frontiers in Blockchain

ORIGINAL RESEARCH
published: 22 October 2020
doi: 10.3389/fbloc.2020.497985

A Blockchain Platform for User Data Sharing Ensuring User Control and Incentives

Ajay Kumar Shrestha, Julita Vassileva and Ralph Deters*

Madmuc Lab, Department of Computer Science, University of Saskatchewan, Saskatoon, SK, Canada

We propose a new platform for user modeling with blockchains that allows users to share data without losing control and ownership of it and applied it to the domain of travel booking. Our new platform provides solution to three important problems: ensuring privacy and user control, and incentives for sharing. It tracks who shared what, with whom, when, by what means and for what purposes in a verifiable fashion. The paper presents a case study of applying the framework for a hotel reservation system as one of the enterprise nodes of Multichain which collects users' profile data and allows users to receive rewards while sharing their data with other travel service providers according to their privacy preferences expressed in smart contracts. The user data from the repository is converted into an open data format and shared via stream in the blockchain so that other nodes can efficiently process and use the data. The smart contract verifies and executes the agreed terms of use of the data and transfers digital tokens as a reward to the user. The smart contract imposes double deposit collateral to ensure that all participants act honestly. The paper also presents a performance evaluation of the new platform by analyzing latency and memory consumption with selected three test-scenarios and measuring the transaction cost for smart contracts deployment. The results show that the node responded quickly in all our cases with a befitting transaction cost.

Keywords: privacy, user modeling, blockchain, smart contracts, data sharing, incentive, proof of concept

OPEN ACCESS

Edited by:
Martin Zeilinger,
Abertay University, United Kingdom

Reviewed by:
Sean T. Manion,
Science Distributed, United States
Sharvari Chandrashekhar Tamane,
Jawaharlal Nehru Engineering College, India

****Correspondence:***
Ajay Kumar Shrestha
ajay.shrestha@usask.ca

Specialty section:
This article was submitted to Blockchain for Good, a section of the journal Frontiers in Blockchain

Received: *14 September 2019*
Accepted: *24 September 2020*
Published: *22 October 2020*

Citation:
Shrestha AK, Vassileva J and Deters R (2020) A Blockchain Platform for User Data Sharing Ensuring User Control and Incentives. Front. Blockchain 3:497985. doi: 10.3389/fbloc.2020.497985

INTRODUCTION

In recent years, there has been a rapid advancement in technological innovation and related research on collaborative approaches for sharing users' data among enterprises (Shrestha and Vassileva, 2016). Research-backed data sharing practices are much needed to strike a balance between user privacy, enhanced user experience and profit for businesses (Tenopir et al., 2011). The questions of when and what data should be shared to whom (Meadows, 2014), and how the data owner should get credit or incentive to share their data are increasingly a matter of intense debate and research. User data is collected by different parties, for example, companies offering apps, social networking sites, and others, whose primary motive is to have enhanced business model while giving optimal services to their customers. However, the collection of user data is associated with serious privacy issues. Some data are contributed voluntarily by the user; others are obtained by the system from observation of user activities, or inferred through advanced analysis of volunteered or observed data (Poslad, 2009). The currently dominant model of ownership over user data, usually

encoded in the service license agreements, presumes that ownership is transferred from the user to the enterprise collecting it and, if shared—to the entire network of businesses.

There are privacy and security problems associated with storing personal data. Even the most prominent online services have experienced security breaches and data theft. When trust resides within a centralized service provider for all the storage of data, it could be affected with centrality issues such as intentionally deleting the user data or not delivering the user data due to a technical failure.

Sharing user data across applications and enterprises helps to improve personalization of functionality, interface and options and thus creates a better user experience. However, there are problems associated with the security, privacy and user control of sharing user data. Security of sharing has been addressed by standard security techniques as well as experimental approaches, for example, carrying out all the communication without trusting anybody and possibly replacing the centralized controlling authority (Shrestha and Vassileva, 2016). Various advanced technologies have been deployed as computational backbones to collect and share user data including cloud computing services, RFID (Radio-frequency identification), as well as various security technologies to protect the collected user data from hackers (Shrestha, 2014). Federated learning (McMahan and Ramage, 2017) allows to mine data scattered in distributed locations.

However, to be compliant with strong privacy legislation (e.g., the EU GDPR), the data owner's consent needs to be asked again, when data is shared to be used by a different system. This is an obstacle since it is obtrusive for a user who does not see what there is to gain. Typically, consent forms are long and opaque; they give no option to users to select the data they are willing to share and prevent sharing other data; it is "take it or leave it." Users do not even read them but scroll down and click "Agree" since otherwise, they cannot use the service. So, it becomes hard or even impossible for users to remember what consent they have given to which enterprise and to keep track of who accesses their data and for what purpose. A flexible mechanism for obtaining and renewing consent for data use and sharing is required that provides appropriate and meaningful incentives for users to capitalize from data sharing and ensures transparency for users to be aware of which of their dataset has been accessed, by whom, for what purpose and under what conditions.

We address all these issues (security, privacy, user-transparency and control, and incentives for data sharing) by proposing a new platform for user modeling using distributed ledgers and smart contracts, relying on user-controlled privacy and data-sharing policies encoded in smart contracts. It also supports creating incentives for users to share their data in terms of rewards (micro-payments or credits) encoded in smart contracts. Thus, users become owners of their data and can decide how their data is collected and used, as well as shared, and benefit not only in terms of improved personalized experience with the service but also directly, for example, by participating in the share of the advertising revenue generated by the service provider. We present an implementation of the new platform to share user data in a decentralized fashion among different businesses providing services in the travel-booking domain and evaluate its performance. This paper presents a significant extension of our previous work (Shrestha et al., 2017), including an implementation of the smart contracts. Throughout this research paper, we have used the term "*new platform*" to represent our proposed system that uses multiple distributed ledger technologies.

A distributed digital ledger (blockchain) system stores encrypted and hashed authenticated data, which is immutable, and any changes or mistakes can be traced back to their source. We used MultiChain (Greenspan, 2013) and Ethereum (Buterin, 2015) blockchains to provide an uneditable private record of all transactions made with user data. MultiChain, being a private blockchain, has the potential to replace traditional centralized databases used to store user data in a decentralized way offering more cryptographic auditing features. Optionally we can store any published item off-chain that saves storage place and bandwidth. MultiChain, however, due to its current architecture, cannot support an access control mechanism, which the proposed platform needs, to provide users means to control how their data is shared and the rewards they get. Therefore, we use Ethereum, which supports smart contracts and commits the contracts' transactions. The smart contracts govern accountability of access and provide incentives to the users and application owners for sharing user data. Combining the security and immutability of data stored in the blockchain, with the specific strengths of two popular blockchains—MultiChain and Ethereum—allows us to combine their advantages: proper data storage and data sharing and smart contracts for access control mechanisms. In this way, our new platform addresses the shortcomings of traditional centralized user models used by internet businesses, which have security vulnerabilities, lack accountability, and take away the ownership, control and incentives for sharing of users over their data.

The rest of the paper is organized as follows: Section II provides an overview of blockchain technology, MultiChain, Ethereum, and smart contracts. A brief analysis of the existing frameworks for user data storage and sharing with their limitations is provided in section III. Section IV presents the proposed platform and its implementation for decentralized data sharing in a travel domain while ensuring users' privacy. Section V presents an evaluation of the performance of the implemented new platform by observing latency and memory consumption and measuring transaction costs while deploying the contracts. In section VI, we provide a descriptive analysis of the result, and in section VII, we conclude our work.

BACKGROUND

This section presents an overview of blockchain and smart contracts.

Blockchain

Blockchain is a data structure used to create a public or private distributed digital transaction ledger which, instead of resting with a single provider, is shared among a distributed network of computers. Distributed ledgers can be used, for example, to store critical assets in the supply chain to track their ownership

and changes in state (Shrestha and Vassileva, 2016). The data is stored in a distributed and redundant fashion in the blockchain, and each node verifies each transaction. Therefore, it is hard for malicious parties to attack and manipulate the data to their advantage. Thus, blockchain does not require a trusted central server or entity and it is often called "trustless." The basic software pattern of blockchain technology was introduced in the source code for the digital cash system, Bitcoin (Nakamoto, 2008), but it is not limited to crypto-currencies. The blockchain software pattern represents a digital ledger—a database with an immutable record of every transaction which has ever taken place. Each block aggregates a timestamped batch of transactions to be placed in the chain. There is also a cryptographic signature to identify each block. Each block refers to the signature of the previous block in the chain, and that chain can be traced back to the very first (Genesis) block created in the chain.

The key idea is that there is no centralized authority to determine what is true or what is false; instead, multiple distributed parties come to a consensus that is entered into the ledger and after that can be accessed by anyone (Shrestha and Vassileva, 2016). Computationally, it is impracticable for anyone (or less than a majority consensus) to go back and alter the history because the blockchain represents a chronological chain of events. Because the data is stored in distributed and redundant fashion and each transaction is verified by each node, it is tough for malicious nodes (corrupted parties) to attack and manipulate the data to their advantage.

Blockchain technology holds promise to transform data management in many domains. Although blockchain was developed as a platform for virtual currency, the applications of the technology have since quickly evolved to numerous fascinating use cases. Many industries, including in the financial technologies (fintech) and banking sectors, commercial supply chains, healthcare, building industries, etc. are now working on incorporating blockchain (distributed ledgers) technology as a core of their data-management systems (Friedlmaier et al., 2016). In contrast to the centralized system, blockchain technology can be totally transparent to the users and very promising to incentivize users for data sharing. It also naturally supports building up incentives for users to share their data, in terms of rewards (micro-payments or credits) encoded in the smart contracts. Details on the smart contracts are presented in section Smart Contracts.

There are three categories of the blockchain, each with a slightly different set of protocols and consensus mechanisms. The consensus is to achieve agreement across validators (or miners) in a network on every new ledger of transactions. The blockchain is usually equipped with consensus protocols to tolerate unreliable involved parties or malicious nodes. The first category of blockchain is public in which anyone can participate in the chain and contribute to the consensus process. The read permission or the right to see the public blockchain is always open to anyone with access to the internet. The second category of blockchain is a consortium in which pre-selected nodes control the consensus process. The right to see the consortium blockchain remains either public or restricted to the participants. The third type is private blockchain in which the transactions are contained within a closed community and are of interest only to the members of the community present in the chain. The private blockchain adopts the core idea of blockchain as a distributed ledger technology (DLT) but assigns the private validator, which is a member of a consortium or separate legal entities of the same organization. The right to see the private blockchain remains restricted to the participants. The blockchain can also be referred to as permissioned or permissionless, each with slightly different properties. Permissioned blockchain is faster, usually, a trusted network offering managed upkeep and private membership such that members can contribute to the consensus process only after meeting some criteria. On the other hand, permissionless blockchain is slower, trust-free, open, transparent, and a public membership network such that any members can contribute to the consensus process without any restriction (Wood, 2016). Therefore, depending upon the consensus mechanism, different blockchains may be suitable for distinct types of business use cases. The next two sections review two major blockchains.

