Pay To Win: Cheap, Crowdfundable, Cross-chain Algorithmic Incentive Manipulation 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

eprint from 2021-03-01 for SoK see https://eprint.iacr.org/2020/1614

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 [7].

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

[7] https://github.com/kernoelpanic/pay2win_artefacts

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 [39].

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 [14,15]. *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 [19,14], various attack approaches, which tamper with the incentives of protocol participants, have been presented for different scenarios and models. As bribing [45,34,36,49] , front-running [31,23,20] Goldfinger [32,29,15] 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 [14,34,36,49] 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 [34]), 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 [32]. 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.

1.1 Contribution

We propose a new design pattern, called *Pay-To-Win* (P2W), for out-of-band algorithmic incentive manipulation (AIM) attacks. To highlight the concepts behind our design approach, we provide instantiations for two *new AIM attacks* (Section 4 and 6).[8] Both are trustless for the attacker and the collaborating miners, rely on out-of-band payments in a different cryptocurrency, and do not require the adversary to control any hashrate. The first instantiation incentivizes deep forks and double-spend collusion. The second instantiation has less capabilities, but is cheaper as it does not require deep-forks, or in the best case even no forks at all. On the technical level, We introduce three crucial enhancements to AIM attacks: (i) *ephemeral mining relays*, as an underlying construction which is required to execute our trustless, time-bounded, cross-chain attacks, (ii) *guaranteed payment* of bribed miners even if the attack fails, which actually reduces the costs of such attacks, and (iii) *crowdfunded attacks*, to further reduce the individual cost of executing such attacks. Summarizing, our contributions are as follows:

- P2W attack method to guarantee payments to participating bribees
- An instantiation for a trustless out-of-band AIM attack to incentivize double-spend collusion.
- An instantiation for a trustless out-of-band AIM attack to incentivize transaction exclusion and/or ordering
- An approach to crowdfund out-of-band double-spending attacks
- Concrete cost estimates for all our attacks, as well as a PoC of our attack smart contract to demonstrate the feasibility and estimate operational costs.

[8] We also describe and evaluate two new in-band attacks targeting transaction ordering and transaction exclusion in the Appendix G. The latter (in-band transaction exclusion) was also described and analyzed in concurrent work by Winzer et al. [49], but no concrete instantiation was given.

All artifacts reaching from calculations and simulations to the PoC are available online [9].

2 Model

We focus on *permissionless* [48] 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, or its backbone protocol [39,26,40], which is sometimes referred to as Nakamoto consensus [22,44]. 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 [34,45,36,14], 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 [46,34,36], we likewise make the simplifying assumption that exchange rates are constant over the duration of the attack.

We split participating miners into three groups and their roles remain static for the attack duration. Categories follow the *BAR (Byzantine, Altruistic, Rational)* [10,33] [33] behavior model. Additionally, we define the *victim(s)* as another group or individual without computational power, i.e, 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 funds $f_B > 0$ that can be in-band or out-of-band, depending on the attack scenario. He 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 such as selfish mining [24].

[9] https://github.com/kernoelpanic/pay2win_artefacts

– **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.

2.1 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 [26], 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 cryptocurrencies.

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 below), therefore we focus on out-of-band attacks in this paper [10] 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

[10] The description of two in-band attacks can be found in the appendix of the paper.

funding cryptocurrency, which fulfills the requirements listed in Section E. 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.,[37,4]) to hinder the traceability of funds and transactions of involved parties, providing another reason why our attacks can be considered more stealthy compared to attacks utilizing in-band payments. 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 [30,32,15], 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.

We present two instantiations of attacks utilizing our P2W approach: *P2W Tx revision/exclusion/ordering* and *P2W Tx exclusion/ordering*. Both attacks differ regarding their capabilities as well as their costs. The first also allows to revise already confirmed transactions and thereby facilitate double spend collusion, while the second is only capably of incentivizing the exclusion as well as the ordering of transactions, but therefore is substantially cheaper. What both instantiations have in common, is that their construction requires a combination of a smart contract based mining pool [47,35] and a chain relay [18,50,1]. We call this underlying construction an *ephemeral mining relay* (EMR). The EMR is introduced and evaluated when explaining our first attack. It takes care that the promised rewards are only payed to complacent bribees which have actively contributed to the attack. Therefore, the two introduced attacks can be considered *trustless*, both for the attacker as well as the collaborating bribed miners. Moreover, our attacks do 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 most powerful attack and evaluated its costs in Ethereum. The source code and all other artifacts related to the evaluation can be found on Github [11].

4 P2W Transaction Revision, Exclusion and Ordering Attack

To illustrate all underlying concepts, we start with a description of our most powerful attack, which allows for transaction revision and thus directly facilitates double-spend collusion. While we focus on transaction revision in our description, the presented attack also bears the possibility for transaction exclusion or ordering.

[11] . [12]

On a high level, miners are offered bribes in a funding cryptocurrency (in our case Ethereum) to mine blocks on the favored branch of a target cryptocurrency (in our case Bitcoin) in which the adversary is executing a double-spend. Moreover, we show how the attack can be constructed to always reward collaborating miners, regardless of the outcome of the attack. Interestingly, this renders our approach significantly cheaper than comparable attacks [34]. As a modification to reduce the costs, we also describe how smart contracts can be used to crowdfund and/or combine multiple double-spending attempts into a single coordinated attack, which further reduces the costs for participants. This implies that theoretically all of the transactions in a target's block could be double-spending attempts by the different entities which performed them.

To execute our attacks, Blofeld must construct a smart contract which temporarily rewards the creation of attacker-defined blocks on the target cryptocurrency. We call this technique an *ephemeral mining relay* (EMR), and evaluate its construction at the end of this section. A EMR requires a main attacker for initialization, after which it 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.

4.1 Description

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 listed above the respective blocks. The different stages are as follows:

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) 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.

[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.

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 on these 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 running of solutions (see Section 4.1). 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 the attacker 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 to get 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.

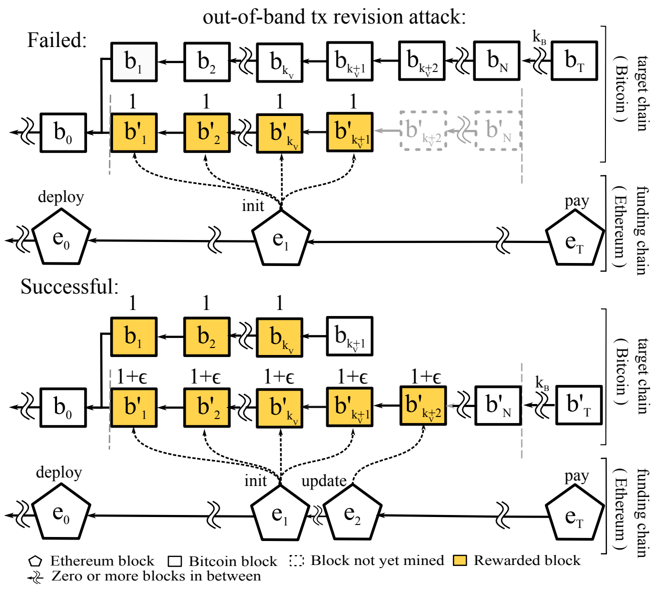

Fig. 1. Example timeline showing blockchain structure and resulting payouts of a failed, and a successful tx revision attack with out-of-band payments. After k_V blocks on the target chain have passed, the attack contract is initialized with (at least) $k_V + 1$ block templates. The double-spend transaction(s) are included in block b'_1. The payouts are performed in block e_T. The colored blocks are rewarded by the attack contract, either with their original value (reward + free normalized to 1) or with an additional ϵ if the attack was successful. The numbers above colored blocks indicated those normalized rewards. If the attack succeeds, the first k_V blocks on the Bitcoin main chain also have to be compensated to provide an incentive for the respective miners (of those blocks) to also mine on the attack chain.

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 $\{b'_1, \ldots, b'_N\}$, Every collaborating miner who mined and successfully submitted a block to the attack contract receives the reward for that block without an additional ϵ.
– *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.

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 Information 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 submit their attack chain blocks to the attack timely. Additionally, Blofeld who initialized the contract and provided the funds has an incentive to submit the relevant part of the main chain, if such a conflicting longer chain ($\{b_1, \ldots, b_T\}$) exists, since he would pay an additional ϵ for every block otherwise. Therefore there is always some actor who has an incentive to submit the correct longest chain to the attack contract.

Ethereum Payout Address Derivation To determine the correct Ethereum payout addresses of collaborating miners, the following approaches are feasible. In the simplest case, all bribed miners directly provide their Ethereum address in the coinbase field of every submitted Bitcoin block on the attack chain. Alternatively, they can disclose their public keys directly via *pay-to-pubkey* outputs

[14] Ideally k_B is specified as an acceptance policy logarithmic in the chain's length as described in [43].

in the coinbase transaction in Bitcoin. The Bitcoin address public key can then be used to derive the corresponding Ethereum address, as described and implemented in the Goldfinger attack example in [36].

For the first k_V main chain blocks, where miners were not yet aware of the attack, they must prove to the contract that they indeed mined the respective block(s). This can be achieved, e.g., by providing the ECDSA public keys corresponding to the payouts in the respective coinbase outputs to the smart contract, such that it can check if they match and then recompute the corresponding Ethereum address.

4.2 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:

$$f_B = k_V \cdot r_b + N \cdot (r_b + \epsilon) + c_{operational} \quad (1)$$

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.

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

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.

$$c_{success} = k_V \cdot r_b + N \cdot \epsilon + c_{operational} \tag{2}$$

The initial k_V compensations are necessary to provide the same incentive for *all* miners that have already produced 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 [34], 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 4.3 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):

$$c_{fail} = N \cdot r_b + c_{operational} \tag{3}$$

Setting the same values for r_b and N amounts to approximately $c_{fail} = 58.25$ BTC in our example.

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

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

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

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 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:

$$c(i) \coloneqq \binom{k_{gap}+2i}{i} - \binom{k_{gap}+2i}{i-1} \tag{4}$$

Assuming the attacker can only fund up to N blocks on the attack chain, the probability of a successful attack is hence given by:

$$\sum_{i=0}^{i \leq N-k_{gap}-1} c(i) \cdot p_{\mathcal{R}}^{k_{gap}+i+1} \cdot p_{\mathcal{A}}^{i} \tag{5}$$

Using formula 5 we can calculate the success probability of the attack. Figure 2 shows the probabilities for different values of rational hashrate $p_{\mathcal{R}}$, as well as different amounts of blocks N these bribed miners can be rewarded/compensated for. The number of confirmation blocks required by victim Vincent is $k_V = 6$. 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.

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

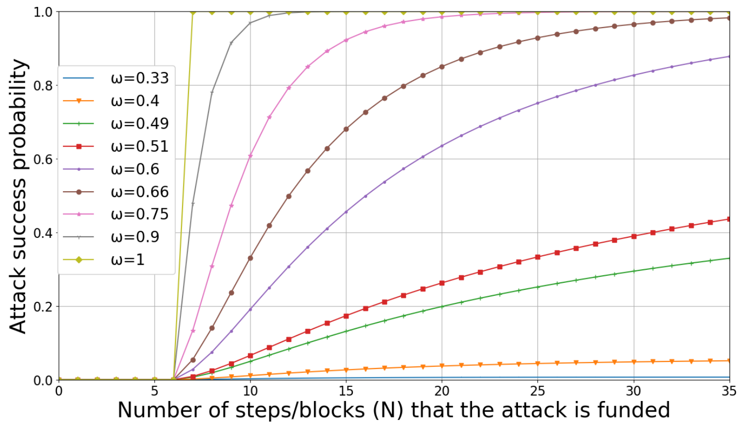

Fig. 2. Attack success probability of a double-spending attack depending on the amount of blocks N that can be compensated/rewarded and different values for the rational hashrate $p_{\mathcal{R}}$. The number of required confirmation blocks by Vincent is set to $k_V = 6$.

