ReviewChain: Untampered Product Reviews on the Blockchain

Daniel Martens
University of Hamburg
Hamburg, Germany
martens@informatik.uni-hamburg.de

Walid Maalej
University of Hamburg
Hamburg, Germany
maalej@informatik.uni-hamburg.de

ABSTRACT

Online portals include an increasing amount of user feedback in form of ratings and reviews. Recent research highlighted the importance of this feedback and confirmed that positive feedback improves product sales figures and thus its success. Online portals' operators act as central authorities throughout the overall review process. In the worst case, operators can exclude users from submitting reviews, modify existing reviews, and introduce fake reviews by fictional users. This paper presents ReviewChain, a decentralized review approach. Our approach avoids central authorities by using blockchain technologies, decentralized apps and storage. It enables users to submit and retrieve untampered reviews. We highlight the implementation challenges encountered when realizing our approach on the public Ethereum blockchain. Then, we discuss possible design alternatives and their trade-offs regarding costs, security, and trustworthiness. Finally, we analyze which design decision should be chosen to support specific trade-offs and present resulting combinations of decentralized blockchain technologies, also with conventional centralized technologies.

CCS CONCEPTS

• **Software and its engineering** → **Software design engineering**; *Software design tradeoffs*; • **Networks** → *Network protocols*;

KEYWORDS

Blockchain, user feedback, product reviews, app reviews, app stores

1 INTRODUCTION

Online portals often allow consumers to rate products on a five-star scale and by writing a review message. Ratings and reviews are used for various purposes, e.g., for products bought on Amazon[1],

---

[1]https://www.amazon.com

trips booked on Tripadvisor[2], or apps downloaded from the Apple App Store[3]. These portals receive an increasing amount of user feedback, e.g., Tripadvsior claims to serve more than 500 million traveler-generated reviews. Consumers use this feature to express their satisfaction and experiences with products. A large body of research analyzes reviews, for example to summarize them [7] and to understand their helpfulness [9]. When deciding between products, reviews have a considerable impact on consumers' choice [3, 5]. Research found that ratings and review correlate with sales and download ranks [6, 11]. Stable and numerous ratings are associated with high quality and lead to higher downloads and sales numbers.

Consumers that rely on reviews when deciding for a product, have to trust at least two parties involved. These are the review authors and the operators of online portals. Untrustworthy reviews of single authors, e.g., an extremely positive review in between negative reviews, can possibly be recognized by consumers. However, this does not apply for larger amounts of reviews modified by online portal operators themselves. The operators act as central authorities throughout the complete review process. In worst case, an operator can exclude consumers from submitting reviews, modify existing reviews, and introduce fake reviews to improve the ratings and rankings of products [4, 8, 10].

In this paper, we aim to resolve the problem of central authorities, being able to influence the review processes, by using the public Ethereum blockchain. Consumers no longer need to rely on central authorities as blockchains operate decentrally across a network of several nodes, in which every user can participate [1]. Thereby, we enable consumers to submit and access untampered and trustworthy reviews. The **contribution** of the paper is twofold:

(1) We propose a decentralized review approach which does not involve central authorities.
(2) We summarize the approach's major implementation challenges on the public Ethereum blockchain, discuss design alternatives, and compare their trade-offs.

The remainder of the papers is structured as follows: Section 2 introduces the most important terminologies used in this paper and presents our ReviewChain approach as well as its design goals. Section 3 discusses the major challenges encountered when implementing the approach on the public Ethereum blockchain. Finally, Section 4 analyzes configurations of design alternatives with regard to their resulting trade-offs.

2 APPROACH

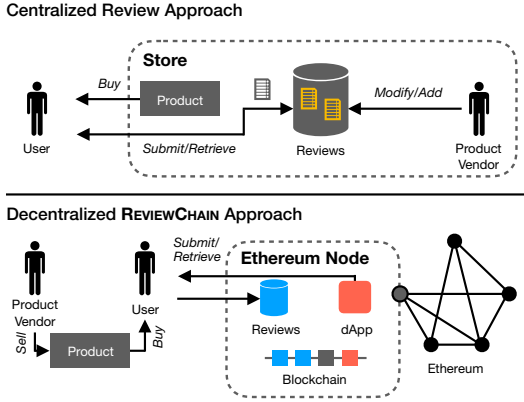We introduce the most important terminologies and present our decentralized ReviewChain approach.