MultiChain

MultiChain is a private permissioned blockchain that provides the privacy and control required in an easy to configure and deploy package (Greenspan, 2013). It supports UNIX and Windows servers and produces a rich JSON-RPC API for easy integration with existing systems. Unlike any other blockchains, MultiChain solves the problems of privacy and openness via integrated management of user permissions with its three main objectives (Greenspan, 2013):

1. To enable only the chosen participants to see the blockchain's activity,
2. To ensure that only selected transactions are permitted,
3. To securely conduct mining without proof of work and its associated costs.

MultiChain being a closed system allows participants to control the maximum block size and set all of the blockchain's parameters in a configuration file, thereby resolves the problems of scalability. Also, it only contains transactions that are of interest to the participants. It can hash up to 1 GB per item (Off-chain data) into the blockchain, with the data itself delivered rapidly over the peer-to-peer (P2P) network. The miner of the Genesis (first) block automatically receives all administrative privileges, including the rights to manage the access permissions of other users (Greenspan, 2013). The administrator can grant other participants any privileges for network transactions containing special metadata.

Ethereum

Ethereum is an open-source, public permissionless blockchain to create decentralized applications (dapps) where users interact with the online services in a distributed peer-to-peer manner that takes place on a censorship-proof foundation. Developers can design interfaces and business logic with any of the known programming languages and tools.

Ethereum has "ether (*ETH*)" as its virtual currency which can be used to pay a transaction fee and to provide a primary liquidity layer for exchanging digital assets. Ethereum also provides a

technology called "smart contract," which is a piece of software that automatically executes the contract. There are "messages" in Ethereum that can be created either by an external entity or internally by a contract, unlike the Bitcoin transaction, which can only be created externally (Buterin, 2015). There is also an explicit option for Ethereum messages to contain data and the recipient of the Ethereum messages to return a response. The Ethereum also has "transactions"—signed data package that stores a message to be sent from an externally owned account. Transactions contain the recipient of the message, a signature identifying the sender, the amount of ether and the data to send, as well as two values called *STARTGAS* and *GASPRICE*. STARTGAS is the maximum number of computational steps the transaction execution can take, and GASPRICE is the fee per computational step which the sender pays in "*wei*" (1 wei = 1 ether/1e18).

The state in Ethereum is made of accounts each consisting of 20 bytes address and state transitions. An account contains four fields (Buterin, 2015), which are:

(a) The nonce—a counter used to make sure each transaction can only be processed once
(b) The account's current ether balance
(c) The account's contract code, if present
(d) The account's storage (empty by default).

We used the Ethereum blockchain as a semi-financial application such as on-blockchain escrow, which allows users to enter into contracts and manage them using their ether to deal with non-monetary assets such as the user profile data. The issues with the public Ethereum blockchain has always been the scalability, an exceedingly long block validation time and GASPRICE. Until now, Ethereum imposes a proof of work (PoW) consensus algorithm, which uses a high computational power (i.e., electricity). In PoW, each node (miner) on the network competes to solve the cryptographic puzzle and reach consensus. As a node solves the puzzle, it broadcasts the block so that other nodes can validate the correctness of the hash value. The miner solving the puzzle at first gets incentives in the form of cryptocurrency.

It is expected that in the future, Ethereum will completely adopt proof of stake (PoS) consensus mechanism eliminating the problem of high computational power[1] In PoS, the block validators (forgers) are chosen based on the number of virtual coins they possess or put at stake. It supports the fact that more coins at stake give a better chance to the node to be selected as one to validate the block of transactions. However, the malicious nodes lose all the coins they put at stake if they try to include faulty transactions (called slashing). It consumes much less computing power when compared to the PoW consensus model. For our model, high transaction speed is not mandatory, and scalability will not be a factor as well because most of the data storage and sharing parts will be done in the private MultiChain, and only the access control policies will be stored in the public Ethereum blockchain.

[1]https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ

Smart Contracts

The smart contracts are instances of contracts deployed on the Ethereum blockchain (Buterin, 2015) although the term was originally coined earlier (Szabo, 1997) in the context of electronic commerce protocols between strangers on the Internet. A smart contract stores the rules which:

1. Negotiate the terms of the contract,
2. Automatically verify the contract, and
3. Execute the agreed terms.

A smart contract consists of different functions that might be called from outside of a blockchain or by other smart contracts. Blockchain, coupled with smart contract technology removes the reliance on the central system between the transaction parties. Since the smart contracts are stored on the Blockchain, all the connected parties in the network have a copy of them.

A smart contract can execute the agreed stored process when triggered by an authorized or agreed event. All contract transactions are stored in chronological order for future access along with the complete audit trail of events. If any party tries to change a contract or transaction on the Blockchain, all other parties can detect and prevent it. If any party fails, the system continues to function with no loss of data or integrity. It, therefore, creates a single large secure logically computer system without the risks, costs and trust issues of a centralized model.

We used Solidity[2] programming language to write smart contracts because it only allows performing basic operations on its basic types resulting in lightweight code. The EVM (Ethereum Virtual Machine)[3] code is used in the contracts, and that consists of bytes, each representing an operation. The code can access the amount of *Wei* sent in the transaction and data of the incoming message, block header data, and return a byte array of data as an output. For deployment and testing of smart contracts on Ethereum blockchain, a truffle framework[4] can be used since it has built-in smart contract compilation, linking, deployment and binary management. As of now, our smart contract runs on EVM. However, with the implementation of the Ethereum Web Assembly (EWASM)[5] in the future, smart contract development can be done in any other programming languages besides Solidity, and that will also speed up the function call between Web Assembly and JavaScript (JS).

RELATED WORK

The tremendous technological advancement in the last few decades has brought many enterprises to collaborate in a better way while making intelligent decisions. The use of Information Technology tools in obtaining data of people's everyday life from various autonomous data sources allowing unrestricted access to entire data has emerged as an important practical issue and has given rise to legal implications. No longer a hypothetical

[2]https://github.com/ethereum/solidity/blob/v0.5.1/docs/index.rst
[3]https://github.com/ethereum/wiki/wiki/Ethereum-Virtual-Machine-(EVM)-Awesome-List
[4]https://github.com/trufflesuite/truffle
[5]https://github.com/ewasm/design

or occasional occurrence, the use of data by individuals other than those who originally gathered the data, termed "data sharing," is currently encouraged or mandated by parallel efforts in the legislature through the twenty first Century Cures Act, biomedical journal leadership through the draft data-sharing policy of the International Committee of Medical Journal Editors, charitable foundations such as the Wellcome Trust and the Bill and Melinda Gates Foundation, and the National Institutes of Health (NIH) in its recent request for information on data management and sharing strategies (Bierer et al., 2017).

Data sharing models at an enterprise-level are the networked information systems allowing users to create profile and store data and make it accessible to others as per the agreement. We present different mechanisms for sharing user data considering four different systems (1) *Single system*, (2) *Centralized servers*, (3) *Decentralized (agent-based, peer-to-peer, service-based)*, and (4) *Blockchain-based (decentralized + incentives)*.

Big data hoarders such as Google, Facebook, Twitter, Microsoft etc. fall under the category of the *single system*. They use their own ecosystem for both the collection and usage of the user profile data. These single systems have contributed to the data transfer Project (DTP)[6] allowing any other service to use their existing APIs and authorization mechanisms to access data. Users can back up (download) their data, leave one service and try out new services whenever they want. The DTP offers an open-source service-to-service data portability platform[7] that enables users to move their data between online service providers. The contributors to the project have felt that interoperability is central to innovation for data portability. Amazon also has OpenDataRegistry[8] to allow the public to provide/upload and discover/access useful datasets via AWS resources. The datasets to be shared must be documented, actively maintained, and supported by data providers since they are not provided by AWS but are curated or optimized for analysis using AWS tools. Datasets are included at the discretion of the AWS Open Data team, which may remove datasets from the registry at any time. Dataset Search[9] from Google is another free search engine tool for searching 25 million publicly available datasets. Google does not curate or provide direct access to datasets directly, so the dataset publishers must use the open standards of schema.org to describe their dataset's metadata. Google then indexes and makes that metadata searchable across publishers.

The report from Schmidt (2018) outlined several experiments showing the terrifying scope of the collection of user data by Google, which then targets the users with paid advertising. Google collects data whenever the user interacts with Google by using their platforms (e.g., Chrome, Android etc.), applications (e.g., Google Maps, YouTube etc.), publisher tools (e.g., AdSense, G analytics) and other tools (e.g., AdWords, AdMob).

The centralized architecture, in most cases, doesn't collect and share the diverse fragments of user data coming from the autonomous and independent entities (applications, agents, devices, sensors, services) in service-oriented, the mobile and ubiquitous computing environments (Dolog and Vassileva, 2005). Centralized architecture is limiting, since it forces unified logical structure (ontology) for the user model, and thus loses the contextual information that exists in the various applications, closer to the user data. Centralized user models are still predominant in business enterprises due to the prevalence of efficient client-server architectures and web technologies that can be used to implement user modeling servers. The physical and logical centralization are different concepts. The user data can be stored in distributed storage spaces, but using the same data schema and the procedural components of the user modeling might be centralized (Carmagnola et al., 2011). On the other hand, the physical storage of user data at one central point doesn't necessarily imply centralized user modeling, if different semantic representation schemas are used, and/or the user modeling processes are independent of each other (Vassileva et al., 2003).

A few prominent examples of data sharing systems include different frameworks to achieve interoperability of distributed model with a centralized server such as Mypes (Abel et al., 2013), online P2P file-sharing networks and data management systems, collaborative repositories such as Wikidata (Vrandečić, and Krötzsch, 2014) etc. Almost all of these systems implement different architectures, and their evaluation is based on different non-functional requirements, such as efficiency, scalability, or reliability (Davoust, 2015). Accordingly, most of the research papers relevant to these design frameworks are focused on the optimization of those properties. The centralized architecture, at most cases, is not sufficient to collect and share the diverse fragments of user data coming from the enormous variety of autonomous and independent entities (applications, agents, devices, sensors, services) comprising service-oriented, mobile and ubiquitous computing environments and IoT (Dolog and Vassileva, 2005). Often the centralized user modeling technique has a predefined point of access that leads to the central point of failure. Replication of the data via mirroring the servers is used to secure the data, but that usually comes with high communication costs.

