Pay-To-Win: Cheap, Crowdfundable, Cross-chain Incentive Manipulation Attacks on Cryptocurrencies

Aljosha Judmayer
SBA Research
ajudmayer@sba-research.org

Nicholas Stifter
TU Wien

Alexei Zamyatin
Imperial College London

Itay Tsabary
Technion and IC3

Ittay Eyal
Technion and IC3

Peter Gaži
IOHK

Sarah Meiklejohn
University College London

Edgar Weippl
SBA Research

Abstract

The feasibility of bribing attacks on cryptocurrencies was first highlighted in 2016, with various new techniques and approaches having since been proposed. Recent reports of real world 51% attacks on smaller cryptocurrencies with rented hashrate underline the realistic threat bribing attacks present, in particular to permissionless cryptocurrencies.

In this paper, bribing attacks and similar techniques, which we refer to as *incentive manipulation attacks* (IMA), are systematically analyzed and categorized. We show that the problem space is not fully explored and present several new and improved attacks. Thereby, we identify *no- and near-fork* attacks as a powerful, yet largely overlooked, category. To be successful, such attacks require forks of short length that are independent from a security parameter *k* defined by the victim, or even no forks at all. The consequences, such as *transaction exclusion and ordering* manipulation, raise serious security concerns for smart contract platforms.

Further, we propose the first *trustless* out-of-band bribing attack capable of facilitating double-spend collusion across different blockchains that reimburses collaborators in case of failure. Our attack is hereby rendered between 85% and 95% cheaper than comparable bribing techniques (e.g., the whale attack). To demonstrate the technical feasibility we published all artefacts of this paper reaching from calculations to a fully functional implementation of our most powerful out-of-band attack consisting of an Ethereum smart contract and a python client[1].

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 [27].

Despite an ever growing body of research in the field of cryptocurrencies, it is still unclear if Bitcoin, and thus Nakamoto consensus, is actually incentive compatible under practical conditions, i.e., that the intended properties of the system emerge from the appropriate utility model for miners [7]. *Bribing attacks*, in particular, target incentive compatibility and assume that at least some 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.

Most bribing attacks proposed so far focus on optimizing a player's (miner's) utility by accepting *in-band* bribes, i.e., payments in the respective cryptocurrency. Thus, a common argument against the practicality of such attacks is that miners won't participate in these attacks as they would put the economic value of their respective cryptocurrency at risk, harming their own income stream. Another common counter argument against classical bribing attacks is that they are considered quite expensive for an adversary (e.g., costs of several hundred bitcoins for one successful attack [22]), or require substantial amounts of attacker hashrate.

Related to classical bribing attacks, which focus on transaction revision, there also exist attacks aimed at transaction *exclusion* and/or *ordering*. Until recently, the latter was a largely overlooked category [19] that, especially in the context of bribing, has not been explicitly considered yet. Only a specific form of a transaction ordering attack was described i.e., *front running*. In its simplest form clients raise the transaction fee in the hope to front-run a competing transaction, which can also be viewed as an unsophisticated bribing attack. Recent research highlights and analyzes front running in the context of the Ethereum platform [10, 12].

Another form of bribing is the *Goldfinger* attack, where the goal of an attacker is to destroy a competing cryptocurrency to gain some undefined external utility [20]. The attack is named after the James Bond movie villain Goldfinger who seeks to destroy the gold reserves stored in Fort Knox to increase the value of his own holdings. The first practical example of such an attack was suggested in [7] and implemented in [24]. Goldfinger attacks inherently require some external utility

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

e.g., that the payments have to be performed out-of-band since, if successful, the value of the targeted cryptocurrency is intended to drop. In comparison, classical bribing attacks directly aim at gaining in-band profit.

The attacks presented in this paper bridge this gap, as they can either be executed driven by in-band profit, or as Goldfinger-style attacks. Thereby, the use of bribes may break the mechanism design and cause rational players to deviate from the prescribed protocol. To systematically expose the body of research on bribing-, front-running- Goldfinger- and other related attacks, we jointly consider them under the general term *incentive manipulation attacks* (IMA), as they all intend to tamper with the incentives of rational actors in the system.

1.1 Contributions

In this paper we show that sophisticated trustless *out-of-band* attacks can readily be constructed, given state-of-the-art smart contract platforms. Such attacks pose an even greater threat to cryptocurrencies, as the argument that miners won't harm their own income stream can no longer be readily applied in this case. Moreover, we show that the cost for an attacker can be significantly reduced by guaranteeing that participating bribees are reimbursed, as well as aligning the interests of multiple attacks (crowdfunding) in a trustless manner, i.e., through smart contract code.

Furthermore, we demonstrate that ordering attacks can also be executed as targeted bribing attacks using smart contracts. Such attacks do not require any attacker hashrate and can work without inducing deep blockchain forks or *near forks*, and may even be successful with *no forks* at all. 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.

We begin our analysis by outlining general assumptions of the attack model most analyzed and newly proposed incentive manipulation attacks have in common (Section 2). We then present a detailed analysis of related work in this area, which allows us to compare and categorize the varying system/attack models of previously proposed attacks (Section 3). This is a prerequisite to identify related attacks and compare them against each other. Thereby, we also show that the problem space is not yet fully explored. By classifying along two different dimensions i.e., intended impact and required interference, we find that incentive manipulation attacks on transaction ordering can be performed as a byproduct of other analysed attacks, but no dedicated IMA has so far been proposed that explicitly targets this area.

We propose two **new incentive manipulation attacks** (Section 5 and 6) to fill some of the gaps outlined by our analysis[2]. Both are trustless for the attacker and the collaborating miners, rely on out-of-band payments using a different cryptocurrency and do not require the adversary to control any hashrate. The first lies in the previously underrepresented category of no-/near-fork attacks and targets transaction ordering and exclusion. The second attack incentivizes deep forks and double-spend collusion.

On the technical level, We introduce three **crucial enhancements to incentive manipulation attacks**: (i) *ephemeral mining relays*, as a mechanism for executing 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 IMA.

Summarizing our contributions are as follows:

- A trustless out-of-band IMA to incentivize transaction exclusion and/or ordering which can be executed without deep blockchain forks
- A trustless out-of-band IMA to incentivize double-spend collusion
- A construction for an ephemeral mining relay to facilitate the proposed out-of-band attacks
- Methods to guarantee payments to participating bribees and an evaluation of the attack cost reduction
- A method to crowdfund out-of-band double-spending attacks

2 Model

For all analyzed and presented incentive manipulation attacks (IMA) we adopt the following general attack model. If an analyzed attack deviates from this model, it is highlighted in detail when the attack is described.

We consider IMA within *permissionless proof-of-work (PoW) cryptocurrencies*. That is, we assume protocols adhering to the design principles of Bitcoin [27], generally referred to as Nakamoto consensus or Bitcoin backbone protocol [16,28,33].

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. As in previous work on bribing attacks [6,22,24,34], we assume the set of miners to be fixed, as well as 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 a certain amount 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

[2] Two other new in-band attacks which we also identified can be found in the full version of the paper.

valid transactions within the network. Moreover, as in prior work [22, 24, 35], we 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)* [4, 21] rational behavior model.

- **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 $\alpha \geq 0$ in the target cryptocurrency. The attacker may deviate arbitrarily from the protocol rules.

- **Altruistic or honest miner(s)** (**A**lice): Honest miners A 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 $\beta \geq 0$ in the target cryptocurrency.

- **Rational or bribable miner(s)** (**R**achel): Miners R controlling hashrate $\omega > 0$ in the target cryptocurrency aiming to maximize their short term profits. 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 [14].

Additionally, we assume the **victim** (**V**incent) of the bribing attacks to be a client without any hashrate. While other bribing attacks implicitly model the victim as honest, i.e., as strictly following the protocol, we want to emphasise that this is not necessarily the case, especially if economically rational counter attacks by the victim should be considered. Even in a setting where the victim has no hashrate, he can use his funds (f_V) for counter attacks. Therefore we distinguish between rational and honest/altruistic victims to allow for a more fine grained discussion of the presented attacks. If Vincent is to be modeled with possession of some hashrate, it can be considered either to be part of β (if altruistic) or ω (if rational). It holds that $\alpha + \beta + \omega = 1$.

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 IMA 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 [16], 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.

