# Bet and Attack: Incentive Compatible Collaborative Attacks Using Smart Contracts

Zahra Motaqy[1](✉), Ghada Almashaqbeh[1], Behnam Bahrak[2], and Naser Yazdani[2]

[1] University of Connecticut, Storrs, USA
{raha,ghada.almashaqbeh}@uconn.edu
[2] University of Tehran, Tehran, Iran
{bahrak,yazdani}@ut.ac.ir

**Abstract.** Smart contract-enabled blockchains allow building decentralized applications in which mutually-distrusted parties can work together. Recently, oracle services emerged to provide these applications with real-world data feeds. Unfortunately, these capabilities have been used for malicious purposes under what is called criminal smart contracts. A few works explored this dark side and showed a variety of such attacks. However, none of them considered collaborative attacks against targets that reside outside the blockchain ecosystem.

In this paper, we bridge this gap and introduce a smart contract-based framework that allows a sponsor to orchestrate a collaborative attack among (pseudo)anonymous attackers and reward them for that. While all previous works required a technique to quantify an attacker's individual contribution, which could be infeasible with respect to real-world targets, our framework avoids that. This is done by developing a novel scheme for trustless collaboration through betting. That is, attackers bet on an event (i.e., the attack takes place) and then work on making that event happen (i.e., perform the attack). By taking DDoS as a usecase, we formulate attackers' interaction as a game, and formally prove that these attackers will collaborate in proportion to the amount of their bets in the game's unique equilibrium. We also model our framework and its reward function as an incentive mechanism and prove that it is a strategy-proof and budget-balanced one. Finally, we conduct numerical simulations to demonstrate the equilibrium behavior of our framework.

**Keywords:** Collaborative attacks · Mechanism design · Criminal smart contracts · Blockchain model

## 1 Introduction

Cryptocurrencies and blockchain technology continue to build innovative computing models and economic tools that can reshape the services and systems

Z. Motaqy—Most work done while at University of Tehran.

around us. Fueled by the huge interest this technology received, researchers and practitioners alike are racing to build new applications and improve existing ones. Smart contracts facilitate this process; individuals can deploy arbitrary code on a blockchain, allowing for trustless collaboration between participants under terms enforced by the contract execution. More recently, and to achieve the quest of allowing blockchain to react to real-world events, the concept of oracles has been introduced [1,3,9]. These are external services that supplement a smart contract with information about specific real-world events.

However, these new capabilities have been used for malicious purposes as well. This falls under what is called criminal smart contracts (CSCs), a term that was first coined by Juels et al. [17]. They showed that CSCs can be used for cryptographic key theft, leakage of confidential information, and real-world crimes such as murder. Since then, several studies have proposed new smart contract-based attacks mainly related to bribery to disrupt the mining process [12,15,18,25]. Nonetheless, these works were limited to (collaborative) attacks on the cryptocurrency/blockchain itself rather than outsider real-world targets. Juels et al. [17] considered real-world targets, but only for solo attackers.

Collaborative attacks are known to be more devastating [4–6,19,26,27,29]. This raises the question of whether CSCs can facilitate such attacks in the real world. Addressing this question is challenging; (pseudo)anonymity of blockchain users leads to incomplete information about the attackers. Also, dealing with attackers in the real world makes it hard to measure and verify their contribution. Attackers are represented by random-looking public keys in a CSC who tend to hide their involvement in a given attack to avoid legal consequences. This is different from attacks with targets within the blockchain ecosystem itself, where usually cryptographic primitives comes for the rescue to prove a claimed contribution [15]. This is even easier in the context of a solo attacker; the fact that an attack took place at the promised time/location implies that the work has been done [17]. Verifying contribution is essential to ensure that self-interested parties cannot collect their rewards without doing the required work.

**Contributions.** In this paper, we bridge this gap by showing how smart contracts can be used to perform collaborative attacks against real-world targets. We develop an incentive compatible framework that allows a sponsor to orchestrate a collaborative attack among (pseudo)anonymous attackers and reward them for that. Our framework mitigates the requirement of quantifying attackers' individual contributions by introducing a betting-based technique to allow trustless collaboration. In particular, attackers bet on an event (i.e., the attack takes place) and then work on making that event happen (i.e., perform the attack). By designing an incentive-compatible reward mechanism, these attackers will be motivated to deliver the work represented by the value of their bets, and hence, obtain their rewards. To the best of our knowledge, our work is the first generic CSC-based collaborative attack framework against real-work targets.

As a use case, we consider distributed denial of service (DDoS) attacks.[1] We design a smart contract to perform this attack, and we formulate attackers' inter-

[1] We note that [22] dealt with smart contract-based DDoS, but the work is very high level and lacks many important details, making it hard to assess its feasibility.

action as a game. Then we formally prove that these attackers will collaborate in proportion to the amount of their bets in the game's unique equilibrium. We also model our framework and its reward function as an incentive mechanism and prove that it is strategy-proof (i.e., attackers will not misrepresent the amounts of their bets) and that it is budget-balanced (i.e., the total rewards allocated to the attackers doesn't exceed the deposited attack rewards). Through numerical simulations, we study the impact of several parameters, including the reward function, total amount of bets, and attack cost, on the attack outcome. We show that in a typical scenario, the proposed incentive mechanism provides individual rationality and fairness for the collaborating attackers.

We argue that showing feasibility of collaborative attacks using CSCs is essential to identifying such threats, and devising secure countermeasures. Moreover, and although our focus is on attacks in this paper, we believe that our framework can facilitate benign collaboration between users. For example, it can be employed in blockchain-based decentralized systems offering digital services, such as content delivery [8] and file storage [7], to incentivize peers to act honestly while serving others. Such a feature has a huge impact on system efficiency. That is, incentive compatibility mechanisms (when applicable) can be used to defend against potential threats instead of (usually computationally-heavy) cryptographic mechanisms.

**Related Work.** Many incidents were reported on cybercriminals collaboration in the context of malware and massive DDoS attacks [4–6,19]. Attackers trade goods, services, and money through the phishing marketplace and even advertise their demands [10]. Moreover, botnets can be even rented for spam campaigns and DDoS attacks [2].

Usually these attacks involve some trust assumption that a sponsor will reward attackers for participation. CSCs can replace such assumption by providing an automated and transparent way for coordination and sponsorship [11,17,20]. As mentioned before, this observation was first investigated by Juel et al. [17]. They proved how the CSCs they developed can be commission fair, meaning that neither the sponsor nor the attackers can cheat. Another work that considered real world targets is [14] who proposed a semi-autonomous (file) ransomware architecture using CSCs. Their scheme allows paying for individual files or reimbursing the victim if decryption is invalid. Despite handling real-world targets, these works focused only on solo attackers.