---

[2]https://www.tripadvisor.com
[3]https://itunes.apple.com/us/genre/ios/id36?mt=8

Figure 1: Centralized Reviewing/REVIEWCHAIN Approach.

2.1 Terminology

Ethereum[4] is a blockchain, developed in 2014 by Vitalik Buterin. In comparison to the Bitcoin blockchain, which only handles accounts and transactions, Ethereum also stores programming logic. When paying for its execution, any turing-complete script can be run on Ethereum. Thereby, it enables decentralized apps without any possibility of downtime, censorship, or third-party interference.

Transactions are digitally signed messages persisted on the blockchain. Each transaction is associated with an action, such as transferring cryptocurrency units. Transactions have a sender and receiver address, that represent public keys belonging to specific users.

Ether (ETH) is the cryptocurrency traded on the Ethereum blockchain. It is used to pay nodes of the network for executing requested operations, such as transactions.

Smart Contracts were first introduced by Szabo [12] in 1994. In Ethereum, these can be written by, e.g., using the Solidity programming language [13]. Contracts are executed on several nodes of the network within Ethereum virtual machines (EVM). After executing a contract, nodes must reach a consensus of the calculated result.

2.2 REVIEWCHAIN Approach

REVIEWCHAIN focuses on creating and providing persistent access to untampered product ratings and reviews. Its fundamental goal is to avoid central authorities. Therefore, our approach utilizes technologies, such as the blockchain as decentralized and immutable data storage, and decentralized apps to provide access to reviews.

A comparison of the conventional centralized review approach and our decentralized REVIEWCHAIN approach is shown in Figure 1. In the upper part of the figure, central authorities are able to exclude specific consumers from submitting reviews. Further, authorities store reviews in a centralized database. This enables them to modify exisiting reviews and introduce fake reviews by fictional consumers that did not buy the reviewed product. In the decentralized approach, shown in the bottom part of the figure, this is no longer possible. Once stored on the blockchain, reviews cannot be modified, neither can consumers be excluded from providing reviews. We refer to these reviews as *untampered reviews*.

---

[4]https://www.ethereum.org

Our REVIEWCHAIN approach must fulfill four requirements. These are extracted from existing review functionalities used for product reviews on Amazon or app reviews within the Apple App Store:

(1) Authors must have purchased the reviewed product.
(2) Review authors must be distinguishable.
(3) Each author can only submit one review per product version.
(4) Review submissions must not result in any costs for authors.

In the following, we apply our approach to the real-world scenario of mobile app reviews. Therefore, we enable authors to submit reviews within native apps, and provide access to reviews via a web service. By doing so, we aim to highlight additional challanges which result from the usage of different technological environments. Moreover, we realize our approach on Ethereum to demonstrate the applicability and limitations of a blockchain technologies.

3 IMPLEMENTATION CHALLENGES

We discuss major implementation challenges of our approach, as well as possible design alternatives and their trade-offs.

3.1 Submitting Transactions via Mobile Apps

After publishing a smart contract on the blockchain, allowing users to store reviews, it is executed by sending a transaction to the contract address. Each transaction needs to be signed with the user's private key. Figure 2 shows two alternatives to submit transactions via mobile apps. The first alternative implements the transaction handling within a backend. App and backend communicate using a REST API. The transactions, as well as the private keys, are generated by the backend. The keys are stored within a centralized database. After signing, the transactions are sent to an Ethereum node. Nodes can run on the backend itself. In the second alternative, the creation of transactions and private keys are contained within the app. Signed transactions are directly sent to a node.

