Smart Contracts for Bribing Miners

Patrick McCorry, Alexander Hicks, and Sarah Meiklejohn

University College London
{p.mccorry,alexander.hicks.16,s.meiklejohn}@ucl.ac.uk

Abstract. We present three smart contracts that allow a briber to fairly exchange bribes to miners who pursue a mining strategy benefiting the briber. The first contract, `CensorshipCon`, highlights that Ethereum's uncle block reward policy can directly subsidise the cost of bribing miners. The second contract, `HistoryRevisionCon`, rewards miners via an in-band payment for reversing transactions or enforcing a new state of another contract. The third contract, `GoldfingerCon`, rewards miners in one cryptocurrency for reducing the utility of another cryptocurrency. This work is motivated by the need to understand the extent to which smart contracts can impact the incentive mechanisms involved in Nakamoto-style consensus protocols.

1 Introduction

Cryptocurrencies such as Bitcoin and Ethereum have collectively achieved a market capitalisation of over $600bn in January 2018. The success of cryptocurrencies relies on an append-only public ledger called the blockchain, and on Nakamoto consensus, a mechanism to reward honest participants (miners) for updating the blockchain. The consensus protocol is designed with the idea of "one-cpu-one-vote" as miners compete to solve a computationally difficult cryptographic puzzle. The first miner to present a valid solution wins the authority to append his block containing a list of recent transactions to the blockchain. Thus, the security and reliability of the blockchain is dependent on the assumption that a majority of the network's computational power is honest. If not, an adversary is able to control the content of the blockchain.

Since the introduction of Bitcoin in 2009, the mining process has changed drastically, with advances in graphic processing units (GPUs), field-programmable gate arrays (FPGAs) and application-specific integrated circuits (ASICs) offering much greater performance than a single CPU. Thus, today's miners must invest in expensive hardware before competing meaningfully in the consensus protocol. Similarly, pooled mining allows a single appointed pool master to decide which transactions to include in a block and how to distribute any earned block rewards amongst a co-operative group of miners. Solutions such as P2Pool [28] and SmartPool [14] allow an algorithm to play the role of the pool master, but have not yet gained widespread use. The combination of these two factors has undeniably led to a decrease in the number of participants in the consensus protocol underlying Bitcoin. In fact, a panel session at Scaling Bitcoin 2015 was made

up of eight participants who together controlled 80% of the Bitcoin network's computational power [20].

Whilst it is assumed that miners will honestly follow the consensus protocol, the assumption that the honest mining strategy is the most rewarding has been criticised. Eyal and Sirer [9] proposed selfish-mining strategies that can be deployed by rational miners with at least 25% of the network's computational power to gain more rewards than they deserve. This work was extended by Sapirshtein et al. [22] and Nayak et al. [18]. Although selfish-mining strategies theoretically weaken the 51% honest mining assumption, there is no evidence that any miners are engaging in them in major deployed cryptocurrencies. This suggests that miners are indeed honest and will not deviate from the honest mining strategy. Yet mining is not always profitable[1] in the short-term [6] and in August 2017 miners have exhibited rational behaviour in order to boot short-term profit. For example some mining pools including ViaBTC mined either Bitcoin or Bitcoin cash depending on which cryptocurrency was more profitable in the short-term. On the other hand influential members of the Bitcoin community have also suggested that miners may be accepting out-of-band bribes to mine Bitcoin Cash [24].

This type of mining behaviour reflects a tragedy of the commons first identified by Bonneau [3]. Individually rational miners have an incentive to maximise their profit (i.e., accept bribes to mine an alternative fork), but collectively share a concern for the network's long-term health. Liao and Katz [13] extend the work of Bonneau by proposing "whale" transactions, which use anomalously large fees to bribe miners. A new whale transaction is authorised for every new block in the alternative fork in order to reward the bribed miners for their continuous support. While Bonneau concludes that bribery attacks should be considered when attempting to prove that a Nakamoto-style consensus protocol is incentive compatibility, so far bribery attacks have not been seen as practical by the wider community. Although the question of how to attack or disrupt a minority chain in the event of new Bitcoin forks is increasingly an active area of discussion [1,5,30]. In fact, the founder of the mining pool BTC.TOP (which held 13% of the network's computational power as of October 2017) stated in an interview:

> *"We have prepared $100 million USD to kill the small fork of CoreCoin, no matter what proof of work algorithm, sha256 or scrypt or X11 or any other GPU algorithm. Show me your money. We very much welcome a CoreCoin change to POS."* [19]

Looking beyond Bitcoin, it is important to consider whether the additional functionality provided by alternative cryptocurrencies can be used to enable new forms of bribery, and in particular whether a platform like Ethereum that supports smart contracts enables the automated (and thus fair) payment of bribes to miners who change their mining strategy. For pooled mining, Velner et al. [27] demonstrate a smart contract that rewards miners in a pool who perform

[1] A surge in the Bitcoin price this year made mining profitable in the short-term for "hobbyist miners" [21].

so-called block withholding attacks[2] and later provide the contract with a proof-of-stale-work. Teutsch et al. [25] show that a briber can set up a script puzzle that diverts the network's mining power, thus removing competition for the briber's mining power. They demonstrate that a briber with at least 38.2% of the network's hashrate can achieve a positive pay-off that also covers the cost of each script puzzle.

