A Reputation Scheme for a Blockchain-based Network Cooperative Defense

Andreas Gruhler, Bruno Rodrigues, Burkhard Stiller
Communication Systems Group CSG, Department of Informatics IfI, University of Zurich UZH
E-mail: [gruhler,rodrigues,stiller]@ifi.uzh.ch

***Abstract*—Distributed Denial-of-Service (DDoS) attacks rise in frequency, diversity, and intensity. Often, centralized defense approaches lack hard- and software capabilities. A cooperative, multi-domain DDoS mitigation system provides defense services on top of an existing, distributed infrastructure. However, participants in these systems lack incentives for cooperation and reputation. Thus, reward systems can fill this gap by providing necessary incentives for cooperation among service providers and consumers. This paper presents the design, implementation, and evaluation of a reputation scheme for the Blockchain Signaling System (BloSS). A smart contract-enabled process automates reputation management to diminish malicious behavior by incentive design. Among other metrics, Beta reputation provides intelligence to identify and reward honest participants.**

***Index Terms*—Blockchain, Cooperative Defense, Reputation and Security Management, Trust**

I. Introduction

Distributed Denial-of-Service (DDoS) attacks are a significant threat to the Internet's availability that remain unmitigated despite many commercial and research efforts. A large number of unsecured devices that range from connected cameras to smart fridges are being connected to the Internet-of-Things, and their growing processing capacity allows attackers to take control of a vast amount of resources to launch malicious attacks. Many of these devices are insecure by design and in many cases impossible to be secured due to their hard- and software constraints.

To prevent or reduce damages caused by these DDoS attacks, different detection and mitigation methods are available, being organized in a single-domain defense or in-house. These defense methods are mainly implemented based on dedicated ASIC-based (Application Specific Integrated Circuit) appliances to analyze flow records exported from edge routers, and further filtering or load balancing traffic. Similarly, cloud services such as the ones offered by CloudFlare [6] or Akamai [1] can take away the burden of detection and mitigation, serving as a proxy being able to load balance, reroute, or drop the traffic in case of DDoS attacks.

However, in-house defense systems have shown to lack hard- and software capabilities to detect and mitigate the attacks themselves [12]. These trend attacks can easily exceed the defense capacity of a single system with respect to volume and frequency. Thus, as DDoS attacks become progressively sophisticated and coordinated, the defense from such attacks likewise needs distribution and coordination. By utilizing other organization's resources, the burden of the protection can be shared, and defense capabilities can be extended through the different protection systems participating in the distributed defense. The Internet Engineering Task Force (IETF) drafted a Distributed-Denial-of-Service Open Threat Signaling (DOTS) architecture for this purpose [12].

While providing many benefits, a cooperative defense also poses many challenges. For example, why shall organizations help each other? In a competitive environment, trust needs to be established, thus, reputation needs to be managed. Solely relying on voluntary contribution (*i.e.*, accepting defense requests) creates a favorable environment for free riding peers (consuming resources without contributing). DDoS attacks might be originated in remote networks without a close relationship between the governing Internet Service Providers (ISP). Thus, a reputation system is to encourage truth-telling and discourage free-riding between providers without a trust-relationship. This situation and the social dilemma that business partners find themselves in is illustrated in Figure 1. First, the attack target publishes malicious IP addresses. Second, multiple mitigators adjust the configuration of their network devices to filter and drop malicious packets. This second step is referred to as the actual "mitigation service". In a third step, the attack target evaluates the effectiveness of the mitigation service. By advertising reputation in a public manner one can provide incentives, foster desirable behavior, and shape the environment for mitigators and attack targets to naturally follow the social norms and rules of the system. Thus, the actions of the participants cannot remain hidden.

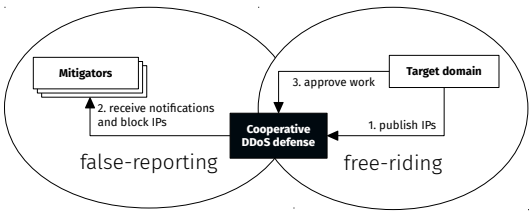

Fig. 1: Problem of false-reporting and free-riding.

This work here on the reputation management scheme tackles the problem of false-reporting and free-riding in a cooperative defense by proposing a reputation and a basis to reward schemes. It extends the work in [22, 25] with

978-3-903176-15-7 © 2019 IFIP

a reputation scheme, however, focusing on the reputation mechanisms solely and abstracting DDoS mitigation service delivery, payments or financial incentives. We assume policies for cooperation are expressed in a smart contract and the system in [22, 25] considers these network and security policies in a more detailed manner. This reputation system builds on established concepts and could be readily integrated in the area of cooperative defense.

A reputation scheme allows contributors and consumers of the network to rate entities that request protection in a cooperative defense. These systems have already been proven useful for e-commerce Web-sites, incentivizing peers to contribute with relevant information and establishing fairness among peers. Moreover, similar social dilemmas exist in other areas, *e.g.* crowdsourcing [33]. Even in large Peer-to-Peer (P2P) networks, peers maintain lasting business relations and transact repeatedly [29]. This increases the potential benefit of such type of systems in P2P related domains. In Mobile Ad hoc NETworks (MANET), researchers identified the same need, to provide incentives and credit-based mechanisms for cooperation among peers [8, 15]. However, the DOTS draft does not discuss these issues, which are of administrative nature or part of business agreements between the cooperating domains. Thus, incentive or reward mechanisms are required.