If more than one cryptocurrency is involved in the considered scenario, for example when out-of-band payments should be performed in another cryptocurrency, an additional *funding cryptocurrency* is assumed. While the attack is performed on a *target cryptocurrency*, the funding cryptocurrency is used to orchestrate and fund it. In such a case, we also assume that the difficulty and thus the mean block interval of the funding chain is fixed for the duration of the attack. Further, no additional attacks are concurrently being launched against either cryptocurrencies.

2.2 Incentive Manipulation: Impact and Interference

IMA represent a generalized form of bribing attacks [6], comprising adversarial strategies aimed at manipulating the incentives of rational participants. Hereby, we first introduce a general classification along two different dimensions, namely by the *intended impact* an attack has on transactions and their ordering and the *required interference*, i.e., the depth of blockchain reorganizations caused by forks for the attack to be successful. Combined with other important characteristics, we systematically analyze and categorize the body of research on incentive manipulation attacks.

2.2.1 Intended Impact on Transactions

A core goal for permissionless PoW cryptocurrencies is to achieve an (eventually) consistent and totally ordered log of transactions that define the global state of the shared ledger. We differentiate between three states a transaction can be in from the perspective of a participant (miner or client):

- *proposed/ published/ unconfirmed*, the transaction has been broadcasted in the respective P2P network;

- *confirmed*, the transaction has been confirmed by at least one block, i.e., has been included in a block;

- *agreed*, the transaction has been confirmed by at least k blocks, where k is defined by the recipient of the transaction. We denote $k_{participant}$ to refer to the confirmation policy of a participant if it is not clear from the context e.g., k_V denotes the confirmation policy of the victim.

We separate between the following three main categories of IMA aimed at manipulating transactions and their ordering:

- **transaction revision**, change a previously proposed, possibly confirmed or agreed transaction;
- **transaction ordering**, change either the proposed, confirmed or already agreed upon order of transactions;
- **transaction exclusion/censorship**, exclude a specific transaction, or set of transactions, from the log of transactions for a bounded amount of time i.e., the transaction remains unconfirmed.

Some IMA may allow multiple types of transaction manipulation at the same time (see Table 1). Depending on the state of the targeted transaction(s) (proposed, confirmed, agreed) the attack might vary in cost and in the required level of interference with consensus.

2.2.2 Required Interference with Consensus

While the previous classification of transaction manipulation attacks describes the intended impact, here we consider the *required interference* with consensus by which they can be achieved. Specifically, we introduce three different fork requirements:

- **Deep-fork required**, where a fork with depth of at least ℓ exceeding a security parameter k_V is necessary (i.e., $\ell > k_V$). The victim defines k_V [15,32] and it refers to its required number of confirmation blocks for accepting transactions.
- **Near-fork required**, where the required fork depth is *not* dependent on k_V, but forks might be required. In other words, the attacker defines the gap k_{gap} he wants to overcome, which can be smaller than k_V.
- **No-fork required**, where no blockchain reorganization is necessary at all (i.e., $\ell = 0$).

No-fork attacks distinguish themselves from the other two categories by aiming to manipulate miner's *block proposals* rather than (preliminary) consensus *decisions*, i.e., already mined blocks. Deep- and near-fork attacks seek to undo state-updates to the ledger that are already confirmed by subsequent proof-of-work.

Some attacks, such as front-running or transaction revision where the victim accepts $k_V = 0$ (zero confirmation), may be executable as no-fork attacks. Others, such as performing a double spend where the victim has carefully chosen k_V [32], may require deep-forks because they need to substantially affect consensus and violate the security assumption that a common prefix of the blockchain remains stable, except with negligible probability [15] (see Section 6). Transaction censorship may require near-forks to exclude the latest blocks which include the respective transaction (see Section 5).

3 Related Work

Equipped with our attack model and the classification by intended impact and required interference, we consider related work on the topic of incentive manipulation attacks within this section. Table 1 presents an overview of our categorization of previous proposals, as well as our new pay-to-win attacks. Each row represents a different attack (in chronological order) and columns outline respective properties.

Tx revision / Tx ordering / Tx exclusion are outlined in subsection 2.2.1. In the first bribing attack proposed by Bonneau [6] the use of *lock time transactions* is suggested, which are only valid on the attacker's chain but there they can be claimed by anyone (anyone-can-spend outputs). Miners are hence expected to be incentivized to mine blocks on the attacker's chain to collect these bribes. A variation of this attack using high fee transactions (*whale transactions*) to provide incentives for miners to join the attack was described by Liao and Katz [22]. In [24] they proposed a smart contract (`HistoryRevisionCon`) which pays additional in-band rewards to miners of the attacker's desired Ethereum chain branch, iff the effects of the double-spending transaction have occurred on this branch. The mentioned attacks ([6,22,24]) are designed to replace or *revise* a specific transaction, i.e., perform a single double-spend. As a consequence, they do not consider defining the order or exclusion of arbitrary transactions. Except for the double-spending transaction itself, the block content of subsequent blocks can freely be defined by the bribed miners. Therefore, it would be possible for such miners to also perform a double-spend of one of their transactions for free by piggybacking on the attack financed by the original attacker.

`GoldfingerCon` [24] can be seen as a special case of the transaction exclusion attack which rewards Bitcoin miners for mining empty blocks with the help of an Ethereum smart contract. Similarly, `Pitchforks` [18] leverages merged mining to subsidize the creation of empty (or specially crafted) blocks in the attacked chain [18].

The `Script puzzle` 38.2% [34] and `CensorshipCon` attack [24] distract hashrate of bribable miners to gain an advantage over the remaining honest miners from the main chain towards puzzles which promise more rewards than honest mining. The goal of both attacks is that the attacker gains the majority of the hashrate in the respective chain, and he can hence arbitrarily order and exclude transactions. Although, the attack does not explicitly aim to allow the specific ordering of certain transactions, this capability is achieved as a byproduct. Neither attack is reverting blocks to change history, which is a different scenario and requires further analysis in this context, as reverting blocks would change the incentives of miners which have produced them.

There are only two previously proposed attack methods to theoretically achieve all three properties: `Script Puzzle double-spend` [34] and *negative-fee mining pools* [6]. A

	Tx rev.	Tx ord.	Tx excl.	Required chain reorganization	Attacker hashrate α	Rational hashrate ω	Distracts hashrate	Requires smart contract	Payment	Trustless for attacker	Trustless for collaborator	Subsidy	Compensates if attack fails
Checklocktime bribes [6]	✓	✗	✗	Deep fork	✗	$\approx[\frac{1}{2},1]$	✗	✗	in-band	✓	$\sim$	✗	✗
Negative fee miningpool [6]	✓	✓	✓	Near-/No- /Deep forks	✗	$\approx[\frac{1}{2},1]$	✗	✗	out-of-band	✗	✗	✗	✓
Whale Transactions [22]	✓	✗	✗	Deep fork	✗	$\approx[\frac{1}{2},1]$	✗	✗	in-band	✓	$\sim$	✗	✗
Script Puzzle double-spend [34]	✓	$\sim$	✓	Deep fork	$(0,\frac{1}{2})$	$1-\alpha$	✓	✗	in-band	$\sim$	✗	✗	$\sim$
Script Puzzle 38.2% attack [34]	✗	$\sim$	✓	Near-/No forks	$[0.382,\frac{1}{2})$	$1-\alpha$	✓	?†	out-of-band	?†	?†	✗	✓
Proof-of-Stale blocks [23, 36]	-$\star$	-$\star$	-$\star$	-$\star$	✗	-	✓	✓	out-of-band	$\sim$	✓	✗	✓
CensorshipCon [24]	✗	$\sim$	✓	Near-/No forks	$[\frac{1}{3},\frac{1}{2})$	$[\frac{1}{3},\frac{2}{3})$	✓	✓	in-band	$\sim$	✗	✓	✗
HistoryRevisionCon [24]	✓	✗	✗	Deep fork	✗	$\approx[\frac{1}{2},1]$	✗	✓	in-band	✓	$\sim$	✓	✗
GoldfingerCon [24]	-	-	✓all	No fork	✗	$\approx[\frac{1}{2},1]$	✗	✓	out-of-band	✓	✓	✗	✓
Pitchforks [18]	-	-	✓all	No fork	✗	$(\frac{1}{3},1]$	✓	✗	out-of-band	✓	✓	✓	✗
Front-running [10, 12]	✗	✓	✗	No fork	✗	$(0,1]$	✗	✗	in-band	✗	✓	✗	✓
Pay per Miner Censorship [37]	✗	✗	✓	No fork	✗	1	✗	✓	in-band	$\sim$	$\sim$	✗	✗
Pay per Block Censorship [37]	✗	✗	✓	No fork	✗	1	✗	✓	in-band	$\sim$	$\sim$	✗	✓
Pay per Commit Censorship [37]	✗	✗	✓	Near-/No fork	✗	1	✗	✓	in-band	$\sim$	$\sim$	✗	✗
P2W Tx Excl.& Ord.	✗	✓	✓	Near-/No forks	✗	$[\frac{1}{2},1]$	✗	✓	out-of-band	✓	✓	✗	✓
P2W Tx Rev. & Excl. & Ord.	✓	✓	✓	Deep fork	✗	$[\frac{1}{2},1]$	✗	✓	out-of-band	✓	✓	✗	✓