Another line of work on CSCs focused on bribery to attack the cryptocurrency or blockchain itself. [25] showed an attack to allow a mining pool manager to destroy competing mining pools, and [15,18] showed how a sponsor can bribe miners to pursue a mining strategy that benefits him (e.g., to allow him to revert a transaction or to double spend). CSC-based attacks on mining have been systematically analyzed in [16] under the umbrella of algorithmic incentive manipulation attacks. On the game theoretic front, Chen et al. [12] introduced a game modeling of a bribery contract. They showed that in any Nash equilibrium, a sponsor cannot win the majority of the votes unless he/she controls more than 20% of the total bribing budget. Although the aforementioned bribery attacks consider collaborative attackers, their target is the blockchain/cryptocurrency itself. Also, these works implement techniques to enable attackers to provide cryptographic proofs that the desired mining strategy has been performed. As we mentioned before, such aspect is hard to achieve when the target resides outside the blockchain ecosystem. Our work handles these issues by showing the feasibility of CSC-based collaborative attacks against real-world targets without measuring the individual contribution of attackers.

## 2 A Model for a CSC-Based Collaborative Attack

In this section, we present the blockchain and threat models we adopt in this work, along with a security notion for CSCs. After that, we introduce the proposed CSC-based collaborative attack model with DDoS attacks as a usecase.

### 2.1 Blockchain Model

We deal with permissionless public blockchains that support smart contracts, such as Ethereum [28]. Hence, the code of the smart contract, its state, and all messages (or transactions) sent to this contract are logged in the clear on the blockchain. Any party needs an account in order to interact with a smart contract, referred to as externally owned account (EOA) in Ethereum. This account is identified by the public key of its owner, and has a public state that is mainly concerned with the account currency balance. Anyone can create any number of EOAs and participate in a CSC, and no real identities are required. Although we consider Ethereum in our CSC model, any other smart contract infrastructure can be used given that its scripting language can represent the CSC functionality.

We require the blockchain to have access to oracles that provide authenticated real-world data feeds. A popular example of such oracle services is Provable [9]. In the context of a CSC, the attack sponsor (who is also the CSC creator) will specify the metrics that the CSC will query from the oracle that will quantify the outcome of the attack. For example, in our DDoS usecase, we use the response delay as the attack outcome metric, where excessive delays means an overwhelmed (or a down) server. Provable can measure this metric by sending http requests to the target server and report the response to the CSC for which parameters like delay can be measured.

### 2.2 Threat Model

We adopt the following threat model in this paper:

– The blockchain is secure in the sense that the majority of the mining power is honest. So the confirmed state of the blockchain contains only valid transactions, and that any attempt to rewrite the blockchain will fail with overwhelming probability. We also assume liveness, meaning that messages cannot be blocked or delayed beyond some bounded duration, and availability in the sense that the blockchain records are accessible at anytime.

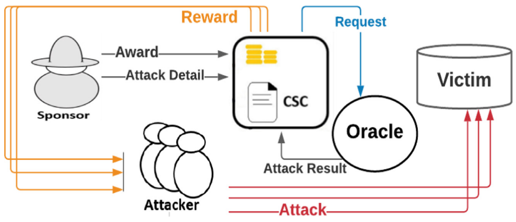


**Fig. 1.** A model for a CSC-based collaborative attack.

– The oracle service is secure (i.e., provide valid digitally signed data), always available, and capable of measuring the attack result in terms of some predefined metrics.[2]
– No trusted party of any type exists. Also, attackers involved in any CSC are mutually-distrusted and rational so they will act based on what maximizes their individual profits. Moreover, these attackers are regarded as independent risk-neutral decision-makers; they will be interested in maximizing the expected value of their utility.[3]

A *fully successful attack* in our model is defined as meeting the desired attack result set by the CSC sponsor. The difference between the actual and desired attack result denote the success level of an attack. It should be noted that in some attacks such success level is not applicable; it is either the attack has been done or not (e.g., steal a cryptographic key).

For security, we adopt the notions of *correctness* and *commission-fairness* proposed in [17]. A correct CSC is one that implements the attack protocol as designed by the sponsor. And a commission-fair CSC is one that guarantees that neither the sponsor nor the perpetrator of the crime can cheat; attack sponsor cannot avoid paying attackers for the work they have done, and attackers cannot collect rewards for work they have not done. Note that *work* here refers to pursuing the attack as per the sponsor's desired metrics.

## 2.3 Attack Model

At a high level, and as shown in Fig. 1, a CSC-based collaborative attack is composed of three phases: a CSC design and deployment phase, an attack phase, and a reward allocation phase. In order to make the discussion easier to follow, we present these phases with a running example. In particular, we consider one of the devastating cybersecurity attacks, namely, DDoS attacks. The sequence diagram of this attack is captured by Fig. 2, while the following paragraphs elaborate on the attack protocol according to our framework phases.

[2] It is the responsibility of the attack sponsor to pick metrics for which there is a secure oracle service that cam measure and report them.

[3] While it is true that attackers are not always risk-neutral, we assume that is the case here for simplicity. For an analysis of non-risk neutral attackers, see [21].

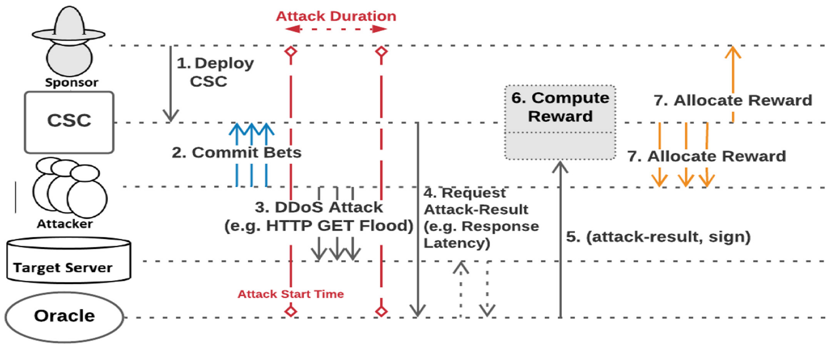


**Fig. 2.** Sequence diagram for a DDoS attack using CSC

**Phase 1. Design and Deployment of CSC.** In this phase, an attack sponsor designs the CSC functionality. This includes defining all APIs and methods needed to orchestrate the attack and distribute the rewards. This in addition to attack parameters such as attack target, duration (measured in rounds, where a round is the time needed to mine a block on the blockchain), the metrics used to measure the attack result, and the reward allocation function. For DDoS, the target can be a specific server, and the metric that we use is response delay. The oracle service Provable can measure such quantity by simply sending HTTP requests to the specified target and report the response back to the CSC for which parameters like delay can be measured [9].

The sponsor then creates a smart contract implementing the CSC functionality and publishes it on the Ethereum blockchain. He will also deposit an award for the attackers in the CSC account. Once published, and since the contract code is public on the blockchain, interested attackers can evaluate the terms and assess if it is feasible for them to participate in the attack. For example, they check their availability during the attack duration and whether they have the resources required to achieve the attack goal. If feasible, each attacker will submit a bet to the CSC's account during the betting period. This period starts when the contract is confirmed on the blockchain until the attack starting time.

In terms of DDoS, an attacker checks if he can reaches the target server, and that he can afford sending traffic to overwhelm the server. In this context, this attacker can, for example, rent botnets to achieve that [2]. Based on the amount of traffic this attacker can afford, which represents his amount of contribution in the attack, the attacker will choose his bet value and deposit in the DDoS CSC.

