Beaver: A Decentralized Anonymous Marketplace with Secure Reputation

Kyle Soska*
CMU
ksoska@cmu.edu

Albert Kwon*
MIT
kwonal@mit.edu

Nicolas Christin
CMU
nicolasc@cmu.edu

Srinivas Devadas
MIT
devadas@mit.edu

ABSTRACT

Reputation systems play a crucial role in establishing trust online, especially in e-commerce settings. Users in reputation systems provide feedback for other users, thereby incentivizing good behavior and disincentivizing bad behavior. With growing concerns of government surveillance and corporate data sharing, it is increasingly common that users on the web demand tools for preserving their privacy without placing trust in a third party. Unfortunately, existing centralized reputation systems need to be trusted for either privacy, correctness, or both. Existing decentralized approaches, on the other hand, are either vulnerable to Sybil attacks, present inconsistent views of the network, or leak critical information about the actions of its users.

In this paper, we present Beaver, a decentralized anonymous marketplace that is resistant against Sybil attacks on vendor reputation, while preserving the anonymity of its customers. Beaver allows its participants to enjoy free open enrollment, and provides every user with the same global view of the reputation of other users through public ledger based consensus. Our use of various cryptographic primitives allow Beaver to offer high levels of usability and practicality, along with strong anonymity guarantees.

1. INTRODUCTION

Reputation systems play a crucial role in establishing trust in online communities and drive many modern online businesses, ranging from auction markets to transportation companies. A typical reputation system features a collection of actors executing a protocol that allows users to leave reviews for their interactions with each other. Reviews, or feedback, usually consist of numeric ratings (e.g., 1–5 stars) and/or a short message. Feedback accumulates over time, and can be queried by other users in the system.

Many of the best known e-commerce businesses (e.g., eBay, Amazon Marketplace, Uber, AirBnb) operate as centralized marketplaces. Users need to trust the marketplace operator to maintain the correctness of the reputation state. In addition, users also need to trust the marketplace operator to maintain the confidentiality of sensitive information such as payment history (i.e., that it will not leak it or sell it to third-parties). Stated differently, centralized marketplaces provide extremely weak privacy guarantees, which require the users to place full trust in the marketplace itself. Surveys of user sentiment suggest users are increasingly reluctant to putting such blind faith in commercial entities whose privacy policies have frequently shown to be questionable [35].

An interesting new development in the realm of online reputation is that of online anonymous marketplaces [11, 34], frequently referred to as "darknet marketplaces." These marketplaces are built on the idea of anonymous commerce—they attempt to provide strong anonymity guarantees to buyers, sellers, and even marketplace operators. While online anonymous marketplaces have so far been primarily used for contraband and illicit items, the more interesting point is that they strive to avail reputation systems with strong privacy and anonymity guarantees, and have proven to be economically viable. To achieve the desired anonymity properties, online anonymous marketplaces build on a combination of network-level anonymity—often running as Tor hidden services [15] or i2p "eep sites" [1]—and payment-level anonymity, using pseudonymous digital payment systems such as Bitcoin [29].

However, similar to "traditional" online marketplaces such as eBay, an online anonymous marketplace remains a centralized service. It thus needs to be trusted for availability as customers cannot query item listings or reviews if it is not online; it needs to be trusted for correctness, i.e., not inject fake reviews or suppress real ones; and it needs to be trusted not to link transaction history with private identifiers (e.g., shipping addresses communicated to vendors). Takedowns—e.g., of the Silk Road or Silk Road 2.0 marketplaces—and the associated arrests of some of their patrons have evidenced that such centralized marketplaces often fail to provide the level of trust their users expect. Besides takedowns, "exit scams" frequently occur [34]: in an exit scam, a marketplace unexpectedly closes, absconds with money left in escrow (collected from a customer, but not yet paid to a vendor), and destroys all reputation information in the process. These shortcomings motivate the search for a solution that can provide strong anonymity without trusting a third party: in other words, a decentralized anonymous marketplace. The recently proposed OpenBazaar prototype [2] is one such distributed effort, but it currently does not provide strong anonymity properties. For instance, OpenBazaar relies on the UDP protocol and does not readily support network-level anonymization techniques such as Tor.

More fundamentally, decentralizing reputation systems has proven to be a challenging task. Early works (e.g., [12, 22]) present peer-to-peer/sensor network algorithms in which a node queries its peers to obtain the reputation for another node in the network. These approaches come with the drawback that each node's view of the network is biased by that of its peers. Another important challenge in decentralizing any reputation system, especially a system that protects users' anonymity, comes from the threat of Sybil attacks [17]. In a Sybil attack, an adversary creates a large number of identities in the network (customer accounts, nodes, etc.) and uses them to either inflate her own reputation or damage the reputation of her competitors. Intuitively, there seems to be a fundamental tension between the ability to identify a Sybil attack and the requirement that customers remain anonymous: *How can one be sure feedback is legitimate without knowing any information about its source?*

*Joint first authors.

In this paper, we introduce a formal model for a decentralized anonymous marketplace (DAM), and design Beaver, a Sybil-resistant DAM. Beaver is designed with e-commerce in mind, and consists of three types of participants: customers, vendors, and network miners. Unlike most existing approaches, participation in Beaver is free, open, and does not use a trusted third party. From the perspective of customers and vendors, Beaver behaves nearly identically to existing e-commerce systems such as Amazon Marketplace and eBay. It allows vendors to establish reputation by selling items to customers while ensuring that vendor reputation has not been adversely modified either positively or negatively. Beaver simultaneously provides strong anonymity to its customers, in that, unless the customer explicitly provides this information, no adversary can learn which purchases a customer has made or associate reviews with particular transactions better than by randomly guessing.

Beaver builds on an anonymous payment system (e.g., Zerocash [4]), consensus protocol (e.g., Bitcoin "blockchain"), and various cryptographic primitives to present a globally consistent view of the network to all of its users without sacrificing anonymity. Due to this consensus construction, Beaver is also able to avoid attacks where a few customers are targeted, and convinced of incorrect statements about another user's reputation.

All interactions in Beaver are performed via the consensus protocol. Concretely, item listings created by vendors as well as payments made or reviews left by a customer to a vendor are publicly available as part of the consensus. With this, customers are able to freely and accurately enumerate all listings and feedback in the system, while deriving strong guarantees about the credibility of these reviews. Customers can also purchase products and leave their own reviews without fearing censorship or retribution. The major innovation in Beaver is that, although transactions and reviews are made public, the relationship between the transactions and reviews are kept private and the customers in Beaver always remain anonymous.

One of the key properties of Beaver is the mitigation of Sybil attacks. Traditional defenses against Sybil attacks rely on knowing the users' identities or their interaction history [28]. When the participants and their interactions are anonymous, as is the case with DAMs, such defenses cannot be deployed. Instead, we *anonymously* link reviews to transactions, by using non-interactive zero-knowledge proofs [7] and linkable ring signatures [24], which guarantees that there is a valid transaction for every review, and institute a small cost for each transaction. As a consequence, we can better understand and compute a notion that we call *credibility*, the lower bound on the cost to an adversary for generating feedback, and thus the trustworthiness of the current state of reputation. While Beaver is not Sybil-proof, we claim that it is Sybil-resistant under modest assumptions about the economic rationality of its participants.

Shortly stated, we make the following contributions in this paper: (1) we formalize the requirements for a decentralized anonymous marketplace (DAM), (2) we design a Sybil-resistant DAM, Beaver, which makes use of novel applications of consensus, anonymous payments, zero knowledge, and linkable ring signatures, (3) we analyze the security properties of Beaver.

In Section 2 we discuss background into anonymous reputation systems and Sybil attacks on reputation systems. In Section 3 we formalize a model for decentralized anonymous marketplaces (DAM). We then introduce, in Section 4, the high level design of Beaver and some preliminaries for understanding the full system design. We delve in the details of Beaver in Section 5, perform a security analysis in Section 6. Finally, we discuss some of our design choices in Section 7, related work in Section 8, and conclude in Section 9.

2. BACKGROUND

Reputation systems are used to collect, maintain, and distribute the performance scores of its users which can then efficiently be queried by other participants in the system. These performance scores can be used as a basis for establishing initial trust among users. Reputation plays an important role in online communities, especially in the context of e-commerce such as Amazon Marketplace and eBay. In this setting, a customer assumes risk both when ordering a product for which she cannot verify the quality and when purchasing directly from a vendor that she has no reason to trust.