Table 1: Comparison of our P2W and existing incentive attacks on cryptocurrencies. A property is marked with ✓ if it is achieved and with ✗ otherwise, - is used if a property does not apply. The symbol $\sim$ means that the property cannot be clearly mapped to any of the previously defined categories without further details or discussion. The symbol $\star$ means that this attack aims against mining pools and hence is not intended to manipulate the chain. The symbol † means that the paper does not explicitly specify the out-of-band payment method but assumes its correctness.

negative-fee mining pool is like a classic mining pool, except that it pays out an above-market return. "*Because such a pool would lose money on expectation, no honest pool should be able to match this reward*" [6]. As with most classic mining pools[3] the pool operator can define the content of a block proposal and hence forge arbitrary attack blocks. Even if miners are rational and hence willing to actively participate in such operations, this approach has at least two major limitations: First, miners would still have to trust the pool owner to pay out the promised rewards. Second, miners could report only blocks which are below the current difficulty target (shares) to prove that they are working for the pool, but withhold blocks which actually match the difficulty target. Thereby, they would potentially gain profits by pretending to participate in the attack/pool without actually doing so. This *miner's dilemma* is a general problem for mining pools [13]. In `Script Puzzle double-spend` [34] PoW like puzzles, offering in-band rewards, are published within the respective cryptocurrency with the intent to distract the hashrate of rational miners. Using the gained advantage to overtake the main chain requires attacker hashrate and transaction ordering merely comes as a byproduct and was not an explicit design goal. Moreover, upon successful execution rational miners are deprived of their bribes, rendering the attack non-repeatable.

Required chain reorganization is outlined in 2.2.2 and classifies if an attack can be realized without, with a near- or with a deep-fork. A classical double-spending attack scenario [30, 32] requires *deep forks* ($\ell > k$) to reorganize the chain. Since the attacker has full control over the required hashrate to perform the attack, he can also arbitrarily order and exclude transactions from the longest chain.

Depending on the scenario and the desired attack outcome, e.g., if only ordering is relevant, deep forks are not necessarily required. For instance, the order of unconfirmed transactions can be manipulated without necessitating a fork, such as performing *front-running* [12]. Ordering attacks on smart contract cryptocurrencies have not been intensively studied [31]. In the paper at hand, we generalize this ability in the context of IMA and analyze how it can be realized (Section 5).

Requires attacker hashrate α for the attack to be successfully executed. As observable in Table 1 there are three attacks which require $\alpha > 0$. The `Script Puzzle 38.2% attack` allows an adversary with appropriate hashrate to establish a computational majority and gain a net profit without considering double-spending attacks. In `Script Puzzle double-spend` the adversary has no explicit minimum hashrate requirement, however low hashrate has to be compensated with more puzzle funds. Moreover, it is designed as a single-shot double-spending attack that, if successful, deprives rational miners of their bribes. `CensorshipCon` uses a smart contract to offer in-band bribes for mining uncle blocks to distract hashrate. Thus, it requires attacker hashrate to include uncle blocks from rational miners in the main chain. Since it has to include all mined uncle blocks, it requires the hashrate of the attacker to be larger than $\frac{1}{3}$ and the hashrate of the bribable miners to be between $[\frac{1}{3},\frac{2}{3})$.

It makes sense to bound the attacker hashrate below $\frac{1}{2}$ since otherwise the attacker has no need to perform bribing attacks as he could overtake the chain single handedly.

Required minimal rational miner hashrate ω for the attack to have a chance to succeed as described and evaluated in the respective paper. Generally, all bribing attacks have to assume that at least some of the miners are rational and hence bribable. Both `Script Puzzle` attacks require all miners to be rational, i.e., $\alpha + \omega = 1$, as well as the `Pay per ...` attacks ($\omega = 1$).

Distracts hashrate from the valid tip(s) of the attacked blockchain to some other form of puzzle or alternative branch that does not contribute to state transitions, e.g., Ethereum uncle blocks in case of `CensorshipCon` or another cryptocurrency in the case of `Pitchforks`.

Requires smart contracts holds true for all attacks which necessitate the use of smart contracts to operate as expected.

Payment specifies where the payments to the bribees are performed. Rewards are either in-band, i.e., in the respective

[3] In P2Pool for example, there is no single operator which can define the content of a block proposal.

cryptocurrency under attack or out-of-band, e.g. in a different cryptocurrency. It can be argued that miners will try not to harm the value of their own cryptocurrency by accepting in-band bribes, hence out-of-band IMA are of particular interest.

Trustless for attacker specifies if the attack itself can be exploited by allowing collaborating/bribed miners to profit without adhering to the attack. For example, `script puzzle` attacks require some form of freshness guarantee to prevent bribees from intentionally waiting until the attack fails before computing puzzle solutions to obtain rewards. It is also possible to claim rewards for stale honest blocks that are later on submitted as uncles to the `CensorshipCon`. Also in naive front-running attacks the attacker has no guarantee that the desired ordering will be achieved by paying a high fee. The `Pay per ...` attacks are modeled only theoretically and no concrete instantiation is given, so it cannot be evaluated if they are constructed trustless for the attacker or collaborator.

Trustless for collaborator specifies if bribees have to trust the attacker that they will receive their payments, if they adhere to the attack. In `Checklocktime bribes` the adversary can try to cheat by creating a conflicting/racing transaction. However, this attempt is only possible if the attacker is under control of some hashrate $\alpha > 0$. The same holds true for `Whale Transactions`, since the attacker has to provide new high fee transactions for each block on the attack chain at each step of the attack. While `HistoryRevisionCon` does not explicitly consider trustlessness for collaborating miners, an augmentation is possible[4], `CensorshipCon` requires that the attacker includes blocks produced by collaborating miners as uncle blocks and thus is not trustless. The `Script Puzzle double-spend` attack is designed as a one-shot attack that defrauds collaborators. The `Script Puzzle` 38.2% attack does not specify how payments are performed and assumes a working trusteless out-of-band payment method.

Subsidy means that the attack leverages some characteristic of the cryptocurrency or environment to become cheaper. In case of `CensorshipCon` the rewards from uncle blocks are used to subsidize the attack, whereas in `Pitchforks` the additional income from merged mining is used as an incentive.

Compensates if attack fails refers to the property that at least a portion of the bribe is paid irrespective of the outcome. To successfully engage rational miners, attacks such as `Checklocktime bribes` [6], `Whale Transactions` [22] and `HistoryRevisionCon` [24], must pay high rewards in case of success to compensate the financial risk faced by bribees if the attack fails despite of their participation. So far no attack facilitating transaction revision achieves this property. `Script Puzzle double-spend` defrauds the bribed miners if successful and hence actually only pays out rewards if it fails.

In front-running attacks, high transaction fees are usually incurred even if the desired ordering effect is not achieved. Thus, in this case it is an undesirable property for the attacker. The same holds true for negative-fee mining pools as rewards have to be paid for performed work even if no attack block fulfilling the difficulty target has been submitted by a miner.

3.1 Main Observations

3.1.1 Ordering attacks underrepresented

Ordering attacks on smart contract cryptocurrencies are still not well understood and discussed [31], yet can be observed in practice [10, 12]. It can be observed that most bribing attack scenarios focus either on *transaction revision* or *transaction exclusion*, and allow for *transaction ordering* merely as a byproduct. All such currently available attacks (`CensorshipCon,Script Puzzle ...`) require that the attacker is in possession of some hashrate. Therefore, the ability to order transactions arbitrarily comes as a byproduct of the ability as a miner to freely define the order and set of transactions to include in their own block proposals as long as a valid block is produced.