**Phase 2. The Attack.** The second phase of our framework is the attack phase. Once the deposits are made by both the sponsor and the attackers, and the attack period starts, these attackers will launch the attack. Each attacker may choose any strategy given that it achieves the attack goal set by the sponsor. For DDoS, an attacker sends appropriate traffic to overwhelm the target.[4] Since all

[4] While it is common that the target defends itself by identifying and filtering the attack traffic, for simplicity we assume that attackers generate effective attack traffic (traffic that passes the defense walls and gets to the target server).

attackers are sending traffic during this period, and although they do not know and do not trust each other, they are collaborating against the same target. CSC allows this automated collaboration coordination without placing trust in anyone. The attack phase continue until the end of the attack period specified in the contract.

**Phase 3. Reward Allocation.** In the third phase, the CSC queries the oracle to obtain the attack result. This can be done by having the sponsor send a transaction to invoke a function in CSC that sends a request to the oracle. The oracle then fetches the attack result and return it by executing a callback function in CSC. Based on the attack goal, this can happen either when the specified attack duration is over (e.g., check that the target server is down) or during that period (i.e., check that the server response delay is long enough). If multiple measures are reported, then the average, minimum or maximum can be computed (recall this is part of the terms that the sponsor specifies).

After that, the CSC computes the total currency value of the sponsor award and the attackers' bets. Then, it distributes this amount among the sponsor and the attackers based on the reward allocation function and the attack result. If the attack is fully successful, the total currency will be distributed among the attackers. If the attack is unsuccessful, then this currency will go to the sponsor (i.e., attackers are punished by taking away their bets). For attacks where there is a success level some where in between, as in DDoS, both attackers and sponsors will get part of the total currency. The distribution of this amount will be based on a reward allocation function that the sponsor chooses. In Sects. 3 and 4 we thoroughly discuss the details of the reward allocation mechanism.

At the end, the CSC sends payment transactions to the sponsor and/or attackers' accounts (these accounts are defined by the attackers public keys used when registering the bets).

We note that despite advances in devising countermeasures against DDoS [23], these attacks cannot be fully prevented. Nonetheless, it is important to point out that given that the CSC is public on the blockchain, the target could be aware of the attack (assuming he is inspecting the blockchain regularly). Hence, the target can employ defense strategies during the specified attack period. This will increase the cost of performing the attack as we will show in Sect. 5.

**The Contract.** The CSC contract for the above collaborative DDoS attack is outlined in Fig. 3. We model a data feed (oracle) as a sequence of pairs $(m, \sigma)$. Where $m$ is the attack result reported by the oracle, and $\sigma$ is the oracle's digital signature over $m$. Thus, the oracle has an private/public key pair $(pk_O, sk_O)$ used to sign/verify signatures. In the figure, $bal[X]$ denotes the balance of users' account $X$ on the blockchain.

## 3 Game Theoretic Model and Analysis

Dealing with self-interested attackers is problematic; they do not care about the sponsor's goals, and they would lie to collect more rewards if they can. Attacking

```
Upon Deploying CSC by an Attack Sponsor $\mathcal{S}$
Init: $award$, $target\_server_{url}$, $start\_time$, $attack\_duration$, $bet_{min}$, $Sponsor_{address} := [\mathcal{S}]$, $pk_O$
Create:
  – Profile of each attacker: Struct $Attacker$ {$account$, $bet$, $reward$}
  – List of all bets placed: $Attacker$ [ ] public $Bets$

Deposit Award:
Assert $bal[\mathcal{S}] \geq award$
Transfer $award$ coins from $\mathcal{S}$ to CSC's account

Upon Submitting a Bet by an Attacker $\mathcal{A}$
Input: $bet = msg.value$
Description:
If: $(current\_time < start\_time)$ and $(bet > bet_{min})$
Then:
Assert $bal[\mathcal{A}] \geq bet$
Transfer $bet$ coins from $\mathcal{A}$ to CSC's account
$Bets$.push(
  – $Attacker.account$ : $msg.sender$,
  – $Attacker.bet$ : $bet$,
  – $Attacker.reward$ : 0)

Upon End of Attack Duration
Description:
Assert $current\_time > start\_time + attack\_duration$
$(m, \sigma) :=$ requestMeasure($oracle, target\_server_{url}$)
Assert sigVer($pk_O, m, \sigma$)
\* Update the $Attacker.reward$ property for all attackers in $Betts$ list based on the reward allocation function *\
ComputePayments($award, Bets, m$)
\* Pay attackers their reward and transfer the rest of the deposited money, if any, to the sponsor's account *\
TransferPayments($Bets$, $Sponsor_{address}$)
```

**Fig. 3.** CSC pseudocode

real world targets complicates the problem since attackers will tend to hide such information to avoid any legal consequences. This raises the question of how a CSC can measure the individual contribution of each attacker. Our goal is to identify and analyze factors that influence the attacker's behavior and use that to configure incentives properly and encourage faithful collaboration. Towards this goal, in this section, we formulate our framework for CSC-based collaborative attacks as an incomplete information game, and we show that it has a strong dominant strategy equilibrium. For simplicity, we present this modeling and analysis in the context of DDoS attack, but it can be generalized to any other attack type using the proper metrics.

## 3.1 Attackers Contribution

Our solution utilizes betting to avoid direct measuring of individual contributions. The attack sponsor places an initial award and each attacker places a bet, all deposited in the CSC account. An attacker is supposed to contribute in the attack in proportion to his bet value. Based on how successful the attack is (as reported by the oracle), the total amount of currency in the CSC account will be distributed among the attackers and/or the sponsor (the sponsor gets a refund only if the attack is not fully successful).

Recall that all CSC information is public on the blockchain as part of the CSC code and state. However, and given that attackers are known only by their random-looking public keys, we cannot guarantee that each attacker will place only one bet. In fact, an attacker may use that to gain privacy, as well as increase his utility (if possible), by dividing the bet into multiple smaller bets, then post these bets using several accounts all controlled by this attacker. Thus, each attacker privately knows his true bet (consisted of several smaller bets), even though all the transactions are public.

Therefore, in our framework, attackers have to decide on two actions. First, before the attack phase starts, they should decide on a betting strategy: each attacker can submit multiple bets that sum to his true bet or one bet that is equal to his true bet. We call the latter *truthful betting*. Afterward, in the attack phase, each attacker should decide on the amount of contribution to the attack.

Later in this section, we show that each attacker's relative contribution is independent of others' contributions and bets and is a function of his own bet value. In the next section, we also show that truthful betting is the best strategy for an attacker.

## 3.2 Interdependent Attackers Game (IAG)

We study the interaction among a set of interdependent strategic attackers as a game with independent private values and strict incomplete information. Independent private values means the utility of an attacker depends entirely on his own private information. Strict incomplete information means we have no probabilistic information in the model, i.e., we consider a worst-case scenario for missing information.