A distributed architecture for sharing and re-using multi-application life logs is presented in Iyilade and Vassileva (2013). User-data, e.g., life logs from different systems are gathered by agents and forwarded to a centralized broker responsible for the user modeling process that comprises request analysis, source selection, source connection, semantic mapping, data integration and response transformation.

In the IoT domain, there are systems such as MobiTribe (Thilakarathna et al., 2014), which has a distributed but logically centralized user model. It focuses on sharing data between mobile devices and applications and uses a centralized content management system as a moderator for the exchange of information. PersonisAD (Assad et al., 2007) is an active, distributed, scrutable model developed for mobile and ubiquitous computing environments. It gathers information from different sensors associated with users and their environment and combines the data logically to infer their preferences and adapt the functionality of user services to provide a richer experience. Like in Dim and Kuflik (2012), the distributed user model is

[6]https://github.com/google/data-transfer-project
[7]https://datatransferproject.dev/
[8]https://github.com/awslabs/open-data-registry/
[9]https://datasetsearch.research.google.com/

generated by single method standalone agents, each storing a unique attribute of a user in a logically centralized vector model.

Furthermore, it is observed that accommodating the conflicting interests among the users is not separable from the architectural design that applies optimization of the specific system properties and involves trade-offs with the participants' autonomy (Carmagnola et al., 2011). In Niu et al. (2004) and Vassileva et al. (2003), the user models are logically and physically decentralized, held by different agents, and the information is gathered from different agents only temporary, on-demand, for a given adaptation purpose.

User data sharing mechanisms have evolved also from centralized, e.g., FTP-servers, to peer-to-peer, where the data is hosted and exchanged directly by users. Peer-to-peer architectures can also be facilitated by centralized indexes (Fanning et al., 1999) to speed up the search, but a competing non-functional requirement of ensuring anonymity has led to the development of alternative decentralized and hashed ways of storing data. In a structured peer-to-peer design such as Chord (Stoica et al., 2003), participants are not given the privilege to store their data at the other peers of their choice; instead, they have to store their data with arbitrary other peers.

More recently, the practice of sharing events from our everyday lives on social networks has led to massive amounts of user data collection, especially by internet giants, such as Facebook and Google. Web 2.0 or participative web, which facilitates sharing of user-generated content has removed traditional bottlenecks to publication and public communication by using platforms for "mass self-communication" (Castells, 2007).

The OECD (Graham and Sacha, 2007) has defined the notion of the participative web as an internet-based concept highly influenced by the intelligent web services that entitle users to contribute in further development, rating, collaboration, distribution and customization of the internet contents and applications. As the Internet has become the global means of communication in people's everyday lives, users draw on new social media applications to express themselves via user-created content. There is a plenitude of various digital devices connected to the web (internet)—from desktop computers to small appliances and portable devices at multiple locations. This has led to an ever-increasing need for users to access online services and various user data from different computing devices (Schilling et al., 2012). The development of cloud-computing technologies has allowed users to share their data conveniently both across devices and with each other. However, due to the need for applications to understand the context of the user to be able to provide good services, there is now rapidly increasing the need for user data sharing and use for a variety of changing purposes, to improve cloud-based services personalization across mobile, and Internet of Things (IoT).

About data sharing, well-developed incentive mechanisms positively motivate the users in virtual communities to willingly engage in knowledge sharing with others (Chen et al., 2012). An incentive mechanism was introduced in Liu et al. (2017) for rational secret sharing schemes and a fair data access control scheme for cloud storage. In the scheme, the decryption key reconstruction activity is to be formalized, and then its security, fairness, and correctness are defined. Afterwards, the decryption key obfuscation is performed with a generation of many fake keys over the shared data. During the exchange of shares, users must send their shares when they deviate from the prescribed scheme and thereby access the shared data together.

In the medical field, the research community is increasingly recognizing the importance of sharing patients' data from clinical trials to maximize the knowledge gain from the research effort (Lo and DeMets, 2016). European Medicines Agency[10] (EMA), several drug companies and one other trial funder have already implemented clinical data sharing. However, the issue with them is to address appropriate and meaningful incentives to capitalize on the promise of data sharing, and of course, they rely on the centralized system for data storage and management. MedRec (Azaria et al., 2016) uses blockchain technology to handle electronic medical records (EMRs). The participating medical stakeholders (researchers, public health authorities, etc.) in the network act as blockchain "miners." It provides the participants with access to aggregate, anonymized data as mining rewards, in return for sustaining and securing the network as miners.

For data sharing in the scientific research domain (Shrestha and Vassileva, 2018), proposed a usable blockchain-based model for a collection of researchers' data, providing accountability of access, maintaining the complete and updated information, and a verifiable record of the provenance, including all accesses/sharing/usages of the data. It enables researchers/data owners to be incentivized either with digital tokens (ether) or with acknowledgment for their efforts in collecting the data during the publication of the research article. The initial adoption of such blockchain-based systems is necessary for continued use of the services, so the user acceptance behavioral model of such a system has been investigated in Shrestha and Vassileva (2019). They used an extended TAM-based model and implemented the descriptive statistic, measurement models and structural models to uncover the relationship between perceived usefulness, perceived ease of use, quality of system, perceived enjoyment and intention to use the blockchain-based data-sharing system.

Additionally, there are some proposals for using blockchain incorporating access control measures to ensure the privacy of data (Zyskind et al., 2015b). The authors also proposed a scheme named Enigma based on multi-party computation (MPC) (Zyskind et al., 2015a). It theoretically resolves the issues on access control but possesses a substantial computational burden with very high communication overhead. It also does not support the general-purpose computation of blockchain smart contracts.

To motivate users for data storage and retrieval services, Interplanetary File System (IPFS) (Benet, 2015) proposed a scheme to incentivize the participating nodes with Filecoin (cryptocurrency) (Protocol Labs, 2017) for hard drive space instead of computing power. It uses the proof of the replication consensus model where miners are required to prove to a verifier that they have created different copies of the files on the network. Similarly, Siacoin (Vorick and Champine, 2014) and

[10]https://clinicaldata.ema.europa.eu/web/cdp/home

Storjcoin (Wilkinson, 2014) are other similar schemes to support distributed data storage by shredding the user-uploaded file, encrypting each segment and spreading the file ciphertext to the participating nodes across the network (Vorick and Champine, 2014; Wilkinson, 2014).

In Molina-Jimenez et al. (2019), the authors mentioned some of the challenges present in the proposals adopting Blockchain as part of their solutions. The authors discussed that the high cost of having all the operations performed over the Blockchain makes them impractical. Also, they proposed approaches to minimize this cost, for example, by "hybrid implementation," which separates contractual operations into decentralized operations for Blockchain interactions and centralized operations for a trusted third party (TTP) interactions. As expected, they argue that all the operations that are not crucial to be in the Blockchain should be moved away from it and directed to the TTP, which is a gateway assistant giving insights about allowing or denying attempts of data access. We adopt this suggestion in our proposed platform.

Furthermore, research in the smart contract technology has also evolved from the conceptual-based architecture to the application-oriented scenarios. According to the systematic mapping study conducted by Alharby et al. (2018), 64% of the total 188 relevant papers they discovered on smart contracts were from the applications category compared to a very few application-oriented academic papers which they discovered in 2017. In 2017, they extracted only 24 papers on smart contracts in total, and their study shows that about 66% of the papers focused on the conceptual level identifying and tackling smart contract issues. We have now seen many academic researchers taking up smart contract technologies in actual building applications on the top of the blockchain.

The next section describes our proposed platform, which ensures the preservation of user control over their data while supporting data sharing among different enterprises in a decentralized manner. Some of the issues of data sharing are associated with serious privacy and security factors. Even the most prominent online services have experienced security breaches and data theft. The sharing platform also needs an effective incentive mechanism that can realize the transparent access to the user data while distributing fair incentives. The emerging literature on the topic includes decentralized user data sharing approaches. However, there is no universal method to keep track of who shared what, to whom, when, for what purpose and under what condition and in exchange of what in a verifiable manner until recently when the distributed ledger technologies came to become the most effective means for designing decentralized P2P network. This research includes an engineering approach of specifying the operations for designing incentives and user-controlled data sharing mechanism. We propose new blockchains- and smart contracts-based decentralized user data sharing platform conceptualizing user-controlled data sharing practices. With the smart contracts stored in the blockchain, users can retain the ownership of data and are incentivized as per the agreed terms. To enable effective user data sharing among enterprises, we used as an example domain of the travel industry that covers travel and tour agencies, hotels and resorts, airlines, restaurants, etc. The travel industries within the hospitality domain usually want to compete successfully, and they must do so by using technologies to drive value to all the parties associated with them (Cassidy and Chae, 2006). By sharing real-time data about users, which is being updated simultaneously by the different participating entities of the consortium, such as travel agencies, hotels, resorts, airlines, restaurants, shopping malls etc., they will able to offer personalized services after analyzing their users' preferences. In addition to privacy, user control and incentives for sharing, our proposed platform supports immutability, authenticity, enhanced security, trusted records and is a promising means to share user data in other different domains including e-commerce, research community etc.