2.1 Anonymous reputation system

Customers leaving public reviews that can be traced leads to negative consequences; based on review history, adversaries can learn a lot of information. Resnick and Zeckhauser [31] showed that vendors on eBay discriminate customers based on their review history. Moreover, if a vendor can associate reviews with transactions (which are often tied to sensitive information such as shipping addresses), then the vendor may try to harm the customer in the future. This type of behavior has been observed on darknet marketplaces and encourages customers to misreport their experiences as positive in order to avoid harassment. To establish a fair market and a useful reviewing system, it is essential that we ensure customers cannot be coerced into leaving inaccurate reviews. To achieve coercion resistance, we design a system where the reviews provably cannot be linked to particular transactions, and thus prevent adversaries from associating reviews with individual customers.

2.2 Decentralized reputation system

Given its importance, there has already been multiple proposals for anonymous reputation systems in the literature. While these systems provide anonymous reputation within their threat model, they rely on either a trusted third party [3, 5, 6, 14, 20], or require a trusted node among a set of powerful servers, known as the anytrust model [40]. Unfortunately, having a centralized network of a small number of nodes, even if only one needs to be honest, is undesirable as the adversary can perform targeted attacks. Moreover, in these systems, user enrollment is often expensive, requiring the user or the trustees to perform an expensive operation to incorporate new members into the system.

On the contrary, modern cryptocurrencies like Bitcoin [30] support a fully decentralized model. That is, users are allowed to join and leave the network at any time, and the security is guaranteed as long as some fraction of the network (rather than a small set of trustees) is honest. This makes it significantly harder for adversaries, especially those with limited resources, to undermine the security of the system. While Bitcoin and related cryptocurrencies do not support reputation or marketplace specific features, we draw inspiration from the way that they handle decentralization.

2.3 Sybil attacks

In general, a Sybil attack is an attack on distributed systems, where many nodes controlled by a few real entities cause the system to misbehave. In reputation systems, this attack concretely means that an adversary controls a large number of nodes and uses them to (1) generate positive reviews for himself to boost his reputation, and (2) leave negative reviews for others (e.g., his competition) to lower their reputation. Both scenarios are common in systems such as Yelp where users are free to review buisnesses without proving that they have ever been a customer. It has been shown by Mayzlin et al. [25] that allowing users to leave reviews without verifying that they have purchased the item generally leads to misbehavior. These attacks are also found in systems like eBay, where malicious ven-

dors purposefully increase their ratings via numerous fake transactions with positive reviews or decrease other vendors' ratings.

Sybil attacks in practice are mitigated by having (1) a central authority (e.g., CAs, Amazon, eBay) verify the identities of the users making it harder to sign up (e.g., require a phone number for account creation), (2) making sure a review came from a valid transaction, and (3) analyzing behaviors [38] to identify malicious users. Unfortunately, monitoring and limiting account creation is at odds with our goals for free open enrollment without a trusted third party, and associating reviews with transactions or tracking behavior is at odds with providing customer anonymity.

A general defense against Sybil attacks that does not require a central authority is to increase the cost of adversarial actions in the system, thereby discouraging an economically rational adversary. This is commonly done via expensive operations such as proof-of-work [21], CAPTCHAs, or by adding a fee to reputation generating actions. In this paper, we show that our system can charge fees (in a way that is particularly natural for marketplaces, and in many cases cheaper than existing markatplaces) to deter Sybil attacks, without sacrificing our goals of anonymity and decentralization.

3. DECENTRALIZED ANONYMOUS MARKETPLACE

In this section, we present definitions and goals for decentralized anonymous marketplace (DAM).

3.1 Definitions and notations

A DAM consists of different components that interact with each other through transactions, which are described here.

3.1.1 Components of DAM

Customers. Customers in a DAM make purchases from vendors, and leave reviews for their purchases if they choose to do so. We will let $\mathcal{C}$ be the set of honest customers $\{c\}$, and $\tilde{\mathcal{C}}$ be the set of malicious customers $\{\tilde{c}\}$.

Vendors. Vendors sell *items*, which could be any sort of good of monetary value. Similar to customers, we will let $\mathcal{V}$ be the set of honest vendors $\{v\}$, and $\tilde{\mathcal{V}}$ be the set of malicious vendors $\{\tilde{v}\}$.

Items. Items are any goods sold by a vendor in a DAM. We let $\mathcal{I}$ be the set of all available items, and let $i \in \mathcal{I}$ be an item. Each vendor v may control several items. Let $\mathcal{I}_v$ be the set of item listings controlled by v.

Reviews. Reviews are left by customers for an item, and we let $\mathcal{R}$ be the set of all reviews in the system.

Ledger. The ledger is a record of all transactions that have happened in the network, denoted $\mathcal{L}$. In certain DAMs, this maybe an abstract notion, rather than a concrete record.

3.1.2 Basic transactions

Customers and vendors in a DAM interact with each other via *transactions*. At minimum, a DAM needs to support (1) payment transactions and (2) review transactions, which eventually become part of the $\mathcal{L}$.

Payment transaction. This transaction moves funds from a customer to a vendor. A payment in a DAM is specified for purchase of a particular item:

- INPUTS:
  - Vendor's payment information
  - Item description
  - Coin
- OUTPUTS:
  - Payment transaction $\mathbf{p}$

Similar to reviews, we let $\mathcal{P}$ be the set of all payment transactions, and $\mathcal{P}_{(C,i)}$ be the set of all transactions for item i by customers $C \subset \mathcal{C}$. $\mathbf{p} \in \mathcal{P}$ indicates a particular transaction.

Review transaction. A customer generates this transaction when she wants to leave a review for an item. Because the reviews are exclusively generated by review transactions, we use the two terms interchangeably. Each review must be associated with a valid transaction.

- INPUTS:
  - Payment transaction
  - Reviews (numeric rating, messages, etc.)
- OUTPUTS:
  - Review transaction $\mathbf{r}$

We call $\mathcal{R}$ the set of all reviews, $\mathcal{R}_{(C,i)}$ the set of reviews for i by customers $C \subset \mathcal{C}$, and $\mathcal{R}_{(c,\cdot)}$ the set of all reviews left by c for any item listing. $\mathbf{r} \in \mathcal{R}$ indicates a particular review.

3.2 Properties of DAM

Apart from a decentralized model of trust, a DAM must also satisfy the following properties for functionality and security.

P1. Correctness: Any customer c who performs a correct payment transaction $\mathbf{p}_{(c,i)}$ for an item listing i can leave a review for item i with probability 1. This property ensures no customer is tricked into paying a vendor without the ability to review her experience. Note that we cannot in general promise fair trade for physical goods in a digital marketplace; we can only ensure that there is a way to report such behavior.

P2. Soundness: A customer c cannot leave a review for an item i without having performed a valid transaction $\mathbf{p}_{(c,i)}$, and is only able to leave exactly 1 review per correctly formatted transaction. This ensures that customers cannot falsely lower or boost reputation of vendors beyond the influence of their transaction.

P3. Item listing completeness: Any customer or vendor can query $\mathcal{I}$ efficiently. This implies that it is not possible to hide any item listings from c and that it is impossible to convince c of the existence of invalid or fake item listings. This prevents, for instance, adversaries from leaving out competitor's item listings to unfairly attract customers.

P4. Review completeness: Any customer or vendor can efficiently enumerate $\mathcal{R}_{(\cdot,i)}$ $\forall i$. This implies that it is not possible to hide any reviews from any c and that it is not possible to convince c of the existence of invalid or fake reviews. This guarantees that adversaries cannot manipulate existing reviews.

P5. Review-payment unlinkability: Given a review $\mathbf{r}$ from an honest customer and any k payment transactions for an item listing i that includes the payment associated with $\mathbf{r}$, γ of which are from malicious customers, the probability that adversary $\mathcal{A}$ learns $\mathbf{p}$ for which the review was generated from is negligibly close to guessing at random from the $k-\gamma$ payments from honest customers. A similar result must hold true for linking a payment to reviews as well. In other words,

$$\Pr\left[\begin{array}{l|l} \mathcal{A}\big(\{\mathbf{p}_j\}_{j\in[k]}, \mathbf{r}_c, & \mathbf{p}_j \in \mathbf{p}_{(\mathcal{C},i)} \\ \quad \mathcal{I}, \mathcal{P}, \mathcal{R}, & \forall j \in [1, k-\gamma], \\ \quad \mathcal{C}, \tilde{\mathcal{C}}, \mathcal{V}, \tilde{\mathcal{V}}\big) = c & \mathbf{r}_c \in \mathcal{R}_{(\mathcal{C},i)}, \\ & \mathbf{p}_c \in \{\mathbf{p}_j\}_{j\in[k]} \end{array}\right] \leq \frac{1}{k-\gamma} + neg(\lambda),$$

and

$$\Pr\left[\begin{array}{l|l} \mathcal{A}\big(\{\mathbf{r}_j\}_{j\in[k]}, \mathbf{p}_c, & \mathbf{r}_j \in \mathcal{R}_{(\mathcal{C},i)} \\ \quad \mathcal{I}, \mathcal{P}, \mathcal{R}, & \forall j \in [1, k-\gamma], \\ \quad \mathcal{C}, \tilde{\mathcal{C}}, \mathcal{V}, \tilde{\mathcal{V}}\big) = c & \mathbf{p}_c \in \mathbf{p}_{(\mathcal{C},i)}, \\ & \mathbf{r}_c \in \{\mathbf{r}_j\}_{j\in[k]} \end{array}\right] \leq \frac{1}{k-\gamma} + neg(\lambda)$$

for implicit security parameter λ.