A notable exception are *front-running* attacks. However, the attacks observed in practise provide no guarantees for the attacker that the desired ordering is achieved even if the highest transaction fee has been paid as the resulting game is an all pay auction [10]. Moreover, we argue that front-running is only a subset of possible (re-)ordering attacks. For instance, it can be desirable to position a transaction precisely between two other transactions. An example where such a constellation would result in a successful attack can be found in [31], where a vulnerability in the `BlockKing` contract is described. In the paper at hand, scenarios where the ordering of transactions can be manipulated by attackers who themselves are not miners are of particular interest.

3.1.2 No out-of-band attacks facilitating transaction revision

Moreover, we also observe insufficiencies of existing *out-of-band* IMA. The only available technique beyond Goldfinger attacks (`GoldfingerCon, Pitchforks`), is the `Script Puzzle` 38.2% attack, which requires substantial attacker hashrate. `Proof-of-stale-blocks` [23] represents a special case aimed at mining pools. So far there exists no out-of-band attack which facilitates transaction revision. Theoretically, all attacks in which the payment is performed *out-of-band* can be used to launch Goldfinger-style attacks, as the reward of the bribee is not directly bound to the value of the respective cryptocurrency under attack. The question of whether or not such attacks are profitable depends on the external utility that can be generated from the failing cryptocurrency.

In the following, we propose two new IMA aimed at different scenarios to fill the outlined gaps.

[4] The issue stems from the fact that the bribing contract checks the balance of the Ethereum account which should receive the bribing funds before issuing any bribes, but without any additional locking constraints these funds can be moved be the attacker once received.

4 Pay-To-Win Incentive Attacks

We introduce two new *pay-to-win* IMA that are *trustless*, both for the attacker and the collaborating miners. Our attacks do not require the adversary to control any hashrate, i.e., we assume $\alpha = 0$. The payment is performed *out-of-band* therefore, 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 use Bitcoin as target and Ethereum as funding cryptocurrency to instantiate and describe our attacks[5]. When relying on trustless out-of-band payments, the assumption that miners of the target cryptocurrency would not harm their own revenue channel does not necessarily hold true anymore. In a world where more than one cryptocurrency with a certain PoW algorithm exists, this is a even more compelling argument.

Sections 5-6 each describe one of the introduced attacks in detail and follow the same structure: (i) a general overview of the attack, (ii) a step-by-step description, (iii) evaluation of the attack. Note that we also describe two new in-band IMA: *in-band transaction ordering* and *in-band transaction exclusion* in the full version of this paper[6]. The latter was also described and analysed in concurrent work by Winzer et al. [37].

5 Transaction exclusion and ordering attack (out-of-band)

In this section we describe how out-of-band incentive attacks, which facilitate both transaction exclusion and ordering, can be constructed. This might 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 [11, 26, 29]. To execute the attack, we describe how an attacker can 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*, as it combines elements from a mining pool and a chain relay. The attack presented here can also be viewed as a form of the *feather forking* attack proposed by Miller [8]. 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.

5.1 Description

Initialization. The attacker's goal is to prevent an unconfirmed transaction tx_V from being included within N newly mined Bitcoin blocks [7] . The adversary initializes an attack smart contract, which provides the functionality of an ephemeral mining relay, by specifying *block templates*. 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 transactions. For each block template, the corresponding bribe is also 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 the case of Bitcoin block templates, the adversary publishes incomplete block headers to the attack contract, as well as the corresponding coinbase transaction. The latter is necessary to allow collaborating miners to include their own Ethereum payout addresses within the block template, as this is later used by the smart contract for reimbursement if a valid block is submitted. Miners joining the attack can only freely change the *nonce* (used to iterate over PoW solutions) and the *coinbase field* (include Ethereum address) in the generated Bitcoin blocks.

We point out that it is the attacker who must receive the Bitcoin block rewards and not the collaborating miners. Instead, collaborators are reimbursed the value of the Bitcoin block reward as part of the bribing payouts in the Ethereum attack contract. This is required as an additional payout guarantee for the bribee in order to render the attack trustless for collaborators. Thereby, rational miners are not required to verify if the block template they are bribed to mine on will result in a valid block

Attack. Rational miners submit valid Bitcoin blocks, based on the attacker's block templates, to the attack smart contract on Ethereum via the ephemeral mining relay attack contract, which verifies that they form a valid chain. As multiple miners may race to claim the rewards for the same block template, they are incentivized to publish any valid PoW solutions they find in a timely manner. An additional incentive for the bribee to publish a solution promptly, comes from the fact that the attack contract pays an additional ε for each solution if the bribing attack as a whole is successful. The incentive of the attacker to publish the solutions together with the associated full block in Bitcoin comes from the rewards he receives in any case, plus the gain from a successful attack.

At each step, the attacker adds 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 halts. It is possible to include more than one block template in a single block, as shown in e_3 (for details

[5] Theoretically, the attack can be funded on any smart contract-capable funding cryptocurrency which fulfills the requirements listed in Appendix C.

[6] Blinded for review

[7] Additional information can be found in Figure 1 which provides a visualization of an attack.

see Appendix E).

Payout. 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.

5.2 Evaluation with Rational Miners Only ($\omega = 1$)

As previously outlined, the attacker locks up a bribe per submitted block template, to ensure miners face no payout risk and are incentivized to join the attack. For an attack duration of N blocks, we can derive a lower bound for the financial resources (*budget*) for Blofeld in Ether (f_B) required 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[8] (r_b) , for every Bitcoin attack chain block in Ether[9], 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 financed:

$$f_B = N \cdot (r_b + \varepsilon) + c_{operational} \tag{1}$$

There, $c_{operational}$ specifies the operational costs for smart contract deployment and execution (e.g., gas costs on Ethereum). Compared to the current block rewards, the operational costs for managing the smart contract are insignificant given the measurements in [24] and Section D. Although, costs currently being below 100 USD (see D), we decided to set $c_{operationl} = 0.5$ 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 (12.5 BTC) including approximated[10] fees (1 BTC) amounting to $r_b = 13.5$ BTC. Providing an additional $\varepsilon = 1$ BTC, yielding approximately 87.5 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 $\omega < 1$ 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 $N \cdot \varepsilon$, which amounts to approximately $c_{fail} = 81.5$ BTC in our example:

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 $N \cdot (r_b + \varepsilon)$ block templates. Therefore, in our example the costs for a successful attack would be approximately 6.5 BTC, which amounts to roughly 48 000 USD at current exchange rates[11]:

$$c_{success} = N \cdot \varepsilon + c_{operational} \tag{2}$$

Since we assume only rational miners, the attack in this scenario is always successful iff $\varepsilon > 0$ and *no fork* is required. Theoretically the bribe can be much smaller than in our example. For a successful attack to be profitable, the amount (v_a) gained from ordering, or transaction withholding, must exceed $c_{success}$.

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 the other bribing scenarios. Other previously proposed incentive attacks aiming at transaction exclusion require the attacker to have a sizeable portion of the overall hashrate (in the target cryptocurrency) under their direct control to even stand a chance. At least 1/3 for `CensorshipCon`, or at least 38.2% for `Script Puzzle 38.2%`. Acquiring or sustaining the required amount of hashrate already bares large costs, not to mention the additionally required bribes. The costs for renting 1/3 of Bitcoin's total hashrate with NiceHash[12] for the duration of one hour are approximately 470 000 USD.

The `Pay per ...` attacks proposed in concurrent work [37] operate in a comparable setting as our described attack and also highlight the economic feasibility without going into

[8] 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.

[9] For simplicity we assume a fixed exchange rate between cryptocurrencies.

[10] According to https://blockchain.com/charts the average transaction fees per Bitcoin block over the last year are 0.69 BTC. Accounting for standard deviation of fee and produced blocks per day the value varies between 0.75 BTC and 0.64 BTC. To provide a permissive margin we round to 1 BTC.

[11] Exchange rates from end of October 2019.

[12] cf. https://www.crypto51.app/

detail how such attacks can 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.

5.3 Evaluation with Altruistic Miners ($\omega + \beta = 1$)

We now discuss a more realistic scenario where not all miners switch to the attack chain immediately, i.e., some of them act altruistically. Altruistic miners follow the protocol rules and only switch to the attack chain if it becomes the longest chain in the network – but do not attempt to optimize their revenue, contrary to economically rational or bribable miners[13].

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, *near-forks* are required.

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). Each new block is appended to the main chain with probability β, and to the attack chain with probability ω respectively ($\beta + \omega = 1$). 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 $\omega^{k_{gap}+i+1} \cdot \beta^{i}$.

