Pay-To-Win:
Incentive Attacks on Proof-of-Work Cryptocurrencies

Aljosha Judmayer
SBA Research
ajudmayer@sba-research.org

Nicholas Stifter
TU Wien
nicholas.stifter@tuwien.ac.at

Alexei Zamyatin
Imperial College London
a.zamyatin@imperial.ac.uk

Itay Tsabary
Technion and IC3
itaytsabary@gmail.com

Ittay Eyal
Technion and IC3
stanga@gmail.com

Peter Gaži
IOHK
peter.gazi@iohk.io

Sarah Meiklejohn
University College London
s.meiklejohn@ucl.ac.uk

Edgar Weippl
SBA Research
eweippl@sba-research.org

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 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 attacks*, are systematically analyzed and categorized. Thereby, we show that the problem space is not fully explored and present several new and improved incentive attacks. We identify *no- and near-fork incentive attacks* as a powerful, yet largely overlooked, category. In particular *transaction ordering and exclusion attacks* raise serious security concerns for stateful cryptocurrencies, such as 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). We implement the basic building blocks of all our out-of-band attacks as Ethereum smart contracts to demonstrate their feasibility.

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 [25].

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 and the intended properties of the system emerge from the appropriate utility model for miners [10]. *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. The economic feasibility of obtaining a computational majority, in particular for small PoW cryptocurrencies, is hereby already effectively highlighted through websites such as Crypto51[1] and real-world incidents[2].

Another serious cause for concern are transaction *ordering* and *exclusion* attacks (e.g. [19]) that can be performed as *near- or no fork* incentive attacks. Hereby, the adversary's goal is to bribe miners such that they construct new (valid) blocks in a way that benefits the adversary. A specific form of this attack, namely *front running*, is highlighted and analyzed in recent research, focusing on its occurrence in the Ethereum platform [12, 14]. The 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.

Most incentive attacks proposed so far focus on optimizing a players utility. In this paper, we also consider how bribes may break the mechanism design and cause rational players to deviate from the prescribed protocol. To do so, we first systematically expose the body of research on bribing-, front-running- Goldfinger- and other related attacks. These techniques are summarized under the general term *incentive attacks* as they all intend to tamper with the incentives of rational actors in the system.

We present, to the best of our knowledge, the **first comprehensive systematization** of incentive attacks which allows to compare and categorize the varying system models of previously proposed attacks. Thereby, we show that the problem space is not yet fully explored.

We propose **three new incentive attacks** [3] to fill some of the gaps outlined by our systematization. Two thereof lie in the previously underrepresented domain of no/near fork incentive attacks. The third is the first attack to incentivize trustless double-spend collusion in an out-of-band scenario.

We introduce three **crucial enhancements to incentive attacks** i.e., (i) *ephemeral mining relays*, as a mechanism for executing trustless, time bounded, cross-chain incentive attacks, (ii) *guaranteed payment* of bribed miners even if the attack fails, which

[1]See: https://www.crypto51.app
[2]See: https://www.gate.io/article/16735
[3]One of which is also described in concurrent work [35].

actually reduces the costs of such attacks, (iii) *crowdfunded attacks*, to further reduce the individual costs of executing incentive attacks.

1.1 Outline of this Paper

We begin our analysis by outlining general system model assumptions most analyzed and newly proposed incentive attacks have in common (Section 2). Literature on bribing- and related attacks is then systematically analyzed and compared in Section 3.

Our new pay-to-win attacks are first outlined at a high level in Section 4, including the main technical requirements that have to be fulfilled. Sections 5, 6 and 7 describe in detail how the respective attacks can be constructed in current cryptocurrencies such that the previously stated requirements are satisfied.

The paper is completed with a general discussion in Section 8 and concluding remarks in Section 9. We provide extensive details on the implementation (ephemeral mining relay) and evaluation of the individual attacks in the Appendix.

2 GENERAL SYSTEM MODEL

For all analyzed and presented incentive attacks we adopt the following general system model. If an analyzed attack deviates from this model, it is highlighted when the attack is described. Additional assumptions and augmentations relevant for specific attacks are introduced where they become necessary.

We consider incentive attacks within *permissionless proof-of-work (PoW) cryptocurrencies*. That is, we assume protocols adhering to the design principles of Bitcoin [25], generally referred to as Nakamoto consensus or Bitcoin backbone protocol [17, 26, 31].

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 [9, 21, 23, 32], 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 this value by creating and broadcasting valid transactions within the network. Moreover, as in prior work, e.g., [21, 23, 33], we make the simplifying assumption that exchange rates are constant over the duration of the attack.

Participating miners are categorized into three groups and their roles remain static for the attack duration. Categories follow the *BAR (Byzantine, Altruistic, Rational)* [5, 20] rational behavior model.

- **Byzantine miners or attacker(s) (Blofeld):** 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 hash rate $\alpha \geq 0$ in the target cryptocurrency. The attacker may deviate arbitrarily from the protocol rules.
- **Altruistic or honest miner(s) (Alice):** 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 hash rate $\beta \geq 0$ in the target cryptocurrency.
- **Rational or bribable miner(s) (Rachel):** Miners R controlling hash rate $\omega \geq 0$ in the *target cryptocurrency* aiming to maximize their short term profits. We consider miners "bribable" if they follow strategies that deviate from the protocol rules as long as they are expected to yield higher profits than being honest. Bribable miners have some hash rate $\omega > 0$ in the cryptocurrency under attack. For our analyses we assume bribable miners do not concurrently engage in other rational strategies such as selfish mining.

Additionally, we assume the **victim (Vincent)** of the bribing attacks to be a client without any hash rate. While other bribing attacks model the victim as an honest miner, we also distinguish between a rational victim to allow for a more fine grained description and subsequent analysis. If Vincent is to be modeled with possession of some hash rate, it can be considered either to be part of β or ω. 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 incentive attacks 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. We further assume that all miners in the *target cryptocurrency* have **perfect knowledge** about the attack once it has started. Analogous to [17], 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, i.e., an additional *funding cryptocurrency* is used to orchestrate and fund the attack on a *target cryptocurrency*, then we assume their respective mean block interval and mining difficulties to remain fixed for the duration of the attack as well. Further, no attacks are launched against the funding cryptocurrency concurrently.

3 INCENTIVE ATTACK SYSTEMATIZATION

Incentive attacks represent a generalized form of bribing attacks [9], 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 and approaches, we systematically analyze and categorize the body of research on incentive manipulation attacks.

3.1 Intended Impact on Transactions

A core goal for permissionless PoW cryptocurrencies, is it to achieve an (eventually) consistent and totally ordered log of transactions that define the global state of the shared ledger. We differentiate between the following three main categories of incentive attacks aimed at manipulating transactions and their ordering:

- **transaction revision**, change a previously published, possibly confirmed transaction;
- **transaction ordering**, change either the proposed or already agreed upon order of transactions;
- **transaction exclusion**, exclude a specific transaction from the ordered log of transactions, either for a bounded amount of time or indefinitely.

Some incentive attacks may allow multiple types of transaction manipulation at the same time (see Table 1). The ability to invalidate a transaction can be considered the result of successfully performing one or more of the above transaction manipulation attacks and does not necessitate a separate category.

3.2 Required Interference with Consensus

While the previous category 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 l exceeding a security parameter k is necessary (i.e., $l > k$). The victim defines k [16, 30] 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 a k defined by the victim (i.e., $l \leq k$).
- **No-fork required**, where no blockchain reorganization is necessary at all (i.e., $k = 0$).

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 sequential proof-of-work.

Some attacks, such as front-running or transaction revision where the victim accepts $k = 0$, may be executable as no-fork attacks. Others, such as performing a double spend where the victim has carefully chosen k [30], may need to substantially affect consensus and violate the security assumption that a common prefix of the blockchain remains stable, except with negligible probability [16].

3.3 Categorization and Comparison of Attacks

Equipped with our classification by intended impact and required interference, we consider related work on the topic of incentive manipulation attacks. Further properties are introduced as part of the discussion. Table 1 presents our categorization of previous proposals, as well as our new pay-to-win attacks. Each row represents a different attack and columns outline respective properties.

Tx revision / Tx ordering / Tx exclusion are outlined in subsection 3.1. In the literature, several bribing attacks are designed to replace or *revise* a specific transaction, i.e., perform a single double-spend [9, 21, 23]. As a consequence, they do not consider defining the order or exclusion of arbitrary transactions. Despite of the double-spending transaction 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` [23] 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] leverage merged mining to subsidize the creation of empty (or specially crafted) blocks in the attacked chain [18].

The *Script puzzle* 38.2% [32] and `CensorshipCon` attack [23] distract hash rate of bribable miners to gain an advantage over the remaining honest miners. Both attacks allow arbitrary transaction ordering and exclusion, but require that the adversary controls more hash rate than the remaining miners. Deep forks and transaction revision are not directly considered and require further analysis.