PROPOSED PLATFORM

The privacy-related legislation, General Data Protection Regulation (GDPR) as of May 25, 2018, regulates the processing of the personal data of the individuals and demands personal data erasure. However, data on the blockchain are always immutable. This could be a challenge while adopting blockchain as part of the solution. Therefore, we have a general architecture of the user data sharing system based on blockchain and off-chain data storage as shown in **Figure 1**. The general architecture ensures that the actual user data is never exposed to the blockchain. The user data is first hashed and encrypted before uploading it into the off-chain storage. The data owners from their client applications can directly store them on the off-chain storage. The terms and conditions regarding the access to user data are encoded in smart contracts along with the metadata and hash of the data and published on a blockchain platform (Ethereum). The hashes of the data on the blockchain prevent the middleware from tampering the data. The content-based addressing makes hashes of data serve as their identifier for retrieval. When the data consumer invokes the smart contracts for accessing the user data, only the successful invocation of the contracts results in the release of the key for decrypting the user data. Then the trusted program (Ning et al., 2018) extracts the hash from the blockchain, uses this hash to retrieve the data from the off-chain storage, decrypts it and releases the data to the data consumer while settling the incentives for the data owner. Blockchain and smart contracts support users by giving the users full transparency over who accesses their data, when and for what purpose, allowing the users to specify a range of purposes of data sharing, kinds of data that can be shared, and classes of applications/companies that can access the data, and providing an incentive to the users for sharing their data (in terms of payment for the use of the data by applications, as specified by the contracts). The general architecture presents the underlying off-chain user data storage mechanism which could be a centralized data store hosted by a trusted party. When trust resides within a centralized service provider for all the storage and management of data, it is hard to mitigate the various risks, for example, of data being misused, hacked or sold to any other bodies without user consent and even destroyed when the

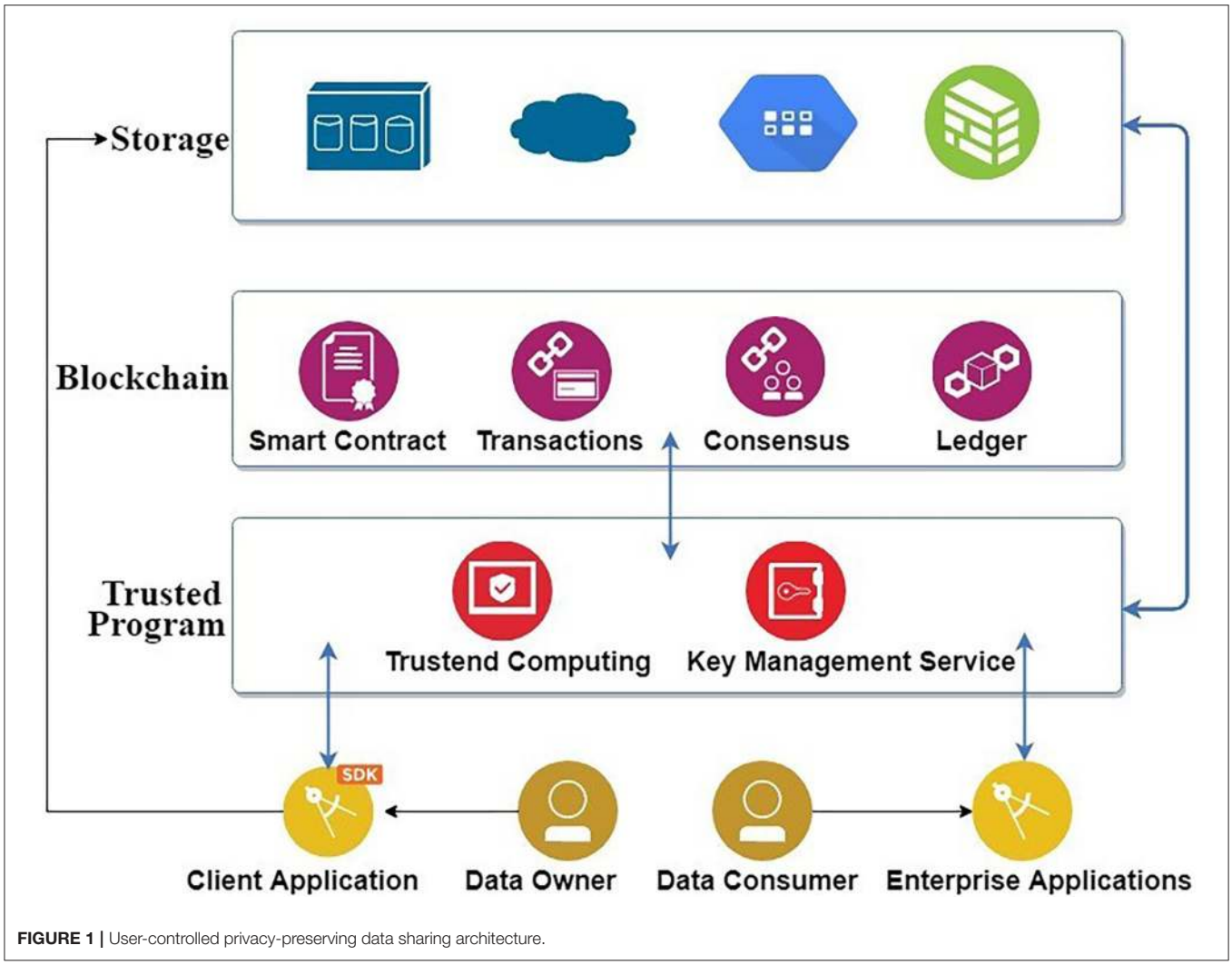

FIGURE 1 | User-controlled privacy-preserving data sharing architecture.

company defaults. Therefore, we present a new platform with separate private permissioned blockchain called MultiChain as a solution to both on-chain and off-chain data storage, encryption, hashing and tracking of data, together with Ethereum (for access control). Off-chain blockchain implementation with user data storage can be successfully achieved with the limited number of peers in the MultiChain (Greenspan, 2013). Users can optionally store any published data in off-chain that saves storage place and bandwidth. The similar idea of storing off-chain data and accessing them via MultiChain has also been proposed in our previous research works (Shrestha et al., 2017; Shrestha and Vassileva, 2018) and others' works such as Yang et al. (2019) and Ferrer-Sapena and Sánchez-Pérez (2019). MultiChain nodes handle key operations such as hashing and encrypting the user data, storing the encrypted file locally (outside of blockchain), committing the hash of the file on the blockchain, searching the required data, verifying the data and delivering the data.

We, therefore, introduce two blockchains: MultiChain to share user profile information among enterprises in their private network, and Ethereum to store securely the access-control policies as smart contracts. A user can register into the system by providing her basic profile information and activating the smart contracts on an Ethereum network node. The Ethereum node automates the functionality to support the user-controlled privacy regarding (a) with whom the users' data will be shared, and (b) how the user will be incentivized once her data are being shared with other third parties. The communication with the Ethereum network node is made directly with our own hosted enterprise nodes. Since the smart contracts are stored on Ethereum, the users can have their own ether addresses which they use to safely store ether (ETH) in their own cold wallet (offline wallet) as well. Once the users' data are being used by any other participating organization, the corresponding users (data provider) will be incentivized with ether. We explain the workflow in section User Incentives for Sharing.

For sharing the user data among enterprises, MultiChain blockchain is installed on each participating enterprise end, which can publish the user data as items into the stream and share them in the network according to the smart contracts set by the users.

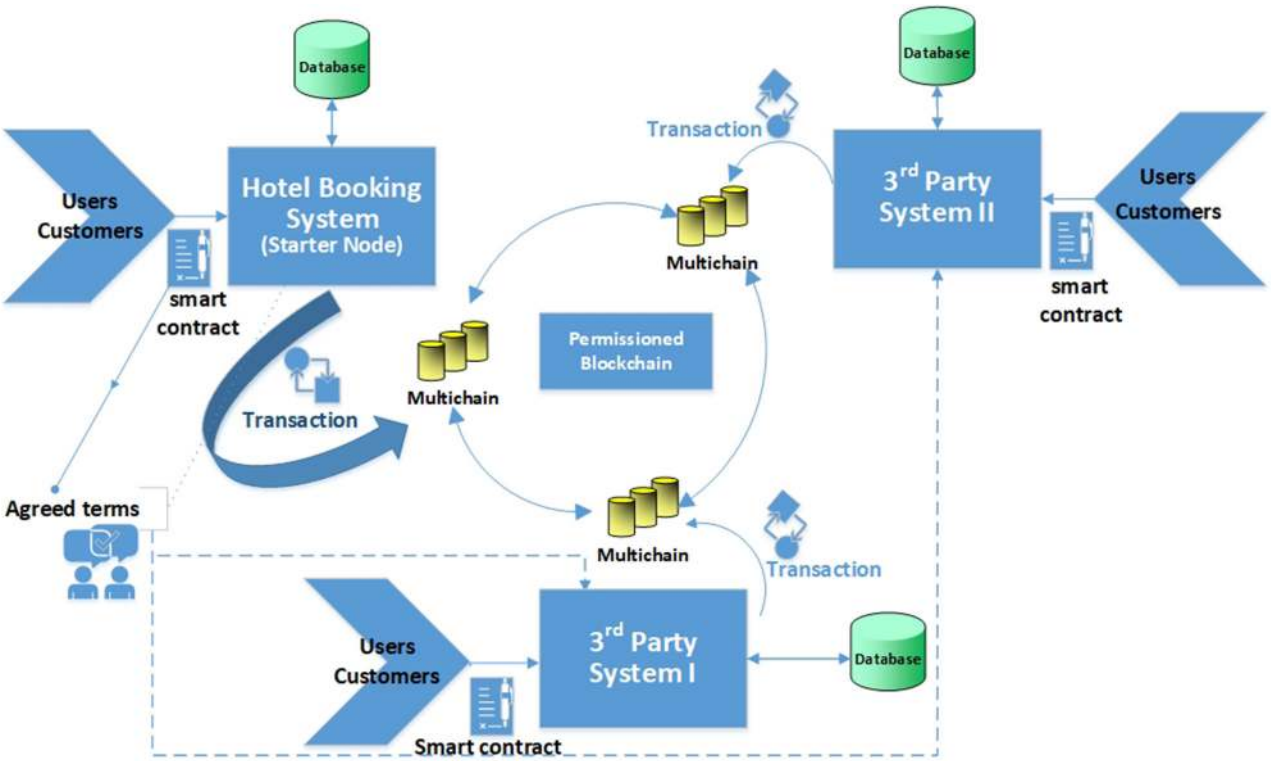

FIGURE 2 | User-controlled privacy-preserving data sharing platform in the travel booking domain.