In this game, let $N$ represent the attackers set such that $|N| = n$, $\omega_S$ be the award of the sponsor, $bet_i$ be the true bet value of the $i^{th}$ attacker such that $0 < bet_i < \omega_S$, and $t_i = \frac{bet_i}{\omega_S}$ be the private information that this attacker has and it represents his type.[5] Let $e_{th}$ represent an estimation of the total traffic

[5] We assume dealing with homogeneous agents in terms of the cost of contributing to the attack (all have same $\alpha$ in Eq. 3). Also, we show later that an attacker's bet value represents his actual contribution in the attack, and hence, his cost.

needed to satisfy the desired DDoS attack result set by the sponsor, and $e_i$ be the relative contribution of the $i^{th}$ attacker in this amount, i.e., $0 \leq e_i \leq 1$. Thus, if attacker $i$ is a free-rider and makes no effort to contribute in the attack, then $e_i = 0$. While if $e_i = 1$, this means that attacker $i$ has launched a fully successful attack on his own. Let $T$ denote the set of all type profiles $\hat{t} = (t_1, \ldots, t_n)$ and $E$ denote the set of all action profile $\hat{e} = (e_1, \ldots, e_n)$. Based on his type $t_i$, attacker $i$ chooses the action $e_i \in [0, 1]$. The strategy function $S : T \to E$ maps each attacker type to an action $e_i$. Rational attackers will select a strategy that will maximize their own utility as possible.

The reward allocation function is one of the mechanism rules that has a significant impact on the attacker's preferred strategies and attack result. In our framework, the sponsor defines the reward allocation function $R : [0, 1] \times [0, 1] \to \mathbb{R}$ in the smart contract. We propose a simple allocation function that allots reward to attackers according to the amount of their bets and the their total relative contributions in the attack denoted as $e_{tot}$, i.e., $e_{tot} = \sum_{i \in N} e_i$.[6] Let $M$ be the total money amount deposited in the CSC's account, so $M = \omega_S + \sum_{i \in N} bet_i$. After the attack phase ends, the CSC distributes $M$ among the attackers and the sponsor, and automatically generate payment transactions to transfer the allocated rewards to their accounts.

We consider the following reward allocation function (where $bet_{tot} = \sum\limits_{i \in N} bet_i$):

$$R(bet_i, e_{tot}) = M \cdot e_{tot} \cdot \frac{bet_i}{bet_{tot}} \tag{1}$$

That is, the share of each attacker from $M$ is proportional to his bet value. As noted, if $e_{tot} < 1$, meaning that the attack is not fully successful, then the residual of $M$ will go back to the sponsor. For our analysis, we want the formulation of $R$ to depend on the attacker type $t_i$. As such, Eq. 1 can be converted into an equivalent formulation as follows:

$$R(t_i, e_{tot}) = M \cdot t_i \cdot e_{tot} \cdot \left(\frac{bet_{tot}}{\omega_S}\right)^{-1} \tag{2}$$

In order to compute the utility of an attacker, we need also to characterize the cost of preforming an attack. Let $C : [0, 1] \to \mathbb{R}^+$ be a cost function that maps a value $e_i$ to the cost expended in achieving the attack contribution profile $e_i$. We assume that $C$ is strictly increasing and convex on its domain. In particular, we use the following general form cost function:

$$C(e_i) = \alpha \cdot \frac{\exp(e_i) - 1}{e_{max} - e_i} \qquad \forall i = 1, \ldots, n \tag{3}$$

[6] Note that in reality we cannot compute $e_i$. Hence, $e_{tot}$ is computed by using a suitable function to convert the delay reported by the oracle into the proper total traffic relative value. For example, if the measured delay meets the desired value, i.e., fully successful attack, then $e_{tot} = 1$. If it is 50% the value of the desired attack result, then $e_{tot} = 0.5$, and so on.

where $e_{max}$ is the max possible attack traffic (the one beyond any attacker capability), such that the cost of generating attack traffic will approach infinity when $e_i$ approaches $e_{max}$. And $\alpha$ is the average cost factor for generating attack traffic, where its value depends on the type of attack, the target's defensive power, and attackers' resources. The resulting utility function $U : [0,1] \times [0,1] \rightarrow \mathbb{R}$ for the $i^{th}$ attacker can be computed as $U\left(bet_i, e_i, e_{tot}\right) = R(t_i, e_{tot}) - C(e_i) - bet_i$, which again we convert in terms of $t_i$ as follows:

$$U\left(t_i, e_i, e_{tot}\right) = R(t_i, e_{tot}) - C(e_i) - t_i \cdot \omega_S \tag{4}$$

Let $\hat{e}_{-i}$ denotes the vector components consisting of elements of $\hat{e}$ other than the $i^{th}$ element, so $e_i + \hat{e}_{-i} = e_{tot}$. Based on that, we can write the utility function as $U(t_i, e_i, \hat{e}_{-i})$, which emphasizes that the $i^{th}$ attacker only has control over his own attack effort, $e_i$. A static strict incomplete information game is then defined by a tuple $IAG = <N,\ E,\ T,\ U>$. In the $IAG$, each attacker (aka player) maximizes its own utility in a distributed fashion. Formally, the non-cooperative $IAG$ is expressed as:

$$\max_{e_i \in [0,1]} \ U\left(t_i, e_i,\ \hat{e}_{-i}\right) \quad \forall i \in N \tag{5}$$

## 3.3 Equilibrium Analysis

It is necessary to characterize a set of attack efforts where each player is satisfied with the utility he receives (which maximizes his utility), given the attack efforts of other players. Such an operating point is called an equilibrium. The equilibrium concept offers a predictable, stable outcome of a game where multiple players with conflicting interests compete through self-optimization and reach a point where no player wishes to deviate. The condition for stability we aim for here is that of strong dominant strategy equilibrium. Note that rules of the game, including the utilities of all players, are public knowledge but not their private information, namely, $t_i$ and $e_i$.

First, we derive the best-response strategy of a player in $IAG$. Then we prove that the described game among attackers in this setting has a dominant strategy equilibrium. The $i^{th}$ attacker's best response strategy $S^*(t_i, \hat{e}_{-i})$ to a given strategy profile $\hat{e}_{-i}$ is given as the unconstrained maximizer of his utility, where $AW$, $bet_{tot}$ and $e_{th}$ are fixed:

$$S^*(t_i, \hat{e}_{-i}) = \arg\max_{e_i \in [0,1]} U\left(t_i, e_i,\ \hat{e}_{-i}\right) \tag{6}$$

To find the maximizing $S^*(t_i)$, we take the first derivative of $U$ with respect to $e_i$ and equate it to 0:

$$-\alpha \cdot \frac{\exp\left(e_i\right)}{e_{max} - e_i} - c \cdot \frac{\exp\left(e_i\right) - 1}{\left(e_{max} - e_i\right)^2} + \frac{\omega_S \cdot t_i \cdot \left(\omega_S + bet_{tot}\right)}{bet_{tot}} = 0 \tag{7}$$

So, the best response strategy $S^*(t_i, \hat{e}_{-i})$ of the $i^{th}$ player (with a value denoted as $e_i^*$) does not depend on the attack efforts of the other players $\hat{e}_{-i}$.

Therefore, we can represent the best response strategy function by $S^*(t_i)$. Furthermore, the only parameters (other than $t_i$) that determine $S^*(t_i) = e_i^*$ are the cost of the required attack traffic $\alpha$ and the quantity $\frac{bet_{tot}}{\omega_S}$.