The only previously proposed attack to achieve all three properties is the *Script Puzzle double-spend* [32]. However, upon successful execution rational miners are deprived of their bribes, rendering the attack non-repeatable.

Requires chain reorganization is outlined in 3.2 and classifies if an attack is realized without-, with a near- or with a deep-fork. A classical double-spending attack scenario [28, 30] requires *deep forks* ($l > k$) to reorganize the chain. Since, the attacker has full control over the required hash rate to perform the attack, he also can arbitrarily order- and exclude transaction from the longest chain.

Depending on the scenario and the desired attack outcome, e.g., 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* [14]. Ordering attacks on smart contract cryptocurrencies have not been intensively studied [29]. In the paper at hand, we generalize this ability in the context of incentive attacks and analyze how it can be realized (Section 5).

Requires attacker hash rate 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 hash rate to establish a computational majority and gain a net profit without considering double-spending attacks. In (*Script Puzzle double-spend*) the adversary has no minimum hash rate requirement, however it is designed as a single-shot double-spending attack. `CensorshipCon` also requires attacker hash rate to include uncle blocks from rational miners. Since it has to include all mined uncle blocks, it requires that the hash rate of the attacker is larger than $\frac{1}{3}$ and the hash rate of the bribable miners to be between $[\frac{1}{3}, \frac{2}{3})$.

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

Required minimal rational miner hash rate 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. Note that the *Script Puzzle* attacks require all miners to be rational, i.e., $\alpha + \omega = 1$.

Distracts hash rate 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.

	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 [9]	✓	✗	✗	Deep fork	✗	$\approx [\frac{1}{2}, 1]$	✗	✗	in-band	✓	~	✗	✗
Whale Transactions [21]	✓	✗	✗	Deep fork	✗	$\approx [\frac{1}{2}, 1]$	✗	✗	in-band	✓	~	✗	✗
Script Puzzle double-spend [32]	✓	✓	✓	Deep fork	$(0, \frac{1}{2})$	$1 - \alpha$	✓	✗	in-band	~	✗	✗	~
Script Puzzle 38.2% attack [32]	✗	✓	✓	Near-/No forks	$[0.382, \frac{1}{2})$	$1 - \alpha$	✓	?†	out-of-band	?†	?†	✗	✓
Proof-of-Stale blocks [22, 34]	-★	-★	-★	-★	✗	-	✓	✓	out-of-band	~	✓	✗	✓
CensorshipCon [23]	✗	✓	✓	Near-/No forks	$[\frac{1}{3}, \frac{1}{2})$	$[\frac{1}{3}, \frac{2}{3})$	✓	✓	in-band	~	✗	✓	✗
HistoryRevisionCon [23]	✓	✗	✗	Deep fork	✗	$\approx [\frac{1}{2}, 1]$	✗	✓	in-band	✓	~	✓	✗
GoldfingerCon [23]	-	-	✓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 [12, 14]	✗	✓	✗	No fork	✗	$(0, 1)$	✗	✗	in-band	✗	✓	✗	✓
P2W Tx Ord. IB	✗	✓	✗	No fork	✗	$(0, 1)$	✗	✓	in-band	✓	✓	✗	✗
P2W Tx Excl.& Ord. OOB	✗	✓	✓	Near-/No forks	✗	$[\frac{1}{2}, 1]$	✗	✓	out-of-band	✓	✓	✗	✓
P2W Tx Rev. & Excl. & Ord. OOB	✓	✓	✓	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 is is achieved and with ✗ otherwise, - is used if a property does not apply. The symbol ~ means that the property cannot be clearly mapped to any of the previously defined categories without further details or discussion. The symbol ★ 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.

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 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 incentive attacks are of particular interest.
Trustless for attacker specifies if the attack itself can be exploited by 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`.
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 hash rate $\alpha > 0$. 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 Puzzles double-spend* attack is designed as a one-shot attack that defrauds collaborators. The *Script Puzzles* 38.2% attack does not specify how payments are performed and assumes a trusteless out-of-band payment method. In front-running attacks the attacker has no guarantee that the desired ordering will be achieved by a high fee.
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 unconditional of the outcome. To successfully engage rational miners, attacks such as Checklocktime [9], whale [21] and `HistoryRevisionCon` [23], must compensate the financial risk faced by participants in case it fails. So far no attack facilitating transaction revision achieves this property.

Script Puzzle double-spend defrauds the bribed miners if successful and 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.

3.4 Main Observations

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 by-product. A notable exception are *front-running* attacks. 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 for such an attack can be found in [29], where a vulnerability in the `BlockKing` contract is described.

Generally, any miner can freely define the order and set of transactions to include in their own block proposals as long as a valid block is produced. In this paper, scenarios where the ordering of transactions can be manipulated by attackers that are not themselves miners are of particular interest. Ordering attacks on smart contract cryptocurrencies are still not well understood and discussed [29], yet can be observed in practice [12, 14]. A notable exception specifically designed to exclude transactions is `CensorshipCon` [23].

Moreover, we also observe an insufficiency of *out-of-band* incentive attacks. The only available techniques, beyond Goldfinger attacks, require substantial attacker hash rate. Note hat *Proof-of-stale-blocks* [22] represents a special case aimed at mining pools.

[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.

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 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 new incentive attacks aimed at different scenarios to fill some of the gaps that were outlined by our systematic comparison and subsequent observations.

4 PAY-TO-WIN INCENTIVE ATTACKS

We introduce three new *pay-to-win* incentive attacks that are *trustless*, both for the attacker and the collaborating miners. We differentiate between *in-band* attacks, i.e. funded and executed within the same cryptocurrency, and *out-of-band* attacks, where the attacks are funded and coordinated on a different cryptocurrency. Our attacks do not require the adversary to control any hash rate, i.e., we assume $\alpha = 0$.

This section provides a high level overview of the attacks, introduced techniques and operational requirements. Sections 5-7, 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, (iv) attack evaluations, and, (v) analysis of attack properties.

4.1 P2W Attack Overview

In-Band Attacks. We introduce a new in-band attack, executed and coordinated on a smart contract capable proof-of-work blockchain.

In-Band Transaction Ordering. The attack (Section 5) incentivizes no-fork transaction ordering in-band: colluding miners are rewarded if unconfirmed transactions are ordered as desired by the adversary. Compared to front running [12], this attack utilizes a smart contract to directly reward miners iff the correct ordering condition is upheld. Current front running attacks can be considered an *all pay auction* [12], where the losing transaction – even if its execution fails – unnecessarily pays a high fee.

Out-of-Band Attacks. 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 smart contracts (e.g. Ethereum), there are no such requirements for the *target cryptocurrency* (e.g. Bitcoin). Out-of-band attacks are arguably more difficult to detect, as this would require monitoring multiple smart contract capable blockchains.

Out-of-Band Transaction Exclusion/Ordering. This attack (Section 6), presents an out-of-band transaction exclusion attack where the attacker is also able to specify the order of the included transactions. This might be used to censor certain transactions (e.g., closing of payments channels) or to perform multiple front-running attacks at once. 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 (see the end of this Section).

Out-of-Band Transaction Revision. Finally, we describe an out-of-band transaction revision attack (Section 7) that directly facilitates double-spend collusion: miners are bribed to mine blocks on the branch favored by the adversary, *on another blockchain.* The previously introduced techniques introduced are hereby combined to demonstrate that they can together form more powerful attacks. We show how it can be constructed to *always reward collaborating miners*, regardless of the outcome of the attack. Interestingly, this renders attacks significantly cheaper, as the necessary compensation to colluders when the attack fails is reduced.

4.2 Technical Requirements

The technical requirements for all three introduced attacks to be considered trustless are summarized as follows:

(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 has not taken place (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 funding cryptocurrency 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.

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.* An ephemeral mining relay is a smart contract that combines the functionality of a chain relay [2, 11, 36] and mining pool [22, 34]. However, in contrast to previous proposals, the mining relay is capable of fully validating the consensus rules of the target cryptocurrency by restricting the allowed block structure. Furthermore, it tracks all ongoing blockchain branches, which is a necessary features for correct verification of incentive attacks. We provide a more detailed description of the ephemeral mining relay construction, a proof of concept implementation deployed on Ethereum for verifying the Bitcoin blockchain, and a cost analysis in the Appendix G.1. The additional verification costs of our implementation amount to less than USD 1.00 per Bitcoin block - negligible compared to the potential economic impact of incentive attacks.

5 TRANSACTION ORDERING IN-BAND

This *no-fork* attack pays additional rewards to miners for reordering *unconfirmed transactions*, comparable to front-running attacks [12, 14]. In front-running attacks, the adversary increases the chance of their 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 [12]. 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.

5.1 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 mined, (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* [4], tx_V remains valid and can alternatively be included by miners. 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 blocks (k 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.