The design of reputation and reward systems is still a challenge because they are vulnerable to gaming attempts by malicious users [9, 28]. These attacks against reputation systems are also referred to as "rating fraud" and can occur in different forms [4]. Attackers can use these strategies to inflate their own reputation (*i.e.* ballot-stuffing) or cause the defamation of others (*i.e.* bad-mouthing) [4]. In collusion and Sybil attacks, multiple identities are misused to manipulate reputation [9].

When using multi-dimensional and subjective user-ratings, a bad rating can dilute the reputation of other peers, independent of their actual behavior [9]. Therefore, self-balancing and credible reputation metrics based on well-defined inputs other than subjective ratings are needed [32]. Similarly, a self-policing reputation system can discipline the participants' behavior without the need for a central authority [14]. Another problem identified in [9] is, that reputation systems are prone to re-entry attacks if the cost to obtain a new identity is too low.

A carefully chosen initial reputation score and reputation function can present newcomers trustworthy in their first transactions, but still discourage from whitewashing the identity [27]. The cost to create a new identity must be higher than benefits that come with the initial reputation score. The blockchain-based attack signaling architecture proposed in [25] already leverages IP verification to proof IP ownership. Using the limited IPV4 address space has shown to be an excellent way to raise the cost barrier of re-entry and to decrease the risk of Sybil and whitewashing attacks [4, 9].

Blockchain technology not only offers new possibilities in attack signaling, but also emerges as a trust-worthy and distributed solution for reputation management [25, 30]. Since reputation is earned in interactions between peers, it can be attached to transactions. Decentralized ledger technologies such as the New Economy Movement (NEM) [19] calculate importance scores for peers. According to NEM, "importance cannot be arbitrarily manipulated or gamed, and importance scores are useful for purposes other than just blockchain consensus, being interpreted as a form of reputation" [19].

This paper is organized as follows. Section II overviews general concepts applied in current reputation systems. Section III describes how the reputation scheme is embedded into the mitigation protocol. The architectural design and implementation are described in Section IV. Section V explains methodology and configuration, including the execution of major experiments. The discussion of experimental results, limitations, and robustness against different forms of rating fraud follows in Section VI. Finally, Section VII concludes the work.

II. Reputation System Concepts

Reputation systems have been widely deployed in different business areas and application contexts. There exists a multitude of use cases for reputation and incentive mechanisms that ranges from crowdsourcing and sensing [7, 33], MANETs (Mobile and Ad-hoc Networks) [8, 15], Border Gateway Protocol (BGP) routing [11], E-commerce [3], file-sharing [29], mobile data plan sharing [18], prediction markets [21], various P2P and blockchain networks and cooperative DDoS defense [22, 25]. General properties of reputation systems are identified in Section II-A and, in the following, probabilistic reputation engines are introduced in Section II-B.

A. System Properties

Reputation systems have a few common properties [11]:

- **Foundation of trust**: Repeated interactions and a clear interaction history build a reliable foundation of trust.
- **Self-policing nature**: The system should be self-policing, with the social norms defined by the users and not by a central authority [14].
- **Carrots and sticks**: With incentives and penalties, users are more likely to behave according to the social norm.
- **Robustness**: The system should be robust against gaming attempts in reputation systems [9]. It needs not be foolproof, but reasonably secure against collusion and Sybil attacks, bad-mouthing (bad ratings), ballot stuffing and identity whitewashing through re-entry. Current literature examines rating fraud, as a type of information fraud (see Figure 2). The goal of an attacker domain is to either increase the reputation of itself (ballot stuffing) or to decrease the reputation of others (bad-mouthing). The attacker can achieve its goals using a variety of attack models. A constant attacker behaves consistently evil, whereas a disturbing agent camouflages its actions skilfully.
- **Accurate and verifiable scoring engine**: The scoring engine should provide accurate and verifiable metrics.

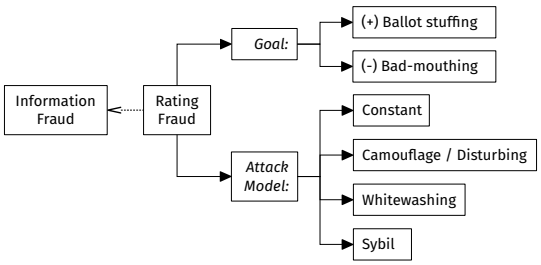

Fig. 2: Fraudulent rating classified by goal and attack [4, 23]

- **Anonymity and privacy**: The personal user feedback should be collected anonymously or bound to a pseudonym, to ensure honest feedback and guarantee some degree of privacy [7]. In a decentralized, anonymous marketplace (DAM) [24] the reputation system requires additional properties to ensure a fair and complete listing of items, payments, and reviews. Most importantly, such a marketplace ensures the unlinkability of reviews and payments (with associated customer information), without compromising the legitimacy of the review [24]. The fully anonymous reputation system developed for the DAM "Beaver" [24] makes use of advanced cryptographic techniques, such as ring-signatures and zero-knowledge proofs, to uphold stringent security and anonymity requirements. These techniques allow raters to stay anonymous by veiling the source of the reputation claim, but without compromising the validity of the rating.

B. Probabilistic Reputation Engines

Surveys and classification about different types of reputation engines are presented by Jøsang *et al.* in [13] and Schlosser *et al.* in [23]. This work focuses on a probabilistic engine, because this type of engine is suitable for the design of binary reputation systems [2]. In probabilistic engines, reputation is expressed as a probability [2, 27]. The expected value of the beta distribution forecasts the probability of a positive, future event x, based on the past binary events x and $\bar{x}$. The binary events represent positive and negative historical reputation ratings. The expected value of this distribution can be interpreted as a reputation score [2].