This property is the most important anonymity property. Payments (i.e., purchases) may entail sensitive information, such as the address of the customer. The adversary learning the identity of the reviewer *and* the content of the reviews, in particular negative reviews, may have bad consequences for the customer. A DAM should protect the customers by ensuring unlinkability.

P6. Payment (review) unlinkability: An adversary $\mathcal{A}$ given two payment transactions $\mathbf{p}_c$ and $\mathbf{p}_{c'}$ generated by honest customers should be not able to tell if they were left by the same customer or different customers negligibly better than random guessing. A similar property should hold for any two reviews as well. That is,

$$\Pr\left[\begin{array}{l|l} \mathcal{A}(\mathbf{p}_c, \mathbf{p}_{c'}, \mathcal{I}, \mathcal{P}, \mathcal{R}, & \mathbf{p}_j \in \mathbf{p}_{(\mathcal{C},i)} \\ \quad \mathcal{C}, \tilde{\mathcal{C}}, \mathcal{V}, \tilde{\mathcal{V}}) = b & \forall j \in \{c, c'\} \\ \quad \wedge\, b = (c \stackrel{?}{=} c') & \end{array}\right] \leq \frac{1}{2} + neg(\lambda),$$

and

$$\Pr\left[\begin{array}{l|l} \mathcal{A}(\mathbf{r}_c, \mathbf{r}_{c'}, \mathcal{I}, \mathcal{P}, \mathcal{R}, & \mathbf{r}_j \in \mathbf{r}_{(\mathcal{C},i)} \\ \quad \mathcal{C}, \tilde{\mathcal{C}}, \mathcal{V}, \tilde{\mathcal{V}}) = b & \forall j \in \{c, c'\} \\ \quad \wedge\, b = (c \stackrel{?}{=} c') & \end{array}\right] \leq \frac{1}{2} + neg(\lambda),$$

where $c \stackrel{?}{=} c'$ evaluates to 1 if $c = c'$, 0 otherwise. This protects customers' identities, in case collection of payments or reviews reveal information about the customer.

Finally, we have a few optional properties that, while not crucial to functionality or security, benefit the customer and improve usability.

O1. Open enrollment: Anyone can efficiently join or leave the marketplace as a customer or a vendor at any time. This is a requirement for a truly decentralized marketplace, as there will not be a trusted third party monitoring membership. This does not imply that the vendor can create an item listing i at no cost.

O2. Selective review linkability: A customer c leaving a review $\mathbf{r}$ should have the option to link $\mathbf{r}$ to a set of reviews she has left for other items $\mathcal{PR}_c$ for any $\mathcal{PR}_c \subset \mathcal{R}_{(c,\cdot)}$ in an efficient publicly verifiable way. This allows the customers to build reputation as well, by showing good reviews she has left previously.

O3. Review exculpability: Related to the previous property, a customer c leaving a review $\mathbf{r}$ should not be able to link $\mathbf{r}$ to any $\mathbf{r}' \notin \mathcal{PR}_c$. This means that a customer c should not be able to link their review with other reviews which are not theirs in order to benefit from others' good reviews or purposefully link another customer to set of bad reviews.

4. Beaver PRELIMINARIES

Beaver uses existing block chains and anonymous payment schemes, such as Zerocash [4], to instantiate a DAM. In this section, we present the high-level design (§4.1), and threat model of Beaver (§4.2). We then discuss the idea of consensus (§4.3), and cryptographic primitives used in Beaver (§4.4).

4.1 High-level design

There are three types of participants in Beaver: customers, vendors, and a distributed network of miners, all of whom enjoy open enrollment (i.e., no trusted third party verifying identities). The distributed network of miners and the public ledger ensure the customers' ability to leave reviews for their interactions with vendors, and allow anyone to enumerate the feedback. In Beaver, the customers and vendors could be miners as well and vice-versa.

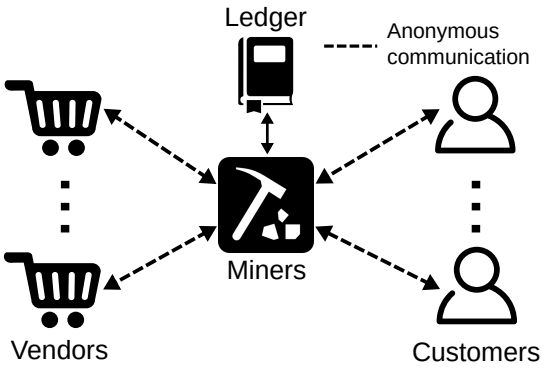


Figure 1: Beaver Architecture

At a high-level, Beaver works as follows. The vendors first register themselves to the network (i.e., the ledger) by publishing their pseudonyms. The customers are then able to enumerate the list of vendors, and purchase a product by making an anonymous transaction to the vendor. To leave a review, the customer privately ties the review to the transaction she made earlier, and submits the review to the network. Beaver, by using cryptography, guarantees that the clients cannot use the same transaction twice to sign a review. Finally, anyone can check the block chain to enumerate the reviews. Figure 1 shows the system architecture.

One key insight of Beaver is that with a public ledger, there is irrefutable public evidence that a valid transaction has taken place, and only the customer knows the secret information regarding the origin the transaction (i.e., private key used to sign a transaction). Using this, we can prevent (1) situations where a customer could be tricked into sending money but be unable to leave a review, and (2) anyone other than the real customer from leaving the review.

4.2 Threat model and assumptions

Beaver relies on reaching global consensus on the ledger, which contains important information such as financial transactions and reviews. We therefore require the underlying consensus protocol to be secure. In the case of a block chain based ledger, similar to that used by Bitcoin [30], we require at least that the adversary does not control a majority of the computational resources in the network, and that the majority of the computational resource behave rationally. A recent work on selfish mining, however, has shown that a simple majority may not be sufficient, and one may need as much as 75% of the network to be honest [18]. In any case, our security assumption will be identical to the security assumption for the underlying public ledger scheme. Beaver also assumes that the customers and vendors are rational, and do not behave maliciously if the cost of doing so is significant. Apart from these two assumptions, we do not limit the adversary's power. The adversary could, for instance, control many vendors and customers that collude with each other, and try to boost its own ratings or lower competitors' reputations.

We also assume existence of an ideal anonymous payment system, which allows anonymous transactions and anonymous transfers of coins back and forth from a regular cryptocurrency like Bitcoin. Zerocash [4], for instance, is a candidate for such currency. Finally, we assume that any communication, especially that of the customers, is done via a truly anonymous communication to ensure anonymity. In practice, the customers may use Tor [16] or other stronger anonymous communication systems [23, 39]. These systems may not behave as an ideal anonymous communication in reality, but addressing this issue is outside the scope of this paper.

4.3 Consensus

Beaver makes use of a consensus protocol for establishing network-wide agreement about the state of the marketplace such as item listings, reviews, and any other actions of its members. In particular, we make extensive use of a public ledger like that of Bitcoin [30], which uses proofs of work to arrive at a global consensus. While there are several other consensus protocols in the literature, the public ledgers in Bitcoin make relatively small assumptions about the network, making it a prime candidate for a decentralized application like Beaver.

At the core of Bitcoin-style public ledger is a hash chain that is constructed by a distributed set of miners. For a period of time, miners listen to messages being broadcast in the network by users, such as a transaction to transfer money from one account to another. Miners then aggregate these messages into a block along with the hash of the previous block and enter it into the public ledger by performing a proof-of-work. In Bitcoin, this proof-of-work is finding a nonce, such that the hash $H(\text{BLOCK}||\text{NONCE}) < \alpha$ where α is a parameter that determines the difficulty of the proof-of-work.