In comparison, the first alternative offers several advantages. Computation-intensive operations, such as generating and de-/encrypting private keys can be performed on backend-side instead of using a resource-limited mobile device. Also, as fees have to be paid to execute transactions and our approach requires reviewers not to pay for these, the central database of the backend can be used to create a list of all participating users' addresses. These can be funded beforehand with tokens to pay the transaction fees of review submissions. However, this alternative introduces a major drawback, the backend acts as a central authority. By not authenticating users, operators can exclude those from submitting reviews.
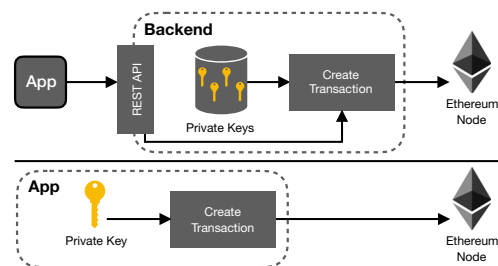

Figure 2: Backend/App-Side Transaction Creation.

Also, the stored private keys can be used to submit reviews without users' behalf. As a result, a decentralized design, as presented in the second alternative, is preferable. In this case, the mobile app must self-contain the private key, as well as the transaction handling.

To realize the second alternative, an official implementation of the Ethereum protocol, called Go Ethereum (Geth)[5], can be cross-compiled to both Android and iOS. Using Java and Swift wrappers, it can be directly called from within the app's native implementation. In addition, Geth generates native wrappers for smart contracts. Geth offers account management, remote node interfacing, and enables interactions with smart contracts. For example, private keys are generated and stored in an encrypted keystore. This keystore offers a standard and light security mode. However, due to the mobile device's resource limitations, the author suggest to use only the light security mode [6]. Although decentralized, this introduces a security risk. The keys are stored in the app's document directory. On regular devices, both Android and iOS, this folder can only be accessed by the application itself. On jailbroken devices apps can access all files on the mobile device. Thereby, the user's private keys can be transmitted to an external server. A server's high computational capacities can be used to brute-force the private key's passphrase and perform transactions afterwards.

3.2 Restricting the Reviewing to App Users

After a user submits a review, the transaction including the review is created, signed, and sent to an Ethereum node. When the nodes of the network reached consensus, the transaction is immutably persisted on the blockchain. Since each person can participate in the network by downloading a copy of the blockchain and running a node, transactions, as well as smart contracts, are publicly visible. This enables participants of the network to copy, modify (e.g., by changing the review text), and resubmit transactions. The participants must not necessarily have downloaded the app the review targets. Thereby, fake reviews can be introduced to the blockchain.

Unfortunately, it is difficult to solve this issue without using a central authority. In the following, we describe three alternatives to exclude non-app users from submitting reviews to the blockchain. First, the smart contract can specify a white-list of addresses allowed to interact with it. Therefore, for each transaction targeting the contract its sender address (msg.sender) is compared with the list. Since all addresses (i.e., app users) are unknown beforehand, the contract must implement a method to register additional addresses. As a drawback, this method can be used to register addresses that did not download the app. Second, an ERC-20 token[7] can be issued and distributed to app users. The smart contract methods must require the sender address to own a token before execution. Further, the contract must ensure that these tokens are untradeable between users. This alternative also requires tokens to be transferred to reviewers by a central authority. Third, a private key to sign transactions can be bundled with the app. By solely white-listing this key's address in the smart contract, only app users are authorized to submit reviews. To use this key, its passphrase must be included in the app as well. This introduces a security risk,

since key and passphrase can be extracted so that fake reviews can be created from outside of the app. Since all users use the same key, we are unable to distinguish them. As a consequence, we cannot ensure that each user only submits one review per product version.

3.3 Reducing Costs of Data Storage

Another challenge is the storage of review data and its associated costs. The Ethereum yellow paper [13] states that a fee of 20,000 gas is required to store a 256-bit word on the blockchain, i.e., 625 gas per byte. Gas is a unit to measure the computational effort required to perform an action within the Ethereum network. The cost of a transaction, e.g., storing data on the blockchain, is calculated by multiplying the amount of gas with the gas price. The gas price is measured in Gwei, where 1 ETH equals 1 billion Gwei. The price is decided by the miners, who perform the transactions, based on the network conditions. If transactions specify a too low gas price, these will not be processed by miners. Vice versa, the higher the price, the faster transactions will be processed. At the time of writing, in February 2018, the gas price to process a transaction below 5 minutes is 5 Gwei, while the median price of the last 1,500 blocks is 22 Gwei. The value of one ETH is $885.