5.2 Evaluation

5.2.1 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 $\epsilon > 0$ in addition to normal mining. Participation in the attack does not require to mine on an alternative fork, hence colluding miners face no additional risk that their blocks will be excluded from the main chain. It is also possible for miners to include an unconfirmed attack contract creation transaction in the same block as the ordering attack itself and still be certain of payment if their block becomes part of the longest chain.

5.2.2 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 C in the Appendix for an analysis where rational miners are additionally incentivized to fork main chain blocks to successfully remove a undesired block from the chain.

5.3 Properties and Analysis

We now analyze possible defensive strategies of the victim (Vincent). Specifically, we consider the possibility of counter bribing.

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* instead of the *all-pay-auction* observed in other front-running [12]. 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 [25]) wallet, he may only recognize the attack after a new block with the intended ordering of the attacker has already been mined. 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, form 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 C.

6 TRANSACTION EXCLUSION AND ORDERING OUT-OF-BAND

In this section we describe how out-of-band incentive attacks, which facilitate both transaction exclusion and ordering, can be constructed. Thereby, we supersede our previous attack in terms of capabilities. For example, 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 [13, 24, 27].

Out-of-band attacks have the advantage, that they can be funded on any smart contract capable *funding cryptocurrency*, while the attack occurs on a different *target cryptocurrency*. These attacks are arguable more difficult to detect and protect against, as the victim would have to monitor multiple, if not all, smart contract capable blockchains. For better readability, we use Bitcoin (target) and Ethereum (funding cryptocurrency) as examples when describing the attack below. As outlined in section 4, we rely on ephemeral mining relays to trustlessly verify the state of the target cryptocurrency, the correct execution of the attack, and also handle payouts to colluding miners (see Appendix G.1 and F for details on the relay).

6.1 Description

Initalization. The attacker's goal is to prevent an unconfirmed transaction tx_V from being included in newly mined blocks within Bitcoin (target). The adversary initializes an attack smart contract by specifying *block templates*, which 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 independent of the 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, since rational miners may not be able to verify if the block template they are bribed to mine on will result in a valid block. We provide more details on block template constructions in Appendix G.1 and F.

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, 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 timely publish any valid PoW solutions they find. An additional incentive for the bribee to publish a solution timely 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 with the associated full block on the target chain comes from the rewards he receives directly, plus the gain from an successful attack.

At each step, the attacker updates the Bitcoin block templates after each submission to the attack contract and, if necessary, can add additional bribes. If no new templates are submitted, the attack halts. Figure 1 provides an explanatory visualization of an ongoing attack.

We note that 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. As such, the attacker may also publish block templates for multiple blocks in advance (leaving references to previous blocks to be filled in by miners).

Payout. Similar to the in-band case, miners can claim payouts in the attack contract once k Bitcoin blocks have been mined after the attack has ended (k 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, compliance with the specified block template, and that all blocks form a valid attack chain. If a submitted block 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 regardless if the attack has failed, plus an extra ϵ if the attack is successful.

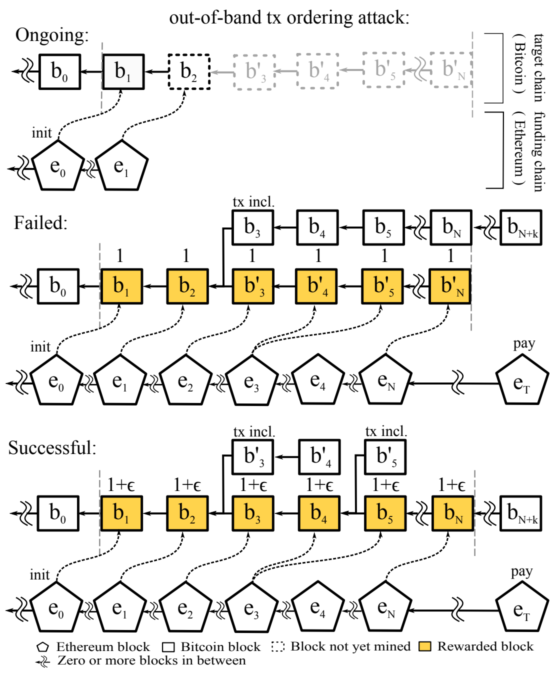

Figure 1: Blockchain structure and timeline of an ongoing, a failed, as well as a successful tx exclusion and ordering attack with out-of-band payments. The attack is initialized when the attack contract is published in block e_0. Block templates are published as transactions in the funding cryptocurrency and refer to blocks in the target cryptocurrency. It is possible to include more than one block template in a single block, as shown in e_3. The payouts are performed in block e_T. The colored blocks are rewarded by the attack contract, either only with their original value (reward + free = 1) or with an additional $1 + \epsilon$ if the attack was successful.

6.2 Evaluation

6.2.1 Evaluation with Rational Miners Only ($\omega = 1$): We assume a scenario where all miners act rationally, i.e., are bribable, and have instant knowledge of the attack once the smart contract has been initialized. 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 the necessary financial resources (*budget*) for Blofeld in Ether (f_B) required for this attack by evaluating the worst case scenario, i.e., the attack runs for N blocks but is still unsuccessful. Note that, N is only known to the attacker.

Necessary attack budget and costs of a failed attack: The budget of the attack contract must cover and compensate all lost rewards for every Bitcoin attack chain block in Ether[5] in case the attack fails. The initial funds of the attacker f_B, as well as the expected reward per Bitcoin block r_b (including fees)[6], define the maximum duration of the attack N in terms of attack chain blocks

[5]For simplicity we assume a fixed exchange rate between cryptocurrencies.
[6]In a concrete attack r_b is not constant, but given by the coinbase output values of every submitted block.

that can be compensated:

$$N = \left\lfloor \frac{f_B - c_{operational}}{r_b} \right\rfloor \tag{1}$$

$$f_B = c_{fail} = N \cdot r_b + c_{operational} \tag{2}$$

Thereby, $c_{operational}$ 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 negligible given the measurements in [23] and Appendix G.1. 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 including fees: $r_b = 14$ BTC, yielding ≈ 84 BTC as a lower bound for the budget.

Costs of a successful attack: Interestingly, the lower bound of the budget is only required to cover potential losses if the attack fails, but if the attack is successful the attacker earns the block rewards on the main chain, thereby compensating the rewards he has to pay to the bribed miners. The costs for a successful attack are thus given by $N \cdot r_b$ main chain blocks, whereas rewards must be paid for $N \cdot (r_b + \epsilon)$ block templates:

$$\begin{aligned} c_{success} &= N \cdot (r_b + \epsilon) + c_{operational} - (N \cdot r_b) && (3)\\ &= N \cdot \epsilon + c_{operational} && (4) \end{aligned}$$

Since we assume rational miners, the attack in this scenario is always successful iff $\epsilon > 0$. For a successful attack to be profitable, the amount gained from ordering or transaction withholding v_a must exceed $c_{success}$.

At a first glance, given that the attacker must pay collaborating miners regardless of the outcome of the attack, one may assume that the costs faced by the attacker are high compared to other bribing schemes. However, this ensures miners face *no risk* from participation – requiring only a *low bribe value* ϵ to incentivize sufficient participation for a successful attack, contrary to existing bribing attacks.

6.2.2 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[7].

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 two blocks for the attack chain to become the longest chain – which altruistic miners will then follow. Hence, the depth of the necessary fork is equal to 1.

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

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

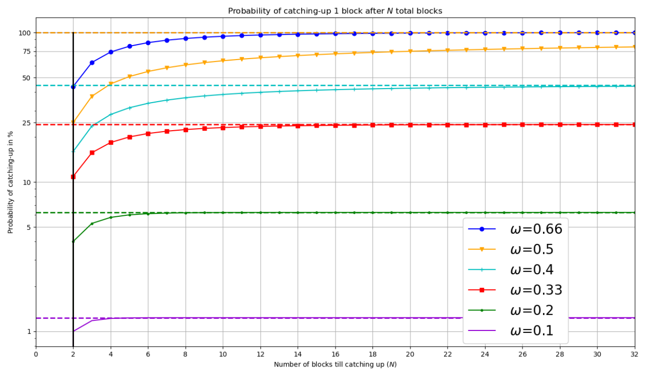

Figure 2: The probability of catching up one block on the y-axis (log scale) within N blocks on the x-axis for different hashrates ω. The dashed line is the maximum probability to catch-up one block after an unlimited number ($N = \infty$) of blocks i.e., $(\frac{\omega}{\beta})^2$.