Miners in such systems are assumed to be economically rational actors, and so they need to be incentivized to spend their computational resources on mining blocks for the network. To do this, Bitcoin has (1) a reward for mining a block, and (2) a transaction fee. Beaver will rely on similar incentives to encourage miners to behave honestly and maintain a healthy ledger.

4.4 Cryptographic primitives

Beaver employs two cryptographic primitives, NIZK and LRSig, which are described in this section.

4.4.1 NIZK

A non-interactive zero-knowledge proof, or a NIZK, of a statement is a zero-knowledge proof of the statement that could be verified easily by anyone without interaction with the party who generated the proof. In Beaver, we use a subcategory of NIZK called non-interactive zero-knowledge proof-of-knowledge (NIZKPoK), which is a NIZK for proving knowledge of a secret value. NIZKPoKs are commonly used to show that, given a blinded version of a secret value (e.g., a commitment), the person who generated the proof knows the secret value underlying the commitment. An example of NIZKPoK is a zero-knowledge proof of possession of discrete logs: given g^x for a generator g of a group $\mathbb{G}$ in which discrete log is hard, NIZKPoK can be used to prove knowledge of x without revealing any informatino on x. This can be done by applying the Fiat-Shamir heuristic [19] to a standard zero-knowledge proof of discrete log as shown in [9].

4.4.2 Linkable ring signatures

Ring signatures (RSig), first proposed by Rivest *et al.* [32], are cryptographic signatures that guarantee anonymity of the signers. Specifically, a RSig algorithm takes as input the private key of the signer, a set of public keys, and a message, and generates a signature that can be verified against the *set* of public keys without revealing *which* key was used to sign the message.

Ring signatures unfortunately do not offer any form of accountability, and there is no way to stop the signer from signing multiple times even when it is not desirable. Linkable ring signatures [24] (LRSig), on the other hand, are accountable variants of ring signatures: All signatures generated by the same signer can be linked to each other, though the identity of the signer is still hidden. LRSigs can be used to prevent signers from signing multiple times, while preserving anonymity.

5. Beaver DESIGN

In this section, we describe the available operations in Beaver. We describe registration transactions (§5.1), special vendor transactions (§5.2), payment transactions, (§5.3), and review transactions (§5.4). In each section, we present the details of the transactions and how the miners could verify them. All transactions are signed with an unforgeable signature scheme by the party generating the transaction, unless otherwise noted. Figure 2 shows a typical workflow of Beaver.

5.1 Registration

Similar to Bitcoin [30] and other cryptocurrencies, Beaver does not require explicit registration for miners. This is also true for customers in Beaver, who by the asymmetry of DAMs, do not need any publicity. A vendor, on the other hand, needs a public identity (pseudonym) that others could refer to for purchases and reviews.

In Beaver, a vendor registers a product that he wants to sell by generating a new public key p_v in the underlying payment system that will be used to receive payments. The vendor then covers the registration fee ρ by moving money into p_v and forms a registration transaction

$$\mathbf{rt} = (\text{REGISTRATION}, \text{TXID}, \text{ITEMINFO}, p_v)\,.$$

REGISTRATION is the type of transaction, and TXID is a unique identifier for the transaction which in practice could the hash of all the other values in **rt**. ITEMINFO is the description of the item being sold, the price, and any other information needed to generate a payment, and p_v is the public key associated with this item.

Once the miners receive this transaction, they run Algorithm 1 to verify the transaction before adding it to the ledger. The miner that successfully adds **rt** to $\mathcal{L}$ claims the fee ρ from p_v. Once added, any customer can find the list of all items sold in Beaver by enumerating all REGISTRATION transactions. The customer can then purchase the product by sending money to the public key in the registration transaction. Moreover, the customers can check all available reviews for this key (item) as shown in §5.4.

Algorithm 1 Registration verification

INPUTS:
1. **rt** $= (\text{REGISTRATION}, \text{TXID}, \text{ITEMINFO}, p_v)$
2. $\mathcal{L}$

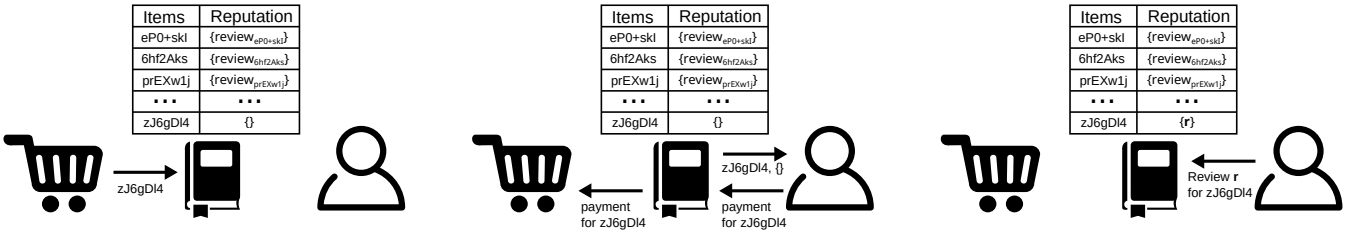
OUTPUT: Miners add **rt** to $\mathcal{L}$ only if all of the steps are satisfied.
1. Check p_v has enough funds to cover the registration fee ρ.
2. Check $\text{TXID} \notin \mathcal{L}_{\text{TX}} \wedge p_v \notin \mathcal{L}_{\mathbf{rt}}$.
3. Check ITEMINFO specifies the price of the item x, and that the price is within minimum and maximum denominations of underlying currency.

5.2 Special vendor transactions

Vendors in Beaver may also perform two special transactions: (1) bootstrapping reputation and (2) updating item listings.

Bootstrapping reputation.

When the item is first listed, there are no reviews for the item, meaning the reputation is `null`. The vendor, however, may have other products with positive reviews and may want to bootstrap the reputation for this item by linking it to these other items with positive reviews. These reviews, while not directly useful for expressing the quality of the particular item, could help establish trust with the vendor. In this case, the vendor can submit a special transaction that includes a NIZKPoK of the private keys of other items he sells.



(a) Vendor registers the public key of item to the ledger.

(b) Customer retrieves item listings and their reviews, and makes a payment transaction for an item.

(c) Customer leaves a review for the item purchased (optional).

Figure 2: Beaver workflow

The customers can then check the reviews of the other items, and be convinced that these items come from the same vendor. Note that this transaction may be submitted at any time, and the vendor may choose when to link the items together.

$$\mathbf{br} = \left(\text{BOOTSTRAP}, \text{TXID}, p_v, (\overrightarrow{p_j, \pi_{s_j}})\right)$$

Here p_v is the public key associated with the item listing that the vendor is interested in bootstrapping and $(\overrightarrow{p_j, \pi_{s_j}})$, is a vector of public keys associated with other item listings, and NIZKPoK for the secret keys to each. and miners use Algorithm 2 to verify this transaction. The miner that successfully adds **br** to $\mathcal{L}$ claims the fee β from p_v.

Algorithm 2 Vendor bootstrap verification

INPUTS:
1. $\mathbf{br} = \left(\text{BOOTSTRAP}, \text{TXID}, p_v, (\overrightarrow{p_j, \pi_{s_j}})\right)$
2. $\mathcal{L}$

OUTPUT: Miners add **br** to $\mathcal{L}$ only if all steps are satisfied.
1. Check p_v has enough funds to cover the fee β.
2. Check $\text{TXID} \notin \mathcal{L}$.
3. From $\mathcal{L}$, verify that p_v and $\vec{p_j}$ are valid public keys for items.
4. For each $\ell \in \left[1, |(\overrightarrow{p_j, \pi_{s_j}})|\right]$, verify π_{s_ℓ} is a valid NIZKPoK for the secret key of p_ℓ. $\vec{\pi_{s_j}}$ is generated using public randomness as a nonce to avoid replay attacks.

Updating listings.

A vendor may want to update an item listing after creating it. For example, the item may be sold out, discontinued, need a description change, or the vendor may wish to hold a promotion or sale. The vendor can issue an update by submitting a special transaction:

$$\mathbf{ut} = (\text{UPDATE}, \text{TXID}, p_v, \pi_{s_v}, \text{ITEMINFO})$$

Here p_v is the public key of the item listing that a vendor wishes to update, π_{s_v} is a NIZKPoK for the corresponding secret key, and ITEMINFO is the new information for the item listing. Miners use Algorithm 3 to verify this transaction, and the miner that successfully adds **ut** to $\mathcal{L}$ claims the fee τ from p_v.