**Definition 1.** *Given an incomplete information game $\Gamma = <N, E, T, U>$, a strategy $S^*(t_i)$ is a strongly dominant strategy, if for every $t_i$ we have that the strategy $S^*(t_i)$ is a strongly dominant strategy in the full information game defined by $t_i$. Formally, for all $t_i$, all $\hat{e}_{-i}$, and all possible values for $e_i$ (denoted as $e_i'$) such that $e_i' \neq S^*(t_i)$, we have*

$$U(t_i, S^*(t_i), \hat{e}_{-i}) > U(t_i, e_i', \hat{e}_{-i}) \tag{8}$$

**Definition 2.** *Given an incomplete information game $\Gamma = <N, E, T, U>$, a strong dominant strategy equilibrium is an action profile $\hat{e}^* = (S^*(t_1), \ldots, S^*(t_n))$ in which each $S^*(t_i)$ is a strongly dominant strategy. The notion of strongly dominant strategy requires that $S^*(t_i)$ is the unique best response to all possible $\hat{e}_{-i}$, i.e., without knowing anything about $\hat{t}_{-i}$ (the type of other attackers).*

**Theorem 1.** *IAG defined above has a strong dominant strategy equilibrium.*

*Proof.* For each player $i$ and for fixed $t_i$ and $e_{-i}$, the reward function $R : T \times E \rightarrow \mathbb{R}$ is linear, and cost function $C : E \rightarrow \mathbb{R}$ is convex with respect to $e_i$. So $U(t_i, e_i, \hat{e}_{-i})$ is concave and has a unique maximum. $S^*(t_i)$ is the unique maximizer of $U(t_i, e_i, \hat{e}_{-i})$[7] and is a strongly dominant strategy that is the best response regardless of $\hat{e}_{-i}$. So for any $i \in N$ and all $t_i$, all $\hat{e}_{-i}$ and all $e_i'$, we have $U(t_i, S^*(t_i), \hat{e}_{-i}) > U(t_i, e_i', \hat{e}_{-i})$ and strategy profile $\hat{e} = (S^*(t_1), \ldots, S^*(t_n))$ is the strong dominant strategy equilibrium of the game. □

## 4 Exploring Incentive Compatibility

As discussed earlier, due to the anonymity characteristic of blockchain users, we can not determine the number of attackers $n$ and the amount of their individual bets $bet_i$. In fact, knowing these bets allows predicting the attack result in the equilibrium of the game, i.e., $\sum_{i \in N} S^*(t_i) = e_{tot}^*$. In this section, we model the rules that govern the interactions in the CSC as a mechanism. By applying mechanism design theory, we prove that despite private information and pure selfish behavior, we can predict the attack result and the conditions that impact it. In addition, we show that under certain reasonable conditions, our mechanism satisfies the necessary constraints of mechanism design, namely, incentive compatibility, individual rationality, budget balance, and fairness.

[7] Note that the utility function is increasing at $e_i = 0$ and decreasing at $e_i = 1$, so the maximum can not occur at end points.

### 4.1 Mechanism Formulation

As we showed in the previous section, the $IAG$ game modeling players' interactions in a CSC has a strong dominant strategy equilibrium. As such, in equilibrium, the contribution of the $i^{th}$ attacker with type $t_i$ will be $S^*(t_i) = e_i^*$, and it is independent of other attackers' types and strategies. Based on that, for an attacker type profile $\hat{t} \in T$, we define the attack result function $AR : T \rightarrow E$ as:

$$AR(\hat{t}) = \sum_{i \in N} S^*(t_i) = \sum_{i \in N} e_i^* = e_{tot}^* \tag{9}$$

We consider the $i^{th}$ attacker's bet $bet_i$ as his payment to CSC, and the reward $R(t_i, AR(\hat{t}))$ is what the mechanism pays him. Accordingly, this attacker receives a payment amount $p_i$, or makes it if $p_i$ is negative, expressed as (note that $t_i \cdot \omega_S = bet_i$):

$$p_i(\hat{t}) = R(t_i, AR(\hat{t})) - t_i \cdot \omega_S \tag{10}$$

Let $G(\hat{t}) = (S(\hat{t}), P(\hat{t}))$ be the outcome function that maps each type profile $\hat{t} \in T$ to an outcome $o = (e_{tot}^*, \hat{p})$ where the payment rule $P(.)$ defines a profile of attackers' payments $\hat{p} = (p_0, p_1, \ldots, p_n)$, and the set of possible outcomes is denoted by $O$.

The attack result $e_{tot}^*$ is the non-monetary part in the outcome. To monetize it, we introduce a valuation function $V : \mathbb{R}^+ \times T \rightarrow \mathbb{R}$ (in terms of some currency) to represent an attackers' preference for a given attack result. In other words, the valuation function expresses the cost that an attacker is willing to tolerate when contributing to the attack (based on his dominant strategy), in addition to the transaction fee paid to post his bets on the CSC (i.e., fees for Ethereum miners). So, if this attacker post his true bet in $k$ small bets, each of which will require $\delta$ transaction fee (since each will be sent in a separate transaction), the total fee will be $k \cdot \delta$. Based on that, $V$ can be expressed as follows:

$$V(e_{tot}^\star, t_i) = -(C(S^\star(t_i)) + k \cdot \delta), \quad \forall i \in N, \forall t_i \in [0, 1] \tag{11}$$

Recall that attacker $i$'s contribution depends only on his type $t_i$, and so is the valuation function shown in Eq. 11. Thus, we can express it as $V(e_{tot}^\star, t_i) = V(t_i)$. As a result, this valuation function can be used to represent the type of attacker, and announcing a type is similar to reporting the attacker's valuation function.

Since we have money/incentive transfer between agents (aka attackers), we work in quasi-linear setting. Each agent has a utility that is the motivating factor behind the selection of his strategy. The preference of attacker $i$ can be captured using his utility function that can be redefined as follows (this is equivalent to the one defined in Eq. 4):

$$U(t_i, o) = V(t_i) + p_i \tag{12}$$

We formalize the incentive mechanism for strategic attackers in the proposed collaborative attack as a direct mechanism.[8] In this setup, each attacker $i$

[8] This mechanism suits our model since we have the space of possible actions is equal to the space of possible types, so an attacker type (which is defined when he bets) is the same as his act (the amount of attack contribution).

announces a type $t'_i$ to the mechanism, which is not necessarily equal to his true type $t_i \in [0,1]$, such that it will lead to an outcome that maximizes his utility. We also have a social choice function $F : T \rightarrow O$ that maps each agents' type profile to an optimal outcome, which is the same as the outcome function.

**Definition 3.** *A direct mechanism in a quasi-linear setting is defined by* $D = (T, G(\hat{t}))$*. The mechanism defines the set of allowable types* $T$ *that each agent can choose and an outcome function* $G$ *which specifies an outcome* $o$ *for each possible type profile* $\hat{t} = (t_1, \ldots, t_n) \in T$*.*

## 4.2 Incentive Compatible Property

Direct mechanisms extract information from agents by motivating them to *tell the truth.* If the best response for all attackers to report their private information truthfully to the CSC-driven mechanism, we say the contract is incentive compatible. Here we prove that the proposed mechanism is cheat-proof, which means that all attackers are motivated to submit their bet truthfully, and any deviation will lead to a utility loss.