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 $k + i + 1$ blocks are added to the attack chain. The probability for a such series is:

$$\omega^{k+i+1} \cdot \beta^i \tag{5}$$

Observe a series of a successful attack with i blocks added to the main chain and $k + i + 1$ blocks added to the attack chain. For any prefix strictly shorter than the whole series, the number of appended blocks to the attack chain is smaller than $k + 1$, as otherwise the attack would have ended sooner. It follows that the last block in a successful attack is always appended to the attack chain.The number of combinations for a such series is derived similarly to the Catalan number, with a difference of k for the starting point:

$$\left(\binom{k+2i}{i} - \binom{k+2i}{i-1}\right) \tag{6}$$

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

$$\sum_{i=0}^{i \leq N-k-1} \left(\binom{k+2i}{i} - \binom{k+2i}{i-1}\right) \cdot \omega^{k+i+1} \cdot \beta^i \tag{7}$$

Figure 2 outlines the probability of catching up one block for different hash rates of ω. It can be observed that N quickly approaches the maximum achievable probability of catching up one block within an unlimited number of blocks i.e., $(\frac{\omega}{\beta})^2$ according to [25, 28]. Based on these calculations, the attacker can decide whether or not to extend the attack period and increase N to win an ongoing race with a higher probability.

6.3 Properties and analysis

We now analyze important properties of this approach.

Counter attack: The most effective counter measure against transaction exclusion is to increase the fee of tx_V such that it surpasses the value promised by the attack contract. The benefit of this attack though, is that it can be executed out-of-band. Thereby, the attack is rendered more stealthy to victims, who may 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 the funding cryptocurrency.

Transaction censorship DoS:. In this section we consider Bitcoin as a target, however in principle the attack is also applicable to other types of cryptocurrencies. (Quasi) Turing complete smart contract capable cryptocurrencies are arguably more resistant to censorship than Bitcoin's more limited UTXO (unspent transaction output) model, as they allow for complex and diverse interaction patterns that can trigger state changes.

We assume, for the remainder of this discussion, that transaction censorship should take place within Ethereum as a target cryptocurrency. Then, even if transactions or their respective side effects can be accurately identified and agreed upon all miners as unwanted behaviour, there exists the possibility of denial-of-service attacks that can be launched by the victim in such a case. The effects of a 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 (as described in this section), 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, calls to known contracts such as ERC20 Tokens etc.

Liveness: In general, 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. The likelihood that attacks are performed on starving chain relays is dependent on the funds at stake.

In our concrete example 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. 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 is only payed if the attack is successful, this further incentivizes rational miners to publish solutions timely. Moreover, in this scenario the attacker can, at any stage, cease publishing new block templates to reduce his losses in case the attack appears likely to fail.

7 TRANSACTION REVISION OUT-OF-BAND

The purpose of this attack is to bribe miners into creating blocks on the blockchain branch of a *target* cryptocurrency favored by an attacker, who executes a double-spend. The novelty of the attack stems from three aspects: (i) The funds used for rewarding dishonest behaviour and collaboration during a double-spending attack are paid on a *funding cryptocurrency* and not the target cryptocurrency itself. (ii) By utilizing a smart contract on the funding cryptocurrency as a platform for our attack, we are able to minimize trust assumptions for the attacker, as well as the risk for miners joining the attack: collaborating miners do not have to trust the attacker to receive their bribes. Instead, the attack smart contract ensures bribed miners receive their payments, even if the attack fails, making the attack cheaper than comparable bribing attacks. (iii) Additionally, the usage of smart contracts also opens up the possibility to crowdfund and/or combine multiple double-spending attempts into a single coordinated attack, which further reduces the costs or participants.

7.1 Description

Figure 3 shows the stages and two different outcomes of the attack. **Initialization phase.** First the attacker (Blofeld) creates the uninitialized attack contract and publishes it on the Ethereum blockchain. This is done with a *deploy* transaction included in some Ethereum block e_0 from an Ethereum account controlled by the attacker. Then, Blofeld creates a conflicting pair of Bitcoin transactions. The spending transaction tx_B is published on the main chain in Bitcoin immediately, and the double-spending transaction tx'_B is kept secret. After the confirmation period of k blocks, defined by the victim, has passed on the Bitcoin main chain, Blofeld releases an initialization transaction which irrevocably 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 has been published.

The contract is initialized with $k + 1$ new Bitcoin block templates, each carrying the transactions from the original chain to collect the fee, but instead of tx_A 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. The attacker 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

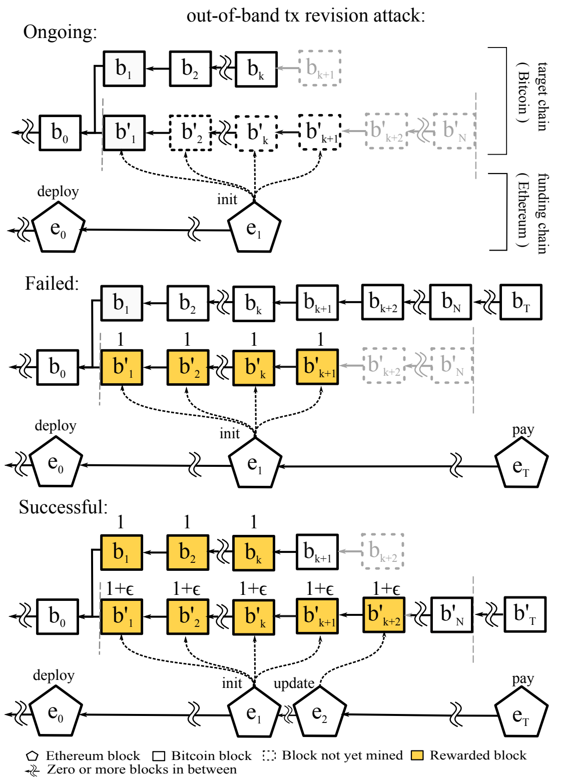

Figure 3: Blockchain structure and example timeline of a tx revision attack with out-of-band payment. The attack contract can be deployed before the actual attack starts. After k blocks on the target chain have passed, the attack contract is initialized with $k + 1$ block templates. The double-spend 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) or with an additional $1 + \epsilon$ if the attack was successful. If the attack succeeds, the first k 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.

attacker wants to get his blocks included in the Bitcoin main chain to receive the Bitcoin block rewards.

It is also possible to deploy and initialize the attack contract at the same time, but publishing an uninitialized attack contract upfront in a *deploy* transaction has the advantage that an attack on the *target chain* could also be crowdfunded before it even has started (see below). In any case, it is important that the double-spend transaction tx'_B is disclosed after block b_k on the main chain, as otherwise Alice may recognize the double-spending attack and refuse to release the goods.

Attack phase: Bribed miners now proceed to mine $k + 1$ 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 $k + 2$ 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 which 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 block b'_x 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 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 distinguishes between the two possible outcomes:

- *Attack fails (Main chain wins).* In this case the contract must fully compensate the bribed miners for their attack chain blocks, 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.
- *Attack succeeds (Attack chain wins).* If the attack chain wins, then the contract executes the following actions: 1) Fully (reward + fees = 1) compensate the miners of k 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+2} in our example, the full block reward plus an additional ϵ as a bribe.

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 b_T is known. This ensures that every participant had enough time to submit information about the longest Bitcoin chain to the contract and that the blocks b_1 to b_N have received sufficient confirmations according to an acceptance policy logarithmic in the chain's length as specified in [30]. 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 [23]. For the first k 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 the then recompute the corresponding Ethereum addresses.

Crowdfunding. The attack described above also opens up the possibility to be crowdfunded. The simplest crowdfunding approach would be to allow the donation of funds after the attack contract has been deployed, but before it is initialized. This methods 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 the collaborators. The main challenges that have to be solved in such a scenario are as follows:

- It has to be ensured that every collaborating attacker, who invests funds to achieve a double-spend attack indeed 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 a participant to cause the attack to fail.
- The attack should not rely on any trusted third party

The details of how such an attack can be constructed on Ethereum as a funding cryptocurrency and Bitcoin as a target cryptocurrency are given in the appendix F. 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 which will be affected. Then, all Bitcoin users which have performed transactions in block b_1 can decide if to invest into the attack to potentially double-spend their transaction. The collaborating attackers, further referred to as *backers*, submit the new transaction to the contract together with some ether to increase the overall funds f_B of the attack. 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 $k + 1$ blocks has been reached, the attack starts as previously described. Since the attacker who initialized the contract has to take care of producing the new blocks for the chain containing the double-spend transactions, some method has to be implemented that the transactions of other attackers are assured to get included in b'_1. In the appendix F we describe a method which requires a collateral from the original attacker as high as the funds he wants to collect i.e., f_B. Thereby, it can be ensured that the other attackers only pay if their transaction was really included in the new chain in block b'_1 – which can be proven to a smart contract. Otherwise they are refunded from the collateral submitted by the initial attacker.

7.2 Evaluation

Analogous to our evaluation in section 6 we now proceed to evaluate the success probability, as well as the costs incurred in the attack. We again distinguish between two cases: (i) we evaluate the attack under the assumption that only rational miners exist and then (ii) relieve this assumption and consider a scenario with altruistic miners.