5.3 Payments

When a user decides to purchase an item, she begins by generating a fresh public key (a pseudonym) p_c. She then transfers funds anonymously (e.g., via Zerocash [4]) to p_c. She finally transfers funds from p_c to the public key associated with the item to make the purchase. She may use the transaction to supply other relevant information such as a shipping address or any special requests for the order, or she may send that information out of band (e.g., through a vendor's website) along with the proof of the purchase via anonymous communication.

Algorithm 3 Item listing update verification

INPUTS:
1. $\mathbf{ut} = (\text{UPDATE}, \text{TXID}, p_v, \pi_{s_v}, \text{ITEMINFO})$
2. $\mathcal{L}$

OUTPUT: Miners add **ut** to $\mathcal{L}$ only if all steps are satisfied.
1. Check p_v has enough funds to cover the fee τ.
2. Check $\text{TXID} \notin \mathcal{L}$.
3. From $\mathcal{L}$, verify that p_v is a valid public key for an item.
4. Verify that π_{sv} is a valid NIZKPoK of secret key for p_v. π_{s_v} is generated using a public randomness as a nonce to avoid replay attacks.

In Beaver, we do not provide transaction privacy for the vendor; i.e., the recipient of the payment transaction is not hidden, unlike Zerocash. This is done so that the users can understand the explicit anonymity set when leaving a review (described in §5.4), and the implications of this design choice are discussed in §7.

A payment transaction **p** looks nearly identical to that of Bitcoin:

$$\mathbf{p} = (\text{PAYMENT}, \text{TXID}, p_c, p_v, v_v, \text{CUSTOMERINFO})$$

where PAYMENT is the type of transaction, TXID is the unique transaction ID, p_c is the customer's fresh nym that holds enough funds to pay amount v_v to the vendor's account p_v as well as any additional fees. If the underlying cryptocurrency supports adding supplementary information to transactions, then CUSTOMERINFO is passed along as information specific to the order such as a shipping address or special requests encrypted under p_v. One key difference between payment transactions in Beaver and payment transactions in Bitcoin is that the transaction fee of the payment is broken into two fees: *tax* f_t and *reviewing fee* f_r. f_t is paid to the miner who adds the payment transaction to the block chain, similar to traditional cryptocurrencies. f_r is paid later to the miner who adds the review associated with this payment to the block chain (§5.4). Upon submission, miners use Algorithm 4 to verify that payment satisfies all the requirements, and the miner that successfully adds **p** to $\mathcal{L}$ claims f_t from p_c. Any future transfer from p_c to another place will be considered invalid, except to claim the reviewing fee, for reasons described in §5.4.

5.4 Reviews

After the payment transaction **p** is added to the block chain, the customer has the option to form a review for the item. The review will contain a message from the customer (e.g., a detailed product review) as well as a numeric rating $a \in \mathcal{Y}$ where $\mathcal{Y}$ is a small set of integers. Once she writes the review, the customer can then sign

Algorithm 4 Payment verification

INPUTS:

1. $\mathbf{p} = (\text{PAYMENT}, \text{TXID}, p_c, p_v, v_v, \text{CUSTOMERINFO})$
2. $\mathcal{L}$

OUTPUT: Miners add $\mathbf{p}$ to $\mathcal{L}$ only if all of the steps are satisfied.

1. Check that the available funds in p_c is larger than $v_v + f_t + f_r$.
2. Check $\text{TXID} \notin \mathcal{L}$.
3. From $\mathcal{L}$, verify that p_v is a valid public key for an item.
4. Find ITEMINFO for p_v in $\mathcal{L}$, and the price x for p_v.
5. Check $v_v \geq x$.

the review with the private key used to make the transaction, and send it to the network to be added.

A naive signature will reaveal the transaction associated with the the private signing key, and therefore ties the transaction to the review. Unfortunately, the transaction may contain sensitive information about the customer, and concern of having the transaction tied to the review limit the customer's ability to leave truthful feedback. Though we wish to hide the exact relationship between reviews and transactions, we must also ensure that there exists a valid payment associated with each review. To achieve this, we use linkable ring signatures. The list of all transactions left for p_v is first broken into groups of size k where k is a public system parameter. The customer p_c who wishes to leave a review for p_v figures out the group that her payment transaction $\mathbf{p}$ belongs to, takes all public keys of customers within that group, and uses them to sign the review with LRSig instead of a regular signature. With LRSig, no one is able to learn which one of the k payments is linked to the review, while guaranteeing that any attempt to submit more than one review per payment will be caught via the linkability of signatures.

With this change, we have to modify the reviewing fee slightly, as the miner who adds the review to the chain cannot figure out which transaction should pay the reviewing fee. Instead, the miners can take from any transaction in the same group that has not yet been claimed, as all valid transactions have been specified to pay the reviewing fee. Furthermore, as previously before, the miners will reject any attempt to transfer the reviewing fee to another account before the fee is legitimately claimed by a miner, ensuring there is enough funds in the account to pay the miner. We note that although there could be a delay in waiting for k transactions to appear on the chain, once there are sufficiently many transactions, the reviews could be submitted at any time (i.e., asynchronously) and still guarantee the anonymity among the set of k transactions.

Customer reputation.

In existing e-commerce systems like Amazon and eBay, it is possible for a customer to build up her reputation as a "good" reviewer or an "expert" of a category of products, by linking all the reviews she leaves to her account. For instance, if one person has left reviews for many different headphones and speakers, then that person's reviews for another headphone may be more important to potential customers. However, a cusomer may also desire the property that reviews for certain products do not link back to other products she has reviewed for sake of privacy.

In Beaver, we allow the customer to choose which reviews to tie together: When leaving a review, the customer also generates a random value r_c that she keeps secret and its commitment $\texttt{Comm}(r_c)$, that she includes in the review. When the same customer wishes to link reviews together, she may refer to a previous review containing $\texttt{Comm}(r_c)$ and include NIZKPoK of r_c in the new review. This also enables the customer to create potentially several groups of reviews (e.g., one for headphones, one for books, etc.), and convince others of her expertise in a category of products without revealing the reviewer's purchases in other categories.

Review enumeration and aggregation.

In the case where a customer wishes to enumerate all reviews for a product along with their contents, she can go through the public ledger. However, it is also common for a customer to want to see an aggregated reputation (e.g., average of all ratings) to get a quick summary of the quality of the product, without sifting through all the reviews. To allow quick summary, after a customer submits a review, the miners compute the total number of ratings for the product thus far denoted N, and the sum of all the ratings denoted s from $\mathcal{L}$. The miners then update these values according to the newly submitted review, adds them to the ledger. Note that N and s are not signed by the reviewer, as there may be reviews for the same product not yet in the ledger, and only the miners who pick the ordering and the reviews to add the ledger can determine N and s accurately. However, N and s cannot be faked, since they are publicly computable from $\mathcal{L}$.

With this, the customer only needs to find the latest review to get an aggregate value which can be done by traversing the block chain backwards. This procedure may take $O(t)$ time where t is the length of the ledger in the worst case, but for vendors with reasonably frequent transactions, it will be much faster.

Review revocation.

The reviewers may want to update their reviews of a product after some time. Often, for example, a product works well in the first few days, and the customer leaves a positive review. Soon after, the product breaks, and the customer wants to change the review to be something negative. In these scenarios, the reviewer simply resigns the review, using Algorithm 5 with the same private key and public keys, and sends the new review to the miners. When enumerating the reviews, if anyone finds reviews that are linked (via LRSig), then he or she simply takes the latest review and ignores older ones. This review, however, must be submitted with a separate reviewing fee to incentivize the additional work needed from the miners.

In summary, a review transaction is the tuple

$$\mathbf{r} = \left(\text{REVIEW}, \text{TXID}, a, M, p'_c, \sigma_{s'_c}, p_v, \texttt{Comm}(r_c), \vec{p_j}, \vec{z_\ell}, \vec{\mathbf{r}_\ell}\right)$$

where REVIEW and TXID are the type and ID of the transaction, a is the numerical rating, M is the detailed review, p'_c is a fresh nym to draw the review fee from if this review is an update, $\sigma_{s'_c}$ is a signature on all other values in $\mathbf{r}$ with the secret key corresponding to p'_c, note that if this review is not an update than $p'_c, \sigma_{s'_c}$ may both be $\texttt{null}$. p_v is the public key of the item of the review, $\texttt{Comm}(r_c)$ is the commitment of a random value used to link reviews if desired later on, $\vec{p_j}$ the set of public keys of size k that includes the customer's public key, and $\vec{z_k}$ are NIZKPoKs of the private keys of the other reviews from the same customer to this review while $\vec{\mathbf{r}_\ell}$ are the reviews to be linked. $\vec{z_\ell}$ and $\vec{\mathbf{r}_\ell}$ may be empty which is denoted by $\texttt{null}$ if the customer does not wish to link any reviews. Customers use Algorithm 5 to generate reviews, and the miners run Algorithm 6 to verify the reviews before adding them to the ledger.