One particular example of a Beta reputation system applied to blockchain is found in the Topl protocol [5]. Topl is a proposed blockchain protocol to create profit-sharing agreements with producers in emerging and frontier markets. The reputation engine "Divine" of the Topl protocol builds upon a Beta reputation engine to facilitate due diligence and reduce counterparty risk [16].

In this work, likewise to [2], Beta reputation for a customer c is calculated as the expected value $E(p) = \frac{\alpha}{\alpha+\beta}$, with $\alpha = positive(c) + 1$ and $\beta = negative(c) + 1$ being the accumulated positive and negative reputation values at one point in time. This value reflects the probability of future positive interaction with customer c, based on its past positive and negative ratings. This Beta function outputs reputation scores in the range of $[0, 1]$, where 0.5 is the initial, neutral score of every new customer.

III. Mitigation Protocol

Figure 3 depicts the protocol for mitigation or defense services by a mitigator M (*i.e.* the requested domain), in response to a DDoS attack at the target domain T (*i.e.* the requester or attack target). It defines how the customers should behave in a cooperative defense. The rational and satisfied customers are referred to as the "desired customer strategies". When M submits proof of blocking, there is no automated way for other peers to build a consensus on the quality of the delivered service. The receipt in a "payment-for-receipt" exchange process cannot be automatically issued by a smart contract, because in the worst case, any upload is accepted as successful delivery and T would pay for a worthless receipt. Attack size and amount of payment are relevant factors that could result in serious financial losses for T. In order to avoid this problem, this reputation protocol here depends on T to rate the outcome of the mitigation service until a validation deadline. Thus, the "nature of the proof" is abstracted, because finding a robust representation and verification thereof presents a current limitation in cooperative defense (see Section VII).

The protocol implements a final rating deadline in addition to the service and validation deadline. A dissatisfied M can still dispute a rejected service as long as the rating deadline has not expired. Afterward, M is assumed to no longer be interested in the payout or having accepted T's rating. The rules of the smart contract define a positive rating by M as the only allowed response to a positive rating (ack) from T. In the same way, receiving negative or no feedback from T cannot (altruistically) be rewarded with a positive rating by M. Nevertheless, a dispute process should be in place for customers to challenge any payout or rating.

As independent observers, we cannot make a statement about the truthfulness or quality of the (proof of) service. Even though blockchain technology can preserve an audit trail for all transactions, it cannot compensate for lack of ground-truth [4]. This holds for the uploaded proof of service as well as for user-defined, subjective ratings. There is no automated way to determine the truthfulness of proof or rating fully. Hence, the correct input of customers remains of paramount importance for the quality of the system as a whole.

Nonetheless, as soon as T decided on the quality of the delivered mitigation service, this information is published by rating and validating the mitigation service. In this sense, T cannot be malicious, only satisfied or dissatisfied. If T is honest, it either rates positively and acknowledges (satisfied) or rates negatively and rejects the service (dissatisfied). Then, a rational M always rates according to the expectation of T. This rational M responds to a rejected service by rating T negatively. However, if the rational M received positive feedback from T, it will also respond positively. When there was no feedback (*i.e.* T selfish), a rational M is only allowed to rate negatively.

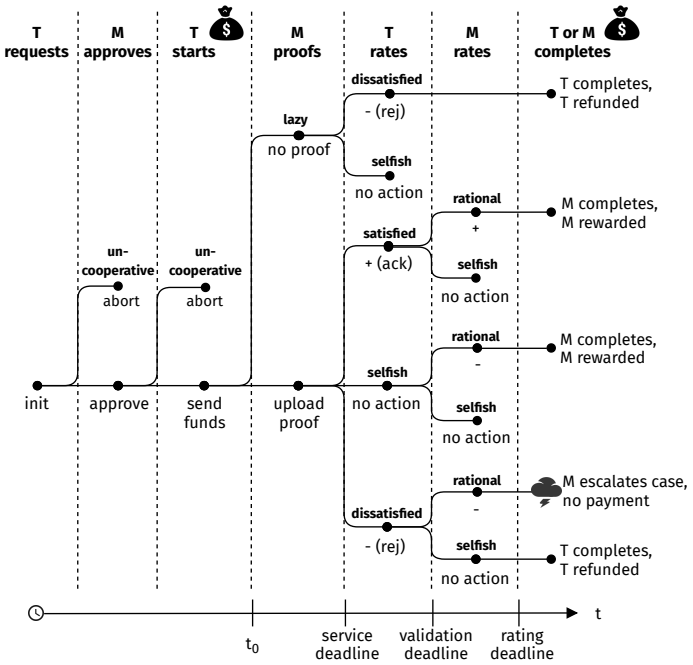

Fig. 3: Customer strategies in the mitigation protocol

IV. Design and Implementation

The software architecture of the prototype in this work is inspired by [31], with slight modifications [1]. Whereas [31] uses PostgreSQL for reputation caching, this work here creates a mitigation task index in a MongoDB database. Based on that data, the reputation API provides reputation metrics to interested consumers over well-defined endpoints. An identity smart contract holds the identities of all participating autonomous systems or domains. A partial **anonymity** (*i.e.* pseudo-anonymity or pseudonymity) is preserved by disguising the real identity of the domain and maintaining a pseudonym in form of an ID (*i.e.* Ethereum account addresses). However, for a particular reputation claim, the claim owner and rated customer and the interaction are traceable within the blockchain.