7.2.1 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 6 when adjusting for k as the security parameter defined by the victim of the attack.

Necessary attack budget and costs of a failed attack: The minimum number of blocks on the attack chain in a successful attack is $k + 1$ i.e., the number of confirmations required on the main chain, plus one to exceed the main chain's length. A lower bound for the budget of the attacker f_B can thus be derived due to the condition $N \geq k + 1$ which has to hold for an attack to be feasible. For Bitcoin, a common choice of $k = 6$ and the current block reward, including fees, is approximately $r_b = 14$ providing a lower bound for the budget of ≈ 98 BTC s.t. the following inequality holds:

$$\frac{f_B}{r_b} > k \tag{8}$$

Costs and profitability of a successful attack: Again, the lower bound of the budget is only required to cover potential losses if the attack fails. However, if it is successful the attack it is cheaper than this lower bound. The costs for a successful attack are given by the $k \cdot r_b$ main chain blocks that have to be compensated on the attack chain plus the additional $N \cdot \epsilon$ bribes.

$$c_{success} = k \cdot r_b + N \cdot \epsilon + c_{operational} \tag{9}$$

The initial k 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 $N \geq k+1$ holds and $\epsilon > 0$. For Bitcoin, this means that the costs of a successful double spend with $k = 6$ and $r_b = 14$ and $\epsilon = 0.0001$ are ≈ 84.0007 BTC. For a successful attack to be profitable, the value of the double-spend v_d has to be greater than the value used for bribing. In Bitcoin transactions carrying more than 84 BTC are observed regularly[8]. This further highlights the dependence of transaction volume to confirmation time as also stated in [30].

7.2.2 Evaluation with Altruistic Miners $\omega + \beta = 1$: Figure 4 shows the attack success probability of the attack for different values of the portion of the hash rate which can be bribed ω, as well as different amounts of blocks N these bribed miners can be rewarded or compensated for. The number of confirmation blocks on the main chain after which the attack starts is set to $k = 6$. Clearly, the attack requires $N > k$ to have a chance of being successful. As with the classical 51% attack, the attack eventually succeeds once the bribable hash rate is above the 50% threshold and the number of payable blocks N grows.

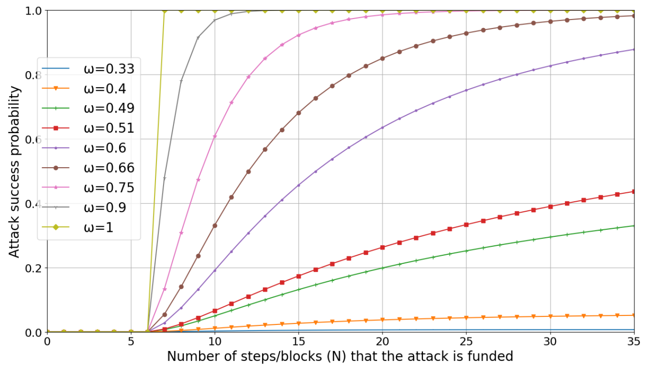

Figure 4: Attack success probability of a double-spending attack depending on the amount of blocks that can be compensated/rewarded N in a bribing attack. Different amounts for the hash rate ω of bribed miners are given. The number of confirmation blocks k is set to 6.

Given these probabilities we can calculate the required number of blocks N we need to reach a success probability of 99.4% given

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

values for k and ω. Table 2 shows a comparison against the whale attack described in [21]. It can be observed that in contrast to the whale attack, our attack becomes cheaper when ω grows large since we faster reach the required probability and therefore have to pay less bribes. Moreover, the costs of our attack if it succeeds are lower than the budget (f_B) required to be available at the beginning of the attack to compensate all bribed miners if the attack would fail. Thereby, we are between $\approx 85\%$ and $\approx 95\%$ cheaper than the whale attack.

ω	whale costs	P2W budget f_B	% whale	P2W costs $c_{success}$	% whale	N
0.532	2.93e+23	7305	0.00	577	0.00	487
0.670	999.79	600	60.01	130	13.00	40
0.764	768.09	330	42.96	112	14.58	22
0.828	1265.14	240	18.97	106	8.38	16
0.887	1205.00	195	16.18	103	8.55	13
0.931	1806.67	165	9.13	101	5.59	11
0.968	2178.58	135	6.20	99	4.54	9
0.999	2598.64	120	4.62	98	3.77	8

Table 2: Comparison of attack costs of the whale attack [21] against our attack for $k = 6$ and a success probability of 99.4% (due to floating point precision). For the calculations we set $r_b = 15$ and defined the additional bribing value $\epsilon = 1$ to account for all operational costs. Costs and budget in BTC.

7.2.3 Available funds: With the possibility to crowdfund attacks, theoretically also double-spends of low value transactions could 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[9] the median value of bitcoins transacted per day (excluding change addresses) is approximately 1 billion USD, whereas the median mining reward per day including transactions fees is approximately 15 million USD.

7.3 Properties and Analysis

The properties of this attack are comparable to the previous out-of-band attack described in section 6. Additionally, the following aspects require some more discussion in context of this attack.

Counter bribing: As already outlined in the previous sections, counter bribing is a viable strategy for the victim against incentive attacks. This also illustrates an important aspect of incentive attacks, namely their visibility. On the one hand, miners of the *target cryptocurrency* have to recognize that an attack is going on, otherwise they wont be able to join it to receive bribes and the attack would 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 incentive attacks were to arise if all rational miners have been informed directly about the attack, while all victims/merchants are not miners themselves and do not monitor all possible funding cryptocurrencies to check if an attack is going on.

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

Cost optimization: The biggest cost driver in the proposed attack is the compensation of k main chain blocks to provide an incentive for all rational miners to switch to the attack chain. In a blockchain with where every block is uniquely attributable to a set of known miners, and where also the overall hashrate of those miners can be approximated, the payout of compensations can be further optimized in various ways. Consider for example a scenario where a small miner, compared to the other miners, is lucky and mines several block within k. Then the attacker can exclude this miner from the payout of compensations since it is unlikely that he will substantially contribute to the attack chain.

8 DISCUSSION

Through our comprehensive systematization of incentive attacks we are able to highlight unconsidered attack types and present new and improved techniques that address several of these open questions.

Hereby, the body of research on incentive attacks demonstrates that not only the hash rate distribution among permissionless PoW based cryptocurrencies plays a central role in defining their underlying security guarantees. The ratio of *rational* miners and available funds for performing bribes also form a key component that demands further study. Our out-of-band attacks also help to highlight that being able to cryptographically interlink cryptocurrencies increases their attack surface.

In our quest for devising *trustless* out-of-band attacks an interesting analogy is also revealed: At an abstract level,the presented attacks require a construction not unlike a mining pool, where the pool owner defines rules for block creation within a smart contract. Moreover, every participant must be able to claim his promised funds in a trustless fashion, based on the submitted blocks and the state of the targeted cryptocurrency. Our *ephemeral mining relay* provides exactly this functionality. Luu et al. [22] also proposes a mining pool (Smart pool) which itself is managed by a smart contract. However, its design and potential applications did not consider use-cases with malicious intent. Smart pool does not enforce any properties regarding the content and validity of a block beyond a valid PoW, as the intrinsic incentive among participants is assumed to collect the respective reward in the mined cryptocurrency, which is only possible if valid blocks have been created.

Smart contract based incentive attacks also introduce the possibility of trustless *crowdfunding* and alignment of interests for multiple attackers, who want to perform double-spends during the same time period. Together with the topic of counter bribing, new research directions are shown that raise fundamental questions on the incentive compatibility of Nakamoto consensus.

An interesting topic that was left unexplored is the idea of employing incentive attack techniques to encourage *desirable* rather than malicious behavior in mining entities and protocol participants, e.g. for quickly reaching a majority during protocol upgrade phases.

9 CONCLUSION

The systematization of incentive 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 three new trustless incentive attacks that achieve new characteristics and are rendered cheaper than previous approaches.

Our contributions underline that incentive attacks on cryptocurrencies remain an open and highly relevant research topic with a variety of unexplored areas.

All previously proposed, and in-the-wild observed incentive attacks, as well as the attacks described in this paper, indicate that the security properties of permissionless PoW based cryptocurrencies are not accurately reflected by assuming only honest and Byzantine actors. 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 systems.

Further game theoretic modeling and analysis of incentive attacks as well as their complex cross-chain interactions is necessary to more accurately assess risks and security guarantees.

REFERENCES

[1] [n.d.]. Average Number Of Transactions Per Block. https://www.blockchain.com/en/charts/n-transactions-per-block. Accessed 2019-05-10.

[2] [n.d.]. BTC Relay. https://github.com/ethereum/btcrelay. Accessed 2018-04-17.