6. ANALYSIS

We first argue that Beaver satisfies all the properties of DAM (§3.2). We then analyze the impact of different parameters.

6.1 Properties of Beaver

P1. Correctness: A payment is completed when the payment transaction $\mathbf{p}$ ends up in the ledger $\mathcal{L}$, which we assume is publicly

Algorithm 5 Review generation

INPUTS:

1. (s_c, p_c): private-public key of the customer
2. p'_c: public key of nym to pay reviewing fee (update only)
3. $\sigma_{s'_c}$: signature of all other fields with private key corresponding to p'_c
4. p_v: public key for the item listing
5. a: rating for this item
6. M: short message for this review
7. $\vec{\mathbf{r}}_\ell$: set of reviews to link to
8. $\vec{r}_\ell$: secret random values in reviews to be linked
9. $\mathcal{L}$: public ledger

OUTPUT: A review transaction and a linkable ring signature σ of the review for item p_v.

1. From $\mathcal{L}$, verify that p_v is a valid public key for an item.
2. Divide payment transactions for $p_v \in \mathcal{L}$, into groups of k.
3. Find the group that p_c belongs to, and extract the k public keys to yield $\vec{p_j}_{j\in[k]}$.
4. Compute signature $\sigma = LRSig\left(\mathbf{r}, s_c, \vec{p_j}_{j\in[k]}\right)$.
5. Generate a random value r_c and its commitment $\texttt{Comm}(r_c)$.
6. Let $\vec{\texttt{Comm}}(r_\ell)$ be the commitments to random values in other reviews to be linked. Use $\mathcal{L}$ as public randomness to compute NIZKPoKs $\vec{z_\ell}$ for $\vec{r_\ell}$. Note that if the customer does not wish to link reviews, then $\vec{z_\ell} = \texttt{null}$, $\vec{\mathbf{r}_\ell} = \texttt{null}$.
7. Output $\mathbf{r} = \left(\text{REVIEW}, \text{TXID}, a, M, p'_c, \sigma_{s'_c}, p_v, \texttt{Comm}(r_c), \vec{p_j}, \vec{z_\ell}, \vec{\mathbf{r}_\ell}\right)$ and σ.

Algorithm 6 Review verification

INPUTS:

1. $\mathbf{r} = \left(\text{REVIEW}, \text{TXID}, a, M, p'_c, \sigma_{s'_c}, p_v, \texttt{Comm}(r_c), \vec{p_j}, \vec{z_\ell}, \vec{\mathbf{r}_\ell}\right)$
2. σ, the linkable ring signature of $\mathbf{r}$
3. s, total numerical rating of p_v thus far
4. N, the number of reviews left for p_v thus far
5. $\mathcal{L}$

OUTPUT: Miners add $\mathbf{r}$ to $\mathcal{L}$ only if all steps are satisfied.

1. Check $\text{TXID} \notin \mathcal{L}$.
2. From $\mathcal{L}$, verify $\vec{p_j}$ is a valid set of public keys of k transactions to p_v.
3. Find the last review left for p_v in $\mathcal{L}$, and verify that s and N correctly computed given this rating a.
4. Verify σ on $\mathbf{r}$.
5. If σ links, then this is an update, and verify p'_c has at least f_r funds to cover the review fee and $\sigma_{s'_c}$ is a valid signature.
6. If $\vec{z_\ell}$ is not $\texttt{null}$, then verify the NIZK.

visible. Upon k completed transactions for an item, customers may generate a review using Algorithm 5, and the signatures will verify as long as a valid payment is in $\mathcal{L}$.

P2. Soundness: Soundess is derived directly from the forgery resistance of linkable ring signatures. Without a payment transaction in $\mathcal{L}$ an adversary would not have a secret key for any of the public keys associated with the k transactions. The forgery resistance of linkable ring signatures implies that this adversary cannot generate a valid signature that will verify. Similarly, if a customer signs two reviews using the same secret key, then the two signatures will be linked, and anyone can detect this misbehavior.

P3. Item listing completeness: Assuming consensus and availability of $\mathcal{L}$, any item listing is publicly visible to everyone since $\mathcal{L}$ holds item listings.

P4. Review completeness: Similar to item listing, assuming consensus and availability of $\mathcal{L}$, any review is publicly visible.

P5. Review-payment unlinkability: As with §3.2, assume that out of the k payment transactions used to sign the review by an honest customer, γ of them were generated by malicious clients. The adversary then knows the honest customer's payment is not part of the γ malicious transactions, so the anonymity set size is effectively $k-\gamma$. The security property of linkable ring signatures ensures that the adversary cannot do non-negligibly better than guessing from $k-\gamma$. Moreover, the optional NIZKPoKs used to link reviews only link reviews together without revealing any information about the payments or the public keys used. We discuss how γ may be determined in §6.2.

P6. Payment (review) unlinkability: For every purchase, customers generate a fresh nym which is unlinked from any other nyms previously used (via security of the underlying payment scheme), therefore no purchases can be linked to other purchases. Similarly, reviews are also signed with the fresh nyms, so the reviews cannot be linked to each other, unless explicitly linked via NIZKPoKs.

O1. Open enrollment: As long as the underlying payment system and consensus protocol allows for open enrollment, customers, vendors, and miners can all join freely.

O2. Selective review linkability: A customer who previously generated a review should know the random value in the commitment used for that review. The customer can then use NIZKPoK to prove possession of the random value in a new review to link two reviews together.

O3. Review exculpability: If the commitment scheme is hiding, then the adversary cannot learn the secret value from the commitment. Since only the honest customer knows the secret value, no other person can generate a correct NIZKPoK to link the reviews.

6.2 Parameters

There are many parameters that impact the performance and security of Beaver. In this section, we describe the impact of each parameter, and provide our recommendations.

Registration fee ρ. ρ is the cost that a vendor must pay to register a new product. This cost mitigates with the problem of vendors with negative reviews simply creating a new listing for the item to reset the reputation. It also mitigates attacks where an adversary attempts to flood the network with item listings to make enumeration and identification of legitimate item listings tedious. It is however important that ρ is not set too high as to discourage new vendors from joining the network.

Transaction tax f_t. f_t is money that is paid to miners when a customer purchases an item from a vendor and directly impacts the credibility of reviews in our system. Since f_t is paid to the network of miners and is lost to both the customer and the vendor, high values of f_t would deter adversaries from leaving fake reviews for sake of influencing someone's reputation thereby mitigating Sybil attacks. For example, this would discourage a vendor from acting as a customer and making a lot of payments to himself in order to leave positive reviews. On the other hand, if f_t is too high, customers may be discouraged from using Beaver and instead seek out cheaper alternatives with less overhead. We believe that f_t should be proportional to the value of the transaction being made, perhaps around 10% of the item price similar to vendor fee charged by Amazon Marketplace. This insight forms basis for the *credibility* of a review, the lower bound on the cost to an adversary of creating for creating it. Finally, f_t also incentivizes the miners to add the payment transaction to the ledger $\mathcal{L}$.

Reviewing fee f_r. f_r is a cost that is paid to the miners for leaving a review, and f_r also helps to establish the lower bound on the

cost of leaving a review in conjunction with f_t. It is important to note that in Beaver, reviews are separate transactions from the payment transactions although practically, a customer needs to supply funds for them both at the same time. We recommend either an absolute fixed fee for f_r, or a fee that is proportional to the size of the review (i.e., length of the message), similar to transaction fees in Bitcoin [30].

Anonymity set size k. k impacts the anonymity of a customer. k should be set in such a way that $k - \gamma$ is sufficiently large with respect to the percieved adversary, and the customer is provided with large enough anonymity set. For an adversary that controls m fraction of all network's resources ($m < .5$), we know that it costs the adversary about $(1-m)f_t\gamma$ on average to create γ transactions.[1] With this fact and an assumption about adversary's financial resources, we could estimate γ, and pick k that will satisfy the customers.