Customer data, including reputation scores, mitigation offers, and tasks are connected to the customers' ID but stored off-chain whenever possible. The reputation contract stores IPFS hashes, that link to reputation data (reputons) about past interactions. Reputation scores and historical data about a peer's behavior in different contracts (*e.g.* peer age, number of tasks solved, number of interactions) can be retrieved from this reputation contract or the public blockchain transaction history. This data creates the **foundation of trust** and is visible to all peers in the system. The attack information broadcast is also assumed to be public. Potential mitigators receive notifications about malicious IPs via blockchain. Based on publicly available reputation data, both contract parties decide whether to enter into a contractual relationship. The reputation API aggregates reputation data and provides it to the interested peers. This API should allow verifying the integrity of the provided information. In this architecture, the API represents the **accurate and verifiable scoring engine**. Since consumers can query the blockchain transaction history independently, the information provided by this API can be proven wrong any time.

The following sections describe how tasks are indexed, how reputation is stored off- and on-chain and how the reputation metrics are presented to consumers of the reputation API.

A. Mitigation Task Index

The reputation API was built with the Ruby on Rails (RoR) web application framework. A scheduler appends new contract information to the local index at an interval of 15 seconds. Since we assume that data on the ledger is immutable (*e.g.* mitigation contract participants never change), new data can be appended to the local database in this periodic manner. The database is updated using a simple counter (`known_cnt`), to mark the position of the last update (see Listing 1). Thus, information appended previously will not be stored or processed twice. The same caching procedure could be applied to cache reputons.

```
def get_latest_tasks
  task_cnt = contract.call.task_cnt
  known_cnt = REDIS.get('known_cnt') || -1
  # append only new tasks
  (known_cnt + 1...task_cnt).each do |task_index|
    # fetch task from the ledger
    task = contract.call.tasks(task_index)
    REDIS.set('known_cnt', task_index)
    # create a new task entry in the database
    MitigationTask.create(
      mitgn: MITIGATION_CONTRACT_ADDRESS,
      _id: task_index,
      target: task[0],
      mitigator: task[1])
  end
end
```

Listing 1: Periodically append tasks to local task index

B. Storage of Reputation Data

Reputation storage is implemented partially off- and on-chain. The actual reputation data is not stored in the database of the API but retrieved directly from the IPFS gateway. The IETF standard media type for reputation exchange [17] enables the interpretation and exchange of reputation claims. Claims are a way to register a reputation statement that a source entity makes about a target entity [10]. The reputon media type represents these claims in JavaScript Object Notation (JSON). These JSON objects are stored on IPFS. Listing 2 shows a reputation statement about a mitigator.

```
{ "application": "mitigation",
  "reputons": [{
     "rater": "0xf43...",
     "assertion": "proof-ok",
     "rated": 20,
     "rating": 0,
     "sample-size": 1 }] }
```

Listing 2: Reputation statement about a mitigator

[1]The prototype is available at https://github.com/in0rdr/prototype

The properties of such a reputon can be interpreted as follows:

- The `rater` records an `assertion` (reputation statement) for a customer. In Listing 2, the reputation statement was made about the mitigation service delivered by a mitigator. The receiving side (*i.e.* the "rated" customer) of the claim is implicitly recorded in the mitigation smart contract, since every mitigation contract is created for exactly two customers.
- The `rated` property of the reputon records the mitigation task ID.
- A `rating` value of zero is a negative feedback. The `rating` value of one is interpreted as a positive reputation statement. Intuitively, when no reputon is supplied for a mitigation task and customer, there exists no `rating` property and the rating for this customer is interpreted as "neutral".
- The `sample-size` specifies "the number of data points used to compute the rating" [17] and is fixed to 1 since the reputation statement originates from exactly one customer. Even though not being a required attribute by the media type specification [17], it improves the information quality of the reputation statement.

The task index of the API contains the task IDs and contract participants. To obtain reputation data, the API controller sets off a read request to the blockchain. An example to fetch all reputon IPFS keys for a customer (in the role of attack target) is given in Listing 3.

```
def get_target_reputons(addr)
  target_tasks = MitigationTask.where(target: addr)
  target_tasks.map do |t|
    # get mitigator claims about target
    { ipfs_key: CONTRACT.call.get_reputon(t._id, 1),
      task_id: t._id }
  end
end
```

Listing 3: Fetch IPFS reputon keys

The reputation smart contract stores the IPFS hash of every reputation claim (see Listing 4). The `reputons` mapping avoids duplicate claims and registers claim owners (*i.e.* the "rater"). The Solidity state variable `interactions` maps every mitigation task ID to an array of size two. Every task can have at most two reputons assigned. First, the reputation statement from the target about the mitigator and second, the reputation statement from the mitigator about the attack target. These reputons hold complete reputation information, including metadata. The `ratings` state variable records the binary reputation value to conclude mitigation tasks.

```
uint256 public reputonCount;
mapping(string => address) reputons;
mapping(uint => string[2]) interactions;
mapping(uint => uint[2]) ratings;
```

Listing 4: On-chain reputation storage

C. Implementation of Reputation Metrics

The reputation API presents reputation summaries to consumers. An example of such a summary for an arbitrary customer is given in Listing 5. The `rating_summary` map reveals, that customer `0x113...` earned more negative than positive reputation for his mitigation service. The average satisfaction with the mitigation services of this customer amounts to $sat(c) = \frac{positive(c)}{negative(c)} = 0/2$ which is 0%. The reputation summary also shows, that this customer received no feedback and thus `neutral` ratings in three of total five interactions. These are only the interactions for which this customer was enrolled as a mitigator. A similar summary can be obtained to learn about the reputation claims obtained in tasks, where customer `0x113...` was enrolled as the attack target. The reputation computation is based on the task IDs in the `rating_source` map of the summary. This information can be used to verify the correctness of the reputation summary.

```
{ "rating_source": {
    "negative": [20, 26],
    "neutral": [2, 8, 14],
    "positive": []
  }, "rating_summary": {
    "negative": 2,
    "neutral": 3,
    "positive": 0 } }
```