This paper proposes three new contracts that reward miners who provide evidence that their mining strategy has changed according to a briber's intention. Our bribery attacks differ to previously proposed contracts, in that they do not focus on disrupting mining pool protocols or attempt to divert miners from solving the network's puzzle. Instead, our contracts facilitate renting hardware by rewarding miners using an in-band bribe (i.e. coins within the cryptocurrency) or an out-of-band bribes (i.e. coins from another cryptocurrency).

Our three smart contracts are as follows:

- In Section 3, we propose `CensorshipCon`, which relies on Ethereum's block reward policy to subsidise a briber wanting full control over the blockchain's content. We provide an analysis to show that a briber with at least 25% of the network's computational power can maximise the subsidy while also earning a small profit.
- In Section 4, we propose `HistoryRevisionCon`, which rewards miners that reverse transactions and computations in the blockchain by mining an alternative fork. It is also the first history-revision bribery attack where the briber and bribee trust only the contract (and not each other).
- In Section 5, we propose and implement `GoldfingerCon`, which rewards miners who can prove their mining strategy has reduced the utility of another cryptocurrency. We provide a proof-of-concept implementation to evaluate the feasibility of our attack and demonstrate that accepting a bribe costs approximately $0.46.

2 Background

In this section we provide an overview of Bitcoin and Ethereum with a focus on the expressiveness of Bitcoin Script, Ethereum smart contracts, and the reward policies in each consensus protocol.

2.1 Bitcoin

Bitcoin [17] is a global public ledger, maintained by a distributed set of miners, that facilitates trading a single asset (bitcoins) in a publicly verifiable manner. All coins are exchanged using transactions that have a list of inputs (the source

[2] In such an attack, a miner sends only partial proofs-of-work to the pool master, discarding all full proofs-of-work. The miner is rewarded by the master for attempting to find a new block but does not contribute to the pool's income as new blocks are discarded.

of the coins) and outputs (the destination of the coins). A mechanism called Bitcoin Script is used to specify conditions that must be satisfied before the coins associated with a transaction output can be spent. The most popular script, *pay-to-pubkey-hash*, requires a digital signature from the corresponding Bitcoin address (i.e., the hash of a public key). Another example of a script is *pay-to-pubkey*, which requires a digital signature from the public key's corresponding private key.

All transactions are recorded on the ledger, also called a blockchain, which is replicated by the peer-to-peer network. In order to update the ledger with a list of recent transactions contained in a block, miners compete to solve a computationally difficult puzzle. The first miner to present a solution wins the right to append his block to the blockchain, and receives 12.5 bitcoins in addition to all transaction fees collected in the block. A block is expected to be found approximately every 10 minutes.

A Bitcoin block is made up of two components. The first component is the block header, which contains the previous block hash, a Merkle tree root committing to all transactions in this block, a nonce to support the proof-of-work puzzle, and a timestamp. The second component is a list of transactions, the first of which is called the coinbase and is used to distribute the block reward. If the block contains no new transactions, the Merkle tree root is replaced with the hash of the coinbase transaction.

2.2 Ethereum

Ethereum [29] was proposed to facilitate users writing, storing and executing expressive, but bounded programs (i.e. smart contracts) within the Ethereum Virtual Machine (EVM). This EVM alongside all storage and computation is replicated across the peer-to-peer network and the blockchain is responsible for storing transactions that authorise state transitions. If all transactions are re-executed by a new peer joining the network, then the peer will eventually discover the contract's most recent state. All computation and storage is measured in units of gas and this is purchased using Ethereum's native currency (i.e. ether) by the user when they are authorising a transaction. As of January 2018, the vast majority of contracts are written in Solidity and like Bitcoin, all users have an Ethereum account which is the hash of a public key (and the corresponding private key is used to sign transactions).

Ethereum has a Nakamoto-style consensus protocol that relies on a distributed set of miners and its blockchain is a variation of GHOST [23], which is a tree-based blockchain. GHOST introduced the concept of an uncle block, which is a competing block that failed to make it into the blockchain but has its block header included in a future block; we call this future block the *publisher block*. For example, consider the case in which there are two competing blocks at height i, b^i_A and b^i_B. If b^i_A is accepted into the blockchain, the block header for b^i_B can still be included in a future block $b^{i+\delta}_{publ}$, at which point we can call it an uncle block $b^i_{B,uncle}$. This new block type was proposed in GHOST to allow

stale blocks to contribute towards the blockchain's overall weight, although uncle blocks in Ethereum did not contribute towards the blockchain's weight until the Byzantium upgrade in October 2017 [4]. This consensus rule was changed in response to an uncle block mining strategy proposed by Lerner [12] that could reward miners more coins than they deserve by exclusively mining uncle blocks.