Big values of k, while providing larger anonymity sets, trade-off the latency of reviews. Since customers cannot review an item until it has been purchased k times, depending on the popularity of the item, customers may have to wait a while, and so this parameter also needs to be balanced for usability.

Reputation bootstrap fee β. Since vendors can only link the items that it knows the secret keys for (i.e., owned by the vendor), bootstrapping reputation should not be a costly operation. β should be set to be the minimal value that would incentivize the miners to add this transaction to the ledger.

Item update fee τ. A vendor should be able to update their items, so τ should be set to be the minimal value that would incentivize the miners sufficiently to add this transaction to $\mathcal{L}$. Note that a vendor may change the price very frequently in an attempt to differentiate prices for different customers, but this is visible to the public, and the price at a given time is same for every customer.

6.2.1 *Review credibility*

Customers who are interested in purchasing an item will first want to look at the reviews have been left by previous customers. In addition to observing the score and message of a previous review, a customer may be interested in evaluating a review's credibility, or what it would have costed an adversary to create the review himself.

While it might be tempting to assume that cost to an adversary for generating a review is the cost to purchase the item plus the associated tax f_t and reviewing fee f_r, it may be the case that an adversary is purchasing the item from himself, and is therefore recovering the price of the item. The cost to an adversary for generating a review is therefore lower bounded by $f_t + (\ell + 1) f_r$, where f_t is the tax paid on the payment transaction, ℓ is the number of times the review has been updated. As discussed in 6.2 later, f_r is recommended to be set to a constant value, and f_t will likely scale with the price of an item. With the ability to change the price of an item (§5.2), f_t may be different for different transactions. Moreover, it is not possible to link a review to any particular transaction. However, customers do know the lowest possible tax for a review since the k transactions that form anonymity set is public. Therefore, customers should be conservative, and assume the lowest f_t out of the k transactions to estimated the credibility of a review.

Though $f_t + (\ell + 1) f_r$ forms the baseline credibility, this may be augmented by other reviews that were linked. A naive augmentation of the credibility would be to add up the credibility of all linked reviews. However, this enables an adversary to pay (out-of-band) a good reviewer to generate NIZK, and all of sudden get a review of very high credibility for potentially much less cost than estimated. There may be other out-of-band attacks that Beaver may not protect against, so customers should be careful when augmenting credibility of a review using linked reviews.

[1] Adversary is expected to win the transaction fee with probability m if everyone follows the Bitcoin protocol. This statement is not true, for instance, under selfish-mining attacks [18].

7. DISCUSSION

We next discuss some additional aspects of Beaver that our current system design does not fully address, or that present room for improvement.

Vendor privacy. Though our system offers high level of privacy and anonymity for buyers, it only offers limited protection for vendors. Beaver provides the ability to hide which items a vendor sells, since it generates a fresh pseudonym for each item; however, the sales number (i.e., the number of transactions) is made public via the ledger. This is necessary in Beaver, as customers need explicit information about their anonymity set before leaving a review. One could argue that such transparency and auditability of the vendors may be good for the market as a whole, but this may not be desirable for vendors who want to conceal their transaction volume. We hope to address this concern in the future.

Query efficiency. By default, to query the reputation score of an item listing, a user needs to go through the whole ledger, which has an $O(n)$ worst-case complexity, where n is the size of the ledger. If each user keeps her own copy of the ledger, she can form an indexed database of the values to do efficient lookups. However, this requires $O(n)$ space from each user, which may be too costly.

Alternatively, customers could let an untrusted service manage an efficient-to-query database that is supposed to be a replica of the ledger, and periodically check the consistency of the ledger and the database. In practice, we imagine that many such services would exist, and that a user may query several of them at the same time and check the consistency of their results to ensure that they are correct. Only one of these services needs to be honest to enable tampering detection. In this way, customers may choose to increase usability at the cost of security. Ideally, we would like such untrusted services to generate a proof that the database is simply an alternate representation of the ledger and that the returned queries are correct so that customers need not trust the database service at all, but we defer such constructions, along with other efficiency improvements, to future work.

Unclaimed reviewing fees. When a payment transaction is approved, a value of at least f_r is left in the customer's account to cover the cost of a review in the future. It may however be the case that a customer forgets to leave a review, has no interest in leaving a review, or loses control of the secret information needed to do so, causing the funds to be effectively orphaned. To deal with this problem, it might make sense for the network to agree on a common policy: for instance, that review fees unused for a long time (e.g., 2 years) can be claimed by miners; that they can be distributed among the customers in the anonymity set; or that they be locked forever and offset by printing new currency.

Tuning the parameter k. The value of k presents a trade-off between anonymity of the customer's review and the latency or delay from the time of purchase before a review can be posted. For example, for $k = 1000$, a popular item that has 1,000 sales per day on average would support reviews with a latency of one day. A less popular item with only 10 sales per day, however, would require 100 days between purchase and review for the same value of k.

The latency may not be a significant issue for customers who are leaving a review because (1) the customer may wish to wait a few days to try out the product before leaving a review anyway, and (2) in practice, a user may interface with the system through an application that can record the user's review immediately and then

send it out as soon as the anonymity set is ready. Unfortunately, high latency is a problem for potential customers that would like to observe reviews, and the vendors who sell these items. Thus, it might make sense for either vendors to select the value of k for their item listing at the time of listing creation, or for the system to support a range of values to accommodate different trade-offs of anonymity and usability. In either of these cases, customers will be responsible for ensuring that k is a sufficiently large number.

Custom algorithms for sybil detection and reputation calculation. Beaver provides users with a large corpus of raw information in the form of the public ledger $\mathcal{L}$. Beaver makes no effort beyond the details of its specification to identify reviews that have been generated as the result of a Sybil attack. Other research [13, 37, 38] attempts to do just this by using a variety of heuristics and machine learning techniques. Customers in Beaver can run some of these algorithms themselves over the ledger to make their own decisions. In a similar fashion, while Beaver provides users with an average review rating, customers may in practice run their own algorithms for distilling the richness of review messages and values down to a concise number that can be easilly compared across items.

Customers who reveal themselves. In the message field of a review, a customer might link her transaction or reveal her true identity. This will have the consequence of reducing the anonymity set of all other customers who bought the item. Beaver cannot guarantee anonymity of honest customers when users behave in this way. We recommend that the applications that customers use to interface with the system make it clear that they should not reveal this sort of information, but ultimately customers revealing themselves falls outside the scope of Beaver.

Economically irrational adversaries. Beaver aims to protect its users against adversaries who are economically rational. Our notion of credibility, for instance, is derived from the lower bound of the cost to an adversary for generating reviews. This unfortunately is less useful against adversaries that are not economically rational. An adversary that is not economically rational for example may spend $1,000 generating fake reviews in order to setup a honeypot item listing, or deanonymize or scam a customer buying a $20 product. Such adversaries are outside the threat model of Beaver.

Despite not being able to provide strong guarantees against economically irrational adversaries, we are less concerned with large scale attacks on customers' anonymity. To deanonymize customers, an adversary would need to make many payment transactions so that the customers' true anonymity sets are very small. Deanonymizing every customer, for instance, would require that the adversary owns at least $\gamma = k - 1$ transactions for *every* k anonymity set, though in practice, the adversary may need to purchase an item more than $k - 1$ times per honest user transaction due network effects. For unpopular item listings, this change in popularity would be easily detectable on the public ledger. For popular listings, this attack would require a tremendous amount of financial resources, which even economically irrational adversaries may not have.

8. RELATED WORK

We now bring attention to notable related efforts for establishing reputation in distributed or anonymous settings as well as anonymous payment schemes that can be used for supporting e-commerce.

8.1 Distributed and anonymous review systems

Dingledine *et al.* [14] explored the idea of using centralized servers to maintain reputation scores for a node's reliability in a decentralized network. Similarly, Gupta *et al.* [20] used a third party that they called a reputation computation agent to facilitate reputation in a peer-to-peer network. Androulaki *et al.* [3] proposed using digital cash schemes where peers mint and send reputation coins using a centralized bank. In this setting, demonstrating possession of reputation coins is sufficient for proving reputation. This paradigm enables tying reputation to users and as opposed to pseudonyms, and in that respect has overlap with Beaver where users can demonstrate possession of reputation across pseudonyms. Unlike Beaver, these proposals all require a centralized agent that must be trusted.

Damiani *et al.* [12] proposed a system where a user queries the reputation of another node in a peer-to-peer network by polling her peers. The EigenTrust [22] algorithm goes a step further and leverages the transitivity of trust, computing a normalized reputation score based off of a network of trust. While the approach of polling peers in the network does not require a trusted third party, each node's view of the network is biased by its peers which stores reputation information in a public ledger using global consensus.