[3] [n.d.]. CoinMarketCap: Cryptocurrency Market Capitalizations. https://coinmarketcap.com/. Accessed 2019-05-10.

[4] [n.d.]. Replace by Fee. https://en.bitcoin.it/wiki/Replace_by_fee. Accessed 2019-05-11.

[5] Amitanand S Aiyer, Lorenzo Alvisi, Allen Clement, Mike Dahlin, Jean-Philippe Martin, and Carl Porth. 2005. BAR fault tolerance for cooperative services. In *ACM SIGOPS operating systems review*, Vol. 39. ACM, 45–58. http://www.dcc.fc.up.pt/~lnes/aulas/1314/SDM/papers/BAR%20Fault%20Tolerance%20for%20Cooperative%20Services%20-%20UIUC.pdf

[6] Adam Back, Matt Corallo, Luke Dashjr, Mark Friedenbach, Gregory Maxwell, Andrew Miller, Andrew Poelstra, Jorge Timón, and Pieter Wuille. 2014. Enabling blockchain innovations with pegged sidechains. http://newspaper23.com/ripped/2014/11/http-_____-___-_www___-blockstream___-com__-_sidechains.pdf Accessed: 2016-07-05.

[7] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. 2012. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. ACM, 326–349.

[8] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. 2013. Recursive composition and bootstrapping for SNARKs and proof-carrying data. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. ACM, 111–120.

[9] Joseph Bonneau. 2016. Why buy when you can rent? Bribery attacks on Bitcoin consensus. In *BITCOIN '16: Proceedings of the 3rd Workshop on Bitcoin and Blockchain Research*. http://fc16.ifca.ai/bitcoin/papers/Bon16b.pdf

[10] Joseph Bonneau. 2018. Hostile blockchain takeovers (short paper). In *5th Workshop on Bitcoin and Blockchain Research, Financial Cryptography and Data Security 18 (FC). Springer*. http://fc18.ifca.ai/bitcoin/papers/bitcoin18-final17.pdf

[11] Vitalik Buterin. 2016. Chain Interoperability. https://static1.squarespace.com/static/55f73743e4b051cfcc0b02cf/t/5886800ecd0f68de303349b1/1485209617040/Chain+Interoperability.pdf Accessed: 2017-03-25.

[12] Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. 2019. Flash Boys 2.0: Frontrunning, Transaction Reordering, and Consensus Instability in Decentralized Exchanges. arXiv preprint arXiv:1904.05234. https://arxiv.org/pdf/1904.05234.pdf

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

[14] Shayan Eskandari, Seyedehmahsa Moosavi, and Jeremy Clark. 2019. SoK: Transparent Dishonesty: front-running attacks on Blockchain. *arXiv preprint arXiv:1902.05164*. https://arxiv.org/pdf/1902.05164.pdf

[15] Uriel Feige, Amos Fiat, and Adi Shamir. 1988. Zero-knowledge proofs of identity. *Journal of cryptology* 1, 2 (1988), 77–94.

[16] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. 2015. The bitcoin backbone protocol: Analysis and applications. In *Advances in Cryptology-EUROCRYPT 2015*. Springer, 281–310. http://courses.cs.washington.edu/courses/cse454/15wi/papers/bitcoin-765.pdf

[17] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. 2016. The Bitcoin Backbone Protocol with Chains of Variable Difficulty. http://eprint.iacr.org/2016/1048.pdf Accessed: 2017-02-06.

[18] Aljosha Judmayer, Nicholas Stifter, Philipp Schindler, and Edgar Weippl. 2018. 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*. https://www.sba-research.org/wp-content/uploads/2018/09/judmayer2018pitchfork_2018-09-05.pdf

[19] Aashish Kolluri, Ivica Nikolic, Ilya Sergey, Aquinas Hobor, and Prateek Saxena. 2018. Exploiting The Laws of Order in Smart Contracts. arXiv:1810.11605. https://arxiv.org/pdf/1810.11605.pdf

[20] Harry C Li, Allen Clement, Edmund L Wong, Jeff Napper, Indrajit Roy, Lorenzo Alvisi, and Michael Dahlin. 2006. BAR gossip. In *Proceedings of the 7th symposium on Operating systems design and implementation*. USENIX Association, 191–204. http://www.cs.utexas.edu/users/dahlin/papers/bar-gossip-apr-2006.pdf

[21] Kevin Liao and Jonathan Katz. 2017. Incentivizing blockchain forks via whale transactions. In *International Conference on Financial Cryptography and Data Security*. Springer, 264–279. http://www.cs.umd.edu/~jkatz/papers/whale-txs.pdf

[22] Loi Luu, Yaron Velner, Jason Teutsch, and Prateek Saxena. 2017. SMART POOL : Practical Decentralized Pooled Mining. http://eprint.iacr.org/2017/019.pdf Accessed: 2017-03-22.

[23] Patrick McCorry, Alexander Hicks, and Sarah Meiklejohn. 2018. Smart Contracts for Bribing Miners. In *5th Workshop on Bitcoin and Blockchain Research, Financial Cryptography and Data Security 18 (FC). Springer*. http://fc18.ifca.ai/bitcoin/papers/bitcoin18-final14.pdf

[24] Andrew Miller, Iddo Bentov, Ranjit Kumaresan, and Patrick McCorry. 2017. Sprites: Payment Channels that Go Faster than Lightning. https://arxiv.org/pdf/1702.05812.pdf Accessed: 2017-03-22.

[25] Satoshi Nakamoto. 2008. Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf Accessed: 2015-07-01.

[26] Rafael Pass, Lior Seeman, and abhi shelat. 2016. Analysis of the Blockchain Protocol in Asynchronous Networks. http://eprint.iacr.org/2016/454.pdf Accessed: 2016-08-01.

[27] Joseph Poon and Thaddeus Dryja. 2016. The bitcoin lightning network. https://lightning.network/lightning-network-paper.pdf Accessed: 2016-07-07.

[28] M. Rosenfeld. 2014. Analysis of Hashrate-Based Double Spending. https://arxiv.org/pdf/1402.2009.pdf Accessed: 2016-03-09.

[29] Ilya Sergey, Amrit Kumar, and Aquinas Hobor. 2018. 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*. 323–338. https://ilyasergey.net/papers/temporal-isola18.pdf

[30] Yonatan Sompolinsky and Aviv Zohar. 2016. Bitcoin's Security Model Revisited. http://arxiv.org/pdf/1605.09193.pdf Accessed: 2016-07-04.

[31] Nicholas Stifter, Aljosha Judmayer, Philipp Schindler, Alexei Zamyatin, and Edgar Weippl. 2018. Agreement with Satoshi - On the Formalization of Nakamoto Consensus. Cryptology ePrint Archive, Report 2018/400. https://eprint.iacr.org/2018/400.pdf

[32] Jason Teutsch, Sanjay Jain, and Prateek Saxena. 2016. When cryptocurrencies mine their own business. In *Financial Cryptography and Data Security (FC 2016)*. https://www.comp.nus.edu.sg/~prateeks/papers/38Attack.pdf

[33] Itay Tsabary and Ittay Eyal. 2018. The gap game. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 713–728. https://arxiv.org/pdf/1805.05288.pdf

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

[35] Fredrik Winzer, Benjamin Herd, and Sebastian Faust. 2019. Temporary Censorship Attacks in the Presence of Rational Miners. Cryptology ePrint Archive, Report 2019/748. https://eprint.iacr.org/2019/748

[36] Alexei Zamyatin, Dominik Harz, Joshua Lind, Panayiotis Panayiotou, Arthur Gervais, and William J. Knottenbelt. 2018. XCLAIM: Trustless, Interoperable Cryptocurrency-Backed Assets. Cryptology ePrint Archive, Report 2018/643. https://eprint.iacr.org/2018/643.pdf https://eprint.iacr.org/2018/643.

A VARIABLES AND SYMBOLS

Used variables and symbols in accordance to [34].

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	Number of confirmation blocks till block is considered as confirmed by the actor which depends on the respective scenario. This could be either the attacker or the victim.
l	The length of the attacker chain since the block causing the fork.
N	Maximum length of the attack chain during the attack.
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.

Symbol	Description
B	The attacker that wants to execute the double-spending attack
V	The victim or merchant, e.g., the actor who would lose money if the double-spending attack is successful
$B_1, B_2, ..., B_x$	Addresses/accounts that are under the control of the attacker
$V_1, V_2, ..., V_x$	Address/accounts that is under the control of the victim
$tx_V, tx_B, tx'_B,$	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	Initial funds of the attacker
r_e, r_b	Funds equivalent to one block reward (including fees)
ϵ	Additional reward paid for a block on the attack chain. The total reward for a block on the attack chain received by a bribed miner hence is $r_b + \epsilon$
ρ	Profit of the attacker
$v, v_d, ...$	Value, e.g., value of the double-spend transaction
$c_{success}$	Total costs of a successful pay-2-win attack
c_{budget}	Total costs of a failed pay-2-win attack, must be equal to f_A.
$c_{operational}$	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 5: Variables and symbols related to actors and costs.

