT 2018
2. Badertscher, C., Maurer, U., Tschudi, D., Zikas, V.: Bitcoin as a transaction ledger: A composable treatment. In: CRYPTO 2017
3. Bonneau, J.: Hostile blockchain takeovers (short paper). In: 5th Workshop on Bitcoin and Blockchain Research, FC 2018
4. Bonneau, J.: Why buy when you can rent? bribery attacks on bitcoin consensus. In: BITCOIN '16: 3rd Workshop on Bitcoin and Blockchain Research (2016)
5. Budish, E.: The economic limits of bitcoin and the blockchain. Tech. rep., National Bureau of Economic Research (2018)
6. Cunicula: Bribery: The double double spend, `https://bitcointalk.org/index.php?topic=122291`, accessed: 2021-1-31
7. Daian, P., Goldfeder, S., Kell, T., Li, Y., Zhao, X., Bentov, I., Breidenbach, L., Juels, A.: Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In: IEEE SP (2020)
8. Eskandari, S., Moosavi, S., Clark, J.: Sok: Transparent dishonesty: Front-running attacks on blockchain. In: FC 2019 - WTSC Workshop
9. Garay, J.A., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol: Analysis and applications. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015
10. Garay, J.A., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol with chains of variable difficulty (2016), accessed: 2017-02-06

11. Judmayer, A., Stifter, N., Schindler, P., Weippl, E.: Pitchforks in cryptocurrencies: Enforcing rule changes through offensive forking- and consensus techniques (short paper). In: CBT'18 (2018)
12. Kolluri, A., Nikolic, I., Sergey, I., Hobor, A., Saxena, P.: Exploiting the laws of order in smart contracts (2018)
13. Kroll, J.A., Davey, I.C., Felten, E.W.: The economics of bitcoin mining, or bitcoin in the presence of adversaries. In: Proceedings of WEIS (2013)
14. Li, H.C., Clement, A., Wong, E.L., Napper, J., Roy, I., Alvisi, L., Dahlin, M.: Bar gossip. In: USENIX OSDI (2006)
15. Liao, K., Katz, J.: Incentivizing blockchain forks via whale transactions. In: FC 2017 (2017)
16. Luu, L., Velner, Y., Teutsch, J., Saxena, P.: Smartpool: Practical decentralized pooled mining. In: USENIX Security Symposium, 2017
17. McCorry, P., Hicks, A., Meiklejohn, S.: Smart contracts for bribing miners. In: FC 2018 - 5th Workshop on Bitcoin and Blockchain Research (2018)
18. Meiklejohn, S., Mercer, R.: Möbius: Trustless tumbling for transaction privacy. Proc. Priv. Enhancing Technol. (2018)
19. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (Dec 2008)
20. Pass, R., Seeman, L., Shelat, A.: Analysis of the blockchain protocol in asynchronous networks. In: EUROCRYPT 2017
21. Sompolinsky, Y., Zohar, A.: Bitcoin's security model revisited (2016), accessed: 2016-07-04
22. Teutsch, J., Jain, S., Saxena, P.: When cryptocurrencies mine their own business. In: FC 2016 (2016)
23. Velner, Y., Teutsch, J., Luu, L.: Smart contracts make bitcoin mining pools vulnerable. In: FC 2017 - BITCOIN Workshop
24. Winzer, F., Herd, B., Faust, S.: Temporary censorship attacks in the presence of rational miners. In: IEEE EuroS&P Workshops 2019 (2019)