The notable difference between GHOST and Ethereum's implementation is the uncle block reward policy. In Ethereum, a miner can include a maximum of two uncle blocks in a newly mined block and the miner receives a publisher reward of $c_{pub} = \frac{1}{32}c_{block}$ for each uncle block included, where c_{block} is the normal block reward. Once included in the blockchain, the uncle block's miner is sent an uncle block reward of $c_{uncle} = (1 - \frac{\delta}{8})c_{block}$, where δ is the distance (number of blocks) between the competing main block $b^i_{A,main}$ and the publisher block $b^{i+\delta}_{publ}$. As of January 2018, the full block reward is 3 ETH and the maximum distance permitted for an uncle block to be included in the blockchain is 6.

Finally, an Ethereum block header can be split into four sections. This includes the previous block hash, gas statistics to highlight the computations involved in this block, a solution to the memory-hard proof-of-work Ethash, and a list of Patricia trie roots for the uncle block headers, transactions, and the global state transaction. This is reflected in the size of an Ethereum block header, which is 480 bytes compared to Bitcoin's 80 bytes.

3 Subsidised Bribery

To set the scene, we consider the case of a briber, Alice, with less than a majority of Ethereum's computational power. Her goal is to control which transactions are accepted into the blockchain. Rather than purchasing or renting new hardware to achieve a majority, she decides to rent hashing power from other miners (which we collectively name Bob) by bribing them. An existing approach for miners to accept in-band bribes without trusting the briber involves whale transactions [13], but this requires Alice to pay the full cost.

Instead, we propose a smart contract that rewards Bob for intentionally mining uncle blocks. As a result, Ethereum's uncle block reward policy is used to directly subsidise bribes paid by Alice. In the best case, Bob is rewarded $\frac{7}{8}$ of the block reward c_{block} for mining an uncle block. If Bob can prove to Alice's smart contract that he mined an uncle block, then this contract will automatically send Bob an additional payout c_{payout}. This second payment covers the remaining fraction of the block reward c_{block} and includes an additional bonus c_{bribe} for accepting the bribe. As a result Bob will always earn more coins than mining honestly, and the uncle block reward subsidises bribes paid by Alice.

3.1 Censorship Contract

The contract `CensorshipCon`[3] rewards Bob for performing an uncle block mining strategy. Bob must withhold a new block until a competing block by Alice is

[3] A partial implementation is available at [15].

accepted into the blockchain, only then publishing his block for inclusion as an uncle block. He must then prove to `CensorshipCon` that his uncle block was included in the blockchain for the contract to send him the bribe. In the following we highlight how to initialise the contract and how to allow bribed miners to accept their subsidised bribe. Afterwards we provide an overview of Appendix A to highlight that the briber requires at least 25% of the network's computational power in order to maximise the subsidy.

Briber assumption. For this contract we assume Alice includes all uncle blocks and transactions that pay Bob his bribe into the blockchain. We consider this a reasonable assumption as accepting these blocks/transactions maximises her subsidy and encourages Bob to pursue the uncle block mining strategy. Finally we assume that Alice has bribed a sufficient portion of miners for the attack to succeed such that all blocks in the blockchain are mined by her.

Contract setup. Alice must set the network's block reward c_{block}, the bribe amount c_{bribe} and also deposit d coins into the deployed contract `CensorshipCon` before publicly advertising the bribe. This contract has a single function called `AcceptSubsidisedBribe()` that we describe below.

Accepting the subsidised bribe. Bob must perform the uncle block mining strategy in order to be eligible for the bribe. If he has mined a new block b^i_B, then he must withhold this block until Alice publishes a competing b^i_A and her block is accepted into the blockchain. Afterwards he can publish his block b^i_B to the network, which allows Alice to include it as an uncle block in one of her future blocks; this future block is the publisher block $b^{i+\delta}_{publ}$. Once his uncle block is accepted into the blockchain he must prove that he is entitled to the payout c_{payout}.

To prove this, Bob creates a transaction that invokes `AcceptSubsidisedBribe`. This function requires as input Bob's uncle block header $b^i_{B,uncle}$, Alice's competing block header $b^i_{A,main}$ and the publisher block $b^{i+\delta}_{publ}$.

Once invoked, the function verifies if Alice's competing block $b^i_{A,main}$ and the publisher block $b^{i+\delta}_{publ}$ are in the blockchain. This involves retrieving the most recent 256 block hashes B_{256},[4] hashing both block headers and checking if $H(b^i_{A,main}) \in B_{256}$ and $H(b^{i+\delta}_{publ}) \in B_{256}$. Next the contract checks if the publisher block $b^{i+\delta}_{publ}$ has indeed included Bob's uncle block $b^i_{B,uncle}$ and if the two competing blocks $b^i_{A,main}$ and $b^i_{B,uncle}$ extend the same block b^{i-1}.[5]

If the above verification is satisfied and Bob has not already received a bribe for $b^i_{B,uncle}$, the contract calculates his payout. To do this it first computes the number of blocks δ between $b^i_{A,main}$ and $b^{i+\delta}_{publ}$, as this is used to calculate Bob's uncle block reward $c_{uncle} = (1 - \frac{\delta}{8})c_{block}$. His uncle block reward is the subsidy

[4] The contract environment provides access via `block.blockhash(uint)` for the latest 256 blocks (except the current block).