Data Sharing Solution

Figure 2 presents a functional model of the new platform. It will be explained below using as an example travel data sharing system in the context of the online travel industry. The parties involved in our travel data sharing scenario are imaginary Saskatoon Grandee Hotel, Saskatoon Travel and Tours, and Saskatoon Shopping Mall. With customers creating valuable data at every stage of their journey, travel industries can leverage this opportunity to collect and share these data and offer better-personalized services to the customers. Although data analytic complexity is a major challenge for travel industries and marketers right now (Fospha, 2019) and that requires collaboration between all teams, from customer relationship management (CRM) to analytics teams (Trainor et al., 2014), we are specifically looking into the blockchain and smart contract technologies to offer an effective incentives solution to the customers for sharing their data, and track who shared what, with whom, when, by what means and for what purposes in a verifiable fashion in this paper. We take user data as the standard data required by hotel reservation systems that include name, nationality, home address, contact email address, the purpose of visit and the duration of stay (check-in and check-out dates). The Hotel Booking System (Saskatoon Grandee Hotel) has the first node in the network that is responsible for the collection of the user data. By default, this node acts as an admin who can grant other nodes admin privileges too. In our case, the permissions for other nodes: "connect," "send," "receive," "issue," "create," "mine," "admin," and "activate" are set by the first admin node. These permissions allow the nodes to perform certain operations. With the *connect* permission, the node can see the blockchain content. The *send* and the *receive* permissions allow a node to send and receive funds, respectively. The *create* permission allows the node to create a stream for sharing data and this is one of the key features of the proposed data sharing solution. The streams' creation and items publication are explained in more detail below. The *issue* permission allows a node to issue assets whereas *mine* permission allows them to take part in providing solutions to adding the validated blocks into the chain. With *activate* permission, the node can grant, or revoke *connect, send,* and *receive* permissions to other nodes that wish to join the consortium blockchain network. Finally, with *admin* permission, the node can change all the permissions for other nodes in the blockchain network. Besides, through the smart contract, only the selected eligible nodes can access or consume the customer data by subscribing to the corresponding published streams, which we explain in the next section. The data provider which includes the customer and the node offering user data are incentivized as per the negotiation made on the options between the two parties. We confined the functions (methods) into our smart contract as per the roles of the participating entities to successively execute different tasks between the data provider and the consumer.

We developed and deployed a general hotel reservation web application on one UNIX machine. The web application was developed in PHP with apache server and MySQL as backend. The machine running MultiChain serves as one of the nodes, that collects customers' data with proper validation. The public Ethereum blockchain stores and executes smart contracts. All the registered customers and data consumers have Ethereum addresses, which are used to transfer the ether while sharing the data as per the smart contracts. Users create their profile in the hotel reservation system as the first node (Node 1) in the blockchain by registering their information and simultaneously choosing which of their data can be shared with third parties. The data stored in the repository is converted into an open data JSON format, which is published via stream in the MultiChain. The stream in MultiChain is used for general data storage and retrieval. Third parties, for instance, Saskatoon Travel and Tours, and Saskatoon Shopping Mall have different nodes that also contain Multichain in their system. They get the addresses and are given various *permissions* to be in the closed network of the blockchain. One node can have multiple addresses to conduct multiple transactions randomly between them, making the process more anonymous. Public key encryption is an underlying technology of MultiChain, so all the connected nodes generate their own pairs of public addresses and private keys.

MultiChain limits the blockchain access to a group of permitted users by expanding the "handshaking" process when two blockchain nodes connect governed by the following Algorithm A.

ALGORITHM A | Peer-to-peer (P2P) connection (Greenspan, 2013)

1. A node is identified as a public address.
2. A node verifies that the other's address is on its own version of the permitted list.
3. A node sends a challenge message to the other party.
4. A node sends back a signature of the challenge message to prove their ownership of the private key linked to the public address they presented. If either node is not satisfied with the results, it aborts the P2P connection.

As shown in **Figure 3**, we first initiate a Multichain in the Hotel reservation system for the first node Grandee Hotel (Node 1) in the blockchain. This node has an administrator role to grant other nodes associated access privileges. In our case, the permissions for other nodes are set by the first admin node, and they can be made true for all the nodes while setting chain parameters. The basic chain parameters set in our Multichain are as below:

#Basic chain parameters

1. chain-protocol = multichain # Chain protocol
2. chain-description = MultiChain model # Chain Description
3. root-stream-name = root # Root stream name
4. root-stream-open = true # Allow anyone to publish in root stream
5. chain-is-testnet = false # Content of the "testnet" field of API responses, for compatibility.
6. target-block-time = 15 # Target time between blocks (transaction confirmation delay), seconds (5–86,400)
7. maximum-block-size = 83,88,608 # Maximum block size in bytes.

The basic chain parameters have been set to limit permissions to any newly added nodes. Similarly, the global permissions in the Multichain are as shown below:

#Global permissions

1. anyone-can-connect = false # private blockchain.
2. anyone-can-send = false # transaction signing is not restricted by address.
3. anyone-can-receive = false # transaction outputs are restricted by address.
4. anyone-can-receive-empty = true #without permission grants, asset transfers and zero na$
5. anyone-can-create = false # selected can create new streams.
6. anyone-can-issue = false # selected can issue new native assets.
7. anyone-can-mine = false # selected can mine blocks (confirm transactions).
8. anyone-can-activate = false # selected can grant or revoke connect, send and receive permissions.
9. anyone-can-admin = false # selected can grant or revoke all permissions.
10. support-miner-precheck = true # Require special metadata output with cached scriptPubKey for input, to support advanced mine$

The multichain daemon is created using the following command with the chain name model:

```
multichain-util create model
multichaind model–daemon
```

This creates the MultiChain Core Daemon such that other nodes can connect to this node using the command:

multichaind model@[ip]:[port] (e.g., multichaind model@192.168.204.132:8353)

We then create two other nodes, Node 2 and Node 3 representing Saskatoon Travel and Tour, and Saskatoon Shopping Mall, respectively, as independent firms in the travel domain. The creation of the nodes offered the individual addresses for those new nodes which were acknowledged by the first node to grant a "connection" permission to them into the MultiChain since it is the private blockchain. Back on the first server, we add connection permissions for another node addresses as:

multichain-cli model grant [address] connect, send, ...

This is the first step in creating the blockchain. While granting the connection permission, further other permissions could also be set for other nodes. As shown in **Figure 4**, Node 2 (Saskatoon Travel and tour) has been given "connect," "send", "receive," "issue," "create," "mine," "activate," and "admin" permissions,

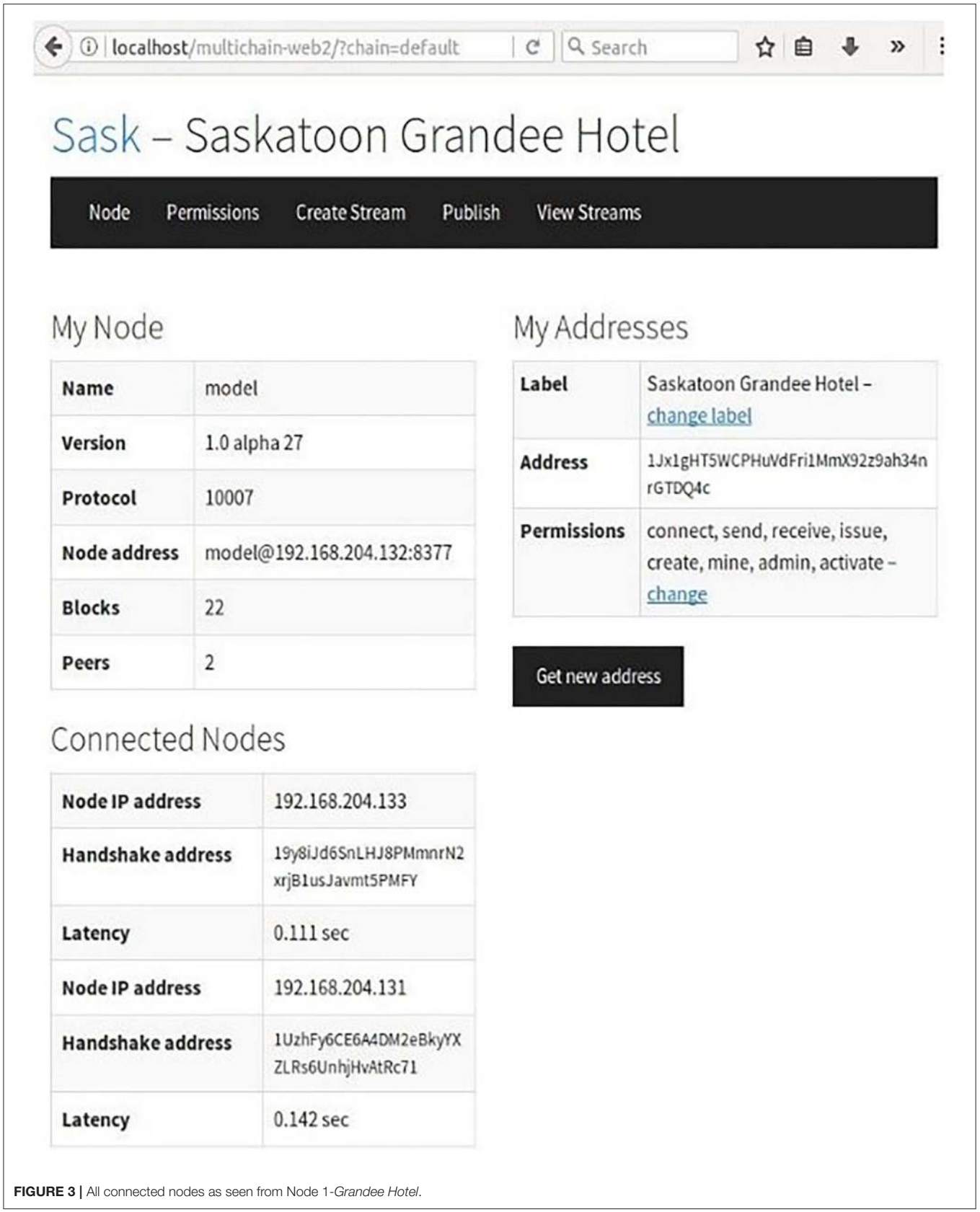

FIGURE 3 | All connected nodes as seen from Node 1-*Grandee Hotel*.