A single 7-day Spotify release for iOS, for example, contains 3,025 reviews with an overall size of 270,110 bytes. To store this amount of data, an amount of 168,818,750 gas is required. Considering a gas price of 5 Gwei the storage costs are 0.844 ETH or $747. This equals $0.247 per review. Using 22 Gwei, the costs are $3,287 or $1.09 per review. The costs can be reduced by defining a low gas price, which consequently requires more time for reviews to be processed.

In the following, we discuss three solutions to reduce storage costs. First, compression algorithms can be used. Alakuijala et al. [2] introduce Brotli, a compression algorithm including a static dictionary and thereby appropriate for short texts. Second, instead of the complete review only its hash can be persisted on the blockchain. The hash length is independent of the review. This reduces costs of longer reviews. The actual review can be stored on a central server. When accessing reviews, their hashes are compared to those on the blockchain. However, this design introduces a single point of failure. Third, a better alternative is to use a a peer-to-peer distributed file system, such as Swarm[8] or InterPlanetary File System (IPFS)[9], see Figure 3. These seek to connect all computing devices with the same system of files. IPFS, for example, provides an immutable, content-addressed block storage model. Each item can be identified by a hash and accessed using an URL. By using a distributed file system, the single authority and point of failure is removed.

---

[5]https://geth.ethereum.org
[6]https://github.com/ethereum/go-ethereum/wiki/Mobile:-Account-management
[7]https://www.ethereum.org/token

[8]https://github.com/ethereum/go-ethereum/tree/swarm
[9]https://ipfs.io

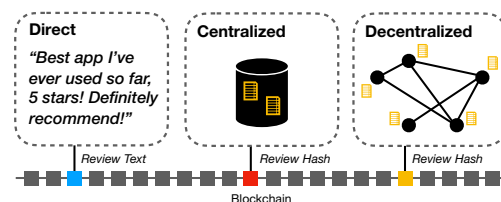**Figure 3: Overview of Data Storage.**

Table 1: Optimized Trade-Offs and their resulting Configurations of Design Alternatives (A1 refers to Alternative 1).

| Trade-Offs | | | Design Alternatives | | | | | Optimi- |
Security	Trust	Costs	Section 4.1: Submission	4.2: Authorization	4.3: Data Storage	4.4: Transact. Fees	4.5: Retrieval	zed for
Good	Medium	Medium	A2: Via App (Directly)	A2: ERC-20 Token	A3: Decentralized	A3: Smart Contract	A2: Local	Security
Poor	**Good**	Medium	A2: Via App (Directly)	A3: Pool Key (App)	A3: Decentralized	A3: Smart Contract	A2: Local	Trust
Good	Poor	**Good**	A2: Via App (Directly)	A2: ERC-20 Token	A2: Centralized	A3: Smart Contract	A2: Local	Costs

3.4 Third-Party Transaction Fee Payments

When performing a transaction, the sender (in our case the reviewer) must pay the fee. As our approch requires that submissions are free of costs for reviewers, we present and discuss three design alternatives. First, as we decided against using a centralized backend to store users' private keys, we do not know their addresses. To collect these, a website can be deployed for users to (automatically) publish their addresses. Afterwards, the addresses can be funded with ETH. This website can be used without submitting a review afterwards. Second, free-of-charge transactions with a gas price set to zero can be used. These transactions are very unlikely to be picked by miners. Therefore, a central authority has to mine all transactions going to the contract's address. As a drawback, the authority could decide against processing single transactions to, e.g., exclude users writing negative reviews. Also, the time to process a review highly depends on the mining capacity of the central authority. Third, to avoid a central authority a smart contract can be used to reward miners. A miner that processed a transaction pointing to the address of the contract has to prove that the gas price was zero. The smart contract can then refund the miner for processing the transaction. The reward is paid from a pool address into which product vendors deposit tokens. It should equal the median transaction fees, normally paid by transaction senders, so that these are as likely to be picked by miners as regular transactions.