[5] This is similar to how a proof-of-stale block is verified by Velner et al. [27]

provided by the network for the briber, and his final payout is calculated as $c_{payout} = c_{bribe} + c_{block} - c_{uncle}$. The contract then sends these coins to the miner's Ethereum account, as stored in $b^i_{B,uncle}$.

3.2 Lower-bound on briber's hashrate.

Appendix A contains an analysis of the computational power required by Alice to maximise the network's subsidy and an overview is presented here. Briefly, we begin by denoting m_A, m_B, and m_H as the network's hashrate shares controlled by Alice, Bob and remaining honest miners. Only Alice and the honest miners compete to create new blocks once Bob has decided to pursue an uncle block mining strategy. His computational power is considered in the network's difficulty calculation [4], but Alice only needs to control more computational power than the honest miners such that $m_A > m_H$ to control the blockchain. To maximise the subsidy she must also ensure that for every new block mined by her, Bob only has the computational power to mine up to two uncle blocks. This final requirement means she must have at least half of Bob's hash rate such that $m_A \geq \frac{1}{2}m_B$. If we combine both requirements then Alice's hashrate must be $m_A > \frac{1}{4}$ to ensure she can out-compete the honest miners and also include all blocks mined by Bob as uncle blocks.

We highlight that there is an issue with the lower bound of $m_A > \frac{1}{4}$ as the briber can potentially exclude all blocks mined by honest miners. As a result it is reasonable to assume that honest miners may defect and also pursue the uncle block mining strategy in order to accept her bribe. If this happens, then the requirement $m_A \geq \frac{1}{2}m_B$ may no longer hold as the bribed miners account for more than half of the network's hashrate such that $m_B > \frac{1}{2}$. As a result it is possible that three uncle blocks are created for every new block by Alice and unfortunately one block must be discarded due to the uncle block limit. In order to satisfy the goal of maximising the briber's subsidy we consider the scenario where miners will only accept the bribe if it is guaranteed that no uncle blocks are discard. This requires Alice's computational power to be $m_A \geq \frac{1}{2}(1 - m_A)$ and leads to a new lower bound of $m_A > \frac{1}{3}$, allowing any value $m_B \in [\frac{1}{3}, \frac{2}{3})$.

4 History Revision Bribery

The contract `HistoryRevisionCon`[6] extends the work of Bonneau [3] and Liao et al. [13], and allows Alice to reward miners for mining on a fork other than the current longest chain. She can also retroactively dictate the starting block for a new fork and also enforce an expressive forking condition. In the following example we use a double-spend transaction to change the balance of two accounts, A_2 and A_3, but in a manner similar to the "hard-fork oracle" outlined by McCorry et al. [16], the forking condition could also depend on other events such as a reversal of the infamous TheDAO theft [10].

[6] A partial implementation is available at [15].

Briber assumptions. The briber is no longer involved in the attack after setting up the contract. For the subsidised bribe we also assume that all bribed miners will include the transactions from other miners. This is reasonable as the bribed miners are collectively working together to ensure the alternative fork becomes the longest (and heaviest) chain.

Contract setup. Alice creates three accounts A_1, A_2, A_3, the first account A_1 creates the bribery contract, A_2 spends coins in the longest chain and A_3 receives the double-spent coins from A_2 in the alternative fork. She then publishes the transaction $T_{A_2,spend}$ that spends her coins from A_2; we denote the block that includes this transaction as b^i. Alice waits until the receiver considers $T_{A_2,spend}$ as confirmed in the blockchain.

Afterwards she publishes the double-spend transaction $T_{A_2,double}$ that sends all coins from A_2 to A_3 and the transaction $T_{A_1,fork}$ which will create the `HistoryRevisionContract` contract. Both transactions must be included in the first bribed block at height i by Bob before he can be rewarded a bribe. Of course the contract will check the balance of A_2, A_3 and that it was created in block i before rewarding any bribes.

Accepting the bribe. An accept bribe transaction must be included in every new block and it calls the `AcceptBribe()` function. The `AcceptBribe()` function requires no inputs and invokes the contract to check that a bribe has not already been paid for this block before sending the full bribe c_{bribe} to the miner's Ethereum account.[7] If his block fails to be included in the blockchain, then he must wait until this block is included as an uncle block.

Accepting the subsidised bribe. Similarly to the mechanism presented in Section 3.1, Bob can call `AcceptSubsidisedBribe()` and be rewarded for mining an uncle block. If the number of uncle blocks remains low (e.g., two or fewer uncle blocks for every block in the blockchain), then the briber can maximise the subsidy and ensure all stale blocks mined are also rewarded.