Listing 5: Reputation summary for mitigator `0x113...`

V. Evaluation

The goal of the experiments was to determine a setting, where constant (see Figure 2) desired customer behavior is distinguishable from undesired or even malicious behavior by looking at reputation values. The simulations were performed in the Communication Systems Network Lab at the University of Zurich. More precisely, the setup comprised two identically equipped Dell XPS machines, with Intel Core i7 CPUs (4 Cores, 3.40 GHz) and 8 GiB of RAM each. The first machine hosted an Ethereum boot node and a mining (validating) peer. The second machine hosted a non-mining `geth` peer and the Node.js simulator script. Also, an Ethereum network statistics dashboard (eth-netstats) has been deployed for monitoring purposes on this second machine. The simulator script was written using the Web3 Ethereum JavaScript API.

A. Evaluation Metrics

Reputation points used to rate target and mitigator are preferably not the same because a task owner with a good rating does not necessarily need to be a good mitigator and vice-versa. Therefore, the reputation earned as mitigator is stored separately from the reputation earned as attack target or task owner. A simple metric gives the analyzing peers a clear understanding of a peers reputation. Added to this, a complex metric might lead to negative feedback loops [10]. The scores earned by ratings from other peers (subjective ratings) and the total amount of transactions (objective metrics) are equally important to have a good understanding of a peers trustworthiness [32]. Therefore, this system presents objective (*e.g.* number of transactions/interactions) and subjective metrics (*e.g.* ratings) to consumers (see Section IV-C). This approach allows peers to calculate a simple trust metric, which helps them to decide

on trustworthy business partners. Beta reputation is assumed to be a suitable metric for this evaluation, since this is a binary reputation system (see Section II-B). According to Jøsang and Ismail (2002), this metric builds "a sound mathematical basis" [2]. A "balance factor of trust" to correct for the trustworthiness of the subjective ratings was not implemented.

B. Customer Strategies

Tables I and II present the numerical evaluation of all customer strategies, which match their analytical specification in Figure 3. These two tables show the state of the simulated world after all customers complete a mitigation contract with each other once. The simulation function takes as an input one mitigator strategy and one target strategy. After the simulation halts, the output analyzed in this evaluation here is the end state of the mitigation tasks in the simulated world (Table I) and the reputation of the customers (Table II). In this way, all 16 possible combinations of customers (4 targets $\times$ 4 mitigators) have been evaluated, to ensure the correct behavior of these customer strategies.

Table I lists all possible combinations of mitigator-target strategies and the mitigation contract ID. Completed tasks are either aborted before payment or paid out successfully. Because selfish and lazy customers never rate, tasks 5 and 6 cannot be completed by either party. Compared to the lazy customer, the selfish one does upload a proof but never rates, being the reason why the end states for task 5 and 6 are different. Task 15 is the typical escalation case, where a dissatisfied target and a rational mitigator would argue about the truthfulness of the proof of service. In this case, no payment is made, and end state is "rejected".

TABLE I: Task configurations and end states

Input: Customer Strategy		**Output: End State**	**Task ID**
Target	**Mitigator**		
Uncooperative	Uncooperative	Completed	0
	Lazy		1
	Selfish		2
	Rational		3
Selfish	Uncooperative		4
	Lazy	Started (funds sent)	5
	Selfish	Proof uploaded	6
	Rational	Completed	7
Satisfied	Uncooperative		8
	Lazy		9
	Selfish		10
	Rational		11
Dissatisfied	Uncooperative		12
	Lazy		13
	Selfish		14
	Rational	Rejected	15

Table II shows the number of positive, neutral and negative ratings for all customer strategies. The satisfied target receives one positive rating from the rational mitigator. The selfish and rational mitigator receive one negative feedback from the dissatisfied target. Since they upload proofs, they also receive one positive rating from the satisfied target. The selfish and dissatisfied target receive one negative feedback from the rational mitigator. The lazy mitigator is rated negatively twice by the satisfied and dissatisfied targets because no proof was uploaded. The data in this table stems from one specific test run but can be replicated by re-running the simulator script. Each simulator test creates customers with their particular strategy in the same deterministic order. The reputation results after another test run in a new environment only differ in customer addresses.

TABLE II: Reputation values and total number of interactions

Input:	**Output: Ratings**			**Output:**
Customer Strategy	**Positive**	**Neutral**	**Negative**	**Interactions**
Satisfied T	1	3	0	4
Selfish M	1	2	1	4
Rational M	1	2	1	4
Uncooperative T	0	4	0	4
Uncooperative M	0	4	0	4
Selfish T	0	3	1	4
Dissatisfied T	0	3	1	4
Lazy M	0	2	2	4
Total:	**3**	**23**	**6**	**32**

C. Evolution of Customer Reputation

When continuous DDoS attacks and the creation of mitigation tasks are simulated, mitigation contracts are created on a regular basis and two randomly chosen agents play the repeated game against (or with) each other. Since, in real-world scenarios, customers are unwilling to interact with unreputable colleagues, there exists an acceptance criteria, such that the two assigned agents start the game considering past reputation scores. The acceptance function takes as input the past reputation of the counterparty [23]. Based on this input, the acceptance function evaluates whether or not to advance the mitigation task. The simulated customers continue with the mitigation actions if the Beta reputation of the counterparty exceeds a value of 30%. For this evaluation we choose 30% to give each customer a chance to change its behavior before exclusion. Beta reputation for a customer c was calculated as defined in Section II-B.