Signatures of Reputation [5] uses a third party, trusted for both privacy and correctness to provide users with monotonic reputation (bad actions cannot be punished). Blömer *et al.* [6] have suggested using primitives such as zero knowledge proofs and ring signatures with the assistance of a third party to enable users to leave feedback for each other and query reputation in a privacy preserving way. Both of these approaches suffer from the limitation of requiring trust in a third party unlike Beaver.

AnonRep [40] uses linkable ring signatures, verifiable shuffles, and homomorphic encryption to build a reputation system where peers can anonymous rate each other, at least one server in a set of powerful servers is honest. This system, however, does not protect again Sybil attacks.

8.2 Anonymous payments

Early on Chaum [10] understood that payment schemes would be a privacy bottleneck for digital commerce. Practical pseudonymous payment schemes such as Bitcoin [30] have paved the way for adoption of anonymous marketplaces. Since then there have been several additional proposals such as Zerocoin [27], Zerocash [4], Mixcoin [8], and Blindcoin [36] that attempt to either fix weaknesses in Bitcoin or replace it altogether with varying security and usability trade-offs.

The anonymity of a customer in a marketplace can only be as strong as the anonymity provided by the underlying payment scheme. While Bitcoin has become a leading choice anonymous transactions, several attacks against its anonymity have been demonstrated such as the work of Meiklejohn *et al.* [26] and the work of Ron and Shamir [33] which demonstrate that in practice it is possible to cluster the aggregate behavior of users. For this reason we advocate the use of payment schemes with strong provable security guarantees such as Zerocash [4].

9. CONCLUSION

We have presented Beaver, a decentralized anonymous marketplace (DAM). Beaver uses anonymous payments, a consensus protocol, zero-knowledge proofs, and linkable ring signatures to instantiate a secure DAM. We showed that our system preserves the anonymity of customers, and incorporates an incentive structure inspired by existing cryptocurrencies, thereby motivating a healthy network while lower bounding the cost of Sybil attacks at the same time.

10. REFERENCES

[1] I2P: The internet invisible project. http://www.geti2p.net.

[2] Openbazaar docs, 2016. https://docs.openbazaar.org/.

[3] E. Androulaki and S. Choi. Reputation systems for anonymous networks. In *PETS*, 2008.

[4] E. Ben-sasson, A. Chiesa, C. Garman, M. Green, I. Miers, and E. Tromer. Zerocash : Decentralized Anonymous Payments from Bitcoin (extended version). In *IEEE Symposium on Security and Privacy*, pages 1–56, 2014.

[5] J. Bethencourt, E. Shi, and D. Song. Signatures of Reputation : Towards Trust Without Identity. pages 1–41.

[6] J. Blömer, J. Juhnke, and C. Kolb. Anonymous and publicly linkable reputation systems. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8975, pages 478–488, 2015.

[7] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 103–112, New York, NY, USA, 1988. ACM.

[8] J. Bonneau, A. Narayanan, A. Miller, J. Clark, J. A. Kroll, and E. W. Felten. Mixcoin Anonymity for Bitcoin with accountable mixes. pages 1–25.

[9] J. Camenisch and M. Stadler. Proof systems for general statements about discrete logarithms. Technical report, 1997.

[10] D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(70), 1985.

[11] N. Christin. Traveling the Silk Road: A measurement analysis of a large anonymous online marketplace. In *Proceedings of the 22nd World Wide Web Conference (WWW'13)*, pages 213–224, Rio de Janeiro, Brazil, May 2013.

[12] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. Managing and sharing servents' reputations in P2P systems. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):840–854, 2003.

[13] G. Danezis. SybilInfer : Detecting Sybil Nodes using Social Networks. *Network & Distributed System Security Symposium(NDSS)*, 2009.

[14] R. Dingledine, N. Mathewson, and P. Syverson. Reputation in p2p anonymity systems. 2003.

[15] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, San Diego, CA, Aug. 2004.

[16] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium*, pages 303–320, August 2004.

[17] J. R. Douceur. The Sybil attack. In *Peer-to-peer Systems*, pages 251–260. Springer, 2004.

[18] I. Eyal and E. G. Sirer. Majority Is Not Enough: Bitcoin Mining Is Vulnerable. *Financial Cryptography and Data Security, Fc 2014*, 8437:436–454, 2014.

[19] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proceedings on Advances in cryptology—CRYPTO '86*, pages 186–194, London, UK, UK, 1987. Springer-Verlag.

[20] M. Gupta, P. Judge, and M. Ammar. A reputation system for peer-to-peer networks. *NOSSDAV*, page 144, 2003.

[21] M. Jakobsson and A. Juels. *Secure Information Networks: Communications and Multimedia Security IFIP TC6/TC11 Joint Working Conference on Communications and Multimedia Security (CMS'99) September 20–21, 1999, Leuven, Belgium*, chapter Proofs of Work and Bread Pudding Protocols(Extended Abstract), pages 258–272. Springer US, Boston, MA, 1999.

[22] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The Eigentrust algorithm for reputation management in P2P networks. *12th International Conference on World Wide Web (WWW)*, page 640, 2003.

[23] A. Kwon, D. Lazar, S. Devadas, and B. Ford. Riffle. *Proceedings on Privacy Enhancing Technologies*, 2016(2):1–20, 2016.

[24] J. K. Liu, V. Wei, and D. Wong. Linkable spontaneous anonymous group signature for ad hoc groups. *Information Security and Privacy*, 2108:325–335, 2004.

[25] D. Mayzlin, Y. Dover, and J. Chevalier. Promotional reviews: An empirical investigation of online review manipulation. *American Economic Review*, 104(8):2421–55, August 2014.

[26] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. Mccoy, G. M. Voelker, and S. Savage. A Fistful of Bitcoins : Characterizing Payments Among Men with No Names. 2013.

[27] I. Miers, C. Garman, M. Green, and A. D. Rubin. Zerocoin: Anonymous Distributed E-Cash from Bitcoin. *2013 IEEE Symposium on Security and Privacy*, pages 397–411, may 2013.

[28] A. Molavi Kakhki, C. Kliman-Silver, and A. Mislove. Iolaus: securing online content rating systems. *Proceedings of the Twenty-Second International World Wide Web Conference*, pages 919–930, 2013.

[29] S. Nakamoto. Bitcoin: a peer-to-peer electronic cash system, Oct. 2008. Available from http://bitcoin.org/bitcoin.pdf.

[30] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Consulted*, pages 1–9, 2008.

[31] P. Resnick and R. Zeckhauser. Trust among strangers in Internet transactions: Empirical analysis of eBay's reputation system. *Advances in applied . . .* , (11):127–157, 2002.

[32] R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *Proceedings of ASIACRYPT*, pages 552–565, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[33] D. Ron and A. Shamir. Quantitative analysis of the full Bitcoin transaction graph. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7859 LNCS:6–24, 2013.

[34] K. Soska and N. Christin. Measuring the longitudinal evolution of the online anonymous marketplace ecosystem. In *Proceedings of the 24th USENIX Security Symposium (USENIX Security'15)*, pages 33–48, Washington, DC, Aug. 2015.

[35] J. Tsai, S. Egelman, L. Cranor, and A. Acquisti. The effect of online privacy information on purchasing behavior: An experimental study. *Information Systems Research*, 22(2):254–268, 2011.

[36] L. Valenta and B. Rowan. Blindcoin: Blinded, accountable mixes for bitcoin. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8976:112–126, 2015.

[37] G. Wang, S. Barbara, T. Wang, H. Zheng, and B. Y. Zhao. Man vs . Machine : Practical Adversarial Detection of Malicious Crowdsourcing Workers. *the 23rd USENIX Security Symposium*, pages 239–254, 2014.

[38] G. Wang, M. Mohanlal, C. Wilson, X. Wang, M. Metzger, H. Zheng, and B. Y. Zhao. Social Turing Tests: Crowdsourcing Sybil Detection. *NDSS 2013 (20th Network*

and Distributed System Security Symposium), pages 1–14, 2013.

[39] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson. Dissent in numbers: Making strong anonymity scale. In *Presented as part of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*, pages 179–182, Hollywood, CA, 2012. USENIX.

[40] E. Zhai, D. I. Wolinsky, R. Chen, E. Syta, C. Teng, and B. Ford. Anonrep: Towards tracking-resistant anonymous reputation. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 583–596, Santa Clara, CA, Mar. 2016. USENIX Association.