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:

$$\left(\binom{k_{gap} + 2i}{i} - \binom{k_{gap} + 2i}{i - 1} \right) \tag{3}$$

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

$$\sum_{i=0}^{i \leq N - k_{gap} - 1} \left(\binom{k_{gap} + 2i}{i} - \binom{k_{gap} + 2i}{i - 1} \right) \cdot \omega^{k_{gap}+i+1} \cdot \beta^{i} \tag{4}$$

It can be observed [14] that N quickly approaches the maximum achievable probability of catching up one block within an unlimited number of blocks [15]. For example if $\omega = 0.66$, 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.

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 5.2).

6 Transaction revision, exclusion and ordering attack (out-of-band)

In the following, we describe an out-of-band transaction revision attack which directly facilitates double-spend collusion. Miners are bribed 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 [22]. To further reduce the costs, we 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. While we focus on transaction revision in our description, the presented attack also bares the possibility for an adversary to exclude and/or order transactions.

6.1 Description

Initialization phase. First the attacker (Blofeld) creates the uninitialized attack contract and publishes it on the Ethereum blockchain [16]. This is done with a *deploy* transaction included in some Ethereum block e_0 from an Ethereum account controlled by the attacker[17] . Then, Blofeld creates a conflicting

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

[14] Figure 2 visualizes the probability of catching up one block for different hash rates of ω.

[15] The Probability to catch up one block within an unlimited number of blocks is $(\frac{\omega}{\beta})^2$ according to [27, 30]

[16] Additional information can be found in Figure 3 which shows the stages and two different outcomes of the attack.

[17] 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

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.

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, where they are allowed to change the nonce and the coinbase field to find a valid PoW and include their payout Ethereum address. Once a solution has been found, it has to be submitted by the miner to the attack contract, which verifies the correctness of the PoW and that only allowed fields (nonce and coinbase) have been changed. If the submitted solution is valid, the contract knows which previous block hash to 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. As in our previous attack, 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

Attack phase: 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.

Payout phase: Once the attack has ended at time T, the miners who joined the attack can collect their bribes from the contract. To accurately pay out bribes, the contract has to determine which chain in Bitcoin has won the race and is now the longest chain. Since collaborating miners are competing for mined blocks, the contract should have received all attack chain blocks $\{b'_1, \dots, b'_T\}$ by them and hence know exactly the state of the attack branch. Additionally, the attacker who initialized the contract and provided the funds has an incentive to feed the main chain, if such a conflicting longer chain ($\{b_1, \dots, b_T\}$) exists, since he would pay an additional ε for every block otherwise. Therefore there is always some actor who has an incentive to feed the correct longest chain to the attack contract.

The attack contract then distinguishes between the two possible outcomes:

I) *Attack fails (Main chain wins).* In this case the contract must fully compensate the bribed miners for their attack chain blocks at most $\{b'_1, \dots, b'_N\}$, which are now stale. Every collaborating miner who mined and successfully submitted a block on the attack chain receives the reward for that block without an additional ε.

II) *Attack succeeds (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 to provide an initial motivation also for them to switch to the attack chain. 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 and a valid chain up to a predefined block height T is known. The delta between b_N and b_T (or b'_N and b'_T respectively) is the confirmation period k_B defined by the attacker. This ensures that every participant had enough time to submit information about the longest Bitcoin chain to the contract and that the blocks 1 to N have received sufficient confirmations according to an acceptance policy logarithmic in the chain's length as specified in [32]. If the acceptance policy is fulfilled, the contract unlocks the payment of compensations and rewards to the miners of the associated blocks.

For blocks on the attack chain, in the simplest case all bribed miners directly provide Ethereum addresses in the coinbase fields or disclose their public keys directly via *pay-to-pubkey* outputs in the coinbase transaction in Bitcoin, as described and implemented in the Goldfinger attack example in [24]. 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 addresses.

6.2 Evaluation with Rational Miners Only ($\omega = 1$)

A lower bound for the required funds of the attacker(s) f_B can be derived analogous to the evaluation in section 5 by also adjusting for the security parameter k_V defined by the victim Vincent.

Necessary attack budget: A lower bound for the attack budget in Ether f_B can be derived due to the condition $N > k_V$ which has to hold for an attack to be feasible. For Bitcoin, a common choice of $k_V = 6$ requiring N to be at least 7. Setting the current block reward, fees, bribe as in 5.2 leads to to a

that potential collaborators can audit it and familiarize themself 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.

budget of 96 BTC:

$$f_B = N \cdot r_b + (N - k_V) \cdot (r_b + \varepsilon) + c_{operational} \tag{5}$$

Costs of a failed attack: The costs of a failed attack are determined by its duration and thus N s.t., $c_{fail} = N \cdot r_b + c_{operational}$, which leads to $c_{fail} = 95$ BTC in our example.
Costs and profitability of a successful attack: Again, if the attack is successful, it is cheaper. The costs for a successful attack are given by the $k_V \cdot r_b$ main chain blocks that have to be compensated on the attack chain plus the additional $N \cdot \varepsilon$ bribes.

$$c_{success} = k_V \cdot r_b + N \cdot \varepsilon + c_{operational} \tag{6}$$

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 iff $N > k_V$ holds and $\varepsilon > 0$. For Bitcoin, this means that the costs of a successful double spend with $k_V = 6$ and $r_b = 13.5$ and $\varepsilon = 1$ are ≈ 88.5 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 88.5 BTC are observed regularly[18]. For comparison, in its cheapest configuration, the whale attack costs approximately 770 BTC [22], but it operates in a setting where not all miners are assumed to be rational.

6.3 Evaluation with Altruistic Miners ($\omega + \beta = 1$)

Adjusting formula 4 for $k_{gap} = k_V$, we can calculate the success probability of the attack. 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 [19].

Given these probabilities we can calculate the required number of blocks N that need to be funded s.t., the probability[20] of success approaches 100%, while fixing the values for k_V and ω. Assuming more than $\omega > 0.5$ 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 probability for success is deemed sufficiently high i.e., 99.5% in our case. Table 6 shows a comparison against the simulation results of the whale attack described in [22]. It can be observed that, in contrast to the whale attack, our attack becomes cheaper when ω grows large since the required probability is reached 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 between $\approx 85\%$ and $\approx 95\%$ cheaper than the whale attack. Additionally, the costs of our attack in case it succeeds are lower than if it fails, even without accounting for the potential gain from the successful double-spend. For comparison, we also provide the *expected* number of blocks after which the attack is expected be successful and the resulting costs if the attack is stopped at that point in time[21].

6.4 Crowdfunding

The attack described above 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.

A solution which incentivizes multiple attackers to perform double-spending attacks concurrently would allow to split the funds for the attack among collaborators. Thereby, theoretically multiple double-spends of low value transactions by different parties could also be made feasible[22] if they together accumulate enough attack funds (f_B). The main challenges that have to be solved in such a scenario are:

- 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 a double-spend attack 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.
- The attack should not rely on any trusted third party.

On a high level, the stages of the attack are as follows. First, the initialization transaction only announces that an attack might happen and the block interval from b_1 to b_{k_V} that 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[23].

[18] cf. https://www.blockchain.com/btc/largest-recent-transactions

[19] Figure 4 shows the probabilities for different values of rational hashrate ω, as well as different amounts of blocks N these bribed miners can be rewarded/compensated for.

[20] We used 99.5% as a target success probability in our calculations.

[21] Further ideas for cost optimization are briefly discussed in Appendix G

[22] See Appendix H for a brief discussion on the available funds.

[23] 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.

If the funding goal of 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 really 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}, the block b_1 to be forked, and 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 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 have been 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. 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 the 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}. Thus, backers are advised not to invest more Ether than Blofeld (excluding the required collateral).

7 Discussion and Mitigations

Through our comprehensive analysis of related work in the area of incentive manipulation attacks (IMA) (see Table 1), we are able to highlight unconsidered attack types and present new and improved techniques that address outlined gaps. Our attacks further serve to highlight the security dependency between transaction value and confirmation time k_V, as also stated in [32]. As with the negative-fee mining pools presented by Bonneau in [6], there exists an interesting analogy between 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 of an *ephemeral mining relay*, presented within this paper, provides exactly this functionality. Luu et al. [23] 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.

A natural question which arises from the presented attacks is how such IMA can be mitigated. We now discuss possible counter measures and limitations of the described attacks.

Counter attacks For the victim(s) counter bribing is a viable strategy against IMA. 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[24]. If defenders 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 IMA, 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 IMA 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 transaction exclusion and ordering attack proposed in the paper does not even produce a fork in the target cryptocurrency when all miners act rationally. But even if forks are induced, participating miners can make use of the fact that the PoW mining process itself 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 could be used to camouflage the real recipient of the funds e.g., [17,25].