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 [34]. 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 [34]. 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].

4.4 Evalution of the attack contract

To verify the outcome of the attack and correctly pay rewards in trustless out-of-band scenarios, our attack contract includes a construction which we call an *ephemeral mining relay* (EMR)[19], as it combines the functionality of a chain relay [18,50,1] and mining pool [47,35].

Chain relays are smart contracts which allow to verify the state of other blockchains, i.e., verify the proof-of-work and difficulty adjustment mechanism, differentiate between the main chain and forks, and verify that a transaction was included within a specific block (via SPV Proofs [11]). 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 Bitcoin, since the rewards they receive for solutions are guaranteed to be paid out by the smart contract in Ethereum.

Liveness The liveness of chain relays in general depends upon the submission of new blocks to advance their state. Therefore, if the relay starves through a

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

[19] 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.

lack of submitted blocks - long range attacks have a higher chance to succeed, as attackers gain additional time to compute long fake chains. In our concrete EMR instantiation liveness is less of an issue, as the duration of the attack is finite and well defined. Moreover, involved actors have an incentive to submit the correct information to the relay in a timely fashion. Consider, for example, a rational miner R who mined a block for the attack chain according to a template. Then R has an incentive to submit the solution to the PoW for this template timely, since he is competing with other rational miners for the offered rewards and bribe. As the additional bribe ϵ is only paid if the attack is successful, this further incentivizes rational miners to publish solutions timely. Our scenario also enables the attacker, at any stage, to cease publishing additional block templates in order to reduce his losses in case the attack appears to fail.

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 [20]. 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. The cost estimates for the identified operations are summarized in Table 2. 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.

Operation	**Approx. costs**	
	Gas	**USD**
Deployment	6 156 688	138.53
Initialization phase	1 364 277	30.70
Attack phase	8 203 136	184.57
Payout phase	64 511	1.45
Total operational costs	15 788 612	355.24
submit one block template	302 228	6.80
submit one block	468 273	10.54

Gas price 45 Gwei, exchange rate 500 USD/ETH [2]

Table 2: Overview of operational costs $c_{operational}$ for each of the main Ethereuem smart contract operations of the attack contract (including the EMR) executing a successful attack on Bitcoin with $k_V = 6$ and $k_B = 6$.

[20] Blinded for review

4.5 Evaluation of Desynchronization

Publishing new block templates timely is a key requirement of this attack. So the question is, can we rely on the assumption that the difference between block intervals on the two chains, namely Bitcoin and Ethereum, is big enough such that before every new Bitcoin block there will be a new Ethereum block announcing the new block template? In other words: What is the probability that the two chains (funding and attack chain) *desynchronize* during an attack, i.e., that two Bitcoin blocks are mined in close succession without an Ethereum block in between. To identify the need to account for such events within the duration of an attack, we analyze the probability that the block intervals fluctuate in a way such that Bitcoin blocks are mined in close succession.

The time between Bitcoin and Ethereum blocks follows an exponential distribution. Assuming constant difficulty and overall hashrate, Ethereum has a mean block interval, i.e., an expected value of 15 seconds ($E_{ETH}(x) = 15$), whereas Bitcoin has a mean block interval of $10 \cdot 60$ seconds ($E_{BTC}(x) = 600$). To approximate the probability that the two chains desynchronize, we first calculate the probability that the time between two Bitcoin blocks is less than the Ethereum mean block interval ($x = 15$):

$$\lambda = \frac{1}{E_{BTC}(x)} \tag{6}$$

$$P(X < x) = 1 - e^{-\lambda \cdot x} \tag{7}$$

$$P(X < 15) \approx 2.47\% \tag{8}$$

The probability that this happens within N Bitcoin blocks i.e. the probability that the time between two Bitcoin blocks is smaller than 15 seconds during N total Bitcoin blocks is given by:

$$P(N) = 1 - \big(1 - P(X < 15)\big)^{N-1} \tag{9}$$

$$P(32) \approx 53.93\% \tag{10}$$

This result already shows that it is necessary to provide templates for more than one Bitcoin block in one Ethereum block when executing long running attacks.

We are now interested in the numbers of block templates the attacker has to provide per Ethereum block. Therefore, we analyze how probable it is that at least n Bitcoin blocks are mined before one Ethereum block. We approximate this value by calculating the probability that at least n Bitcoin blocks are found within the Ethereum mean block interval of 15 seconds. The Bitcoin block discovery is a Poisson point process, where the Poisson distribution parameter $\Lambda = E(X = n) = \frac{t}{E_{BTC}(x)}$ refers to the expected value of the number of events happening within $t = 15$ time. Then the complementary probability of finding at most $n-1$ blocks is given by:

$$P(X > n) = 1 - P(X \leq n-1) \tag{11}$$

$$P(X \leq n) = F(x) = e^{-\lambda} \sum_{i=0}^{n-1} \frac{\lambda^i}{i!} \tag{12}$$

$$P(X > 1) \approx 2.47\% \tag{13}$$

$$P(X > 2) \approx 0.03\% \tag{14}$$

$$P(X > 3) \approx 2.556 \cdot 10^{-4}\% \tag{15}$$

$$P(X > 4) \approx 1.595 \cdot 10^{-6}\% \tag{16}$$

Since both chains start at the same point in time, $n = 1$ already refers to a sequence of two Bitcoin blocks without an Ethereum block in between. We now calculate the probability that at least n Bitcoin blocks are found within the mean Ethereum block interval t during a period of N Bitcoin blocks in total:

$$P(n, N) = 1 - \big(1 - P(N > n)\big)^{\lceil (N-1)/n \rceil} \tag{17}$$

$$P(n = 1, N = 32) \approx 53.930\% \tag{18}$$

$$P(n = 2, N = 32) \approx 0.490\% \tag{19}$$

$$P(n = 3, N = 32) \approx 0.003\% \tag{20}$$

So when providing three Bitcoin block templates, there remains approximately a 0.490% chance that all of them are consumed before the next Ethereum block is published.

To further justify these numbers and account for the fact that Ethereum blocks are exponentially distributed as well, we implemented a tool to simulate such parallel blockchain chain executions. Measuring the probability of desynchronization yields comparable results to our calculations with a mean Ethereum block interval of 15 seconds. After 10,000 runs of our simulation limited to $N = 32$ total Bitcoin blocks each, a chain of at least two consecutive Bitcoin blocks before a corresponding Ethereum block was found in 53.0% of all cases. A chain of at least three consecutive Bitcoin blocks was found in 1.57% of all cases, a chain of at least four consecutive Bitcoin blocks in 0.08% of all cases. Consecutive chains of length 5 or longer have never occurred during 10,000 runs.

Desynchronization Prevention As Section 4.5 shows, the attacker should not rely on the assumption that the difference between block intervals on the two chains, e.g., Bitcoin and Ethereum, is big enough such that before every new Bitcoin block there will be a new Ethereum block announcing the new block template. Therefore, the attacker is advised to publish block templates for multiple blocks in advance (leaving references to previous blocks to be filled in by miners) [21]. In this case, only the first block includes a *previous block hash* field,

[21] Furthermore, in practice collaborating miners would want to have at least a couple of block templates available to ensure that their hardware does not stall.

whereas in subsequent block templates this value is left empty and has to be filled by collaborating miners based on the current attack chain state. Later, the contract can use previously submitted valid attack blocks to check the validity of the submitted solutions, i.e., if they form a valid chain and have sufficient difficulty. This solution is implemented in our PoC.

Other approaches to ensure that new block templates are available to rational miners independently of block intervals in the funding cryptocurrency are also conceivable. Strictly speaking, it is not even necessary that the Ethereum block with the new block template has been mined before the next Bitcoin block for which the template has to be used. This is possible if the attack contract is implemented such that it enables collaborators to provide a valid Ethereum transaction signed by the attacker as a proof that the therein announced new block template for a specific attack was approved, alongside their solution. Then any such transaction can be seen as a guarantee for the collaborating miners that they will receive a reward if they mine a block according to the template. At some later point the transaction defining the target chain block template is included in the funding cryptocurrency and presents proof to the attack contract that the respective block on the target chain was based on a valid template. The drawback of this method is, that it requires some way to prevent equivocation of the attack operator to prevent that more signed block templates are available than actual funds in the contract.

5 Crowdfunding and Multiple Attackers

Our attack from Section 4.1 also opens up the possibility to be crowdfunded. The simplest crowdfunding approach would be to allow donations as soon as the attack contract has been deployed. This method allows to collect funds but does not offer any guarantees for the backers.

Ideally, any solution which incentivizes multiple attackers to perform double-spending attacks concurrently, would allow to split the funds for the attack among collaborators. Thereby, multiple double-spends of low value transactions by different parties could be made feasible if they together accumulate enough attack funds (f_B). The main challenges that have to be solved in such a scenario are as follows:

- It has to be ensured that every collaborating attacker, who invests funds to achieve a double-spend attack, has some chance that his individual double-spend is successful, i.e., if the invested value is used by the contract, then the according double-spend attack for the respective transaction has to be performed.
- It has to be ensured that the attack cannot be poisoned by collaborating attackers such that they are able to sabotage the whole attack for all participants, i.e., it should not be possible for any participant to cause the attack to fail because of their inputs.
- The attack should not rely on any trusted third party.

We now outline an approach to achieve these goals within the framework of our previous attack s.t. its general structure is preserved. On a high level, the stages of the modified attack are as follows. First, the initialization transaction only announces that an attack might happen and that the block interval from b_1 to b_{k_V} will be affected. Then, all Bitcoin users who have performed transactions in block b_1 can decide whether or not to invest in the attack to potentially double-spend their transaction. The collaborating attackers, i.e., the backers, submit their double-spending transaction to the contract, together with some bribing funds in Ether that increase the overall funds f_B of the attack[22]. This crowdfunding attack approach can be viewed as a practical instantiation of an analysis performed in [17], where all payouts of a single block are viewed as the theoretical gain of a of double-spending attack. In [17] such a situation is analyzed from a financial perspective regarding the overall achievable economic security of Nakamoto consensus.

If the funding goal for reverting at least $k_V + 1$ blocks has been reached, the attack starts as previously described. Since the attacker who initialized the contract has to take care of producing new block templates for the chain containing the double-spend transactions, some method has to be implemented that the transactions of other backers are assured to be included in b'_1. We describe a method which requires a collateral from the original attacker (Blofeld) as high as the funds he wants to collect (f_B). In doing so, it can be ensured that the other backers only pay if their transaction was actually included in the new chain in block b'_1, which can be proven to the smart contract. Otherwise they are refunded from the collateral submitted by the initial attacker.

The phases of the attack are as follows:

1) Blofeld who initiates the attack, deploys an attack contract in Ethereum and locks his collateral of value f_B with this contract. Additionally, he publishes his spending transaction tx_{B_1} on the main network.

2) Once k_V blocks on the main chain have been mined, Blofeld initializes the attack contract with his double-spend tx'_{B_1}, his part of the attack funds f_{B_1}, a reference to the block b_1 which is to be forked, as well as a reference to the common ancestor block b_0.

3) Everybody who has included a transaction in block b_1 is then allowed to submit double-spending transactions $tx'_{B_{\{2,\dots,x\}}}$ including some amount of Ether $f_{B_{\{2,\dots,x\}}}$ that he or she is willing to invest in the attack. If these backers reach the funding goal of compensating at least $k_V + 1$ blocks before $k_V + 1$ main chain blocks have been submitted to the attack contract, the attack starts automatically. All invested funds (excluding the collateral f_B) are then free to be used by the EMR as described in the original attack.