5 Goldfinger Bribery

As proposed by Kroll et al. [11], a "Goldfinger attack" can be modeled as a game between an adversary who receives external utility from devaluing (or destroying) a currency and the network that aims to run/maintain the value of a currency. One approach for devaluing a cryptocurrency is to effectively reduce its usefulness such that there is no guarantee a transaction will be accepted (or remain confirmed) in the blockchain. This can be accomplished by performing significant blockchain re-organisations (such as reversing 10 or more blocks in the blockchain) [1] or mining consecutive empty blocks. We propose that a briber can use a smart contract-enabled blockchain to fairly reward miners for mining empty blocks in another cryptocurrency.

[7] This can be accessed in the contract environment as `block.coinbase`.

5.1 Goldfinger Contract

We present `GoldfingerCon`, the first contract to realise a Goldfinger-style attack. This contract rewards miners in a smart contract-enabled blockchain for reducing the utility of another cryptocurrency by mining empty blocks.[8] In the following we discuss how to set up the Goldfinger contract, how Bob can prove he mined an empty block to the contract, the technical hurdles for this attack and our proof of concept implementation.

Contract setup. Alice creates `GoldfingerCon`, deposits d coins and must set the payout for each empty block as c_{bribe}. In order to activate the contract she must set the identification hash of the initial block $H(b_B^i)$ and all miners should begin mining empty blocks from $H(b_B^i)$ onwards. This contract is publicly announced to all miners in the victim cryptocurrency.

Accepting the bribe. Bob must audit the code for `GoldfingerCon` to verify that the contract will reward him for mining empty blocks in the victim cryptocurrency. Once he has decided to pursue the Goldfinger attack, then every new block $b_{B,btc}^i$ he mines should only contain the coinbase transaction and he must wait for the block to achieve a sufficient depth in the blockchain. Next he must publish an accept bribe transaction $T_{B,accept}$ that includes the block header $b_{B,btc}^{i,*}$ and the corresponding coinbase transaction $T_{B,coinbase}^i$in its payload.

This transaction calls the `AcceptBribe` function in `GoldfingerCon` and as a result the contract will verify whether $H(b_B^i)$ is an empty block before sending Bob his bribe. In order to verify that it is indeed an empty block, the contract checks that the identification hash of the coinbase transaction $H(T_{B,coinbase}^i)$ is stored in the block header's Merkle root field. Once the verification is complete, the contract extracts the public key PK_B from the coinbase transaction's output (assuming it is a *pay-to-pubkey* script), computes an Ethereum account B from PK_B and sends B the bribe c_{bribe}.

Validating and propagating empty blocks. In order to verify that the chain of empty blocks represents the most computational weight it is important that `GoldfingerCon` has access to all known forks in the victim cryptocurrency. We recommend that block headers follow a strict pre-defined format to ensure they are valid for both the contract and the victim cryptocurrency. Finally it is also important for Alice to remain online and ensure all empty blocks are propagated throughout the victim cryptocurrency's network. Otherwise a cartel of miners could mine headers for the contract, but they are not used (or potentially be invalid) in the victim cryptocurrency network. Another option is to build an escape hatch into the contract which allows her to terminate the bribe if cheating is detected, but this may also undermine the contract's credibility.

Step	Purpose	Gas Cost	US$ Cost
1.	Create contract	3,505,654	4.21
2.	Submit block header 49,996 (checkpoint)	316,799	0.38
3.	Submit block header 50,000 (out of order)	276,663	0.33
4.	Submit block header 49,999 (out of order)	261,727	0.31
5.	Submit block header 49,998 (out of order)	261,727	0.31
6.	Submit block header 49,997 (in order)	314,017	0.38
7.	Organise orphan blocks	284,206	0.34
8.	Accept bribe for block 50,000	152,579	0.18

Table 1: A breakdown of the gas and financial cost for submitting several existing Bitcoin blocks and accepting a bribe from `GoldfingerCon`.

.

5.2 Proof-of-concept implementation

We have implemented `GoldfingerCon` [15] in Solidity (0.4.10) and performed experiments on an Ethereum private network in October 2017 (before the Byzantium update). Our implementation demonstrates the cost of maintaining the longest chain of block headers and parsing Bitcoin block headers and coinbase transactions. It does not, however, rigorously validate block headers or coinbase transactions according to the network's consensus rules as discussed in Section 5.1. In the following we present the cost of creating the contract, publishing five blocks in the reverse order, computing the current longest fork (i.e. a blockchain re-organisation within the contract) and accepting a single bribe.

Table 1 presents the gas and financial breakdown for each transaction, assuming 1 ETH is worth $300 (a reasonable estimate as of October 2017 [8]) and the gas price is 4 Gwei.[9] The first step involved creating the `GoldfingerCon` contract alongside setting the payout for each bribe and the contract's owner as the briber. The second step required the owner to set the starting block (i.e. the checkpoint) as Bitcoin block 49,996 and thus the contract will only send bribes for new empty blocks that extend this checkpoint.