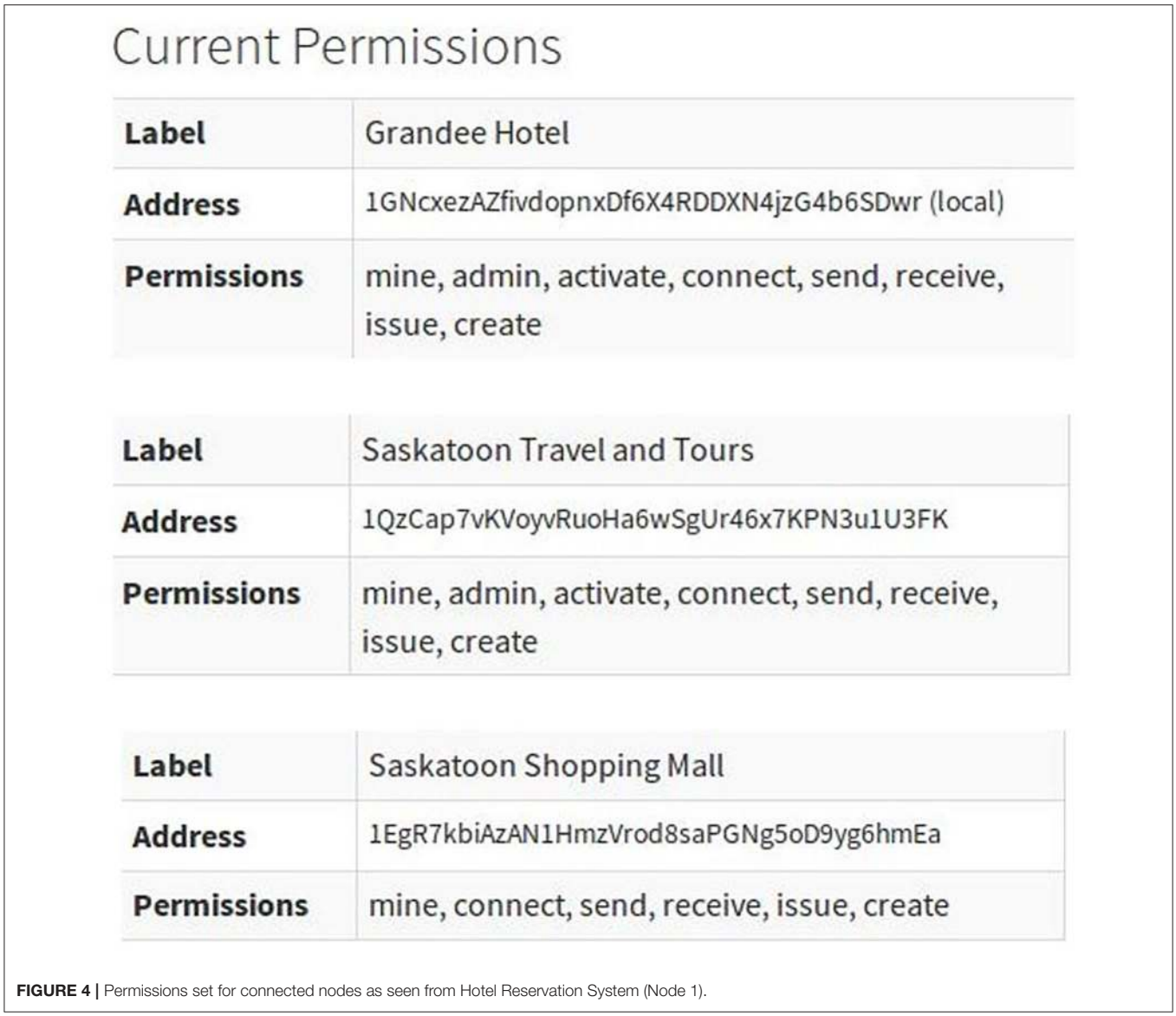

FIGURE 4 | Permissions set for connected nodes as seen from Hotel Reservation System (Node 1).

and Node 3 (Saskatoon shopping Mall) has been given all permissions except 'admin' and 'activate'. This means Node 2 in the blockchain can act as admin but Node 3 cannot. **Figure 5** shows how to publish the stream by uploading the items containing user profile data into the stream and share them with other consortium enterprise nodes.

Figure 6 shows the list of the streams created by the Node 1-Grandee Hotel. The first node was the Hotel Reservation System, which collects the users' data while booking the room in the hotel. The system collects basic user information such as name, the purpose of visit, nationality, duration of stay etc. This information is particularly useful to other tourism enterprises, the shopping mall, and travel and tour operators because they can provide attractive offers to the customers during their stay. The collected useful data from the user profile can be shared among the enterprise consortium as per the predefined agreed terms in the smart contract.

The collected data at Node 1 from the off-chain database is converted into JSON format before being published as items into the stream. There can be any number of streams and the data published in every stream is stored in full or referenced by a hash inside the transactions. Only the nodes with permission can view the contents of the streams.

All other nodes in the network easily can convert and store the received items into their own repositories. Every node in the MultiChain can have access to any stored raw data. To ensure data confidentiality, streams from the Multichain are used with a combination of symmetric and asymmetric cryptography for encrypting the data before being put into the chain. This method utilizes three blockchain streams (more details about this can be found in Greenspan, 2013):

1. Pubkeys stream: To distribute the participants' public keys under the RSA public-key cryptography.

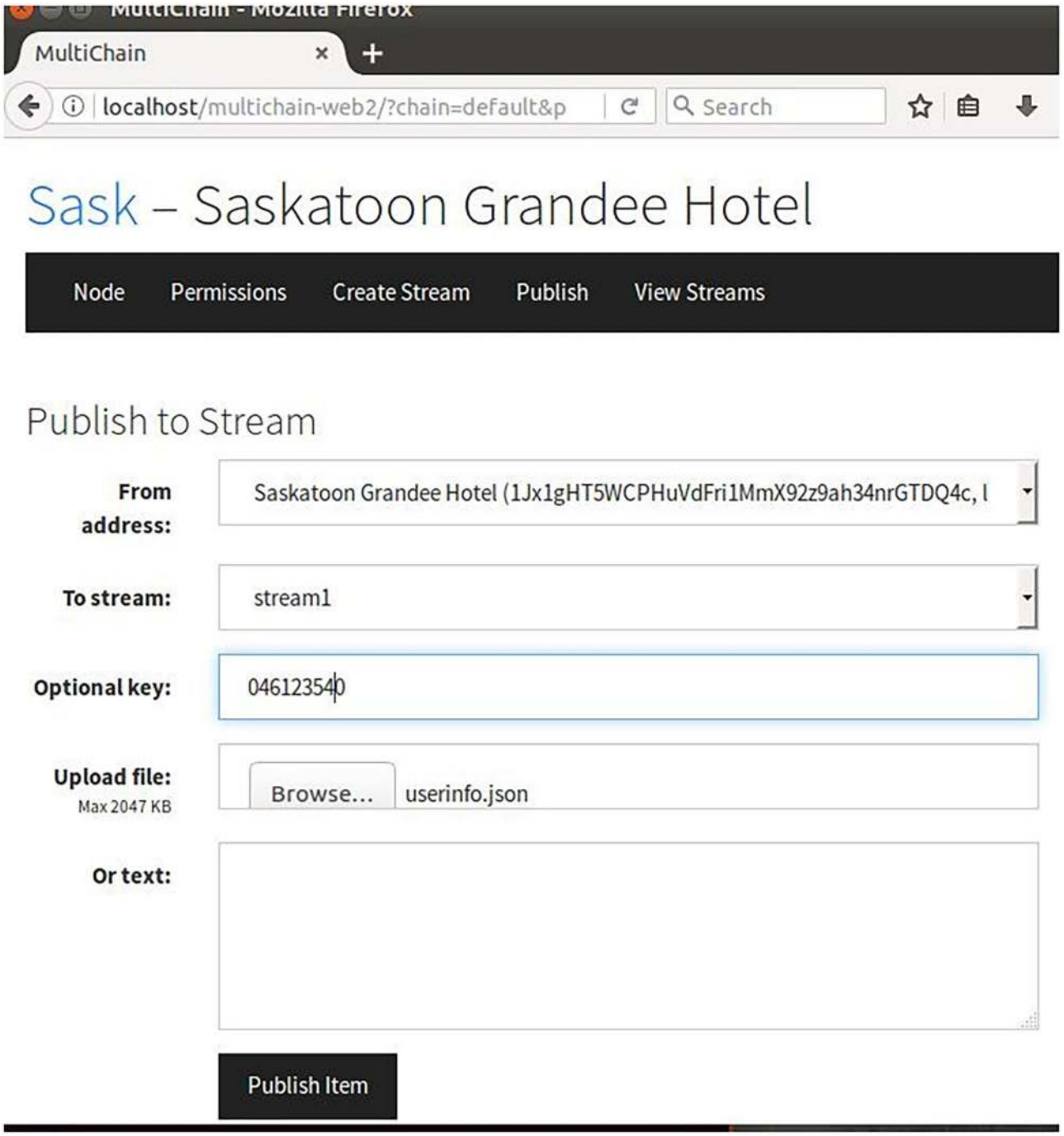

FIGURE 5 | Publishing stream of items.

2. Items stream: To publish the large pieces of data, each of which is encrypted using a symmetric AES cryptography scheme.
3. Access stream: To create stream-entry containing a secret password for data, encrypted with the participant's public key to provide data access. Therefore, only a subset of blockchain participants with the password can read the encrypted data.

All the participants in the system with their Ethereum account addresses are in the MultiChain network. The data provider node puts customer data in the local storage. The MultiChain enables to encrypt it using the data consumer node's public key and slice the large dataset into smaller segments. The encryption process starts with the generation of the RSA private-public key pair on the node (server with running bash shell and xxd installed on)