3) Once the attack has been started by the attack contract, Blofeld publishes a block template to the attack contract. The Merkle branch of this template includes all submitted double-spending transactions $tx'_{B_{\{2,\dots,x\}}}$, which are i) valid

[22] Theoretically, an attacker can also specify a fixed rate of funds he wants to collect, depending on the overall value of the submitted Bitcoin transaction which should be double-spent.

according to information from his full node ii) backed by some ether. Additionally, the attack contract has to require some freshness information such that Blofeld is unable to produce blocks before officially starting the attack to rip compensations increasing his invested value f_{B_1} from his fellow backers. An example for such a freshness guarantee would be the inclusion of the latest funding chain block hash e_1 in the block template.

4) Then the attack proceeds as originally described.

5) When N blocks are mined and published to the attack contract, the backers who have not witnessed that their double-spending transaction was included in the attack chain can now claim their invested Ether back from the attack contract. Therefore, the attack contract automatically allows any backer to reclaim their money if Blofeld cannot submit a valid Merkle inclusion proof for the respective double-spending transaction.

In this approach, Blofeld has to provide a collateral as large as the total funds required for a successful attack f_B. If he behaves honestly, the collateral will be returned to him by the attack contract once the attack has ended – regardless if it was successful or failed. The collateral ensures that the initiator is able to compensate additional backers, in case their funds were used for the attack, but Blofeld did not include their double-spending transaction(s).

Like all other backers, Blofeld is required to invest funds f_{B_1} into the double-spending attack (in addition to the required collateral f_B). This investment by Blofeld should ensure that he is indeed willing to execute an attack and also loses funds if he is not able to provide correct block templates. For example, if he as an initiator purposely stalls the attack e.g., by not producing any block templates, or not forwarding them in time to the Bitcoin main network, the attack will fail. But then he will also lose his invested funds f_{B_1}. Thereby, asymmetric losses in cases where Blofeld intentionally lets the attack fail can be avoided by backers. Thus, backers are advised not to invest more Ether than f_{B_1} provided by Blofeld.

6 P2W Transaction Exclusion and Ordering Attack

In this section, we describe a modification of our attack from Section 4.1, which exclusively targets transaction exclusion and ordering. Thereby, the attack becomes less powerful but also substantially cheaper, as we show in our evaluation. Nevertheless, the resulting attack could be used to perform multiple front-running attacks at once, or/and to censor certain transactions. Such attacks can be profitable for an attacker attempting to falsely close an off-chain payment channel (i.e., publish an old/invalid state) but prevent the victim from executing the usual penalizing measures [41,38,21]. The attack presented in this section can also be viewed as a form of the *feather forking* attack proposed by Miller [16]. In a feather fork the attacker publicly promises that he will ignore any block containing a blacklisted transaction. The attack proposed in the paper at hand uses smart contracts on a funding cryptocurrency to provide a more credible threat.

Initialization Phase (init) The attacker's goal is to prevent an unconfirmed transaction tx_V from being included within N newly mined Bitcoin blocks . As in our previous attack, the smart contract is initialized and updated with *block templates*, which specify the content of the block according to the needs of the attacker. These templates have to be used by the collaborating Bitcoin miners to be eligible for rewards. This allows the attacker to fully control the content of the mined blocks, including ordering and inclusion of only desirable transactions. For each block template, the corresponding compensation and bribe is conditionally locked within the smart contract, ensuring miners will be reimbursed independently of the final attack outcome as long as they provide a valid solution. In contrast to our fist attack, the attack can start immediately after it is initialized and does not have to wait for k_V blocks to pass, nor must it be initialized with $k_V + 1$ blocks.

Attack Phase (update) As in our previous attack, rational miners submit valid Bitcoin blocks, based on the attacker's block templates, to the attack smart contract on Ethereum via the attack contract, which implements an EMR and verifies that they form a valid chain. At each step, the attacker can add new Bitcoin block templates after each submission to the attack contract and, if necessary, can even increase the bribes. If no new templates are submitted, the attack contract switches to the payout phase after k_B blocks. Note that it is possible to include more than one block template in a single block, as shown in Figure 3 for block e_3 (for details see Section 4.5).

Payout Phase (pay) Miners can claim payouts in the attack contract once k_B Bitcoin blocks have been mined after the attack has ended (k_B being a security parameter defined by the attacker). The attack smart contract is responsible for verifying the validity of submitted blocks, i.e., their PoW in compliance with the specified block template, and that all blocks form a valid attack chain. If a submitted PoW is valid, the attack contract rewards miners even if the attack chain did not succeed to become the main chain, i.e., collaborating miners *face no risk*. The first miner to submit a valid PoW for the respective block template will, in any case, receive value equivalent to the full Bitcoin block reward in Ether, regardless if the attack has failed, plus an extra ϵ if the attack is successful.

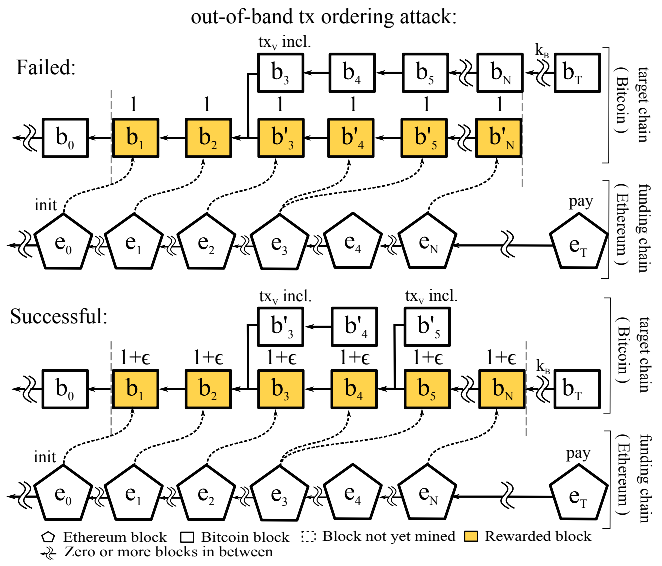

Fig. 3. Example timeline showing blockchain structure and resulting payouts of a failed, and a successful transaction exclusion and ordering attack with out-of-band payments. The attack is initialized when the attack contract is published in block e_0. Block templates are published as transactions in the funding cryptocurrency and refer to blocks in the target cryptocurrency. The payouts can be performed after k_B blocks. The colored blocks are rewarded by the attack contract, either only with their original value (reward + fee normalized to 1) or with an additional ϵ if the attack was successful. The numbers above colored blocks indicated those normalized rewards for the respective block.

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

We now derive a lower bound for the financial resources (*budget*) required from Blofeld in Ether (f_B) for this attack. Let us assume Blofeld wants to run the attack for N blocks (before the attack has finished, N is only known to the attacker).

Necessary Attack Budget The budget of the attack contract must cover and compensate all lost rewards[23], for every Bitcoin attack chain block in Ether in case the attack fails, plus an extra bribe ϵ per block in case the attack was successful. These values together with the funds of the attacker f_B, define the maximum duration of the attack N in terms of attack chain blocks that can be

[23] This encompasses, block rewards including fees. In a concrete attack r_b is not constant, but given by the coinbase output values of every submitted block.

financed:

$$f_B = N \cdot (r_b + \epsilon) + c_{operational} \tag{21}$$

There, $c_{operational}$ specifies the operational costs for smart contract deployment and execution (e.g., gas costs in Ethereum). Compared to the current block rewards, the operational costs for managing the smart contract are insignificant given the measurements in [36] and Section 4.4. Although, costs currently being around 166 USD (see 4.4), we decided to set $c_{operationl} = 0.02$ BTC to provide a future-proof and permissive margin. Assume an attacker wants to specify the transaction ordering and/or exclusion in Bitcoin for the duration of one hour i.e., $N = 6$. A lower bound for the budget of the attacker f_B can thus be derived by the current block reward (6.25 BTC) including approximated[24] fees (1 BTC) amounting to $r_b = 7.25$ BTC. Providing an additional $\epsilon = 1$ BTC, yielding approximately 49.52 BTC as a lower bound for the budget in this example.

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. Note that the actual costs for a failed attack can be much lower, since Blofeld is able to halt the attack by not publishing any further block templates. In the *worst case* the attack duration is N and not one block produced by complacent miners (according to a published block template) made it into the main chain. Then the costs would be close to the maximum budget, reduced by $N \cdot \epsilon$, which amounts to approximately $c_{fail} = 43.52$ BTC with our chosen values for N, r_b and $c_{operational}$.

Costs and Profitability of a Successful Attack If the attack is successful, the attacker 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.

$$c_{success} = N \cdot \epsilon + c_{operational} \tag{22}$$

Therefore, the costs of a successful attack *only depend on the* ***bribe*** *paid per block as well as the* ***operational costs***. Since we assume only rational miners, the attack in this scenario is always successful and *no fork* will be required if $\epsilon > 0$. For a successful attack to be profitable, the amount (v_a) gained from ordering, or transaction withholding, must exceed $c_{success}$.

[24] According to `https://blockchain.com/charts` the average transaction fees per Bitcoin block over the last year are 0.73 BTC. Accounting for standard deviation of fees and produced blocks per day the value varies between 0.79 BTC and 0.67 BTC. To provide a permissive margin we round to 1 BTC.

While the attacker must have the funds to compensate collaborating miners regardless of the outcome of the attack – the attack becomes cheaper than comparable attacks since the additional bribe does not have to account for the risk of getting nothing, faced by rational miners in other bribing scenarios. Other previously proposed AIM approaches require the attacker to have a sizeable portion of the overall hashrate (in the target cryptocurrency) under their direct control to stand a chance. For example `CensorshipCon` [36] which is also aiming at transaction exclusion but specifically for Ethereum, requires $p_{\mathcal{B}} > 1/3$, or `Script Puzzle 38.2%` as described in [45], which would also allow to change the order of subsequent blocks requires $p_{\mathcal{B}} > 38.2\%$. Acquiring or sustaining the required amount of hashrate already bears large costs, not to mention the additionally required bribes. The costs for renting 38.2% of Bitcoin's total hashrate with NiceHash[25] for the duration of one hour alone, amount to approximately 500 000 USD.

The `Pay per ...` attacks proposed in concurrent work [49] operate in a comparable setting as our described attack and also highlight the economic feasibility without going into detail how such attacks can actually be constructed. The main differences to our attack are, that they focus on an in-band setting and only consider a model where all miners are rational.

6.2 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. Blocks of altruistic miners are likely to also include transactions and transaction orderings that are undesirable to the attacker. Therefore, blocks of such miners may have to be excluded by the attacker, i.e., by providing templates which intentionally fork away these blocks. If altruistic miners find a block, the attacker and colluding miners must mine at least two blocks for the attack chain to become the longest chain again – which altruistic miners will then follow. Hence, the security parameter k_{gap} is equal to 1 in this case, as we start our attack immediately after one undesired block has been mined. Therefore, *no* deep forks of some length k_V (defined by the victim) are required in this scenario.

We derive the probability of the attack chain to win a race against altruistic miners, based on the budget of the attacker. The attack chain must find two blocks more than the altruistic main chain – but must achieve this within the upper bound of N blocks (maximum funded attack duration). Assuming the attacker can only fund up to N blocks on the attack chain, the probability of a successful attack is hence given by Formular 5.