The next series of steps (i.e. 2-6) involved an ad-hoc Ethereum account sending the contract four Bitcoin blocks $50,000$ to $49,997$ in the reverse order. This resulted in the contract recognising block $49,997$ as the latest block in the blockchain. Step 7 notified the contract to evaluate the orphan blocks (i.e. blocks $49,998$ to $50,000$) and this resulted in the contract setting block $50,000$ as the latest block in the blockchain. This demonstrates that handling small block re-organisations within an Ethereum contract can be gas-efficient.

Finally step 8 involves simulating a miner publishing the coinbase transaction for block $50,000$ to accept their bribe. The contract verified that the identification hash of the coinbase transaction was stored in the Merkle tree root of block

[8] It is also possible to bribe miners for building an alternative fork by dictating that a block hash $H(b_B^i)$ cannot be in the blockchain.

[9] 1 gwei $= 10^{-18}$ ether

50,000. The public key from the coinbase transaction output is then extracted[10] to construct an Ethereum address. The miner's bribe is sent to the constructed Ethereum address and the bribe for block 50,000 is marked as claimed.

6 Discussion

Countering Goldfinger attacks. Bonneau [3] identified that the intended victims of a Goldfinger attack could counter-bribe the miners in order to protect the blockchain's integrity, but he went further to argue that it is not desirable to rely on wealthy members of the community to protect Nakamoto-style consensus. Another counter-measure that is often suggested by the Bitcoin community is to change the proof-of-work algorithm in response to an attack [7] and effectively punish the miners for participating in the attack by making their mining hardware redundant. We highlight that this is only a viable option if there is not an another cryptocurrency with significant value that also relies on the same proof of work algorithm. Also, it is only a one-shot approach as the briber could then rent the next viable hardware (e.g., GPUs). In terms of a new proof-of-work algorithm it may be useful to select one that is not easily verified within a smart contract environment to hinder our smart contracts. As demonstrated by Luu et al. [14], however, this defence can be overcome.

Removing asymmetrical trust assumption for history-revision bribery attacks The closest mechanism to our history-revision contract is whale transactions, where Alice includes large fees to entice miners to include her transactions, but these have asymmetrical trust assumptions. Briefly, if the bribed miners do not trust the briber, then the briber must sign a list of transactions in advance with incrementing time-locks (to ensure that only a single bribe accepted is included per block). On the other hand, if the briber does not trust the bribed miners, then to ensure that they do not collude and mine a fork without the briber's desired revision, the briber can publish a new whale transaction after every new block [13]. `HistoryRevisionCon` removes this asymmetry, as both the briber and bribed miners can trust only the contract.

Towards a 51% collusion. All bribery attacks require a strategy that persuades the network's computational power to join the attack and accept this bribe. In `CensorshipCon` and `GoldfingerCon`, miners are rewarded for every uncle/empty block mined, whereas in `HistoryRevisionCon` miners are paid only if the attack is successful. One approach for persuading miners to accept this bribe is to provide a greater reward for miners that join the attack early, and a list of deadlines can be set to ensure that there is a proportional increase in bribed blocks over time. For example, `GoldfingerCon` may require 10% of all blocks to be empty by time t_1 and 20% by time t_2. The contract can terminate if a deadline is missed. So far our contracts have assumed that a sufficient share

[10] All early coinbase transactions (including block 50,000) used pay-to-pubkey bitcoin scripts.

of miners have joined the attack. We leave it as future work to devise reliable strategies for ramping up support amongst miners.

In-band vs out-of-band payments. Out-of-band payment for bribery attacks such as `GoldfingerCon`, script puzzles [25] and proof-of-stale blocks [27] are viable as the utility received by miners is external to the cryptocurrency which is being attacked. Bribery attacks that rely on in-band payments like whale transactions [13], `CensorshipCon`, `HistoryRevisionCon` can potentially be viewed as not practical due to their public nature undermining the bribed miner's reward. There have been a few circumstances such as TheDAO fork [10] where in-band bribery could potentially have been used to reward miners for mining an alternative fork. We do not claim that in-band bribery attacks are immediately feasible today, but this may change in the future as the political climate surrounding cryptocurrencies continues to evolve.

Impact on Nakamoto consensus. Our contract `CensorshipCon` demonstrates how subtle changes to Ethereum's implementation of GHOST has directly enabled a subsidy for bribery attacks, whereas `GoldfingerCon` highlights that miners do not need to trust the briber when attacking the consensus protocol of another cryptocurrency. Bonneau [3] argued that bribery attacks are not recognised as a viable attack due to their public (and sometimes trusted) nature and this is reflected in the community as no new Nakamoto-style consensus protocol has considered bribery attacks in their threat model. We argue that with rise of smart contract-enabled blockchains, centralisation of mining hardware [26], politically motivated actors [19,24] and wealthy pseudonymous thieves [2,10], it appears that bribery-style attacks are indeed becoming increasingly viable. We thus also argue that new Nakamoto-style consensus protocols should consider bribery attacks when evaluating whether the protocol is incentive-compatible.