The great benefit of the herein described attacks is that bribes are payed 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 [6] 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 IMA 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. On a high level, the technical requirements for out-of-band attacks to be considered trustless are summarized in Appendix C.

8 Conclusion

The analysis of incentive manipulation attacks presented in this paper forms a necessary prerequisite and basis for the comparison and discussion of related work. We close some of the hereby identified research gaps by describing and evaluating two new trustless IMA that achieve new characteristics and are rendered cheaper than comparable previous approaches. Our new attacks, as well as the existing body of research on IMA, 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 IMA also form a key component, as rational miners can be incentivized to act in a Byzantine manner. The possibility of trustless out-of-band attacks highlights that being able to cryptographically interlink cryptocurrencies increases this attack surface. Further, Smart contract based IMA introduce 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. Together with the topic of counter bribing, new research directions are shown that raise fundamental questions on the incentive compatibility of Nakamoto consensus. All previously proposed, or in-the-wild observed, IMA, as well as the attacks described in this paper, indicate that the security properties of permissionless PoW based cryptocurrencies are neither accurately reflected by assuming only rational actors, nor by ignoring the existence of incentives at all, i.e., only considering honest and Byzantine miners. IMA show that, as soon as rational players are considered, interesting questions arise whether or not the incentive structures of prevalent cryptocurrencies actually encourage desirable outcomes. Additionally, in a world where multiple cryptocurrencies coexist, it is likely not sufficient to model them individually as closed and independent systems.

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

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] Amitanand S Aiyer, Lorenzo Alvisi, Allen Clement, Mike Dahlin, Jean-Philippe Martin, and Carl Porth. Bar fault tolerance for cooperative services. In *ACM SIGOPS operating systems review*, volume 39, pages 45–58. ACM, 2005.

[5] Adam Back, Matt Corallo, Luke Dashjr, Mark Friedenbach, Gregory Maxwell, Andrew Miller, Andrew Poelstra, Jorge Timón, and Pieter Wuille. Enabling blockchain innovations with pegged sidechains, 2014. Accessed: 2016-07-05.

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

[7] Joseph Bonneau. Hostile blockchain takeovers (short paper). In *5th Workshop on Bitcoin and Blockchain Research, Financial Cryptography and Data Security 18 (FC). Springer*, 2018.

[8] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A Kroll, and Edward W Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *IEEE Symposium on Security and Privacy*, 2015.

[9] Vitalik Buterin. Chain interoperability, 2016. Accessed: 2017-03-25.

[10] Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. Flash boys 2.0: Frontrunning, transaction reordering, and consensus instability in decentralized exchanges. arXiv preprint arXiv:1904.05234, 2019.

[11] Christian Decker and Roger Wattenhofer. A fast and scalable payment network with bitcoin duplex micropayment channels. In *Symposium on Self-Stabilizing Systems*, pages 3–18. Springer, 2015.

[12] Shayan Eskandari, Seyedehmahsa Moosavi, and Jeremy Clark. Sok: Transparent dishonesty: front-running attacks on blockchain. 2019.

[13] Ittay Eyal. The miner's dilemma. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 89–103. IEEE, 2015.

[14] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography and Data Security*, pages 436–454. Springer, 2014.

[15] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Advances in Cryptology-EUROCRYPT 2015*, pages 281–310. Springer, 2015.

[16] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol with chains of variable difficulty, 2016. Accessed: 2017-02-06.

[17] Ethan Heilman, Foteini Baldimtsi, and Sharon Goldberg. Blindly signed contracts: Anonymous on-blockchain and off-blockchain bitcoin transactions. Cryptology ePrint Archive, Report 2016/056, 2016. Accessed: 2017-10-03.

[18] Aljosha Judmayer, Nicholas Stifter, Philipp Schindler, and Edgar Weippl. 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.

[19] Aashish Kolluri, Ivica Nikolic, Ilya Sergey, Aquinas Hobor, and Prateek Saxena. Exploiting the laws of order in smart contracts. arXiv:1810.11605, 2018.

[20] Joshua A Kroll, Ian C Davey, and Edward W Felten. The economics of bitcoin mining, or bitcoin in the presence of adversaries. In *Proceedings of WEIS*, volume 2013, page 11, 2013.

[21] Harry C Li, Allen Clement, Edmund L Wong, Jeff Napper, Indrajit Roy, Lorenzo Alvisi, and Michael Dahlin. Bar gossip. In *Proceedings of the 7th symposium on Operating systems design and implementation*, pages 191–204. USENIX Association, 2006.

[22] Kevin Liao and Jonathan Katz. Incentivizing blockchain forks via whale transactions. In *International Conference on Financial Cryptography and Data Security*, pages 264–279. Springer, 2017.

[23] Loi Luu, Yaron Velner, Jason Teutsch, and Prateek Saxena. Smart pool : Practical decentralized pooled mining, 2017. Accessed: 2017-03-22.

[24] Patrick McCorry, Alexander Hicks, and Sarah Meiklejohn. Smart contracts for bribing miners. In *5th Workshop on Bitcoin and Blockchain Research, Financial Cryptography and Data Security 18 (FC). Springer*, 2018.

[25] Sarah Meiklejohn and Rebekah Mercer. Möbius: Trustless tumbling for transaction privacy. Cryptology ePrint Archive, Report 2017/881, 2017. Accessed:2017-09-26.

[26] Andrew Miller, Iddo Bentov, Ranjit Kumaresan, and Patrick McCorry. Sprites: Payment channels that go faster than lightning, 2017. Accessed: 2017-03-22.

[27] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, Dec 2008. Accessed: 2015-07-01.

[28] Rafael Pass, Lior Seeman, and abhi shelat. Analysis of the blockchain protocol in asynchronous networks, 2016. Accessed: 2016-08-01.

[29] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network, 2016. Accessed: 2016-07-07.

[30] M. Rosenfeld. Analysis of hashrate-based double spending, 2014. Accessed: 2016-03-09.

[31] Ilya Sergey, Amrit Kumar, and Aquinas Hobor. Temporal properties of smart contracts. In *Leveraging Applications of Formal Methods, Verification and Validation. Industrial Practice - 8th International Symposium, ISoLA 2018, Limassol, Cyprus, November 5-9, 2018, Proceedings, Part IV*, pages 323–338, 2018.

[32] Yonatan Sompolinsky and Aviv Zohar. Bitcoin's security model revisited, 2016. Accessed: 2016-07-04.

[33] Nicholas Stifter, Aljosha Judmayer, Philipp Schindler, Alexei Zamyatin, and Edgar Weippl. Agreement with satoshi - on the formalization of nakamoto consensus. Cryptology ePrint Archive, Report 2018/400, 2018.

[34] Jason Teutsch, Sanjay Jain, and Prateek Saxena. When cryptocurrencies mine their own business. In *Financial Cryptography and Data Security (FC 2016)*, Feb 2016.

[35] Itay Tsabary and Ittay Eyal. The gap game. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 713–728. ACM, 2018.

[36] Yaron Velner, Jason Teutsch, and Loi Luu. Smart contracts make bitcoin mining pools vulnerable. In *International Conference on Financial Cryptography and Data Security*, pages 298–316. Springer, 2017.

[37] Fredrik Winzer, Benjamin Herd, and Sebastian Faust. Temporary censorship attacks in the presence of rational miners. Cryptology ePrint Archive, Report 2019/748, 2019.

[38] Alexei Zamyatin, Dominik Harz, Joshua Lind, Panayiotis Panayiotou, Arthur Gervais, and William J. Knottenbelt. Xclaim: Trustless, interoperable cryptocurrency-backed assets. Cryptology ePrint Archive, Report 2018/643, 2018. 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
$B_1, B_2, \ldots, B_x$	Other accounts/addresses under the control of the attacker(s)
$V_1, V_2, \ldots, V_x$	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 $r_b + \varepsilon$
ρ	Profit of the attacker
$v, v_d, \ldots$	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 2: Variables and symbols related to actors and costs.

Symbol	Description
α	Hash rate of the attacker
β	Hash rate of all *honest* miners that are not bribable
ω	Hash rate of all rational i.e., bribable miners $\omega = 1-(\alpha+\beta)$ and each mining entity i controls ω_i such that $\omega = \sum_{i=1}^{k} \omega_i$
ω_m	Hashrate of some rational mining entity, which evaluates the profitability of accepting bribes.
ω_α	Estimated hashrate of rational mining entities which will accept bribes and follow the attackers strategy.