Figure 4 visualizes the probability of catching up one block for different hash rates of $p_{\mathcal{R}}$. It can be observed that N quickly approaches the maximum achievable probability of catching up one block within an unlimited number of

[25] cf. `https://www.crypto51.app/`

blocks [26]. For example if $p_{\mathcal{R}} = 0.66$, then there is a 85% probability to catch up one block after six total blocks ($N = 6$) and a 96% probability after twelve total blocks ($N = 12$). This means, the attacker can decide whether or not to extend the attack period and increase N to win an ongoing race with a higher probability.

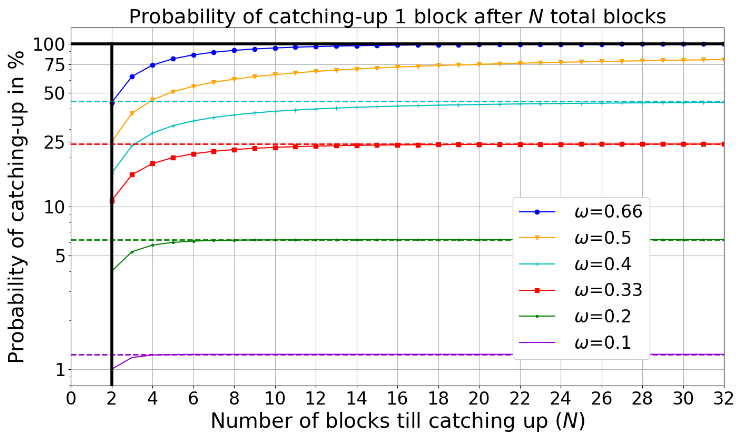

Fig. 4. The probability of catching up one block on the y-axis (log scale) within N blocks on the x-axis for different hashrates $p_{\mathcal{R}}$. The dashed line is the maximum probability to catch-up one block after an unlimited number ($N = \infty$) of blocks i.e., $(\frac{p_{\mathcal{R}}}{p_{\mathcal{A}}})^2$.

Costs of a Successful and Failed Attack The success probability of the attack has an influence on the choice of N and thus on the required budget f_B, but the calculations for the respective bounds in terms of costs are the same as in the previous model with only rational miners (Section 6.1).

7 Discussion and Mitigations

Our AIM attacks serve to highlight the security dependency between transaction value and confirmation time k_V, as also stated in [43]. As with the negative-fee mining pools presented by Bonneau in [14], there exists an interesting analogy between such an incentive manipulation attack and a mining pool. At an abstract level, the presented attacks rely 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

[26] The Probability to catch up one block within an unlimited number of blocks is $(\frac{p_{\mathcal{R}}}{p_{\mathcal{A}}})^2$ according to [39,42]

of an *ephemeral mining relay*, presented within this paper, provides exactly this functionality. Luu et al. [35] 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 the intrinsic incentive among participants is assumed to earn mining rewards in the target cryptocurrency, which is only possible if valid blocks have been created.

The natural questions which arise from the presented attacks are: How likely are such AIM attacks to occur in practice? Can they be efficiently mitigated? We now discuss these questions and provide some directions to explore possible counter measures and limitations of the described attacks.

7.1 Practical possibility

The focus of this paper is to improve upon existing attacks and demonstrate the technical feasibility of advanced bribing attack, 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., 51% attacks on Ethereum Classic [9,7] Bitcoin Gold [8] Bitcoin Cash [6] , as well as incentive manipulation attacks, e.g., Fomo3D [5] and front-running [20]. 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, attacks 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.

7.2 Counter attacks

Counter bribing refers to the technique of countering bribing attacks with other bribing attacks [14,15]. 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. Against transaction exclusion attacks, counter bribing can be performed by increasing the fee of tx_V such that it surpasses the value promised for not including the transaction[27]. If defenders

[27] Another possible counter attack would be to launch a DoS attack against the censor, see Appendix B for details

have imperfect information, they may not be able to immediately respond with counter bribes. In this case, some of the attack chain blocks may have already been mined, or even take the lead, before they are recognized by defenders. Counter bribing then necessitates the incentivization of a fork, and thus a more expensive transaction revision attack, leading to asymmetric costs in the bidding game. 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.

Although our proposed attacks are clearly visible on the funding cryptocurrency, they are not necessarily observable in the target cryptocurrency. In the best case, the proposed transaction exclusion and ordering attack does not even produce a fork in the target cryptocurrency when all miners act rationally. Even if forks are induced, participating miners can make use of the fact that the PoW mining process does not require any strong identity by using different payout addresses. 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., [37,28].

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. 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 [14] 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.

Another interesting aspect of counter bribing is revealed if crowdfunded attacks are assumed. In this case, the funds required to counter bribe can be higher than the invested funds of each individual attacker. In a scenario with multiple victims, organizing coordinated counter bribing is difficult. All victims would be better off if the attack fails, but for an individual victim it is cheaper to not take action and hope that others will fund the counter bribe, leading to a *collective action problem.*

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.

8 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 and are economically feasible in practice. The implications of our proposed method (and related AIM 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 [25,40,13,27], or explicitly exclude bribing [12]. 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 attacks, 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 [17]). 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 [20], demonstrate that the existence of incentives cannot be ignored in PoW cryptocurrencies, i.e., by only considering honest and malicious (Byzantine) miners. 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 methods, as well as the attacks 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.

9 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.

References

1. Btc relay. https://github.com/ethereum/btcrelay, accessed 2018-04-17
2. Coinmarketcap: Cryptocurrency market capitalizations. https://coinmarketcap.com/, accessed 2019-05-10
3. Replace by fee. https://en.bitcoin.it/wiki/Replace_by_fee, accessed 2019-05-11
4. Tornado cash, https://tornado.cash/, accessed: 2020-10-15
5. How the winner got fomo3d prize - a detailed explanation. medium (2018), https://medium.com/coinmonks/how-the-winner-got-fomo3d-prize-a-detailed-explanation-b30a69b7813f, accessed: 2020-09-15
6. Bitcoin cash miners undo attacker's transactions with 51% attack. coindesk (2019), https://www.coindesk.com/bitcoin-cash-miners-undo-attackers-transactions-with-51-attack, accessed: 2020-09-15
7. Ethereum classic 51% attack the reality of proof-of-work. cointelegraph (2019), https://cointelegraph.com/news/ethereum-classic-51-attack-the-reality-of-proof-of-work, accessed: 2020-09-15
8. Bitcoin gold (btg) was 51% attacked. github (2020), https://gist.github.com/metalicjames/71321570a105940529e709651d0a9765, accessed: 2020-09-15
9. Ethereum classic suffers second 51% attack in a week. coindesk (2020), https://www.coindesk.com/ethereum-classic-suffers-second-51-attack-in-a-week, accessed: 2020-09-15
10. Aiyer, A.S., Alvisi, L., Clement, A., Dahlin, M., Martin, J.P., Porth, C.: Bar fault tolerance for cooperative services. In: ACM SIGOPS operating systems review. vol. 39, pp. 45–58. ACM (2005), http://www.dcc.fc.up.pt/~Ines/aulas/1314/SDM/papers/BAR%20Fault%20Tolerance%20for%20Cooperative%20Services%20-%20UIUC.pdf
11. Back, A., Corallo, M., Dashjr, L., Friedenbach, M., Maxwell, G., Miller, A., Poelstra, A., Timón, J., Wuille, P.: Enabling blockchain innovations with pegged sidechains (2014), http://newspaper23.com/ripped/2014/11/http-_____-___-_www___-blockstream___-com__-_sidechains.pdf, accessed: 2016-07-05
12. Badertscher, C., Garay, J.A., Maurer, U., Tschudi, D., Zikas, V.: But why does it work? A rational protocol design treatment of bitcoin. In: Nielsen, J.B., Rijmen, V. (eds.) Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II. Lecture Notes in Computer Science, vol. 10821, pp. 34–65. Springer (2018). https://doi.org/10.1007/978-3-319-78375-8_2, https://eprint.iacr.org/2018/138.pdf

13. Badertscher, C., Maurer, U., Tschudi, D., Zikas, V.: Bitcoin as a transaction ledger: A composable treatment. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10401, pp. 324–356. Springer (2017). https://doi.org/10.1007/978-3-319-63688-7_11, `https://eprint.iacr.org/2017/149.pdf`
14. Bonneau, J.: 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), `http://fc16.ifca.ai/bitcoin/papers/Bon16b.pdf`
15. Bonneau, J.: Hostile blockchain takeovers (short paper). In: 5th Workshop on Bitcoin and Blockchain Research, Financial Cryptography and Data Security 18 (FC). Springer (2018), `http://fc18.ifca.ai/bitcoin/papers/bitcoin18-final17.pdf`
16. Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J.A., Felten, E.W.: Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In: IEEE Symposium on Security and Privacy (2015), `http://www.ieee-security.org/TC/SP2015/papers-archived/6949a104.pdf`
17. Budish, E.: The economic limits of bitcoin and the blockchain. Tech. rep., National Bureau of Economic Research (2018), `https://faculty.chicagobooth.edu/eric.budish/research/Economic-Limits-Bitcoin-Blockchain.pdf`
18. Buterin, V.: Chain interoperability (2016), `https://static1.squarespace.com/static/55f73743e4b051cfcc0b02cf/t/5886800ecd0f68de303349b1/1485209617040/Chain+Interoperability.pdf`, accessed: 2017-03-25
19. Cunicula: Bribery: The double double spend. Bitcoin Forum, `https://bitcointalk.org/index.php?topic=122291`, accessed: 2021-1-31
20. 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: 2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020. pp. 910–927. IEEE (2020). https://doi.org/10.1109/SP40000.2020.00040, `https://arxiv.org/pdf/1904.05234.pdf`
21. Decker, C., Wattenhofer, R.: A fast and scalable payment network with bitcoin duplex micropayment channels. In: Symposium on Self-Stabilizing Systems. pp. 3–18. Springer (2015)
22. Dembo, A., Kannan, S., Tas, E.N., Tse, D., Viswanath, P., Wang, X., Zeitouni, O.: Everything is a race and nakamoto always wins (2020)
23. Eskandari, S., Moosavi, S., Clark, J.: Sok: Transparent dishonesty: Front-running attacks on blockchain. In: Bracciali, A., Clark, J., Pintore, F., Rønne, P.B., Sala, M. (eds.) Financial Cryptography and Data Security - FC 2019 International Workshops, VOTING and WTSC, St. Kitts, St. Kitts and Nevis, February 18-22, 2019, Revised Selected Papers. Lecture Notes in Computer Science, vol. 11599, pp. 170–189. Springer (2019). https://doi.org/10.1007/978-3-030-43725-1_13, `https://arxiv.org/pdf/1902.05164.pdf`
24. Eyal, I., Sirer, E.G.: Majority is not enough: Bitcoin mining is vulnerable. In: Financial Cryptography and Data Security. pp. 436–454. Springer (2014), `http://arxiv.org/pdf/1311.0243`
25. Garay, J.A., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol: Analysis and applications. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30,