7 Conclusion

In this paper, we proposed three contracts to evaluate whether a smart contract-enabled blockchain can have an impact on Nakamoto consensus. Our contracts highlight that Ethereum's uncle block reward policy can be used to directly subsidise a bribery attack, that a briber can dictate the conditions that must be satisfied (i.e. reverse theft) when bribed miners mine an alternative fork and the feasibility of Goldfinger-style attacks that reward miners for reducing the utility of another cryptocurrency. Our contracts (including the work in [3,13,25,27]) are the first steps towards realising practical bribery strategies that overcome the inherent trust issue between a briber and bribee. This is achieved as all contracts self-enforce the bribery agreement and the fair exchange of coins.

Acknowledgements

Patrick McCorry and Sarah Meiklejohn are supported in part by EPSRC grant EP/N028104/1, and Alexander Hicks is supported in part by VASCO Data Se-

curity[11] and UCL through an EPSRC Research Studentship. We would like to thank Joseph Bonneau for discussions around bribery attacks, Ilya Sergey, Changyu Dong, and Abhiram Kothapalli for comments on early drafts of this paper, and Sergio Lerner and Adrian Eidelman for bringing to our attention that Ethereum's Byzantium upgrade changed the network's difficulty calculation.

References

1. G. Andresen. Ways to enhance Post-fork withering of Core chain. *RedditBTC*, Mar. 2017.
2. Bloomberg. Ethereum Bandits Stole $225 Million This Year. *Fortune*, Aug. 2017.
3. J. Bonneau. Why buy when you can rent? In *International Conference on Financial Cryptography and Data Security*, pages 19–26. Springer, 2016.
4. V. Buterin. Change difficulty adjustment to target mean block time including uncles . *Ethereum EIP Github Repository*, Oct. 2017.
5. D. Dinkins. If Hard Fork Happens, Chain Backed By Majority of Miners Will Likely Win. *Cointelegraph*, Oct. 2017.
6. J. Donnelly. Winter is Coming: Bitcoin Mining for Heat (And Profit). *CoinDesk*, Sept. 2016.
7. N. Dorier. Proof-of-Work update is not a threat to miners, it is a necessity for users. *Medium*, Mar. 2017.
8. ETHGasStation. Bribery Contracts. *ETH Gas Station*, Oct. 2016.
9. I. Eyal and E. G. Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *International conference on financial cryptography and data security*, pages 436–454. Springer, 2014.
10. R. Hanson. A $50 Million Hack Just Showed That the DAO Was All Too Human. *Wired*, June 2016.
11. J. A. Kroll, I. C. Davey, and E. W. Felten. The economics of bitcoin mining, or bitcoin in the presence of adversaries. In *WEIS 2013*, 2013.
12. S. D. Lerner. Uncle Mining, an Ethereum Consensus Protocol Flaw. *Bitslog blog*, Apr. 2016.
13. K. Liao and J. Katz. Incentivizing blockchain forks via whale transactions. In *4th Workshop on Bitcoin and Blockchain Research (BITCOIN)*, 2017.
14. L. Luu, Y. Velner, J. Teutsch, and P. Saxena. Smart pool: Practical decentralized pooled mining. *IACR Cryptology ePrint Archive*, 2017:19, 2017.
15. P. McCorry. Bribery Contracts. *GitHub*, Jan. 2017.
16. P. McCorry, E. Heilman, and A. Miller. Atomically trading with roger: Gambling on the success of a hardfork. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pages 334–353. Springer, 2017.
17. S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.
18. K. Nayak, S. Kumar, A. Miller, and E. Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 305–320. IEEE, 2016.
19. A. Quentson. Bitcoin Market Needs Big Blocks, Says Founder of BTC.TOP Mining Pool. *Cryptocoinsnews*, Feb. 2017.
20. J. Redman. The Scaling Bitcoin Workshop Hong Kong Wrap-Up. . *BitcoinComNews*, Dec. 2015.

[11] www.vasco.com

21. B. Reutzel. Bitcoin's Price Surge is Making Hobby Mining Profitable Again. . *CoinDesk*, July 2017.
22. A. Sapirshtein, Y. Sompolinsky, and A. Zohar. Optimal selfish mining strategies in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 515–532. Springer, 2016.
23. Y. Sompolinsky and A. Zohar. Secure high-rate transaction processing in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 507–527. Springer, 2015.
24. J. Song. Why Miners Are Mining Bitcoin Cash and Losing Money Doing It. *CoinDesk*, Aug. 2017.
25. J. Teutsch, S. Jain, and P. Saxena. When cryptocurrencies mine their own business. In *International Conference on Financial Cryptography and Data Security*, pages 499–514. Springer, 2016.
26. J. Tuwiner. Bitcoin Mining in China. *Buy Bitcoin Worldwide*, Mar. 2017.
27. Y. Velner, J. Teutsch, and L. Luu. Smart contracts make bitcoin mining pools vulnerable. *IACR Cryptology ePrint Archive*, 2017:230, 2017.
28. F. Voight. Wiki on P2Pool. . *Bitcoin Wiki*, June 2011.
29. G. Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 151, 2014.
30. J. Zhuoer. [Ending the Soft/Hard Fork Debate]A Safe Hard Fork is the same as a Soft Fork. *Medium*, Oct. 2016.