**Definition 4.** *The social choice function* $F(\cdot)$ *is a dominant strategy incentive compatible (DSIC) (aka strategy-proof or cheat-proof) if and only if*

$$U(t_i, o) \geq \; U(t'_i, o') \qquad \forall i \in N, \; \forall \hat{t} \in T, \; \forall t'_i \in [0,1] \tag{13}$$

*where* $o = G(t_i, \hat{t}_{-i})$ *and* $o' = G(t'_i, \hat{t}_{-i})$*.*

Thus, if the SCF is DSIC, then the best response for agent $i$ is to bet truthfully, i.e., $bet_i = t_i \cdot \omega_S$, regardless of other attackers' bets. By calling a direct mechanism DSIC or strategy-proof, we mean that the mechanism implements an incentive-compatible or strategy-proof social choice function.

**Theorem 2.** *The proposed direct mechanism (or social choice function) modeling our CSC-based collaborative attacks is DSIC.*

*Proof.* We need to show that the utility of attacker $i$, is maximized when he bet truthfully. We use proof by contradiction. Without loss of generality, assume that the true type of attacker $i$ is $t_i$, but he tries to misrepresent his type and submit his bet with two distinct blockchain addresses in two transactions ($k = 2$) containing two bets $bet_{i,1}$ and $bet_{i,2}$, such that $bet_{i,1} + bet_{i,2} = bet_i$ and $bet_i = t_i \cdot \omega_S$. We denote the incorrect types by $t_{i,1}$ and $t_{i,2}$, so we have $t_i = t_{i,1} + t_{i,2}$. We assume that he benefits from this untruthful act which means $\exists t_{i,1}, t_{i,2} \in T$, where $t_i = t_{i,1} + t_{i,2}$, and $\exists \hat{t}_{-i} \in T^{n-1}$ such that

$$U(o', t_{i,1}) + U(o', t_{i,2}) > U(o, t_i) \tag{14}$$

where $G(t_{i,1} + t_{i,2}, \hat{t}_{-i})$ and $G(t_i, \hat{t}_{-i})$ are the values of $o'$ and $o$, respectively.

Note that, from each attacker $i$'s perspective, his dominant strategy $S^*(t_i)$ is a function of his true type $t_i$, which he knows, and so he can calculate his dominant strategy as $S^*(t_i) = S^*(t_{i,1} + t_{i,2})$. Thus:

$$\begin{aligned} AR(t_{i,1} + t_{i,2}, \hat{t}_{-i}) &= S^*(t_{i,1} + t_{i,2}) + \sum_{j \in N, i \neq j} S^*(t_j) \\ &= S^*(t_i) + \sum_{j \in N, i \neq j} S^*(t_j) = AR(t_i, \hat{t}_{-i}) \end{aligned} \tag{15}$$

And for the rewards, we have:

$$\begin{aligned} R(t_{i,1}, e_{tot}) + R(t_{i,2}, e_{tot}) &= (t_{i,1} + t_{i,2}) \cdot (bet_{tot} + \omega_S) \cdot \left(\frac{bet_{tot}}{\omega_S}\right)^{-1} \cdot e_{tot} \\ &= t_i \cdot (bet_{tot} + \omega_S) \cdot \left(\frac{bet_{tot}}{\omega_S}\right)^{-1} \cdot e_{tot} = R(t_i, e_{tot}) \end{aligned} \tag{16}$$

So the attacker's reward does not increase by hiding his type (i.e., hiding his bet). Therefore, from the mechanism point of view, the payments are the same as if the attacker would announce his true type, so:

$$p_i(t_i, \hat{t}_{-i}) = p_i(t_{i,1}, \hat{t}_{-i}) + p_i(t_{i,2}, \hat{t}_{-i}) \tag{17}$$

As we see, rewards and payments are identical in both cases. So the only way for an attacker to obtain a higher utility when lying, is to have a higher valuation than the one for the betting truthfully. This means that we have:

$$V(t_{i,1}) + V(t_{i,2}) > V(t_i) \tag{18}$$

Which means that:

$$C(S^*(t_i)) + 2\delta < C(S^*(t_i)) + \delta \tag{19}$$

This inequality cannot hold since this means that a transaction fee $\delta$ is negative, which is not the case. Thus, we a get a contradiction, meaning that an attacker will not get a higher utility by submitting multiple bets instead of the a single one (i.e., his true bet). $\square$

Note that we assumed attackers do not trust each other, so each attacker at least needs one transaction to submit his bet or he can submit multiple bets that sum to his true bet. However, let's assume that two attackers do trust each other and want to collude and fool the mechanism to increase their profit. The only possible misbehavior is to submit one bet that its amount is equal to the sum of their bets. In this case, the mechanism considers them as one identity (i.e., one attacker) and based on the proposed reward allocation function, this will not increase their reward. The only effect of this collusion is that the actual attack result would be higher than the pre-calculated attack result. This is because the same amount of contribution costs less for two attackers than one attacker (recall that the cost function is convex) and when two attackers assumed as one, their contribution will be considered less.

### 4.3 Budget Constraint

To be economically feasible, an incentive mechanism must be budget constrained. In our framework, the total rewards allocated to the attackers should not exceed $M = \sum_{i \in N} bet_i + \omega_S$, which is the total deposit made to the CSC account. In terms of payments, this means that the total attackers payment should not exceed the sponsor award $\omega_S$.

**Definition 5.** *(Budget Constraint for CSC). A reward mechanism is budget constrained if for $\forall \hat{t} \in T$ we have $\sum_{i \in N} p_i(\hat{t}) \leq \omega_S$.*

**Theorem 3.** *The proposed direct mechanism is budget constrained.*

*Proof.* The total attackers payments can be expressed as:

$$\begin{aligned}\sum_{i \in N} p_i(\hat{t}) &= \sum_{i \in N} \Big(R(t_i, S(\hat{t})) - t_i \cdot \omega_S\Big) \\ &= \sum_{i \in N} \Big((bet_{tot} + \omega_S) \cdot t_i \cdot \Big(\frac{bet_{tot}}{\omega_S}\Big)^{-1} \cdot S(\hat{t}) - t_i \cdot \omega_S\Big) \\ &= (bet_{tot} + \omega_S) \cdot AR(\hat{t}) - bet_{tot}\end{aligned} \tag{20}$$

where $AR : T \to E$ is as given in Eq. 9 and $E = [0, 1]$. So for any type profile $\hat{t} \in T$ we have:

$$-bet_{tot} \leq (bet_{tot} + \omega_S) \cdot S(\hat{t}) - bet_{tot} \leq \omega_S \tag{21}$$

From Eqs. 20 and 21, we get $\sum_{i \in N} p_i(\hat{t}) \leq \omega_S$, completing the proof. □

### 4.4 Voluntary Participation Constraint

Individual rationality or voluntary participation property of a social choice function means that each attacker gains a non-negative utility by participating in the mechanism that implements the social choice function. There are two stages at which individual rationality can be examined. First, when the amount of bet (types) of other attackers $\hat{t}_{-i}$ is unknown to attacker $i$, and therefore predicting attack result (i.e. outcome) is impossible. Second, when before the attack phase (ex-post stage), a choice to withdraw from the mechanism is given to all attackers. That is when all the attackers have announced their bet, and an attack result can be calculated. Note that a truthful attacker who submits his bet in one transaction incurs two transaction fees, $2\delta$, to withdraw from the mechanism (and if an attacker submits $k$ smaller bets, his cost to withdraw will $2k\delta$). This property of *ex-post individual rationality* is stated as follows.