3.5 Retrieval of Untampered Reviews

Our aim is to enable users to access reviews on the blockchain in a usable and untampered manner. Therefore, we deploy a decentralized app (dApp). This app must fulfill several requirements: It must be completely open source, operate autonomously, and data must be decentrally stored to avoid central points of failure. Further, dApps must use tokens for access and to rewards miners, e.g., Gas.

From an implementation perspective, the dApp consists of a frontend and backend. The frontend can be developed in any programming language, such as JavaScript. To fulfill the above mentioned requirements, it should be deployed on a decentralized storage. The backend is implemented as a smart contract and deployed on the Ethereum blockchain. The dApp connects to an Ethereum node using an remote procedure call (RPC) connection. As the reviews are timely distributed, the node has to have a full replica of the blockchain. In case a remote node is used, the user has to trust that it returns unmodified results. To ensure that the dApp reads untampered data, users have to start a node by themselves. This downloads a large amount of data, which requires space and time and might be unappropriate for mobile devices.

4 DISCUSSION

Table 1 list three possible configurations of design alternatives (one per row). Each configuration combines different design alternatives with regard to their trade-offs. The first alternative is optimized for high security. Transactions are submitted directly from within the app, an ERC-20 token is used to authorize consumers to submit reviews, and data is stored decentrally (e.g., using IFPS or Swarm). Further, this option is based on smart contracts to reward miners processing the review transactions. For review retrieval this alternative uses a dApp accessing a local Ethereum node. For the second alternative trustworthiness is optimized. In comparison to the first, it bundles the app with a private pool key to sign transactions, and thereby completely avoids central authorities. As a drawback, this design reduces the implementation's security as the private key and its passphrase can be extracted from the app. The last alternative focuses on optimizing costs. It is the same as the first alternative, except for using a centralized database to store the complete reviews (not hashes). Thereby it reduces storage costs, with the drawback of reducing the trustworthiness of reviews, as the database contents can be modified by its operator. Depending on the trade-offs to be achieved, such as optimizing for security, trustworthiness, or costs, different configurations of design alternatives need to be chosen. This can either lead to a completely decentralized approach, based on blockchain technologies, or a combination of decentralized and centralized design alternatives, such as a blockchain to store review hashes and a conventional database to store its contents.

REFERENCES

[1] Cuneyt Gurcan Akcora, Yulia R Gel, and Murat Kantarcioglu. 2017. Blockchain - A Graph Primer. *CoRR* (2017).
[2] Jyrki Alakuijala, Evgenii Kliuchnikov, Zoltan Szabadka, and Lode Vandevenne. 2015. Comparison of brotli, deflate, zopfli, lzma, lzham and bzip2 compression algorithms. *Google Inc.* (2015).
[3] Patrali Chatterjee. 2001. Online reviews: do consumers use them?. In *Association for Consumer Research 2001 Proceedings*.
[4] Tommaso Fornaciari and Massimo Poesio. 2014. Identifying fake Amazon reviews as learning from crowds. *European Chapter of the Association for Computational Linguistics*.
[5] Ulrike Gretzel and Kyung Hyan Yoo. 2008. Use and Impact of Online Travel Reviews. In *Information and Communication Technologies in Tourism 2008*.
[6] M Harman, Yue Jia, and Yuanyuan Zhang. 2012. App store mining and analysis: MSR for app stores. In *IEEE Working Conference on Mining Software Repositories*.
[7] Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *2004 ACM SIGKDD International Conference*.
[8] Nitin Jindal and Bing Liu. 2008. Opinion spam and analysis. In *International Conference on Web Search and Data Mining*.
[9] Susan M Mudambi and David Schuff. 2010. What Makes a Helpful Online Review? A Study of Customer Reviews on Amazon.com. *MIS Quarterly* (2010).
[10] Venkataraman Vivek Bing Liu Mukherjee, Arjun and Natalie Glance. 2013. What Yelp Fake Review Filter Might Be Doing? *ICWSM*.
[11] Dennis Pagano and Walid Maalej. 2013. User feedback in the appstore - An empirical study. *International Requirements Engineering Conference*.
[12] Nick Szabo. 1994. Smart contracts. (1994). http://szabo.best.vwh.net
[13] Gavin Wood. 2014. Ethereum: A secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper 151. (2014).