A Subsidy Analysis

This section is concerned with determining lower bounds on Alice's hash rate in order to both maximise the subsidy from Ethereum's uncle block reward policy and to provide her with full control over the blockchain. We find the lowest bound she requires is more than $\frac{1}{4}$ of the network's hashrate, but this does not allow all other miners to pursue the uncle block mining strategy and accept the bribe. This can be overcome if she has more than $\frac{1}{3}$ of the network's hashrate as this will allow all other miners to mine uncle blocks and for her to include every uncle block in the blockchain. In the following we present our assumptions for the subsidy analysis before presenting the subsidy value, whether a profit is possible for the briber and how to derive the two lower bounds.

Assumptions We assume that miners do not perform selfish mining strategies and that Alice has control of the minimum hashrate required to execute the attack.

Lower bounds on hashrate Keeping the same model, we denote m_A, m_B and m_H the portion of network's hashrate controlled by Alice, Bob and the remaining honest miners respectively. Alice's hashrate must satisfy two requirements before she out compete the honest miners and maintain the longest chain with only her blocks:

– Alice's hashrate must be at least half of the bribed miner's computational power such that $m_A \geq \frac{1}{2} m_B$.

– Alice's hashrate must be greater than the remaining honest miners on the network such that $m_A > m_H$.

The first requirement ensures Alice can include all new blocks published by Bob as uncle blocks. Recall she can only include up to two uncle blocks per block she published. The second requirement allows Alice to win against the remaining honest miners in creating blocks for the blockchain as she controls a majority portion of the effective hashrate (i.e. the network's hashrate attempting to mine main blocks which excludes the bribed miners). Putting these together gives a lower bound of $m_A > \frac{1}{4}$ on Alice's portion of the hashrate. This is enough for her to support Bob controlling $m_B = \frac{1}{2}$ of the hashrate, which ensures $m_A > m_H$.

The case above assumes that honest miners are not willing to accept the bribe which may be not be realistic as miners are economically motivation. In particular, honest miners may find themselves competing against Alice and if she controls a majority of the effective hashrate then all their blocks will be excluded. It is then necessary to revise our first requirement such that it now takes the form $m_A \geq \frac{1}{2}(1 - m_A)$. This allows Alice to support the inclusion of withheld blocks from all other miners (i.e. the remaining network's hashrate) as uncle blocks. Alice can then bribe the necessary portion of miners $m_B \geq \frac{1}{3}$ which satisfies the requirement $m_A \geq \frac{1}{2}m_B$ and allows $m_A > m_H$ to hold.

Subsidy and expectation values Alice can maximise her subsidy by minimising the number of blocks δ it takes until a bribed block is included in the blockchain as an uncle block. Ideally, she should include uncle blocks in her next block. otherwise δ may range between 1 and 6 if an uncle is delayed entry. Before highlighting the impact of this subsidy for Alice, recall that she also receives an additional publisher reward $c_{pub} = \frac{1}{32}c_{block}$ for each uncle block she includes. Hence for one uncle, Alice only has to pay Bob $c_{payout} > \frac{4\delta-1}{32}c_{block}$ to guarantee him a higher payout than if he had won the full block reward. If she includes two uncle blocks, she has to pay $c_{payout} > \frac{4\delta-1}{16}c_{block}$ to ensure both blocks have a higher payout.

As Alice receives a block reward for every block she mines, she can still profit from the reward while ensuring Bob receives a higher payout. If her payout to Bob is less than the block reward (i.e. $c_{payout} < c_{block}$), she will have earned more than she spent. The values of δ required for her to make a profit are then easily found by solving the inequality with the expressions for c_{payout} previously given. For example, if she only includes a single uncle in her block, then she is guaranteed a profit for any δ as the reward is sufficient to ensure Bob receives the full block reward c_{block} alongside an additional c_{bribe}. If she includes two uncle blocks, then she must be efficient as the δ for both blocks must be less than 4 on average in order to satisfy the condition. Of course this analysis is only concerned with the potential mining profit, and does not take into account the preliminary costs of gaining the required hashrate.

As previously mentioned, the payout c_{payout} should be sufficient for Bob to be guaranteed a higher reward than the block reward and giving him a clear incentive to accept bribes. If the attack is successful, Alice will control the effective

majority of the network's hashrate and honest miners will see their expectation value decrease as their blocks are excluded from the blockchain. They will then be further incentivized to accept bribes.