Table 3: 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 4: Variables and symbols related to blockchain mechanics.

B Figures

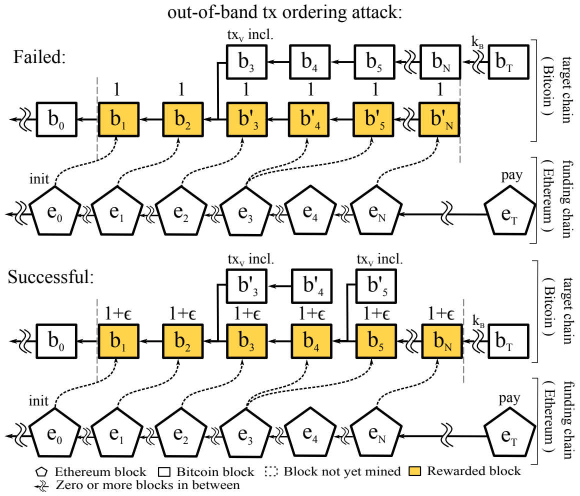

Figure 1: Example timeline showing blockchain structure and resulting payouts of a failed, and a successful tx exclusion and ordering attack with out-of-band payments. The attack is initialized when the attack contract is published in block e_0. Block templates are published as transactions in the funding cryptocurrency and refer to blocks in the target cryptocurrency. The payouts are performed in block e_T. 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.

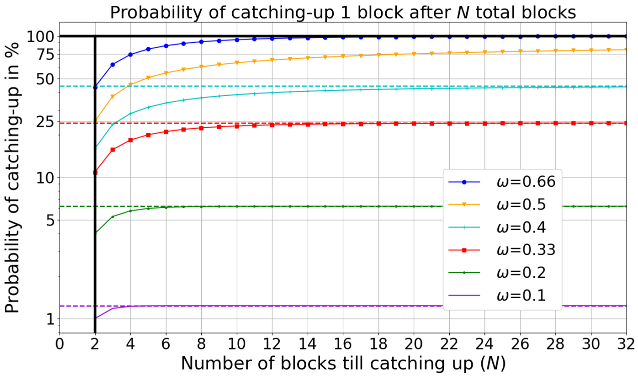

Figure 2: The probability of catching up one block on the y-axis (log scale) within N blocks on the x-axis for different hashrates ω. The dashed line is the maximum probability to catch-up one block after an unlimited number ($N = \infty$) of blocks i.e., $(\frac{\omega}{\beta})^2$.

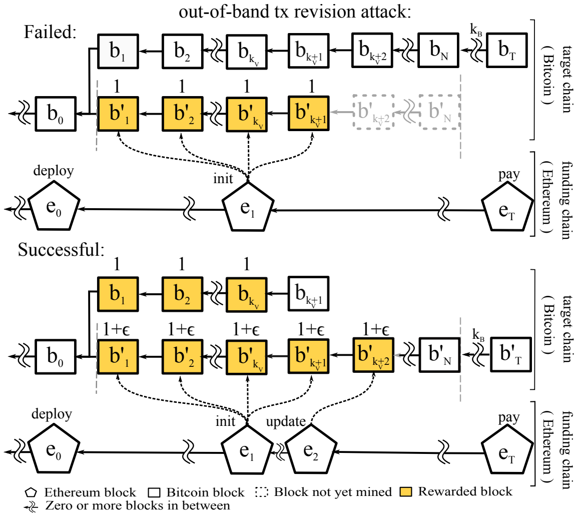

Figure 3: Example timeline showing blockchain structure and resulting payouts of a failed, and a successful tx revision attack with out-of-band payments. The attack contract can be deployed before the actual attack starts. After k_V blocks on the target chain have passed, the attack contract is initialized with $k_V + 1$ 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 to also mine on the attack chain.

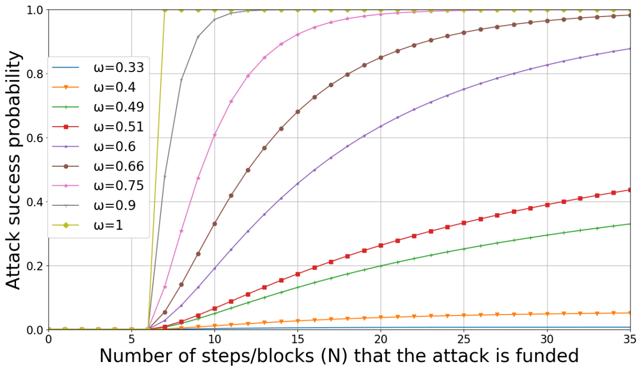

Figure 4: Attack success probability of a double-spending attack depending on the amount of blocks N that can be compensated/rewarded and different values for the rational hashrate ω. The number of required confirmation blocks by Vincent is set to $k_V = 6$.

C Technical requirements

On a high level the technical requirements which would allow to trustlessly execute all our attacks 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 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 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 2).
4. A trustless way to determine the state of the target cryptocurrency 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 in smart contracts on the funding cryptocurrency.
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.

D Evalution of the ephemeral mining relay

To verify the outcome of the attack and correctly pay rewards in trustless out-of-band scenarios, we introduce the concept of *ephemeral mining relays* (EMR)[25]. An ephemeral mining relay is a smart contract that combines the functionality of a chain relay [1, 9, 38] and mining pool [23, 36].

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 [5]). However, a naive chain relay implementation allows only 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.

[25] 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.

In contrast to previous proposals, our EMR is capable of fully validating the consensus rules of the target cryptocurrency by 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.

Furthermore our EMR tracks all ongoing blockchain branches, which is not only a necessary feature to determine the winning branch, but also to correctly compensate the failed branch of an incentive attack.

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 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 feed the correct information to the relay in a timely fashion. Consider, for example, a rational miner R who mined a block template for b'_3 (see figure 1). 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 payed 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 new block templates in order to reduce his losses in case the attack appears likely to fail.

Costs estimates: We implement a fully functional EMR on Ethereum, which is capable of verifying the state of the Bitcoin blockchain. We use Solidity v0.6.2 and use a local instance of the Ethereum blockchain for cost analysis. The cost estimates for the identified operations are summarized in Table 5 We used our EMR to approximate the costs for submitting a block template for a Bitcoin block ($ 0.58 USD), as well as the costs for submitting and verifying a new Bitcoin block ($ 0.90 USD) in the worst case. In total the costs of an example attack on Bitcoin with $k_V = 6$ and $k_B = 6$ and a gas price of 12 Gwei are about $ 30 USD. This confirmes that the costs for maintaining 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 amounts to $ 76 875 USD.

Operation	**Approx. costs**	
	Gas	**USD**
Deployment	6 156 688	11.86
Initialization phase	1 364 277	2.63
Attack phase	8 203 136	15.80
Payout phase	64 511	0.12
Total operational costs	15 788 612	30.41
one submitted block template	302 228	0.58
one submitted block	468 273	0.90

Gas price: Gas price 12 Gwei, exchange rate 160.53 USD/ETH [2]

Table 5: Overview of operational costs $c_{operational}$ for each of the main Ethereuem smart contract operations of the ephemeral chain relay (EMR) executing a successful transaction revsion,exclusion and ordering attack on Bitcoin with $k_V = 6$ and $k_B = 6$.

E Evaluation of Desynchronization

Publishing new block templates timely is a key requirement of this attack. In favor of an easier presentation we chose to 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 Ethereuem block announcing the new block template. Although, it is possible for the target and the funding chain to desynchronize, i.e., that two or more Bitcoin blocks are mined before a single Ethereum block has been found. 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. 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.

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 ($x = 15$):

$$\lambda = \frac{1}{E_{BTC}(x)} \tag{7}$$

$$P(X < x) = 1 - e^{-\lambda \cdot x} \tag{8}$$

$$P(X < 15) \approx 2.47\% \tag{9}$$

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:

$$P(N) = 1 - \big(1 - P(X < 15)\big)^{N-1} \tag{10}$$

$$P(32) \approx 53.93\% \tag{11}$$

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:

$$P(X > n) = 1 - P(X \leq n-1) \tag{12}$$

$$P(X \leq n) = F(x) = e^{-\lambda} \sum_{i=0}^{n-1} \frac{\lambda^i}{i!} \tag{13}$$

$$P(X > 1) \approx 2.47\% \tag{14}$$

$$P(X > 2) \approx 0.03\% \tag{15}$$

$$P(X > 3) \approx 2.556 \cdot 10^{-4}\% \tag{16}$$

$$P(X > 4) \approx 1.595 \cdot 10^{-6}\% \tag{17}$$

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:

$$P(n,N) = 1 - \big(1 - P(N > n)\big)^{\lceil (N-1)/n \rceil} \tag{18}$$

$$P(n = 1, N = 32) \approx 53.930\% \tag{19}$$

$$P(n = 2, N = 32) \approx 0.490\% \tag{20}$$

$$P(n = 3, N = 32) \approx 0.003\% \tag{21}$$

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 section. 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.