B 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.

B.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 transaction can easily be proven to a smart contract in retrospect.

B.1.1 Properties and analysis: The advantage of this approach is that it is easy to implement but it has some drawbacks. First, the transaction hashes of the involved target transactions tx_V might change 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 is not much of an issue since the original transaction tx_V is still valid an can be included by a complacent rational miner.

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 is not covered in the attack condition of the contract. Thus, a naive contract only working with transaction hashes 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.

B.2 Verify operation on certain state

This approach addresses the issue of interfering transactions mentioned in the previous section. Thereby, two general approaches exists: i) *Runtime check:* 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. ii) *Retrospective check:* It is proven to the attack contract in retrospect hat it has successfully operated on the correct workd-/smart contract state before any funds are unlocked.

B.2.1 Retrospective check: Up to Ethereum EIP-150 revision the *transaction receipt* also contained the *post-transaction* state R_σ. [10] 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 generic method for Ethereum around this would be to require that the racing attack transaction has to be a 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 therefore, less generic but more efficient solutions are preferable.

B.2.2 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 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. For example the name is not registered yet. 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. This allows to write customized proxy contracts which are tailored to check the precondition of the attack, e.g., the contract to exploit is still in the desired state. This can for example be achieved by reading publicly accessible variables – for which getter functions are created automatically – or a contracts balance. This does not change the incentives for the miner of the respective block, but additionally ensures that no interaction happens if the race is not won i.e,. the attack is not successful.

Summarizing, it can be said if *runtime checks* are possible, the attack becomes more efficient and more complex attack scenarios can be envisioned.

C 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-/no-fork* attack is it to exclude one or multiple unconfirmed transactions from their generated blocks.

C.1 Description

Initialization. The attacker knows some transaction tx_V which he wants to prevent from getting into the main chain. He then intializes the attack contract at block $e - 1$, specifying the transaction and the duration N (in blocks) of the exclusion attack.
Attack. The attack contract will pay an extra ϵ for every block mined between block e_1 and e_N that (i) does not include transaction tx_V itself and (ii) does not extend any block that included transaction tx_V. That is, if an altruistic miner decides to include tx_V in his block e_i ($i < N$), colluding miners must perform a 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 blocks have passed after the end of the attack, i.e., at a block $e_T \geq e_{N+k}$, where k is the security parameter of the blockchain. 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 Appendix D.

C.2 Evaluation

C.2.1 Evaluation with Rational Miners Only ($\omega = 1$): Estimating the costs of such an attack in an scenario where all miners are

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

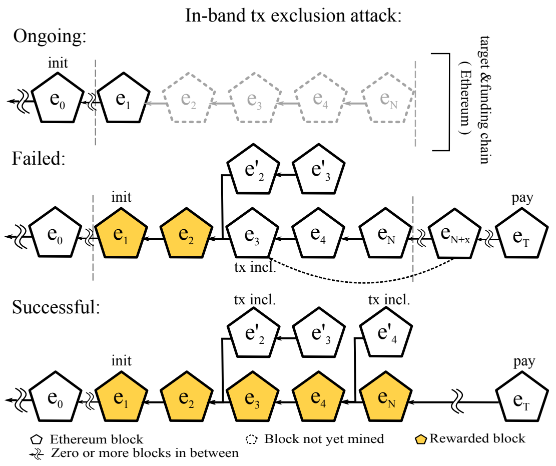

Figure 5: The figure shows a ongoing-, a failed- as well as a successful Transaction exclusion attack with in-band payments. The attack is initialized when the attack contract is published in block e_1. If the unwanted transaction has been included, this can be proven to the attack contract as shown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

rational ($\alpha = \beta = 0$ and $\omega = 1$) and have perfect information about the attack is trivial. In this case, the respective transaction would not be included into the block chain as long as the bribe ϵ for non-inclusion surpasses the fee miners can gain from including transaction, i.e., $\epsilon > fee(tx_V)$.

C.2.2 Evaluation with Altruistic Miners $\omega + \beta = 1$: If a fraction of miners behaves altruistically, i.e., will not joining the attack independent of profit, rational miners need additional incentive to perform *near forks*, excluding blocks containing tx_V. 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 6 consecutive blocks on merely 8.3%. Hence, 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 [21] (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:

$$\rho = \frac{\left(1 - \left(\frac{\alpha+\omega_\alpha}{\beta+\omega_m}\right)^{z+1}\right) \cdot \omega_m}{\beta + \omega_m} \tag{10}$$

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:

$$\rho' = \frac{\left(\frac{\alpha+\omega_\alpha+\omega_m}{\beta}\right)^{z+1} \cdot \omega_m}{\alpha + \omega_\alpha + \omega_m} \cdot (\epsilon + 1) \tag{11}$$

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

$$\epsilon > \frac{\left(1 - \left(\frac{\alpha+\omega_\alpha}{\beta+\omega_m}\right)^{z+1}\right)}{\beta + \omega_m} \cdot \frac{\alpha + \omega_\alpha + \omega_m}{\left(\frac{\alpha+\omega_\alpha+\omega_m}{\beta}\right)^2} - 1 \tag{12}$$

To estimate a worst case lower upper 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 $\epsilon \approx 17 \cdot r_e$, i.e., if a rational miner m assumes no other miners will join the attack, a bribe 17 times the value of a block reward is necessary. We provide a detailed overview of necessary bribing values ϵ for different attack constellations (ω_α and ω_m) in Table 6 in Appendix D. 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 $\epsilon = 0$).

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

C.2.3 Comparison to Existing Attacks. A comparable attack allowing arbitrary transaction exclusion is `HistoryRevisionCon` [23]. 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 $\epsilon \approx 0.603 \cdot r_e$.

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.

C.3 Properties and analysis

We now present some practical issues of the given straw man protocol with their respective fixtures if available in this scenario.

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

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

Table 6: 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.

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 must can but 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.

Counter attack. 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 next section 6

Proof a negative. Since we are in an in-band scenario, the successful execution of the attack relies on a proof that transaction tx_V was included to correctly pay out rewards and detect unwanted inclusion. In 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 some additional 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 ensures 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. A method how such a construction can be implemented in current Ethereum is given in D.

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. Thereby, 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 a more powerful out-of-band attack which uses the technique of *block templates* and also allows for arbitrary ordering in section 6.

D 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. Additionally, we provide details on the costs of in-band transaction exclusion, in the classical bribing model defined in [21].

D.1 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.

D.2 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 in B 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 owne account directly, e.g., as required by certain Tokens. [11]

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 have 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.

D.2.1 Block template in-band. Another way around the previously outlined problems 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.

One way around this is to have a delay of one block, and transactions in the peer to peer network which define the content of blocks. For example, as soon as the attack starts the attacker publishes the attack contract tx_{B_1} which gets initialized with the current block height. Additionally the attacker publishes a singed transaction tx_{B_2} to the attack contract that defines the content of block e_1, i.e., the Merkle Patricia Trie root of block e_1. Since tx_{B_2} cannot define itself, complacent miners which want to collect the rewards have to include tx_{B_2} in e_2. The transaction tx_{B_2} invokes the attack contracts, checks that is had been included in e_2 and stores the intended Merkle Patricia Trie root of block e_1. When the attack stops, anyone can prove to the contract that the current chain indeed only contains specified Merkle Patricia Trie roots by showing the respective headers to the contract within 256 blocks.

E DETAILS AND IMPLEMENTATION OF TX EXCLUSION AND ORDERING OUT-OF-BAND

E.1 Block templates and block intervals

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. Though, 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,

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

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.

E.2 Block Interval Desynchronization

To identify the need for such constructions 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 $E_{BTC}(x)$ of 15 seconds, whereas Bitcoin has a mean block interval of $10 \cdot 60$ seconds.

$$E_{ETH}(x) = 15$$
$$E_{BTC}(x) = 600$$

What is the probability that the time between two Bitcoin blocks is less than the Ethereum mean block interval of 15 seconds?

$$\begin{aligned} x &= 15 \\ \lambda &= \frac{1}{E_{BTC}(x)} \\ P(X < x) &= 1 - e^{-\lambda \cdot x} \\ P(X < 15) &\approx 2.47\% \end{aligned}$$

What is the probability that this happens within N Bitcoin blocks i.e., what is the probability that the time between two Bitcoin blocks is smaller than 15 seconds during N total Bitcoin blocks.

$$\begin{aligned} P(N) &= 1 - \left(1 - P(X < 15)\right)^{N-1} \\ P(32) &\approx 53.93\% \end{aligned}$$

This necessitates that in long running attacks, more than one block template for target cryptocurrency blocks is included in one funding cryptocurrency block.