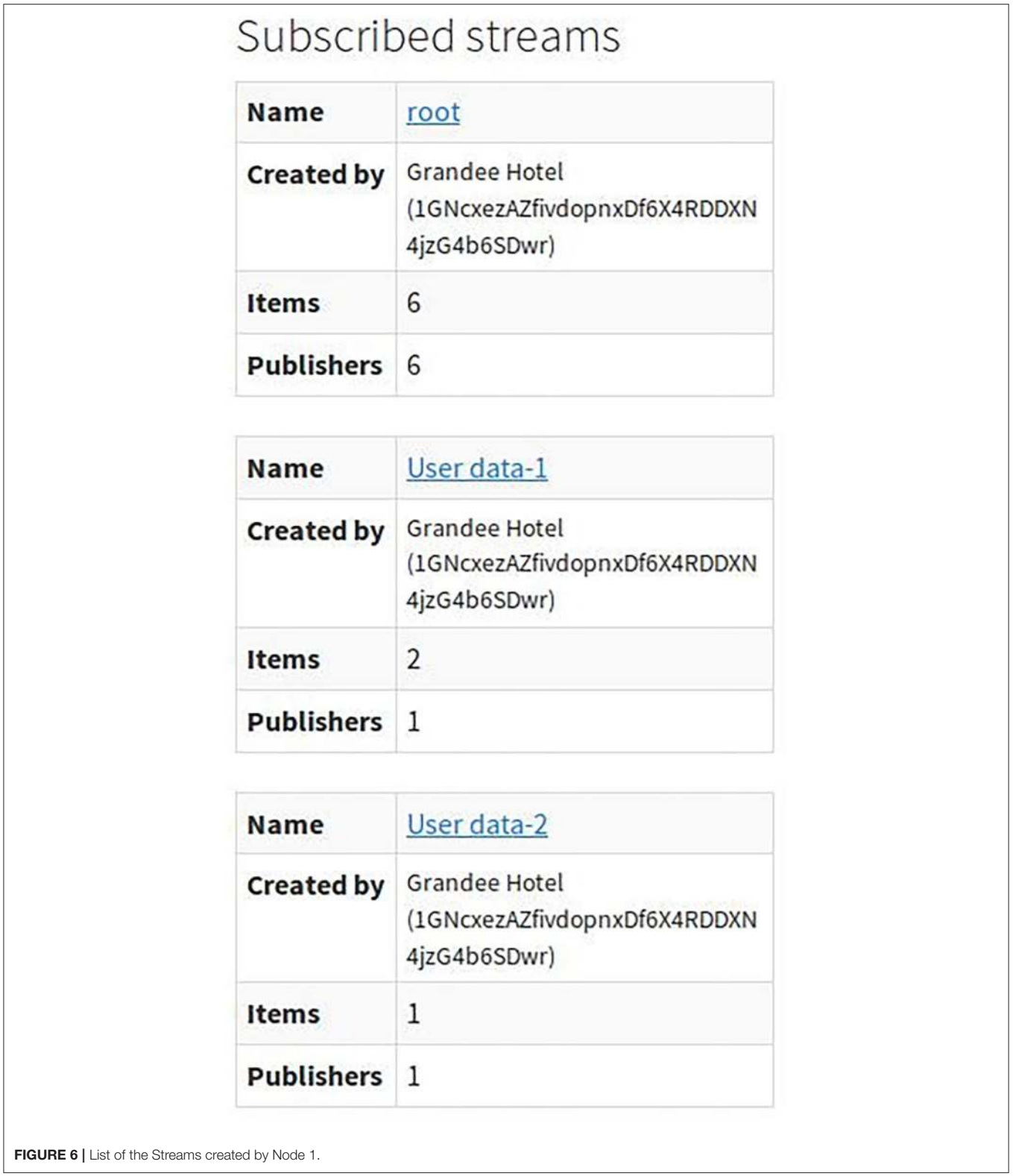

FIGURE 6 | List of the Streams created by Node 1.

that wants to access the customer data, using the openssl and then publishes the public key into the *pubkeys* stream for other data provider nodes to read as:

```
$ openssl genpkey -algorithm RSA -out [address].pem
$ openssl rsa -pubout -in [address].pem | xxd -p -c 9999
```

The data provider node uses the openssl with AES cryptography scheme to encrypt the data and produces the corresponding hexadecimal file as:

```
$ openssl enc -aes-256-cbc -in [datafile] pass:$(openssl rand -base64 48) | xxd -p -c 9999
```

It then publishes the encrypted file into the *items* stream by creating a transaction with their hashes and commits it into the blockchain. In addition to that, it also retrieves the public key of the data consumer node from the *pubkeys* stream to encrypt the secret password for data as:

```
$ [passwordshellvariable] | openssl rsautl -encrypt -inkey [publickeyshellvariable].pem -pubin | xxd -p -c 9999
```

It then publishes the secret password for data into the *access* stream by creating a transaction with their hashes using the data consumer node's address, item's label and commits it into the blockchain. The enterprise data consumer node subscribes to the stream searching for the data and finds the off-chain item with the help of their metadata and hashes. It then places the hash portion in its retrieval queue that queries the data in the P2P network. The node which possesses the data signature responds to the query. At this point, the smart contracts get triggered and with their successful execution, the tokens are transferred from the data consumer's Ethereum address to the customer's account while delivering the requested data (with verified hashes) to the local storage of the consumer node using the same path. The eligible data consumer node uses its private key to retrieve the secret password for the data which was encrypted using its public key with the help of the same openssl and xxd. The application accesses all the MultiChain Community commands using the JSON-RPC API as they are available under open source licenses[11] The smart filters such as stream filters offer callbacks which are also shared with the JSON-RPC API on nodes to examine the validity of the data.

User Incentives for Sharing

We deploy smart contracts on Ethereum that guarantee the user receives the incentive when the user data is consumed by the participating enterprise nodes. All the users (data providers), participating enterprise nodes (data consumers) have Ethereum addresses which interact with the smart contracts. The users can decide which consumers (applications or companies) can access their data. Here, the application sets the terms and conditions that the users agree to allow the enterprise a license to collect and use the contents before using their services and specifies that the user who shared their data retain ultimate ownership to their content, but the enterprise node also receives the limited perpetual license (and right to sublicense) to distribute such contents (see **Appendix A** and **Appendix B** for associated smart contracts). The smart contracts give the users full transparency over who accesses their data, when and for what purpose. So, in the beginning, only the selected enterprises access the user data by subscribing to the published streams in the MultiChain. The incentive is given in the form of a digital token by transferring ethers to users' ether addresses.

We use Ropsten blockchain to implement the contracts on an online Remix IDE[12] because the Ropsten is a test network with the same proof of work (PoW) consensus mechanism for the block's validation as in the mainnet of the Ethereum and the Remix is a free IDE to deploy any untrusted codes before going live. In addition to that, anyone can use Etherscan[13] to explore the Ropsten blockchain for searching for any transactions taking place on the blockchain. Also, we use metamask[14] to deploy the Injected web3 environment to connect the contracts with the ether account addresses. **Figure 7** illustrates a flowchart for the workflow logic with the full process cycle, including the creation of contracts to handling the successful payment for data sharing.

We confine the functions (methods) into our smart contract as per the roles of the participating entities to successively execute different tasks between the data provider and the consumer. The steps to generate the codes are described in *Algorithm B* and *Algorithm C*.

The variables hold the Ethereum addresses, data prices, and contract states. We create a setter function to make its parent or child change the state of the contract if needed, and the compiler automatically generated getter functions for all public variables.

ALGORITHM B | Signed Terms And Conditions

Input: A_{cu}, A_{co}, A_{cc}, deposit, dataPrice, contractState

1. A_{cu}, A_{co}, A_{cc} are the set of all ether addresses of customers, data consumers and contracts, respectively.
2. Grant access to only $a_{cu} \varepsilon A_{cu}$, $a_{co} \varepsilon A_{co}$ who got registered into the system.
3. Change the contract state to Created.
4. a_{cu} deposits e_d.
5. Set data price to ep such that $e_p = ½\ e_d$.
6. Contract balance of a_{cc} is $r_b = e_d$, where $a_{cc} \varepsilon A_{cc}$.
7. Allow a_{co} to choose the customer data of its interest.

ALGORITHM C | Confirmed Data Consume

Input: A_{cu}, A_{co}, A_{cc}, deposit, dataPrice, contractState

1. A_{cu}, A_{co}, A_{cc} are the set of all ETH addresses of customers, consumers and contracts, respectively.
2. a_{co} decides to consume the customer data acu, pays $2e_p$ such that consumer's deposit $= e_p$.
3. Contract balance of acc is $r_b = e_d + 2 * e_p$.
4. Change the contract state to Locked.
5. Grant a_{co} to access the customer data.
6. a_{co} confirms data availability
7. Change the contract state to Inactive.
8. Transfer deposit e_p from a_{cc} back to a_{co}.
9. Remaining Contract balance of a_{cc} becomes $r_b = e_d + e_p$.
10. Transfer r_b to a_{cu}.

[11] https://github.com/MultiChain/multichain/blob/master/COPYING

[12] https://remix.ethereum.org

[13] https://ropsten.etherscan.io/tx/0xa278a0b05cea83b45bc1879246113265e34e72d37f66dfb5bf2ed2660d6d6902

[14] https://metamask.io/

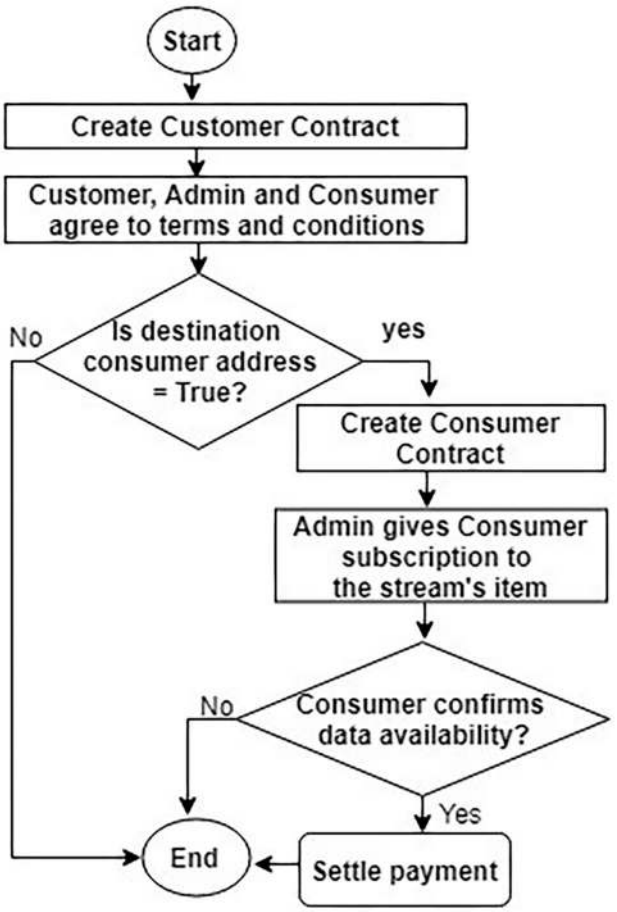

FIGURE 7 | Flowchart for workflow logic of smart contracts.

We added modifiers in the contracts to support inheritance by restricting the functions with some conditions and created events to keep arguments in the transaction logs that notify clients about what is happening with the contracts. This model doesn't include the attestation of the smart contracts which might be required to ensure the reliability of the contracts. However, the attestation authority, as needed, can be added accordingly into the system.

The contracts (see **Appendix C**) following the algorithms B and C were deployed on the Remix. Thus, as seen from the algorithms, after accessing the user data, the corresponding user is incentivized by the data consumer. Our approach delivers a usable blockchain-based model for a collection of user data, providing accountability of access, maintaining the complete and updated information with a verifiable record of the provenance, including all accesses/sharing/usages of the data. The next section will give insights into evaluating the performance of our model.