2015, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9057, pp. 281–310. Springer (2015). https://doi.org/10.1007/978-3-662-46803-6_10, https://eprint.iacr.org/2014/765.pdf
26. Garay, J.A., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol with chains of variable difficulty (2016), http://eprint.iacr.org/2016/1048.pdf, accessed: 2017-02-06
27. Gai, P., Kiayias, A., Russell, A.: Tight consistency bounds for bitcoin. Cryptology ePrint Archive, Report 2020/661 (2020), https://eprint.iacr.org/2020/661
28. Heilman, E., Baldimtsi, F., Goldberg, S.: Blindly signed contracts: Anonymous on-blockchain and off-blockchain bitcoin transactions. Cryptology ePrint Archive, Report 2016/056 (2016), https://eprint.iacr.org/2016/056.pdf, accessed: 2017-10-03
29. 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: Proceedings of the International Workshop on Cryptocurrencies and Blockchain Technology (Sep 2018), https://www.sba-research.org/wp-content/uploads/2018/09/judmayer2018pitchfork_2018-09-05.pdf
30. Judmayer, A., Stifter, N., Zamyatin, A., Tsabary, I., Eyal, I., Gai, P., Meiklejohn, S., Weippl, E.: Sok: Algorithmic incentive manipulation attacks on permissionless pow cryptocurrencies. Cryptology ePrint Archive, Report 2020/1614 (2020), https://eprint.iacr.org/2020/1614
31. Kolluri, A., Nikolic, I., Sergey, I., Hobor, A., Saxena, P.: Exploiting the laws of order in smart contracts. arXiv:1810.11605 (2018), https://arxiv.org/pdf/1810.11605.pdf
32. 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. vol. 2013, p. 11 (2013), https://pdfs.semanticscholar.org/c55a/6c95b869938b817ed3fe3ea482bc65a7206b.pdf
33. Li, H.C., Clement, A., Wong, E.L., Napper, J., Roy, I., Alvisi, L., Dahlin, M.: Bar gossip. In: Proceedings of the 7th symposium on Operating systems design and implementation. pp. 191–204. USENIX Association (2006), http://www.cs.utexas.edu/users/dahlin/papers/bar-gossip-apr-2006.pdf
34. Liao, K., Katz, J.: Incentivizing blockchain forks via whale transactions. In: International Conference on Financial Cryptography and Data Security. pp. 264–279. Springer (2017), http://www.cs.umd.edu/~jkatz/papers/whale-txs.pdf
35. Luu, L., Velner, Y., Teutsch, J., Saxena, P.: Smartpool: Practical decentralized pooled mining. In: Kirda, E., Ristenpart, T. (eds.) 26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017. pp. 1409–1426. USENIX Association (2017), http://eprint.iacr.org/2017/019.pdf
36. McCorry, P., Hicks, A., Meiklejohn, S.: Smart contracts for bribing miners. In: 5th Workshop on Bitcoin and Blockchain Research, Financial Cryptography and Data Security 18 (FC). Springer (2018), http://fc18.ifca.ai/bitcoin/papers/bitcoin18-final14.pdf
37. Meiklejohn, S., Mercer, R.: Möbius: Trustless tumbling for transaction privacy. Proc. Priv. Enhancing Technol. **2018**(2), 105–121 (2018). https://doi.org/10.1515/popets-2018-0015, http://eprint.iacr.org/2017/881.pdf
38. Miller, A., Bentov, I., Kumaresan, R., McCorry, P.: Sprites: Payment channels that go faster than lightning (2017), https://arxiv.org/pdf/1702.05812.pdf, accessed: 2017-03-22

39. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (Dec 2008), `https://bitcoin.org/bitcoin.pdf`, accessed: 2015-07-01

40. Pass, R., Seeman, L., Shelat, A.: Analysis of the blockchain protocol in asynchronous networks. In: Coron, J., Nielsen, J.B. (eds.) Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II. Lecture Notes in Computer Science, vol. 10211, pp. 643–673 (2017). https://doi.org/10.1007/978-3-319-56614-6_22, `https://doi.org/10.1007/978-3-319-56614-6_22`

41. Poon, J., Dryja, T.: The bitcoin lightning network (2016), `https://lightning.network/lightning-network-paper.pdf`, accessed: 2016-07-07

42. Rosenfeld, M.: Analysis of hashrate-based double spending (2014), `https://arxiv.org/pdf/1402.2009.pdf`, accessed: 2016-03-09

43. Sompolinsky, Y., Zohar, A.: Bitcoin's security model revisited (2016), `http://arxiv.org/pdf/1605.09193.pdf`, accessed: 2016-07-04

44. Stifter, N., Judmayer, A., Schindler, P., Zamyatin, A., Weippl, E.: Agreement with satoshi - on the formalization of nakamoto consensus. Cryptology ePrint Archive, Report 2018/400 (2018), `https://eprint.iacr.org/2018/400.pdf`

45. Teutsch, J., Jain, S., Saxena, P.: When cryptocurrencies mine their own business. In: Financial Cryptography and Data Security (FC 2016) (Feb 2016), `https://www.comp.nus.edu.sg/~prateeks/papers/38Attack.pdf`

46. Tsabary, I., Eyal, I.: The gap game. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. pp. 713–728. ACM (2018), `https://arxiv.org/pdf/1805.05288.pdf`

47. Velner, Y., Teutsch, J., Luu, L.: Smart contracts make bitcoin mining pools vulnerable. In: Brenner, M., Rohloff, K., Bonneau, J., Miller, A., Ryan, P.Y.A., Teague, V., Bracciali, A., Sala, M., Pintore, F., Jakobsson, M. (eds.) Financial Cryptography and Data Security - FC 2017 International Workshops, WAHC, BITCOIN, VOTING, WTSC, and TA, Sliema, Malta, April 7, 2017, Revised Selected Papers. Lecture Notes in Computer Science, vol. 10323, pp. 298–316. Springer (2017). https://doi.org/10.1007/978-3-319-70278-0_19, `https://fc18.ifca.ai/bitcoin/papers/bitcoin18-final14.pdf`

48. Vukolić, M.: The quest for scalable blockchain fabric: Proof-of-work vs. bft replication. In: International Workshop on Open Problems in Network Security. pp. 112–125. Springer (2015), `http://vukolic.com/iNetSec_2015.pdf`

49. Winzer, F., Herd, B., Faust, S.: Temporary censorship attacks in the presence of rational miners. In: 2019 IEEE European Symposium on Security and Privacy Workshops, EuroS&P Workshops 2019, Stockholm, Sweden, June 17-19, 2019. pp. 357–366. IEEE (2019). https://doi.org/10.1109/EuroSPW.2019.00046, `https://eprint.iacr.org/2019/748`

50. Zamyatin, A., Harz, D., Lind, J., Panayiotou, P., Gervais, A., Knottenbelt, W.J.: Xclaim: Trustless, interoperable cryptocurrency-backed assets. Cryptology ePrint Archive, Report 2018/643 (2018), `https://eprint.iacr.org/2018/643.pdf`, `https://eprint.iacr.org/2018/643`

A Variables and Symbols

Symbol	Description
B	The attacker that wants to execute the double-spending attack
V	The victim or merchant, e.g., the actor who would lose money if the double-spending attack is successful
$B_1, B_2, ..., B_x$	Other accounts/addresses under the control of the attacker(s)
$V_1, V_2, ..., V_x$	Other accounts/addresses under the control of the victim(s)
$tx_V, tx_B, tx'_B,$	Transactions: i.e., transaction of the victim, transaction of the attacker, conflicting transaction of the attacker.
$fee(tx_V)$	Function that returns the fee of given transaction e.g., tx_V
f_B	Required initial funds of the attacker
r_e, r_b	Funds equivalent to one block reward in Ethereum and Bitcoin respectively (including fees)
ϵ	Additional reward paid for a block on the attack chain. The total reward for a block on the attack chain received by a bribed miner hence is $r_b + \epsilon$
ρ	Profit of the attacker
$v, v_d, ...$	Value, e.g., value of the double-spend transaction
$c_{success}$	Total costs of a successful pay-2-win attack
c_{fail}	Total costs of a failed pay-2-win attack
$c_{expected}$	Total costs of a successful pay-2-win attack finished with the expected number of blocks
$c_{operational}$	Total operational costs for smart contract deployment and gas
$c_{counter}$	Total operational costs to launch a counter bribing attack e.g., transaction fees, gas, etc.

Table 3: Variables and symbols related to actors and costs.

Symbol	Description
$p_{\mathcal{B}}$	Hash rate of the attacker
$p_{\mathcal{A}}$	Hash rate of all *honest* miners that are not bribable
$p_{\mathcal{R}}$	Hash rate of all rational i.e., bribable miners $p_{\mathcal{R}} = 1 - (p_{\mathcal{B}} + p_{\mathcal{A}})$ and each mining entity i controls $p_{\mathcal{R}_i}$ such that $p_{\mathcal{R}} = \sum_{i=1}^{k} p_{\mathcal{R}_i}$
p_M	Hashrate of some rational mining entity, which evaluates the profitability of accepting bribes.
$p_{\mathcal{E}}$	Estimated hashrate of rational mining entities which will accept bribes and follow the attackers strategy.

Table 4: Variables and symbols related to hashrate.

Symbol	Description
k_V, k_B, k_{gap}	Number of confirmation blocks till block is considered as confirmed by the actor which depends on the respective scenario. This could either be the victim, attacker or given by the desired interference.
ℓ	The length of the attacker chain since the block causing the fork.
N	Maximum length of the attack chain during the attack.
$N_{expected}$	The expected length of the attack chain for a successful attack, it holds that $N < N_{expected}$.
e_x	Some funding chain block at (relative) height x. In our examples the funding chain is considered to be Ethereum. The notation $e_x > e_y$ specifies that e_x has been mined after block e_y i.e., e_x has a higher blockheight.
b_x	Some target chain block at (relative) height x. In our examples the target chain is considered to be Bitcoin. The notation $b_x > b_y$ specifies that b_x has been mined after block b_y i.e., b_x has a higher blockheight.

Table 5: Variables and symbols related to blockchain mechanics.

B DoS Against Transaction Censorship

We consider Bitcoin as a target, however in principle our transaction censorship attack is also applicable to other types of cryptocurrencies. Although, we argue that (quasi) Turing complete smart contract capable cryptocurrencies are more resistant to censorship than Bitcoin:

Let's assume, for the remainder of this discussion, that transaction censorship should take place within Ethereum as a target cryptocurrency. Moreover, the respective transactions, or their side effects, can be accurately identified and all miners agree that these transactions exhibit unwanted behaviour. This opens up the possibility of denial-of-service attacks launched by the victim(s) in such a case. The reason for this stems from the fact that the effects of an unwanted transaction can be proxied through multiple layers of smart contract invocations and interactions. Hereby, the problem arises that miners may only learn of the unwanted behavior of a transaction by first evaluating its state changes. If the resulting behavior is to be censored, miners have to roll back all changes and cannot collect transaction fees for their efforts. Therefore, the attacker can waste the resources of every censoring miner without a loss of funds.

It is impossible to directly overcome this issue without changing the consensus rules, however by basing the attack on block templates, the problem is shifted away from the collaborating rational miners toward the attacker. Hereby, the attacker may choose to only include simple transactions for which he is certain that they cannot hide any unwanted activity e.g., all value transfer transactions, or calls to known contracts such as ERC20 Tokens.

C Ideas for Cost Optimizations

The biggest cost in the proposed out-of-band transaction revision attack derives from the compensation of k_V main chain blocks to provide an incentive for all rational miners (which already have contributed blocks to the main chain between block b_1 and b_{k_V}) to switch to the attack chain. In a blockchain where every block is uniquely attributable to a set of known miners, and where the overall hashrate of those miners can be adequately approximated, the payout of compensations can be further optimized in various ways. As an example, consider the scenario where a small miner, compared to the other miners, is lucky and mines several blocks within k_V. Then it may be cheaper for the attacker to exclude this miner from being eligible for compensation since it is unlikely that he will substantially contribute to the attack chain.

D Available Crowdfunding Funds

With the possibility to crowdfund attacks, theoretically multiple double-spends of low value transactions by different parties could also be made feasible if they together accumulate enough attack funds (f_B). The discrepancy between the value transferred in one Bitcoin block and the rewards (including fees) distributed for mining one Bitcoin block, show that the funds for long range double-spending attacks using this technique are theoretically available. Over the last year (2019) the median value of bitcoins transacted per day (excluding change addresses) is approximately 780 million USD, whereas the median mining reward per day including transactions fees is approximately 11 million USD[28] .