We now analyze how probable it is that more than two Bitcoin blocks are mined before one Ethereum block, i.e., what is the probability that $3, 4, 5, \ldots$ Bitcoin blocks are found within the Ethereum mean block interval of 15 seconds. The block discovery is a Poisson point process, where the Poisson distribution parameter $\Lambda = E(N = n) = \frac{t}{E_{BTC}(x)}$ refers to the expected value of the number of events happening within t time.

$$\begin{aligned} P(N = n) &= \frac{\Lambda^n}{n!} \cdot e^{-\Lambda} \\ \lambda &= \frac{1}{E_{BTC}(x)} = 1/600 \\ \Lambda &= E(N = n) = t/600 \\ t &= 15 \\ P(N = 1) &\approx 2.43\% \\ P(N = 2) &\approx 0.03\% \\ P(N = 3) &\approx 0.00025\% \\ P(N = 4) &\approx 1.58 \cdot 10^{-6}\% \\ P(N = 4) &\approx 7.94 \cdot 10^{-9}\% \end{aligned}$$

Thus, a lower bound for this event happening within a certain block interval, i.e., during an attack, can be derived as follows:

$$\begin{aligned} n &= 3 \\ P(N) &= 1 - \left(1 - P(N = 3)\right)^{\lceil N/n \rceil} \\ P(32) &\approx 0.486\% \end{aligned}$$

F DETAILS AND IMPLEMENTATION OF TX REVISION EXCLUSION AND ORDERING OUT-OF-BAND

F.1 Trustless crowdfunding of incentive attacks with collateral

One of the main improvements of this attack is the possibility that the required funds can be crowdsourced. The main challenges that have to be solved in this scenario are outlined in section 7. In the following, we describe one approach for *trustlessly* crowdfunding incentive attacks using a collateral. The phases of the attack are as follows:

- The attacker who initiates the attack, in the following referred to as the *initiator*, deploys an attack contract in the funding cryptocurrency and locks his collateral of value f_B (as described in the original attack) with this contract.
- Then the initiator publishes his spending transaction tx'_B on the main network.
- Once k blocks on the main chain have passed, he initializes the attack contract with his double-spend tx'_B, the block to be forked b_1 and the common ancestor block b_0.
- Then everybody who has included a transaction in block b_1 is allowed to submit a double-spending transactions $t'_{B_{\{2,\ldots,x\}}}$ including some amount of ether that he or she is willing to invest in the attack. In the follows these additional attackers are referred to as *backers*.
- If these so called backers reach the funding goal of compensating at least $k + 1$ blocks before block $k + 1$ blocks have been submitted to the attack contract, then the attack starts automatically. All invested funds are free to be used as described in the original attack.

- Once the attack has been started by the attack contract, the initiator publishes a block header template to the attack contract. The Merkle branch of this template includes all submitted double-spending transactions $tx'_{B_{\{2,\ldots,x\}}}$, 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 the initiator 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.
- Then the attack proceeds as originally described.
- When N blocks have passed and been 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 his money if the initiator has not submitted a valid Merkle inclusion prove for the respective double-spending transaction.

In this approach, the attacker who initiates the attack, i.e., the *initiator*, has to provide a collateral as large as the total funds required for a successful attack f_B. If the initiator as well as his fellow backers are honest, the collateral will be released 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 the initiator did not include their double-spending transaction.

Moreover, the initiator is also required to invest funds into the double-spending f_{B_1} attack which will be consumed by the attack as in the original attack. This investment by the initiator 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 also the initiator will lose his invested funds f_{B_1}. Thus, backers are free to invest ether as long as the amounts stays below f_{B_1}.

G EPHEMERAL MINING RELAY

In this section we outline the functionality of the *ephemeral mining relay* used in out-of-band incentive attacks, and provide cost estimates for an implementation on top of Ethereum, which verifies the Bitcoin blockchain.

G.1 Construction

The out-of-band incentive attacks presented in this paper require the deployment of a so-called ephemeral mining relay on the funding cryptocurrency. An ephemeral mining relay[12] is a combination of a chain relay [2, 11, 36] and a decentralized mining pool smart contract [22].

[12]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.

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 [6]). 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. As such, chain relays are vulnerable to so called *poisoning attacks* [36], where a miner tricks the relay into accepting a chain containing invalid (e.g. double-spent) transactions as valid. Note: such an attack is not possible to be performed against any node that is fully validating consensus rules, as it will simply reject the invalid blocks.

To prevent collaborating miners from executing such poisoning attacks (e.g., in an attempt to collect rewards, without actually executing the attack on the target cryptocurrency), we extend its the functionality with additional consensus rule checks. Thereby, a major challenge is efficiently verifying that each transaction in a block submitted to the relay is indeed valid , e.g. does not double spend any transaction in the same blockchain branch.

To fully check the validity of blocks and transactions submitted to a chain relay, we identify the following approaches:

- *Merkle tree templates.* The adversary funding and coordinating incentive attacks can specify the exact structure of the Merkle tree to be included in the blocks mined by collaborating miners. As such, miners can only freely construct the coinbase transaction of a block - which must be separately parsed and verified by the ephemeral chain relay. Recall: the coinbase transaction is the first transaction in a block, which generates new cryptocurrency units and *does not spend any existing transaction outputs.* As such, a coinbase transcation cannot be double spent and can hence fully verified at low cost.
- *Near-empty blocks.* A simpler variant of the Merkle tree templates described above is to restrict the transactions included in blocks generated by collaborating miners to the (small) set of transactions specified by the adversary. While this reduces the verification costs in the ephemeral mining relay, the adversary may have to pay additional compensation to miners for missed out transaction fees.

We note that other viable approaches may exist to verify transaction validity in chain relays, e.g. utilizing non-interactive Zero Knowledge Proofs [7, 8, 15] may represent an interesting avenue for future work.

G.2 Cost Estimates

We implement a minimal viable ephemeral mining relay on Ethereum, which is capable of verifying the state of the Bitcoin blockchain. We use Solidity v0.5.2 and use a local instance of the Ethereum blockchain for cost analysis.

Specifically, we identify five main operations of an ephemeral mining relay:

Operation	**Approx. costs**	
	Gas	**USD**
Initialization	244 137	0.21
Block parsing and verification	174 929	0.15
Block header storage	141 534	0.12
Transaction parsing	117 253	0.1
Markle tree verification	80 257 - 194 351	0.07 - 0.16

Gas price: 5 Gwei, Exchange rates as per 10 May 2019 (168.01 USD/ETH) [3]

Table 7: Overview of costs for the each of the main operations of the ephemeral chain relay implemented on Ethereum for Bitcoin. Note: Merkle tree verification costs depend on the depth of the tree / transactions in a block. Numbers provided are lower and upper bounds.

- *Initialization*, i.e., storing the first block and necessary templates used as basis for the incentive attack, as defined by the attacker.
- *Block parsing and verification*, i.e., checking that (i) the proof-of-work of a block is valid and meets the necessary difficulty target, (ii) the block extends the correct blockchain branch (attacker fork or main chain).
- *Block header storage*, i.e., permanently storing the necessary block header information in the smart contract, to be used for later verification (e.g. during payouts). Note: it is not necessary to store all transaction included in a block, but only the block header (e.g., 80 bytes in Bitcoin) which contains the root of the transaction Merkle tree.
- *Transaction parsing*, i.e., parsing and verifying the inputs and outputs of a transaction and extracting any additional data (e.g. funding cryptocurrency address to be used for payout of collaborating miners).
- *Merkle tree verification*, i.e., verifying that a given transaction (more specifically, its hash) is included in the Merkle tree of a block at a specific position. Verification of Merkle tree templates follows a similar principle, as it is only necessary to check the inclusion of sub-trees (i.e., check that the root of the attacker's Merkle tree template is included at the correct position in the Merkle tree of the block).

The cost estimates for the above operations are summarized in Table 7. Note: the costs for Merkle tree verification may vary, depending on the depth of the Merkle tree. However, the increase in costs is marginal, as each additional layer merely requires an additional hashing operation (i.e., costs grow in $O(log(n)$, where n is the depth of the Merkle tree). The worst case costs per Bitcoin block in 2018/2019 ($<$ 2048 transactions/block) [1] amount to approximately USD 0.16.

We observe the costs for maintaining a ephemeral mining relay are marginal and *negligible when compared to the potential scale of incentive attacks* described in this paper. As such, in an exaggerated case where an out-of-band attack on Bitcoin via a relay on Ethereum in maintained for 24 hours (144 Bitcoin blocks on average), the costs merely amount to approximately USD 10 in the best and USD 23 in the worst case (if all blocks are full), i.e., between USD 0.4 and USD 1 per hour. For comparison: the reward for a single Bitcoin block (*excluding* transaction fees) at the time of writing amounts to USD 76 875.