Varying time-windows were chosen for the deadlines, such that they might lead to situations, where customers miss to meet a deadline. For example, the code that simulates a mitigator uploading a proof with the minimum service deadline of three blocks might not be executed fast enough, because the block time of the blockchain increases faster due to other transactions being processed. The service, validation and final rating/escalation deadlines are sampled randomly for each task. Service deadlines are chosen in a range of $[3, 13]$ blocks and validation deadlines in range of $[17, 27]$ blocks. Final rating deadlines are sampled in the range of $[32, 42]$ blocks. Before the new mitigation contract is created, all peers are funded with 10 Ethers each. The contract price was fixed to one Ether for every mitigation task.

The simulations were performed using different compositions of customer strategies (see Table III). The experiments only differed in measurement duration (from one up to 10

days) and the number of customers. Figure 4 presents the average Beta reputation by time and customer strategy for the third test run.

TABLE III: Distribution of customer strategies

Strategy	**Number of Customers**		
	First Run (1 day)	**Second Run (5 days)**	**Third Run (10 days)**
Rational M	10	20	50
Satisfied T			
Uncooperative T	10	20	50
Selfish T			
Dissatisfied T			
Uncooperative M			
Lazy M			
Selfish M			
Total Number of Customers:	**80**	**160**	**400**

VI. Discussion

To the best of the author's knowledge there is no concrete (or remotely comparable) implementations of BloSS that produce real, historical transaction data. Therefore, the evaluation of the reputation scheme using simulated data is still limited in meaningfulness. However, all simulated interactions between the different types of customers allow for a discussion about how reputation could evolve in the system proposed and how a customer with a particular strategy would perform.

Overall, Beta reputation scores for attack targets (Ts) allow identifying a satisfied T, because in comparison with the undesired T strategies it develops the highest average reputation value (*cf.* Figure 4). From Figure 4 it also becomes evident, that dissatisfied and selfish Ts are downgraded continuously and will eventually no longer receive help from mitigators (Ms), due to their lousy rating. It is not desirable in all circumstances that satisfied Ts crowd out dissatisfied Ts, since this will incentivize Ts to accept also poorly (or even severely) delivered mitigation services. Because of the current design, no third party checks that a rating indeed matches the quality of the delivered service, this problem remains yet unsolved.

After the first few mitigation contracts, M reputation values in Figure 4 also allow to clearly distinguish a constant rational M from the lazy M. However, unlike for the Ts, it is difficult to distinguish the rational M from the selfish and uncooperative M. The uncooperative M usually shows very low positive and negative reputation, because it aborts mitigation tasks early, never takes any action, thus receives little feedback and its reputation stagnates at 50%, not performing particularly good or bad. Hence, chances of picking an uncooperative M for a transaction is diminished the most by considering raw reputation values, especially the amount of positive and negative ratings. Because both, the selfish and rational M, upload proofs, the only difference between the two strategies is that the selfish M never rates. Selfish M behavior is an irrational strategy, since the selfish M (deliberately or not) deprives itself of payment. Assuming that M rarely forgets to rate services at the end (which is the definition of selfish behavior), one can also assume that there exists more rational than selfish Ms. This leads to the conclusion that chances of picking a rational M compared to a selfish M with the same Beta reputation score are higher, due to the incentive.

Sybil- and Collusion Attacks. The reputation system prototype excludes the possibility where a customer can boost its reputation by creating mitigation contracts with itself. As long as the system is not deployed on a public ledger, customers can be prevented from maintaining multiple account pseudonyms on the blockchain and transacting between them to inflate reputation (Sybil attack). Hence, an authority controlling access to the customer accounts is required. Since trustworthy authorities give or revoke write permissions for a consortium blockchain, these authorities regulate and manage customer data. They can be held responsible for matching real-world entities with customer accounts in order to prevent Sybil situations. The BloSS system already relies on certificates to assert domain ownership [25]. A similar approach can be defined to manage customer accounts and pseudonyms of the reputation module. The impact of authority on whitewashing (*i.e.* re-entry) attacks were not analyzed in this work. Unfortunately, the need for regulating authorities weakens the self-regulatory nature of the reputation system implemented, however the use of certificates required runs along the same argument.

Ballot Stuffing and Bad-mouthing. This system is not immune against ballot stuffing. Besides the transactions recorded on the blockchain, customers can agree on discounts and benefits over alternative communication channels [4]. An example of such an arrangement would be two customers rating each other positively independent of the mitigation outcome (*i.e.* fake service contracts). If they refund the price of mitigation, the price for improved reputation equals the contract setup costs. Contrary to ballot stuffing, the blockchain-based reputation system design impedes bad-mouthing [4]. A customer can only leave feedback for completed transactions. This elevates the cost barrier of bad-mouthing a competitor, because a transaction needs to be committed for every fraudulent reputation statement [4].

Verifiability and Anonymity. The open architecture of the prototype allows consumers to verify reputation metrics by issuing queries to the reputation API. In the current design of the system, verifiability of the reputation metric leads to reduced anonymity. In turn, this only provides limited security for the rater.