E Basic Requirements for the Attack

On a high level the technical requirements which would allow to trustlessly execute our AIM can be generalized by the five main points listed below.

1. Given a block in a block interval (on the target chain) defined by the attacker, a trustless way to verify either that
   (a) a certain state transition was performed (e.g., a transaction was included in the blockchain).
   (b) a certain state transition was not performed (e.g., a transaction was not included).
2. A trustless way to uniquely attribute blocks on the target cryptocurrency to miner addresses, as well as a way to map the latter to corresponding addresses in the funding cryptocurrency.
3. A trustless way to transfer value in the funding cryptocurrency to a uniquely attributed address of a collaborating miner (see point 1).

[28] Numbers retrieved from `https://www.blockchain.com/charts`

4. A trustless way to determine the state of the target cryptocurrency (main chain) after T blocks have been mined on top of a block pre-defined by the attacker, i.e., the longest chain. This implies that it is possible to verify the PoW of the target cryptocurrency on the funding cryptocurrency (e.g., in a smart contract).
5. A trustless way to determine the state of the attack on the target cryptocurrency after T blocks have been mined on top of a block pre-defined by the attacker, i.e., the attack chain anchored at this specific block.

F Transaction ordering attack (in-band)

This *no-fork* attack pays additional rewards to miners for reordering *unconfirmed transactions*, comparable to front-running attacks [23,20]. In front-running attacks, the adversary increases the chance of his transaction being included before others by increasing the transaction fee paid to miners. However, the result is an *all pay auction*: even if the attack fails, the high-fee transaction can be included by miners. As such, the adversary must *always* pay the fee, independent of the attack outcome [20]. In contrast, our attack ensures the adversary pays colluding miners only if the attack was successful, i.e. if the desired transaction ordering was achieved.

F.1 Description

Initialization. The adversary (Blofeld) observes the P2P network and initiates the attack once he sees a victim's (Vincent) transaction tx_V which he wants to front-run (e.g. registering a domain name or interacting with an exchange). First, Blofeld publishes his front-running transaction tx_B. Simultaneously, he publishes and initializes an *attack contract* with the identifiers of the two transactions, the desired order ($tx_B < tx_V$), the block in which the transaction(s) are to be included, and a bribe ϵ. Once the contract creation transaction has been included into a block, (i) the configuration can no longer be changed and (ii) the bribe is locked until the attack times out. This is necessary to prevent the attacker from attempting to defraud colluding miners by altering the payout conditions, after the attack was executed.

Attack. If the attack is successful, colluding miners generate a block which has the desired ordering of transactions. Note: even if the victim attempts to update the original transaction tx_V with tx'_V , e.g. using *replace by fee* [3], tx_V remains valid and can alternatively be included by miners to invalidate tx'_V. Rational miners will hence include tx_B and tx_V in the specified order, fulfilling the payout conditions, as long as this results in the highest reward.

Payout. After k_B blocks (k_B is the blockchain's security parameter defined by the attacker in this case), miners can claim their payouts, whereby the smart contract first checks if the ordering of the two transactions is as specified.

F.2 Evaluation with Rational Miners Only ($p_{\mathcal{R}} = 1$)

First, we assume a scenario where all miners act rationally, i.e., are bribable. Miners are incentivized to collude with the adversary, as the contract guarantees a reward $\epsilon > 0$ in addition to normal mining. Participation in the attack does not require to mine on an alternative fork, hence colluding miners face no additional risk that their blocks will be excluded from the main chain. It is also possible for miners to include an unconfirmed attack contract creation transaction in the same block as the ordering attack itself and still be certain of payment if their block becomes part of the longest chain.

F.3 Evaluation with Altruistic Miners ($p_{\mathcal{R}} + p_{\mathcal{A}} = 1$)

In theory, this attack is practicable with any hash rate of bribable miners $p_{\mathcal{R}} > 0$, however the higher the hash rate, the higher the chances of success. If 2/3 of the hash rate is controlled by rational miners, the attack is expected to succeed in two out of three cases. We refer to the Section G in the Appendix for an analysis where rational miners are additionally incentivized to near-fork main chain blocks to successfully remove a undesired block from the chain.

F.4 Counter Mechanisms

For a list of general counter mechanisms, e.g., require blinded commitments upfront, which can be used to avoid such vulnerability during the design of smart contracts see [20,23]. In this section we focus on counter bribing as a mitigation strategy. Therefore, we distinguish the counter bribing based on the point in time where the counter attack is performed.

Immediate Counter Bribing As long as the new block has not been mined, an effective counter measure against this attack is to immediately perform counter bribing through the same attack mechanism. Hereby, attacker and victim engage in an *English auction*, as only the winner pays the bribe, instead of the *all-pay-auction* observed in other front-running [20]. This defensive strategy assumes that Vincent is actively monitoring the P2P network and immediately becomes aware of the attack.

Delayed Counter Bribing If Vincent only has an SPV (Simple Payment Verification [39]) wallet, he may only recognize the attack after a new block with the intended ordering of the attacker has already been mined. Since, Vincent is not in possession of any hash rate, he cannot directly launch a counter attack to fork the respective block. Thus, the costs for a successful counter bribing attack have become much higher than the costs for the original attacker Blofeld. For an analysis on how much it costs to remove one block from the chain see Appendix G.

F.5 Details and implementation of tx ordering in-band

There are two methods which allow to implement verification of transaction ordering in Ethereum. The first method only relies on proofs over the transaction trie of a given block to verify the desired transaction ordering. The second method tries to verify the desired state.

Verify transaction ordering This methods works via a transaction trie inclusion proof provided to the attack smart contract. Since the key in the trie is the index of the transaction in the block and the value is the transaction hash, the ordering of any two or more transactions can be proven to a smart contract in retrospect.

The advantage of this approach is that it is conceptionally simple, but it bears certain drawbacks. Lets assume the transaction hash of the involved target transaction tx_V changes e.g., if a transaction was updated via *replace by fee*, or a completely different but conflicting transaction form the same address with the same nonce has been issued tx'_V. This case can still be captured by an attack contract which also checks the nonce of the respective transaction. Since the original transaction tx_V is still valid and can be included by a complacent rational miner, all transactions with the same nonce from the same account become invalid.

A problem arises if the victim publishes another transaction tx''_V from a different account which has not been included in the initialization of the attack contract. This transaction might be semantically equivalent to tx_V, e.g., it would register the same name in sENS, but would not be covered in the attack condition of the contract. Thus, a naive contract only working with transaction hashes and nonces of known transaction can be fooled by a victim to pay out bribes although the attack was not successful because tx''_V has been included before tx_B and just tx_V has been included after tx_B.

Verify operation on certain state This approach addresses the issue of interfering transactions mentioned in the previous section in two different ways.

Retrospective check It is proven to the attack contract in retrospect hat it has successfully operated on the correct world-/smart contract state before any funds are unlocked.

Up to Ethereum EIP-150 revision the *transaction receipt* also contained the *post-transaction*[29] state R_σ. This would have allowed to prove to the attack contract the state before any transaction as well as the state after a specific transaction. Unfortunately the post-transaction state was removed from the transaction receipt for performance reasons.

A currently working generic method for Ethereum around this would be to require that the racing attack transaction has to be at index 0 in the new block

[29] The according Ethereum yellowpaper describing this is still available at `http://gavwood.com/Paper.pdf` (accessed: 2019-05-04)

mined by the miner. It would then be possible to prove to the attack contract in retrospect that the specified transaction at index 0 operated on a specific world state i.e., the word state of the previous block, e.g., where the name to register was not registered yet. The only way to also generically prove that the resulting state was indeed the required one without any side effects is that only transactions which are directly relevant to the attack are included in the new block in the respective order, because then the resulting world state can be precomputed. This of course renders the attack more expensive and less generic.

Runtime check During runtime a smart contract in Ethereum does not know at which position the transaction which invoked the contract is location in the current block. Moreover, it is not possible to query the indices of other transactions during runtime. An alternative to working with indices of transactions is working directly with the required states. The attack contract checks if it is operating on the correct world state directly before even performing the attack e.g., check if the name it wants to register is available. If the attack contract would encounter an error while performing an attack it could prevent any future payouts of bribes.

In our front running example, the front running transaction can also be sent to the attack contract directly, which additionally works as a proxy or dispatcher and only forwards i.e., performs the transaction, iff a queriable attack condition is met i.e., the target contract is in a specific pre-defined state. Since the state (storage) of a contract cannot directly be accessed from another contract, only accessible functions, variables and certain state variables like `balance` can be accessed. Note that for publicly accessible variables getter functions are created automatically. These, runtime checks ensure that no payments happen if the race is not won i.e,. the attack is not successful. Summarizing, it can be said if such checks are possible, the attack becomes more efficient and more complex attack scenarios can be envisioned.

G Transaction exclusion (in-band)

To highlight why executing incentive attacks out-of-band may be desirable for an adversary, we describe an in-band transaction exclusion attack. Thereby, we outline challenges an attacker must overcome and describe how existing attacks are evaluated in the classical setting for bribing attacks.

The purpose of this near- or no-fork attack is it to exclude one or multiple unconfirmed transactions from their generated blocks.

G.1 Description

Initialization. The attacker knows some transaction tx_V which he wants to prevent from getting into the main chain. He then intializes the attack contract at block $e-1$, specifying the transaction and the duration N (in blocks) of the exclusion attack.

Attack. The attack contract will pay an extra ϵ for every block mined between block e_1 and e_N that (i) does not include transaction tx_V itself and (ii) does not extend any block that included transaction tx_V. That is, if an altruistic miner decides to include tx_V in his block e_i ($i < N$), colluding miners must perform a near-fork, i.e., extend block e_{i-1} rather than e_i, if they wish to receive rewards.

Payout. Collaborating miners can claim payouts once k_B blocks have passed after the end of the attack, i.e., at a block $e_T \geq e_{N+k_B}$, where k_B is the security parameter defined by the attacker. Most PoW blockchains use accumulators, such as Merkle trees, to store and efficiently prove inclusion of transactions in a block. However, proving non-existence of an element in a such accumulator is often inefficient. To this end, the attack contract will reward any submitted block between e_1 and e_N, *unless the adversary submits an inclusion proof* for tx_V, before the payouts are claimed in block e_T. If the adversary proves that a block e_x included tx_V, any blocks extending e_x, i.e., $e_{x+1}, e_{x+2}, ...$, will not receive any payouts. Figure 5 shows a failed attack where tx_V was included in block e_3 - thus only blocks up to, but not including, e_3 are rewarded.

More information on the technicalities of this attack when implemented in Ethereum are presented in Section G.5.

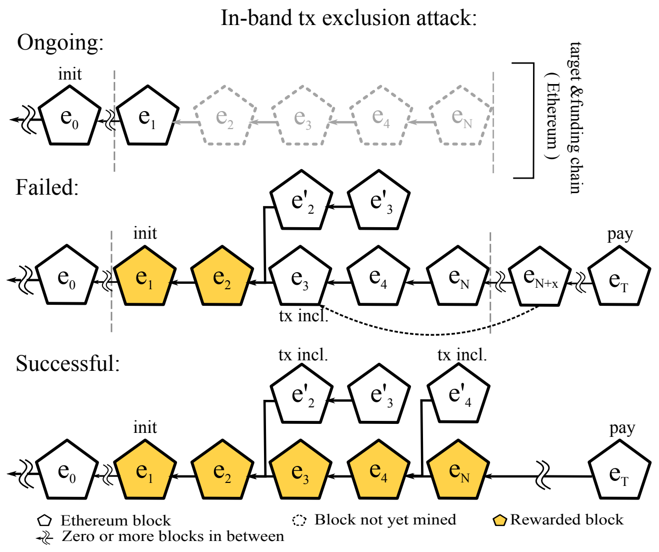

Fig. 5. The figure shows a ongoing-, a failed- as well as a successful Transaction exclusion attack with in-band payments. The attack is initialized when the attack contract is published in block e_1. If the unwanted transaction has been included, this can be proven to the attack contract as shown in the failure case in block e_{N+x}. The payouts are performed in block e_T. The colored blocks are rewarded by the attack contract with an additional ϵ.