**Definition 6.** *(Ex-post Individual Rationality). The utility an attacker $i$ with type $t_i$ receives by withdrawing from a CSC is equal to $-2k\delta$. To ensure attacker $i$'s participation when withdrawal is allowed at the ex-post stage, we must satisfy the following ex-post Individual Rationality (or participation) constraint*

$$U(G(\hat{t}), t_i) \geq -2k\delta \quad \forall \hat{t} \in T \tag{22}$$

A mechanism satisfies ex-post individual rationality if it implements a social choice function that satisfies ex-post individual rationality. In the next section, through numerical analysis, we show that under mild conditions the proposed mechanism satisfies this constraint. Note that these conditions can be encoded as rules in CSC by the sponsor. For instance, he can condition the attack on a specified amount of total bets, or he can restrict attackers' type by specifying upper bound and lower bound on acceptable bets.

### 4.5 Fairness

Different definitions for fairness have been proposed in the literature including proportional fairness, max-min fairness, $\alpha-$fairness [24]. To measure the fairness of the proposed incentive mechanism, we need a metric that captures how close the payment obtained by this mechanism is to fair payment. By fair payment, we mean reward allocation based on the contribution of attackers (instead of the amount of their bets). In other words, if attackers could prove their contribution, then the reward allocation function would simply be a function of their contribution and the attack result.

We denote the fair payment by $p'_i$ and can be computed as:

$$p'_i(\hat{e}) = \omega_S \cdot e_i \cdot e_{tot} \tag{23}$$

Similar to [13], we consider a metric, which we call the fairness-score, based on the root mean square (RMS) of the difference between the mechanism payment and the fair payment. The fairness-score is defined as

$$D_{rms} = \sqrt{\frac{1}{n} \sum_{i \in N} (p_i(\hat{t}) - p'_i(\hat{e}))^2} \tag{24}$$

where $p_i(\hat{t})$ is given by Eq. 10. A value of zero for $D_{rms}$ indicates that the mechanism payment equals to the fair payment, which discourages free-riders and rewards contributors. As the value increases, the fairness of the reward allocation function decreases. In the next section, using numerical simulation for two reward functions, we show that under some mild conditions, the fairness score drops for both rewarding schemes after hitting the desired attack result.

## 5 Numerical Simulations and Discussion

Based on our game model, there are four key parameters that impact the attack outcome (in terms of payment and attack result). These factors are $bet_{tot}$, $\omega_S$, $C(e_{th})$, and type profile $\hat{t} = (t_1, \ldots, t_n)$. Furthermore, the reward allocation function has a significant impact on the attack result and other properties, such as fairness and individual rationality.

In this section, we conduct simulations for a typical DDoS scenario to analyze how various system parameters impact attack outcome. The parameter values

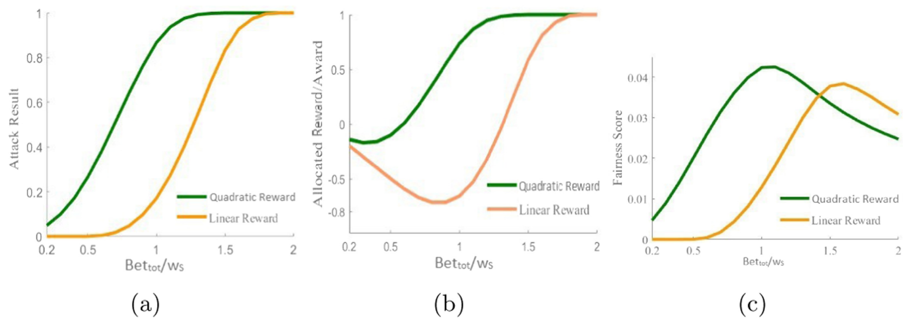


**Fig. 4.** The impact of increasing $\theta$ for fixed $\gamma = 0.35$ on (a) attack result (b) proportion of allocated award to attackers, and (c) fairness-score of payment.

used in this study are $n = 30$, $\frac{bet_{max}}{bet_{min}} = \frac{1}{10}$, $bet_i \sim Uniform(bet_{min}, bet_{max})$, and the results are averaged over 50 randomly chosen attackers' bets. In our simulations, we consider two reward allocation functions: linear and quadratic (in the bet amount). The former is the one given by Eq. 2 and denoted by $R$, while the latter is expressed as:[9]

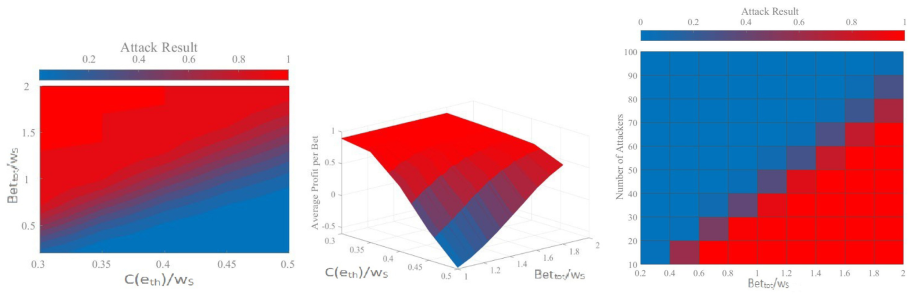
$$R'(\hat{t}, e_{tot}) = M \cdot e_{tot} \cdot \frac{t_i^2}{\sum_{i \in N} t_i^2} \tag{25}$$

As for the attack result, it will have an upper bound based on the contract terms. That is, putting more effort beyond what is needed to reach the desired attack result will not increase the rewards for the attackers. Thus, rational attacker will stop when the measured attack result reaches the desired value.

We define two parameters: $\theta = \frac{bet_{tot}}{\omega_S}$, and $\gamma = \frac{C(e_{th})}{\omega_S}$. Here, $\theta$ gives an indication of the total amount of bets with respect to the sponsor award, and $\gamma$ is an indicator of the cost of performing the attack with respect to this award. We use these parameters in our simulations to study the impact of total bet and cost on the attack result.

Comparing the impact of reward allocation functions, as Fig. 4(a) shows the quadratic scheme attracts significantly more contribution than the linear reward scheme with the same value of $\theta$. In this scheme, having $\theta = 1$, one can estimate that the expected attack result $e^*_{tot}$ would be around 90% of the desired attack result $e_{th}$. On the other hand, we observe that the linear reward allocation function takes $\theta \approx 1.7$ to attract such contribution however this scheme allocate the rewards fairer compared to the quadratic scheme. Moreover, Fig. 4(b) shows that at that point the quadratic reward scheme pays 75% of the considered award while it costs attackers around 80% of the total cost of a successful attack. The remaining 25% of the award will be paid for reaching the desired attack result $e_{th}$. As expected, the attack result, the allocated reward to award ratio, and fairness

[9] Note that the theoretical (game and mechanism) analysis conducted for $R$ holds for $R'$, too.