E.1 Block template distribution

Given the above probabilities, the attacker is advised to publish block templates for multiple blocks in advance (leaving references to previous blocks to be filled in by miners). Also in practice collaborating miners would want to have at least a couple of block templates available to ensure that their hardware does not stall. To ensure that new block templates are available to rational miners, independently of block intervals in the funding cryptocurrency, several approaches are possible. The attacker could, for example, publish a sequence of block templates where only the first includes the *previous block hash* and the other previous block hash values are filled and checked automatically by the smart contract based on the previously submitted valid attack blocks.

Other approaches can also be envisioned. In theory, 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 in a way that accepts any valid Ethereum transaction signed by the attacker as a proof that the therein announced new block template for a specific attack was approved, and is rewarded accordingly. 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 indeed the respective block on the target chain was based on a valid template.

F 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.

G 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.

H 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[26] 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.

I Transaction revision (out-of-band) attack costs

ω	Whale costs	P2W costs c_{failed}	P2W costs $c_{success}$	N	P2W costs $c_{expected}$	N expect.
0.532	2.93e+23	6953	596	515	136	55
0.670	999.79	554	122	41	95	14
0.764	768.09	298	104	22	92	10
0.828	1265.14	216	98	16	90	9
0.887	1205.00	176	94	13	89	8
0.931	1806.67	149	92	11	89	8
0.968	2178.58	122	90	9	89	7
0.999	2598.64	108	90	8	89	7

Table 6: Comparison of attack costs given in BTC for the whale attack [22] and our out-of-band tx. revision attack with $k_V = 6$, $r_b = 13.5$, $c_{operational} = 0.5$ and $\varepsilon = 1$.

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

J Transaction ordering attack (in-band)

This *no-fork* attack pays additional rewards to miners for reordering *unconfirmed transactions*, comparable to front-running attacks [10, 12]. 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 [10]. 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.

J.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.

J.2 Evaluation with Rational Miners Only ($\omega = 1$)

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 $\varepsilon > 0$ 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.

J.3 Evaluation with Altruistic Miners ($\omega + \beta = 1$)

In theory, this attack is practicable with any hash rate of bribable miners $\omega > 0$, 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 K 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.

J.4 Counter Mechanisms

For a list of general counter mechanisms, e.g., require blinded commitments up-front, which can be used to avoid such vulnerability during the design of smart contracts see [10, 12]. 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 [10]. 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 [27]) 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. Moreover, among the previously described bribing attacks in Section 3, no attack is directly applicable by Vincent in this scenario. For an analysis on how much it costs to remove one block from the chain see Appendix K.

J.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.

J.5.1 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.

J.5.2 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*[27] 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 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 pre-computed. 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.

K 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.

K.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

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

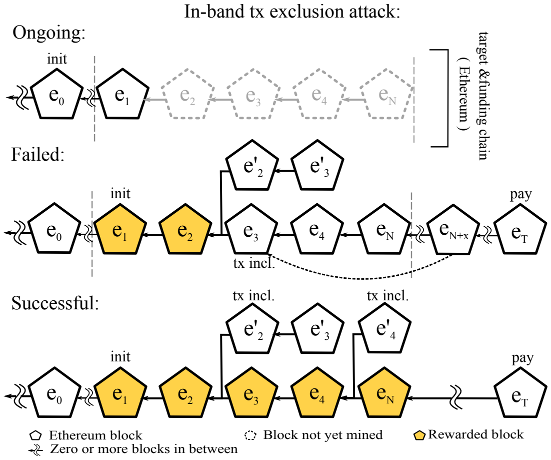

Figure 5: The figure shows a ongoing-, a failed- as well as a successful Transaction exclusion attack with in-band payments. The attack is initialized when the attack contract is published in block e_1. If the unwanted transaction has been included, this can be proven to the attack contract as shown 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.

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., $e_{x+1}, e_{x+2}, \ldots$, 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 K.5.

K.2 Evaluation with Rational Miners Only ($\omega = 1$)

Estimating the costs of such an attack in a scenario where all miners are rational ($\alpha = \beta = 0$ and $\omega = 1$) 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., $\varepsilon > fee(tx_V)$.

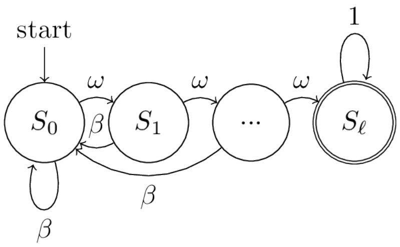

Figure 6: Finite Markov chain for calculating the probability of mining at least ℓ consecutive blocks with hashrate ω.

K.3 Evaluation with Altruistic Miners ($\omega + \beta = 1$)

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 ω, the likely hood of rational miners finding chains of consecutive blocks decreases exponentially in their length ℓ. For example, given $\omega = \frac{2}{3}$ 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 $\omega = \frac{2}{3}$ 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 ω by computing the matrix of the finite Markov chain depicted in 6 with N as exponent as shown in formula 22.

$$P = \begin{bmatrix} \beta & \omega & 0 & \cdots & 0 \\ \beta & 0 & \omega & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \beta & 0 & 0 & \cdots & \omega \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}^N \cdot \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \end{bmatrix} \quad (22)$$

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 [22] (specifically, we extend the analysis after Equation 4 in the aforementioned paper).

A rational miner with hashrate ω_m will mine on the attack chain if he his expected profit is higher than with honest mining. To make a rational decision on which chain to mine, he must estimate and compate the hashrate of (i) all miners expected to join the attack ω_α, and (ii) the hashrate of all altruistic miners extending the conflicting main chain branch β. Note that $\omega = \omega_\alpha + \omega_m$. 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 ω_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:

$$\rho = \frac{\left(1 - \left(\frac{\alpha+\omega_\alpha}{\beta+\omega_m}\right)^{z+1}\right) \cdot \omega_m}{\beta + \omega_m} \tag{23}$$

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:

$$\rho' = \frac{\left(\frac{\alpha+\omega_\alpha+\omega_m}{\beta}\right)^{z+1} \cdot \omega_m}{\alpha + \omega_\alpha + \omega_m} \cdot (\varepsilon + 1) \tag{24}$$

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

$$\varepsilon > \frac{\left(1 - \left(\frac{\alpha+\omega_\alpha}{\beta+\omega_m}\right)^{z+1}\right)}{\beta + \omega_m} \cdot \frac{\alpha + \omega_\alpha + \omega_m}{\left(\frac{\alpha+\omega_\alpha+\omega_m}{\beta}\right)^2} - 1 \tag{25}$$

To estimate a worst case lower bound for the necessary bribe, we set $\omega_\alpha = 0$ and a calculate ε for a small rational miner with hashrate $\omega_m = 0.05$. We receive $\varepsilon \approx 17 \cdot r_e$, 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 (ω_α and ω_m) in Table 7 in Section K.5. We observe that once $\omega_m + \omega_\alpha$ 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 $\varepsilon = 0$).

Table 7 shows the costs for incentivizing in-band transaction exclusion if blocks that include the respective transaction should be forked by rational miners.

K.3.1 Comparison to Existing Attacks

A comparable attack allowing arbitrary transaction exclusion is `HistoryRevisionCon` [24]. 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 ($\alpha > \frac{1}{3}$). For comparison: if we assume $\omega = 0.33$ s.t., $\omega_\alpha = 0.28$ and $\omega_m = 0.05$, our attack would require $\varepsilon \approx 0.603 \cdot r_e$.

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

K.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 J.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 5

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

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

Table 7: Comparison of minimum bribing attack costs ε for certain attack hashrates ω_α and undecided individual miners ω_m. The table also shows the expected reward of m if ω_m would be directed towards the attack chain ρ', as well as the expected reward ρ if ω_m would be directed towards the main chain.

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 5.

K.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.

K.5.1 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 J.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[28].

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.

K.5.2 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.

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