G.2 Evaluation with Rational Miners Only ($p_{\mathcal{R}} = 1$)

Estimating the costs of such an attack in a scenario where all miners are rational ($p_{\mathcal{B}} = p_{\mathcal{A}} = 0$ and $p_{\mathcal{R}} = 1$) and have perfect information about the attack is trivial. In this case, it is a no-fork attack and the respective transaction would not be included into the block chain as long as the bribe ϵ for non-inclusion surpasses the fee miners can gain from including transaction, i.e., $\epsilon > \mathit{fee}(tx_V)$.

G.3 Evaluation with Altruistic Miners ($p_{\mathcal{R}} + p_{\mathcal{A}} = 1$)

If a fraction of miners behaves altruistically, i.e., will not join the attack independent of profit, rational miners need an additional incentive to perform near-forks, excluding blocks containing tx_V.

Probability of success without a fork As rational miners find a block with probability $p_{\mathcal{R}}$, the likely hood of rational miners finding chains of consecutive blocks decreases exponentially in their length ℓ. For example, given $p_{\mathcal{R}} = \frac{2}{3}$ the probability of generating a chain of $\ell = 6$ consecutive blocks is merely 8.3%. But what if the attack of delaying a certain reoccurring transaction or set of such transactions at some point in time within the next N total blocks. Like for example deny all transaction to a smart contract token to manipulate the price. The probability for a miner with hashrate $p_{\mathcal{R}} = \frac{2}{3}$ to mine at least $\ell = 6$ consecutive blocks at least once within the next $N = 100$ total blocks is approximately 97.2%. This can be calculated for different values of N, ℓ and $p_{\mathcal{R}}$ by computing the matrix of the finite Markov chain depicted in 6 with N as exponent as shown in formula 23.

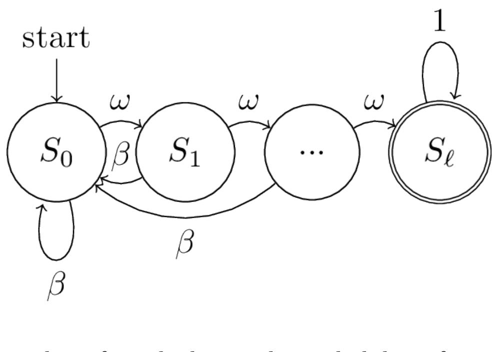

Fig. 6. Finite Markov chain for calculating the probability of mining at least ℓ consecutive blocks with hashrate $p_{\mathcal{R}}$.

$$\mathtt{P} = \begin{bmatrix} p_{\mathcal{A}} & p_{\mathcal{R}} & 0 & \cdots & 0 \\ p_{\mathcal{A}} & 0 & p_{\mathcal{R}} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{\mathcal{A}} & 0 & 0 & \cdots & p_{\mathcal{R}} \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}^N \cdot \begin{bmatrix} 1 \, 0 \, 0 \cdots 0 \end{bmatrix} \tag{23}$$

Probability of success and costs with near-forks To increase the chance of success, the adversary must increase the bribe ϵ paid to colluding miners, to reimburse the risk of loosing block rewards r_e due to a failed fork. Assume a block containing tx_V was mined by altruistic miners. In this scenario, the *attack chain*, i.e., the fork produced by collaborating miners which must not contain tx_V, is only one block behind the main chain. As such, the required bribing funds are significantly lower, when compared to deep fork bribing attacks. To estimate the bribing costs of this attack, we revisit the analysis of Whale Transactions from [34] (specifically, we extend the analysis after Equation 4 in the aforementioned paper).

A rational miner with hashrate p_M will mine on the attack chain if his expected profit is higher than with honest mining. To make a rational decision on which chain to mine, he must estimate and compare the hashrate of (i) all miners expected to join the attack $p_{\mathcal{E}}$, and (ii) the hashrate of all altruistic miners extending the conflicting main chain branch $p_{\mathcal{A}}$. Note that $p_{\mathcal{R}} = p_{\mathcal{E}} + p_M$. For simplicity, we normalize the block reward (incl. transaction fees) to $r_e = 1$. The expected revenue of a rational miner m with hash rate p_M for mining on the main chain is given by the probability that the main chain wins multiplied with his share of mining power on the main chain:

$$\rho = \frac{\left(1 - \left(\frac{p_{\mathcal{B}}+p_{\mathcal{E}}}{p_{\mathcal{A}}+p_M}\right)^{z+1}\right) \cdot p_M}{p_{\mathcal{A}} + p_M} \tag{24}$$

where z is the number of blocks the attacker chain is behind the main chain - in our case $z = 1$. In contrast, the profit from mining on the attack chain is given:

$$\rho' = \frac{\left(\frac{p_{\mathcal{B}}+p_{\mathcal{E}}+p_M}{p_{\mathcal{A}}}\right)^{z+1} \cdot p_M}{p_{\mathcal{B}} + p_{\mathcal{E}} + p_M} \cdot (\epsilon + 1) \tag{25}$$

A rational miner m will only join the attack if $\rho' > \rho$. We hence derive the necessary bribe ϵ as follows:

$$\epsilon > \frac{\left(1 - \left(\frac{p_{\mathcal{B}}+p_{\mathcal{E}}}{p_{\mathcal{A}}+p_M}\right)^{z+1}\right)}{p_{\mathcal{A}} + p_M} \cdot \frac{p_{\mathcal{B}} + p_{\mathcal{E}} + p_M}{\left(\frac{p_{\mathcal{B}}+p_{\mathcal{E}}+p_M}{p_{\mathcal{A}}}\right)^2} - 1 \tag{26}$$

To estimate a worst case lower bound for the necessary bribe, we set $p_{\mathcal{E}} = 0$ and a calculate ϵ for a small rational miner with hashrate $p_M = 0.05$. We receive $\epsilon \approx 17 \cdot r_e$, i.e., if a rational miner m assumes no other miners will join the attack, a bribe 17 times the value of a block reward is necessary. We provide a detailed overview of necessary bribing values ϵ for different attack constellations ($p_{\mathcal{E}}$ and p_M) in Table 6 in Section G.5. We observe that once $p_M + p_{\mathcal{E}}$ exceeds 38.2%, a rational miner m is always incentivized to mine on an attack chain with $z = 1$, independent of the bribe value ϵ (i.e., necessary $\epsilon = 0$).

Table 6 shows the costs for incentivizing in-band transaction exclusion if blocks that include the respective transaction should be forked by rational miners.