(a) (b) (c)

**Fig. 5.** The impact of $\theta$ and $\gamma$ on (a) the attack-result, and (b) ratio of average profit over average bet size, with fixed number of attackers $n = 30$. (c) the impact of $\theta$ and number of attackers on attack result ($\gamma = 0.35$)

score increase with the increase of $\theta$. Therefore, the attack sponsor can condition the attack on a minimum amount of total bet to ensure the success of the attack and a given fairness score. Also, Fig. 4(c) demonstrates that after hitting the desired attack result, in both rewarding schemes, the fairness-score drops which means after this point, the more the total bet, the fairer the allocation will be.

As Fig. 5(a) shows, the required $\theta$ for providing the incentive of launching a successful attack increases as the cost factor $\gamma$ increases. Therefore, as the attack cost goes higher, a larger value of $bet_{tot}$ is needed to provide the required incentives to launch a successful attack. Accordingly, knowing the total amount of the bets, the number of attackers, and the award, the target server can estimate and change the attack result through increasing the cost of that attack (be deploying proper defenses against DDoS).

As discussed in Sect. 4.4, attackers are incentivized to participate when a non-negative utility is expected. Figure 5(b) shows that under some mild conditions, in terms of $\theta$ and $\gamma$, the proposed mechanism satisfies the desirable property of individual rationality. The red region denotes the situations that the average attackers' profit is positive. That is knowing the cost of the desired attack and the minimum total bet, the attack sponsor or the attack target can adjust the award or the cost to incentivize or deincentivize attackers to participate.

As shown in Fig. 5(c), the larger the number of attackers the larger $bet_{tot}$ required for launching a successful attack. In other words, increasing the number of attackers alone would not lead to more collaboration rather they should be incentivized enough to collaborate. Therefore accepting small bets decreases the attack result and a rule of minimum acceptable bet in the CSC can be helpful.

## 6 Conclusion

In this paper, we introduced a framework for employing CSCs to orchestrate real-world targeted attacks. These attacks are launched by several collaborating attackers without any knowledge of each other or any trust between them. To study the feasibility of this idea, we considered DDoS attack as a usecase. By using thorough game theoretic analysis and mechanism design, we showed that the attack sponsor can design a cheat-proof and budget-balanced mechanism to encourage collaboration of selfish rational attackers. Furthermore, the sponsor can predict and adapt the attack result, i.e., determine under what conditions attackers will participate in the attack. Simulation results show that, under some mild conditions on the attack cost and total amount of bets, the proposed incentive mechanism provides individual rationality and fair allocation of rewards. Being the first to study CSC-based collaborative attacks against real-work targets, we believe that our work will contribute in promoting the foundational understanding of these attacks, an important step towards developing effective countermeasures.

## References

1. Aeternity oracles. https://aeternity.com/documentation-hub/protocol/oracles/oracles/
2. Botnet economy runs wild. https://www.networkworld.com. Accessed 18 Sept 2020
3. Chainlink. https://chain.link/
4. Cyber criminal collaboration intensifies. https://www.computerweekly.com. Accessed 18 Sept 2020
5. Cybercriminals are increasing efficiency with coordinated attacks. https://www.enisa.europa.eu. Accessed 18 Sept 2020
6. Evidence found of malware families collaborating. http://www.darkreading.com. Accessed 18 Sept 2020
7. Filecoin. https://filecoin.io/
8. Noia. https://noia.network/
9. Provable. https://provable.xyz/
10. Abad, C.: The economy of phishing: a survey of the operations of the phishing market. First Monday **10**(9), 1–11 (2005)
11. Brunoni, L., Beaudet-Labrecque, O.: Smart contracts and cybercrime: a game changer. Math. Struct. Model. **4**, 136–142 (2017)
12. Chen, L., et al.: The game among bribers in a smart contract system. In: Zohar, A. (ed.) FC 2018. LNCS, vol. 10958, pp. 294–307. Springer, Heidelberg (2019). https://doi.org/10.1007/978-3-662-58820-8_20
13. Da, B., Ko, C.C.: Resource allocation in downlink MIMO-OFDMA with proportional fairness. J. Commun. **4**(1), 8–13 (2009)
14. Delgado-Mohatar, O., Sierra-Cámara, J.M., Anguiano, E.: Blockchain-based semi-autonomous ransomware. Future Gener. Comput. Syst. **112**, 589–603 (2020)
15. Judmayer, A., et al.: Pay to win: cheap, crowdfundable, cross-chain algorithmic incentive manipulation attacks on pow cryptocurrencies. In: Workshop on Trusted Smart Contracts (2021)
16. Judmayer, A., et al.: SoK: algorithmic incentive manipulation attacks on permissionless pow cryptocurrencies. In: Workshop on Trusted Smart Contracts (2021)
17. Juels, A., Kosba, A., Shi, E.: The ring of gyges: investigating the future of criminal smart contracts. In: ACM CCS, pp. 283–295 (2016)

18. McCorry, P., Hicks, A., Meiklejohn, S., et al.: Smart contracts for bribing miners. In: Zohar, A. (ed.) FC 2018. LNCS, vol. 10958, pp. 3–18. Springer, Heidelberg (2019). https://doi.org/10.1007/978-3-662-58820-8_1
19. Nazario, J.: Politically motivated denial of service attacks. In: The Virtual Battlefield: Perspectives on Cyber Warfare, pp. 163–181 (2009)
20. O'hara, K.: Smart contracts - dumb idea. Internet Comput. **21**(2), 97–101 (2017)
21. Qian, Y., Haskell, W.B., Tambe, M.: Robust strategy against unknown risk-averse attackers in security games. In: AAMAS (2015)
22. Rodrigues, B., Trendafilov, S., Scheid, E., Stiller, B.: SC-FLARE: Cooperative DDoS signaling based on smart contracts. In: IEEE ICBC, pp. 1–3 (2020)
23. Zargar, S.T., Joshi, J.: A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks. EEE Commun. Surv. Tutor. **15**(4), 2046–2069 (2013)
24. Trichakis, N.K.: Fairness in operations: from theory to practice. Ph.D. thesis, Massachusetts Institute of Technology (2011)
25. Velner, Y., Teutsch, J., Luu, L., et al.: Smart contracts make bitcoin mining pools vulnerable. In: Brenner, M. (ed.) FC 2017. LNCS, vol. 10323, pp. 298–316. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70278-0_19
26. Vogt, R., Aycock, J.: Attack of the 50 foot botnet. Technical report, Department of Computer Science, University of Calgary (2006)
27. Vogt, R., Aycock, J., Jacobson Jr., M.J.: Army of botnets. In: NDSS (2007)
28. Wood, G.: Ethereum: a secure decentralised generalised transaction ledger (2014)
29. Xu, S.: Collaborative attack vs. collaborative defense. In: Bertino, E., Joshi, J.B.D. (eds.) CollaborateCom 2008. LNICST, vol. 10, pp. 217–228. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03354-4_17