Limitations. The baseline scenario presented in Section III, where exactly two customers work together under exactly one mitigation contract and where they act following this protocol, might be too simplistic for real-world scenarios. In some instances, a contract between multiple parties with additional steps during attack mitigation could be required. A $1 : n$ defense situation with one attack target and n mitigators can be solved with n "$1 : 1$ mitigation contracts", for instance, if large distributed CDNs support DDoS attack mitigation by temporarily serving the content to users. Also, a different mitigation smart contract can replace or extend the mitigation module. Thus, the understanding and investigations of the basic set-up will help to combine findings of more complex

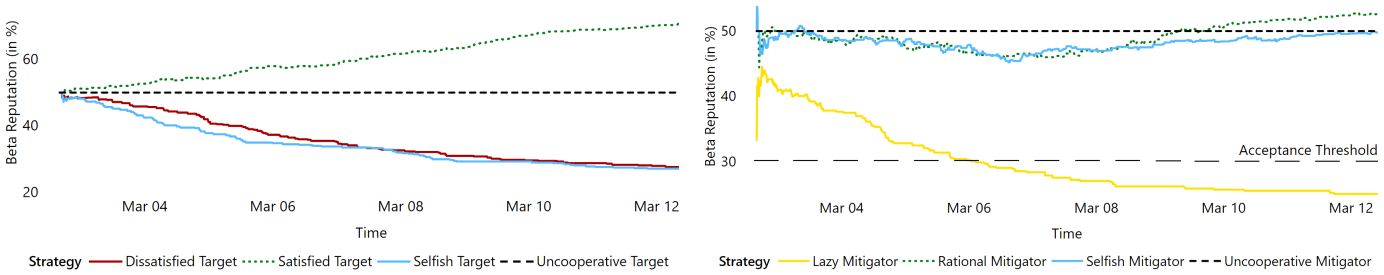

Fig. 4: Average Beta reputation for targets and mitigators

settings.

The blockchain used for prototyping the system relies on Proof-of-Work (PoW) consensus to confirm transactions. This has the disadvantage of varying block times, which among other determinants also depends on the current difficulty of the math problem. Since parties to a mitigation contract arrange fixed deadlines in blocks, the entire time span in seconds varies. Employing another consensus algorithm, for example, Proof-of-Authority (PoA), can solve this problem. No mining (*i.e.* solving complex math problems) is required in PoA systems, since blocks are appended to the chain by authorized peers only [20]. This comes with the benefit of predictable block times and thus steady time intervals (*i.e.* predictable deadlines) [20].

The evaluation with the target strategies as described in Figure 3 assumes that there are no lazy Ts that never start a task they initiated. This would be irrational behavior, since a rational T is motivated to see the issue resolved as soon as possible. Nonetheless, there could be a T that forgets about an approved task. Such behavior is undesired, since mitigator M has no feedback until the task is either started or aborted.

The simulator assumes that a customer participates in all contracts as either M or T, but not as both. In a real-world scenario, today's M can be tomorrow's T and vice-versa. The simulator was programmed in a simplified version in order to attribute reputation scores better to one customer address.

VII. Final Considerations and Future Work

This work presented a mitigation protocol for a cooperative defense, fostering trust and establishing a baseline toward rewards to be distributed. Thus, the protocol discourages free-riding (attack targets) and false-reporting (mitigators) in the long run by incentivizing the rational behavior between cooperative entities. System evaluation tested different customer (*i.e.,* cooperative entities) strategies over 10 days of simulation, showing that non-rational customers had reputations points diminished over time in contrast to rational/satisfied customers. The system prevents Sybil and collusion attacks by mapping customer accounts to real-world identities, preventing customers from creating several identities to manipulate reputation scores. However, ballot stuffing and bad-mouthing are not prevented, but discouraged due to the cost to deploy a mitigation contract only to manipulate reputation scores.

As future work, it is intended to further investigate protocols or tools that can be used to ensure a reliable and automated proof-of-mitigation in a collaborative defense [26], serving as a basis to rate a mitigation service. Also, it is intended to investigate approaches to automate the resolution of escalation cases to decide whether a proof of mitigation satisfies the requested mitigation service. Further, assumptions about different types of customers and their respective strategies can deviate from the situation in a proper, cooperative DDoS defense, which only can be investigated based on real-life traces.

References

[1] Akamai. *How to Protect Against DDoS Attacks - Stop Denial of Service*. 2016. URL: https://goo.gl/EDTEim (visited on Dec. 4, 2017).

[2] Audun Josang and Roslan Ismail. "The Beta Reputation System". In: *Proceedings of the 15th Bled Electronic Commerce Conference*. Vol. 5. 2002, pp. 2502–2511.

[3] Alexander Schaub, Rémi Bazin, O. Hasan, L. Brunie. "A Trustless Privacy-Preserving Reputation System". In: *ICT Systems Security and Privacy Protection*. IFIP Advances in Information and Communication Technology. May 2016, pp. 398–411.

[4] Yuanfeng Cai and Dan Zhu. "Fraud Detections for Online Businesses: A Perspective from Blockchain Technology". In: *Financial Innovation* 2.1, Dec. 2016, p. 20.

[5] Christopher Georgen. *Topl, Empowering Growth by Enabling Investment*. 2017. URL: https://goo.gl/Z4uhL7.

[6] CloudFlare. *CloudFlare Advanced DDoS Protection*. 2016. URL: https://goo.gl/FNJwLP (visited on Dec. 4, 2017).

[7] Sergi Delgado-Segura, Cristian Tanas, and Jordi Herrera-Joancomartí. "Reputation and Reward: Two Sides of the Same Bitcoin". In: *Sensors* 16.6, May 2016, p. 776.