	$p_M = 0.05$	$p_M = 0.1$	$p_M = 0.2$	$p_M = 0.3$	$p_M = 0.33$	$p_M = 0.382$	$p_M = 0.4$
$p_{\mathcal{E}} = 0.00$	$p_{\mathcal{A}} = 0.950$ $\rho = 0.050$ $\epsilon = 17.050$ $\rho' = 0.050$ $P = 0.003$	$p_{\mathcal{A}} = 0.900$ $\rho = 0.100$ $\epsilon = 7.100$ $\rho' = 0.100$ $P = 0.012$	$p_{\mathcal{A}} = 0.800$ $\rho = 0.200$ $\epsilon = 2.200$ $\rho' = 0.200$ $P = 0.062$	$p_{\mathcal{A}} = 0.700$ $\rho = 0.300$ $\epsilon = 0.633$ $\rho' = 0.300$ $P = 0.184$	$p_{\mathcal{A}} = 0.670$ $\rho = 0.330$ $\epsilon = 0.360$ $\rho' = 0.330$ $P = 0.243$	$p_{\mathcal{A}} = 0.618$ $\rho = 0.382$ $\epsilon = 0.000$ $\rho' = 0.382$ $P = 0.382$	$p_{\mathcal{A}} = 0.600$ $\rho = 0.400$ $\epsilon = 0.000$ $\rho' = 0.444$ $P = 0.444$
$p_{\mathcal{E}} = 0.05$	$p_{\mathcal{A}} = 0.900$ $\rho = 0.052$ $\epsilon = 7.503$ $\rho' = 0.052$ $P = 0.012$	$p_{\mathcal{A}} = 0.850$ $\rho = 0.105$ $\epsilon = 4.056$ $\rho' = 0.105$ $P = 0.031$	$p_{\mathcal{A}} = 0.750$ $\rho = 0.210$ $\epsilon = 1.362$ $\rho' = 0.210$ $P = 0.111$	$p_{\mathcal{A}} = 0.650$ $\rho = 0.315$ $\epsilon = 0.267$ $\rho' = 0.315$ $P = 0.290$	$p_{\mathcal{A}} = 0.620$ $\rho = 0.346$ $\epsilon = 0.062$ $\rho' = 0.346$ $P = 0.376$	$p_{\mathcal{A}} = 0.568$ $\rho = 0.401$ $\epsilon = 0.000$ $\rho' = 0.512$ $P = 0.578$	$p_{\mathcal{A}} = 0.550$ $\rho = 0.420$ $\epsilon = 0.000$ $\rho' = 0.595$ $P = 0.669$
$p_{\mathcal{E}} = 0.10$	$p_{\mathcal{A}} = 0.850$ $\rho = 0.055$ $\epsilon = 4.286$ $\rho' = 0.055$ $P = 0.031$	$p_{\mathcal{A}} = 0.800$ $\rho = 0.110$ $\epsilon = 2.512$ $\rho' = 0.110$ $P = 0.062$	$p_{\mathcal{A}} = 0.700$ $\rho = 0.219$ $\epsilon = 0.792$ $\rho' = 0.219$ $P = 0.184$	$p_{\mathcal{A}} = 0.600$ $\rho = 0.329$ $\epsilon = 0.000$ $\rho' = 0.333$ $P = 0.444$	$p_{\mathcal{A}} = 0.570$ $\rho = 0.362$ $\epsilon = 0.000$ $\rho' = 0.437$ $P = 0.569$	$p_{\mathcal{A}} = 0.518$ $\rho = 0.419$ $\epsilon = 0.000$ $\rho' = 0.686$ $P = 0.866$	$p_{\mathcal{A}} = 0.500$ $\rho = 0.439$ $\epsilon = 0.000$ $\rho' = 0.800$ $P = 1.000$
$p_{\mathcal{E}} = 0.20$	$p_{\mathcal{A}} = 0.750$ $\rho = 0.059$ $\epsilon = 1.637$ $\rho' = 0.059$ $P = 0.111$	$p_{\mathcal{A}} = 0.700$ $\rho = 0.117$ $\epsilon = 0.914$ $\rho' = 0.117$ $P = 0.184$	$p_{\mathcal{A}} = 0.600$ $\rho = 0.234$ $\epsilon = 0.055$ $\rho' = 0.234$ $P = 0.444$	$p_{\mathcal{A}} = 0.500$ $\rho = 0.352$ $\epsilon = 0.000$ $\rho' = 0.600$ $P = 1.000$	$p_{\mathcal{A}} = 0.470$ $\rho = 0.387$ $\epsilon = 0.000$ $\rho' = 0.623$ $P = 1.000$	$p_{\mathcal{A}} = 0.418$ $\rho = 0.448$ $\epsilon = 0.000$ $\rho' = 0.656$ $P = 1.000$	$p_{\mathcal{A}} = 0.400$ $\rho = 0.469$ $\epsilon = 0.000$ $\rho' = 0.667$ $P = 1.000$
$p_{\mathcal{E}} = 0.30$	$p_{\mathcal{A}} = 0.650$ $\rho = 0.058$ $\epsilon = 0.408$ $\rho' = 0.058$ $P = 0.290$	$p_{\mathcal{A}} = 0.600$ $\rho = 0.117$ $\epsilon = 0.050$ $\rho' = 0.117$ $P = 0.444$	$p_{\mathcal{A}} = 0.500$ $\rho = 0.233$ $\epsilon = 0.000$ $\rho' = 0.400$ $P = 1.000$	$p_{\mathcal{A}} = 0.400$ $\rho = 0.350$ $\epsilon = 0.000$ $\rho' = 0.500$ $P = 1.000$	$p_{\mathcal{A}} = 0.370$ $\rho = 0.385$ $\epsilon = 0.000$ $\rho' = 0.524$ $P = 1.000$	$p_{\mathcal{A}} = 0.318$ $\rho = 0.445$ $\epsilon = 0.000$ $\rho' = 0.560$ $P = 1.000$	$p_{\mathcal{A}} = 0.300$ $\rho = 0.466$ $\epsilon = 0.000$ $\rho' = 0.571$ $P = 1.000$
$p_{\mathcal{E}} = 0.33$	$p_{\mathcal{A}} = 0.620$ $\rho = 0.057$ $\epsilon = 0.144$ $\rho' = 0.057$ $P = 0.376$	$p_{\mathcal{A}} = 0.570$ $\rho = 0.113$ $\epsilon = 0.000$ $\rho' = 0.132$ $P = 0.569$	$p_{\mathcal{A}} = 0.470$ $\rho = 0.226$ $\epsilon = 0.000$ $\rho' = 0.377$ $P = 1.000$	$p_{\mathcal{A}} = 0.370$ $\rho = 0.339$ $\epsilon = 0.000$ $\rho' = 0.476$ $P = 1.000$	$p_{\mathcal{A}} = 0.340$ $\rho = 0.373$ $\epsilon = 0.000$ $\rho' = 0.500$ $P = 1.000$	$p_{\mathcal{A}} = 0.288$ $\rho = 0.432$ $\epsilon = 0.000$ $\rho' = 0.537$ $P = 1.000$	$p_{\mathcal{A}} = 0.270$ $\rho = 0.452$ $\epsilon = 0.000$ $\rho' = 0.548$ $P = 1.000$
$p_{\mathcal{E}} = 0.38$	$p_{\mathcal{A}} = 0.568$ $\rho = 0.050$ $\epsilon = 0.000$ $\rho' = 0.067$ $P = 0.578$	$p_{\mathcal{A}} = 0.518$ $\rho = 0.100$ $\epsilon = 0.000$ $\rho' = 0.180$ $P = 0.866$	$p_{\mathcal{A}} = 0.418$ $\rho = 0.200$ $\epsilon = 0.000$ $\rho' = 0.344$ $P = 1.000$	$p_{\mathcal{A}} = 0.318$ $\rho = 0.300$ $\epsilon = 0.000$ $\rho' = 0.440$ $P = 1.000$	$p_{\mathcal{A}} = 0.288$ $\rho = 0.330$ $\epsilon = 0.000$ $\rho' = 0.463$ $P = 1.000$	$p_{\mathcal{A}} = 0.236$ $\rho = 0.382$ $\epsilon = 0.000$ $\rho' = 0.500$ $P = 1.000$	$p_{\mathcal{A}} = 0.218$ $\rho = 0.400$ $\epsilon = 0.000$ $\rho' = 0.512$ $P = 1.000$
$p_{\mathcal{E}} = 0.40$	$p_{\mathcal{A}} = 0.550$ $\rho = 0.046$ $\epsilon = 0.000$ $\rho' = 0.074$ $P = 0.669$	$p_{\mathcal{A}} = 0.500$ $\rho = 0.093$ $\epsilon = 0.000$ $\rho' = 0.200$ $P = 1.000$	$p_{\mathcal{A}} = 0.400$ $\rho = 0.185$ $\epsilon = 0.000$ $\rho' = 0.333$ $P = 1.000$	$p_{\mathcal{A}} = 0.300$ $\rho = 0.278$ $\epsilon = 0.000$ $\rho' = 0.429$ $P = 1.000$	$p_{\mathcal{A}} = 0.270$ $\rho = 0.306$ $\epsilon = 0.000$ $\rho' = 0.452$ $P = 1.000$	$p_{\mathcal{A}} = 0.218$ $\rho = 0.354$ $\epsilon = 0.000$ $\rho' = 0.488$ $P = 1.000$	$p_{\mathcal{A}} = 0.200$ $\rho = 0.370$ $\epsilon = 0.000$ $\rho' = 0.500$ $P = 1.000$

Table 6: Comparison of minimum bribing attack costs ϵ for certain attack hashrates $p_{\mathcal{E}}$ and undecided individual miners p_M. The table also shows the expected reward of m if p_M would be directed towards the attack chain ρ', as well as the expected reward ρ if p_M would be directed towards the main chain.

Comparison to Existing Attacks A comparable attack allowing arbitrary transaction exclusion is `HistoryRevisionCon` [36]. While `HistoryRevisionCon` only requires bribing amounts ϵ between $0.09375 \cdot r_e$ and $1.4375 \cdot r_e$ (depends on how effective uncle block inclusion can be optimized), it also requires a substantial attacker hashrate ($p_{\mathcal{B}} > \frac{1}{3}$). For comparison: if we assume $p_{\mathcal{R}} = 0.33$ s.t., $p_{\mathcal{E}} = 0.28$ and $p_M = 0.05$, our attack would require $\epsilon \approx 0.603 \cdot r_e$.

The only other comparable transaction exclusion attack is the *Script Puzzle 38.2% attack*, which requires $p_{\mathcal{B}} > 38.2\%$ (in Bitcoin). For comparison, if we assume $p_{\mathcal{R}} = 0.382$, our attacks requires a bribe value ϵ close to zero: mining on the attacker chain becomes the highest paying strategy independent of the bribe.

G.4 Counter Mechanisms

Unique transaction specification: To deny some transaction from getting into the blockchain, the respective transaction has to be known. We made the simplifying assumption that the transaction hash is known to the attacker and wont change. Although, in practice this might not hold true because of several ways around this restriction: Even if transaction malleability is not possible for any third party, transactions can be recreated by the sender s.t. they are semantically equivalent but their transaction hash differs. Ethereum actively supports this as *replace-by-fee*, when a new transaction from the same account with a higher gas value is available it will be preferred by miners. The new transaction is not even required to be semantically equivalent to the original one.

Therefore, the victim can evade the attack if the attack contract relies on transaction hashes. A possible but less generic way around this is to evaluate contract states instead of transaction hashes to determine if the effects of some unwanted transaction have made it into the blockchain. Although, this seams like a promising approach, the feasibility of this solution highly depends on the individual case as outlined in Section F.5.

Counter Bribing The most effective counter measure against the attack is to increase the fee of tx_V s.t. it surpasses the value promised by the attack contract. Since the transaction exclusion incentives have to be made public, the attack cannot be considered stealthy in the target cryptocurrency. This motivates that the incentivization of the attack happens out-of-band on a distinct funding cryptocurrency and thus hidden from clients which only operate and monitor the target cryptocurrency. Such an attack is described in the Section 6

Proof a negative Since we are in an in-band scenario, the successful execution of the attack relies on a proof that transaction tx_V was included to correctly pay out rewards and detect unwanted inclusion. It can be argued that rational miners would be disincentivised to include this proof and collect the rewards for mined blocks anyway. Moreover, the exact same incentive attack can be used to keep this proof transaction out of the blockchain. We now show that this is not an efficient counter attack by introducing and additional cost gap. To introduce this cost gap between the attack and its counter attack, the stabilization period between e_N and e_T can be increased s.t. it is larger than the period between e_1 and e_N. Thereby, the counter attack gets more expensive than the original attack. This leverages the fact that the victim has to get his transaction into the blockchain before e_N, whereas the attacker of can choose a longer stabilization period.

Nevertheless, an approach that poses more convincing evidence of transaction absence is desirable. An in-band method that relies on a proof that the transaction tx_V was indeed *not* included in the chain in the respective interval would be ideal. Thereby, the attacker can be sure that the payment only happens if the requested condition is fulfilled. In practice such proves are less efficient in current cryptocurrencies like Ethereum. A possible way around this is to provide a *block template* for every block, which must be used by the miners to be later able to collect the associated additional reward ϵ. Thereby, it can be ensured by the attacker that only wanted transactions are included as well as their order. The block template can be provided in a transaction to an attack contract which encompasses all transaction hashes in their respective order which should be included in the next block, excluding his own hash.

Another alternative would be to use out-of-band techniques and launch the attack form a different smart contract capable funding cryptocurrency whose miners are not affected by the attack. Moreover, if the set of miners is distinct, the incentives of the miners to not include a inclusion prove of tx_V are less of an issue. We describe an out-of-band attack which uses the technique of *block templates* and also allows for arbitrary ordering in Section 6.

G.5 Details and implementation of tx exclusion in-band

The two important aspects of this attack are: i) Determine if the unwanted transaction tx_V was included, and if so in which block ii) Correctly reward complacent miners.

To collect the reward, a rational miner has to submit the block header he mined in the respective range to the attack contract. The attack contract then checks if this block really lies in the respective interval in the recent history of the chain. In Ethereum, the last 256 block hashes can be accessed from within a smart contract, thereby the smart contract can verify if a submitted block header really is part of the recent history. From the submitted block header the contract can also extract the beneficiary / coinbase address of the respective miner directly.

Transaction inclusion proof The naive way of determining if tx_V has been included in a block is to request a Merkle patricia trie inclusion prove, as described in Section F.5, that the respective transaction is part of a given block header which lies in the defined interval. This approach has the drawback that it will not detect other semantically equivalent transactions with a different hash.

A way around this in an in-band scenario on Ethereum is to define state conditions which must be met depending on the use-case at hand. For example, if you can show me a transaction to a certain address / contract that is part of a block in the specified interval than I consider this as a prove that an unwanted interaction with the respective address / contract has taken place and do not reward the miners from that block on. Thereby, care has to be taken to account for transaction obfuscation via proxy contracts which perform message calls on behalf of a transaction from an externally owned account. These, cannot easily be proven to a contract since the respective transaction has to be evaluated on the EVM with the correct world-state. Thus, this variant is only error free if the unwanted transaction has to come from an externally owned account directly, e.g., as required by certain Tokens[30].

Therefore, the safest variant is do check if the state change or condition which should have been triggered by an unwanted transaction has occurred or not. For example if the balance of a contract has been raised/decreased, or if certain public accessible state variable has changed in an undesired way. If this can be checked by the attack contract before performing any payouts, it is not possible to collect rewards if the requested condition has not been fulfilled.

Block template in-band Another way around the previously outlined problem of proving that an unwanted operation / transaction has not taken place is to specify exactly what transactions are allowed to take place. Interestingly, this is easier in an out-of-band scenario than in an in-band scenario since the attacker has to convincingly ensure the collaborating rational miners that they will receive their bribes while defining the content of all blocks in a way that can be proven to the attack smart contract. At the same time the content of the blocks also has to define those blocks, which leads to a recursive dependency since the transaction to the attack contract cannot define itself because their hash is not known in advance.

[30] Interestingly, a UTXO model would also be easier to censor if the output which has to be spent in an unwanted transaction is known.