[8] Mieso K. Denko. "Detection and Prevention of Denial of Service (DoS) Attacks in Mobile Ad Hoc Networks Using Reputation-Based Incentive Scheme". In: *Journal of Systemics, Cybernetics and Informatics* 3.4, Jan. 2005, pp. 1–9.

[9] Richard Dennis and Gareth Owen. "Rep on the Block: A next Generation Reputation System Based on the Blockchain". In: *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*. Dec. 2015, pp. 131–138.

[10] F. Randall Farmer and Bryce Glass. *Building Web Reputation Systems*. 1st ed. 2010.

[11] Harlan Yu, Jennifer Rexford, E. W. Felten. "A Distributed Reputation Approach to Cooperative Internet Routing Protection". In: *1st IEEE ICNP Workshop on Secure Network Protocols, 2005. (NPSec).* Nov. 2005, pp. 73–78.

[12] IETF. *Distributed Denial of Service (DDoS) Open Threat Signaling Requirements*. 2017. URL: https://datatracker.ietf.org/wg/dots (visited on Nov. 23, 2017).

[13] Audun Jøsang, Roslan Ismail, and Colin Boyd. "A Survey of Trust and Reputation Systems for Online Service Provision". In: *Decision Support Systems*. Emerging Issues in Collaborative Commerce 43.2, Mar. 2007, pp. 618–644.

[14] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. "The Eigentrust Algorithm for Reputation Management in P2p Networks". In: *Proceedings of the 12th International Conference on World Wide Web*. 2003, pp. 640–651.

[15] Rizwan Khan and Avimanyou Vatsa. *Detection and Control of DDOS Attacks over Reputation and Score Based MANET*. Nov. 2017.

[16] Matt Kindy. *Divine: A Blockchain Reputation System For Determining Good Market Actors*. June 2017. URL: https://goo.gl/7wU7CY (visited on Nov. 22, 2017).

[17] Murray Kucherawy and Nathaniel Borenstein. *A Media Type for Reputation Interchange*. URL: https://tools.ietf.org/html/rfc7071 (visited on Nov. 3, 2017).

[18] Tuo Yu, Zilong Zhou, D. Zhang, X. Wang, Y. Liu, S. Lu. "INDAPSON: An Incentive Data Plan Sharing System Based on Self-Organizing Network". In: *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*. Apr. 2014, pp. 1545–1553.

[19] *NEM: Technical Reference*. Tech. rep. 2015.

[20] *Parity, Proof of Authority Chains*. URL: https://goo.gl/v7UDhu (visited on Feb. 12, 2018).

[21] Jack Peterson and Joseph Krug. "Augur: A Decentralized, Open-Source Platform for Prediction Markets". In: *arXiv preprint arXiv:1501.01042*, 2015.

[22] Bruno Rodrigues, Thomas Bocek, and Burkhard Stiller. "Enabling a Cooperative, Multi-Domain DDoS Defense by a Blockchain Signaling System (BloSS)". In: *Semantic Scholar*, 2017.

[23] Andreas Schlosser, Marco Voss, and Lars Brückner. "Comparing and Evaluating Metrics for Reputation Systems by Simulation". In: *Proceedings of the IEEE Workshop on Reputation in Agent Societies*. 2004.

[24] Kyle Soska et al. "Beaver: A Decentralized Anonymous Marketplace with Secure Reputation." In: *IACR Cryptology ePrint Archive* 2016, 2016, p. 464.

[25] Bruno Rodrigues, Thomas Bocek, A. Lareida, D. Hausheer, S. Rafati, B. Stiller. "A Blockchain-Based Architecture for Collaborative DDoS Mitigation with Smart Contracts". In: *Security of Networks and Services in an All-Connected World*. Vol. 10356. July 2017, pp. 16–29.

[26] Stephan Mannhart, Bruno Rodrigues, E. Scheid, S. Kanhere, B. Stiller. "Towards Mitigation-as-a-Service in Cooperative Network Defenses". In: *The 3rd IEEE Cyber-Science and Technology Congress*.

[27] Thomas Bocek, Michael Shann, D. Hausheer, B. Stiller. "Game Theoretical Analysis of Incentives for Large-Scale, Fully Decentralized Collaboration Networks". In: *2008 IEEE International Symposium on Parallel and Distributed Processing*. Apr. 2008, pp. 1–8.

[28] Matthew Buechler, Manosai Eerabathini, C. Hockenbrocht, D. Wan. *Decentralized Reputation System for Transaction Networks*. Tech. rep. Dept. of CIS - Senior Design, University of Pennsylvania, Philadelphia, PA, 2015.

[29] Yao Wang and Julita Vassileva. "Trust and Reputation Model in Peer-to-Peer Networks". In: *Proceedings Third International Conference on Peer-to-Peer Computing (P2P2003)*. Sept. 2003, pp. 150–157.

[30] Zibin Zheng, Shaoan Xie, H. Dai, X. Chen, H. Wang. "Blockchain Challenges and Opportunities: A Survey". In: *Work Pap*, 2016.

[31] *Work.Nation: Decentralized Skill Attestations Using uPort, Ethereum and IPFS*. Oct. 2017. URL: https://github.com/worknation/work.nation.

[32] Li Xiong and Ling Liu. "Building Trust in Decentralized Peer-to-Peer Electronic Communities". In: *International Conference on Electronic Commerce Research (ICECR-5)*, Feb. 2003.

[33] Yu Zhang and Mihaela van der Schaar. "Reputation-Based Incentive Protocols in Crowdsourcing Applications". In: *arXiv:1108.2096 [physics]*, Aug. 2011.