PERFORMANCE EVALUATION

The system performance evaluation was carried out by analyzing the performance metrics that mostly affect user experience (UX).

Table 1 | Test scenario description.

Scenario	Descriptions
S1	Two enterprise nodes connected
S2	Three enterprise nodes connected
S3	Eight enterprise nodes connected

We did successive experiments on the freshly created node to evaluate the performance of the system by setting four goals to find out:

1. How long it takes an enterprise Multichain node to get connected to the network.
2. How long it takes the enterprise Multichain node to respond to the actions (like starting stream, viewing a stream item, loading or publishing the items into the stream).
3. How much memory the node consumes when blockchain daemon gets started, and
4. How much gas the transactions use (validation cost) to complete the execution.

Since there are two blockchain platforms in our system, we considered measuring latency and memory consumption parameters among the private blockchain nodes because the data sharing is performed in the private network which requires very low latency for optimal performance and the storage delay can also play a role in the increased latency and poor performance. For the smart contracts' execution, we always try to make transaction costs low, so we measured the efficiency of our smart contracts in terms of the transaction cost. To evaluate the implementation prototype, we performed an evaluation plan to simulate real-world interactions. We categorized the first three goals as implementation under the data-sharing model and the last goal as the implementation under the user incentive model.

Experimental Setup 1

To achieve the first three goals, the evaluation involved three scenarios to simulate different levels of concurrency while monitoring latencies in the Windows and the UNIX machines. The three scenarios are shown in **Table 1**. We stopped all extra processes except the basic OS processes to run in the background alongside the Multichain daemon to ensure that no other process would affect our experiments. The experiment was carried out on the newly created Multichain nodes. Since the MultiChain uses the cryptography mechanism, it restricts block index and chainstate access to the list of permitted users; so, we created blockchain nodes as fresh ones. The block index maintains information for every block, and where it is stored on disk. The chain state maintains information about the resulting state of validation as a result of the currently best-known chain. Node parameters were set up as stated before to store the key-value pairs of all the block and state hashes.

The theoretical peak bandwidth of a network connection is fixed as per the technology used. However, the actual number of packets to be sent over the network is affected by higher and lower latencies. Excessive latency prevents data

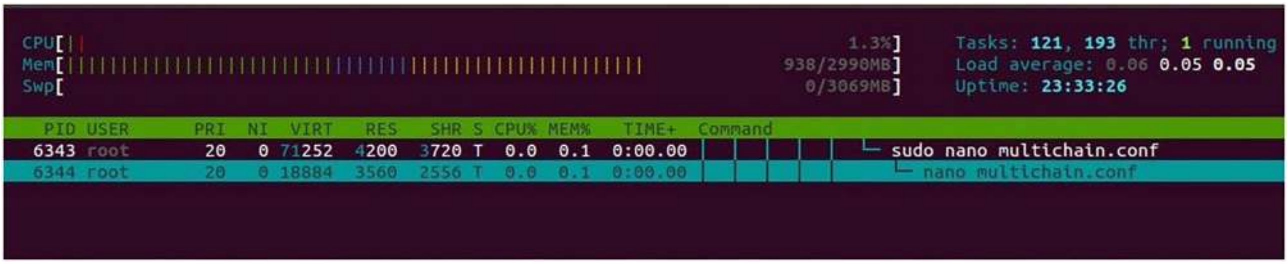

FIGURE 8 | Memory status at normal state.

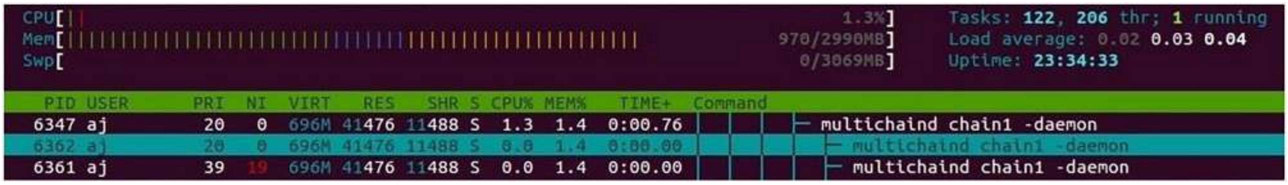

FIGURE 9 | Memory status after multichain daemon started.

from filling the network pipe, thus decreasing throughput and limiting the maximum effective bandwidth of a connection. Therefore, we set our goal of the evaluation to retrieve the latency alongside memory consumption in each case.

To observe the effect of the multichain core daemon being stopped and reconnected into the network, we made the scripts that run with the gap of 1 min for every new observation with the following Multichain commands:

multichain-cli model stop
multichaind model daemon

We observed the latency from the first node Node 1, when it got connected to another single-node N2 for scenario S1, next when it got connected to the other two nodes N2 and N3 for scenario S2 and similarly with other seven nodes N2–N7 for scenario S3 in a total of 20 observations.

For S3, we first recorded the latency from Node 1 to connect it with the other 7 nodes in the network and then finally took their average to find the mean latencies for connecting 7 different nodes from Node1.

In addition to that, we carried out another experiment to observe the memory consumption for the nodes when the corresponding multichain core daemon got started on that particular node. A total of five observations was carried out, one of which is shown in **Figures 8**, **9**. The figures show the total memory usage during the pre- and post-activation of multichain Daemon.

Experimental Setup 2

To achieve the last goal, the evaluation involved deploying the codes with the Remix IDE on the Ethereum Ropsten testnet. To this date, the smart contracts require some gas to store its code and commit the transactions into the Ethereum blockchain, and the actual cost is paid in ether. ETH Gas Station[15] provides three categories of gas prices. They are SafeLow (<30 min), Standard (<5 min), and Fast (<2 min). The gas limit is helpful to optimize the gas used to provide a safety mechanism, as sometimes code with bugs might keep on consuming unnecessary gas for the execution. We used a gas price of 25 Gwei (2.5e-8 ether) which was the then price for achieving a faster transaction. The cost of a transaction always increases when the gas price goes higher. We are providing one of the instances of the contract creation in this paper with the following transaction hashes:

H1: 0xe7b67820af62d3dc5c41357a6de1192de14f8c39cc0416a2830c57c28e3c32b6
H2: 0x8b38cc9b56f584ccec022678f61b16b166e32e581fd0329a394599000af8f226
H3: 0xd333d232646a571be438cda52548503f07047fa07dcce026f18b8df2c88870d0

In this case, a contract was first created at an address "0x2aadf80E4CE7Fc2Db5d57dD975e0337D373e1C50" with the transaction hash H1. After that, 2 ether were transferred into the contract from a customer, which was at an address "0x5378fa11529725ccc491bb6708f9e2f06a1639d5." The Data price was set as 1 ether. So, the customer must receive 3 ether at the end of the transactions. The data consumer at an address "0x923c1eDfAdB6332254C83BCbAE85B2cA6b9Bb36e" with the transaction hash H2 transferred 2 ether to the contract in which 1 ether was the deposit. Finally, the data consumer got the customer data, and then the contract transferred 1 ether to the data consumer and 3 ether to the customer with the transaction

[15]https://ethgasstation.info/

FIGURE 10 | Logs of successful transaction and payment.

Table 2 | Latency (ms) summary for three test scenarios.

Scenarios	N	Min	Max	Avg.	SD
S1	20	85	159.5	122.57	19.32
S2	20	80	156	126.2	24.24
S3	20	106.86	144.7	127.22	11.01

hash of H3. The corresponding logs are given in **Figure 10**. More of the details can be obtained using the corresponding transaction hashes on the Ropsten website[16].

[16]https://ropsten.etherscan.io/

RESULT ANALYSIS

The data on latency for the first part of the observation is shown in **Table 2** and **Figure 11**. All the scenarios have the minimum and maximum latency around 100 and 150 ms, respectively, giving the average latency time <125 ms.

It can be concluded that there is no scalability limit in terms of node count because each node does not need to connect to every other to create a fully connected peer-to-peer network in the private blockchain. Moreover, before conducting the experiments on the real machines, we had initially conducted similar tests on a large number of virtual machines, and we found that our results are similar in both cases. The only difference here with the real physical nodes is that it reflects the realistic scenario

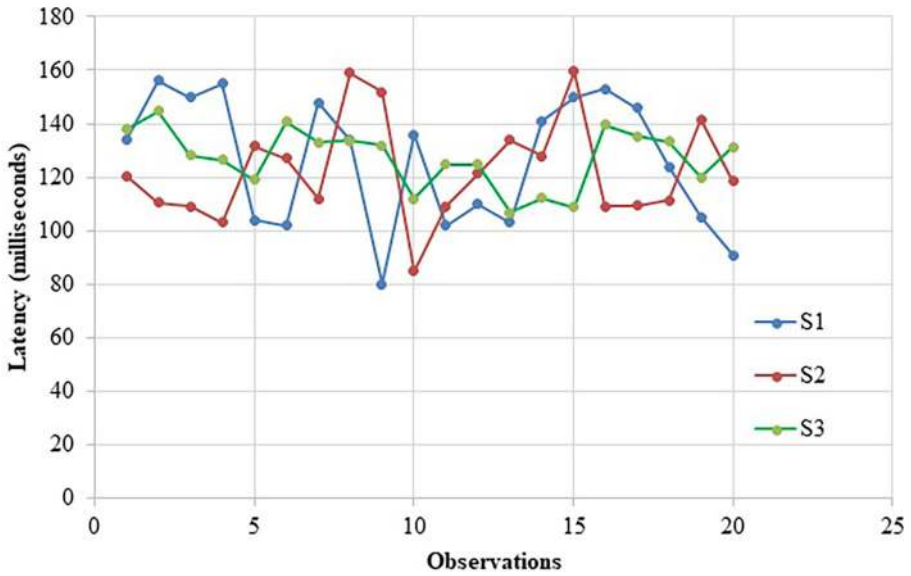

FIGURE 11 | Latency Test results in a chart for three test scenarios.

with no network factor influencing the experiments. However, for all the node catch-up time, new nodes joining the chain must replay all transactions from the beginning, and so it can take them significant time before they are up-to-date. The exact amount of time will depend on how many blocks and transactions are in the chain.

Our experiment had been carried out with only 10 streams having 100 items in total, which was below 100 MB. It is because we were only concerned with the latency. In addition to that, since no smart contracts are committed into the Multichain, unlike in the Ethereum, there is no execution of any automated program for every message on every blockchain node. That surely contributed to the low latency that we observed here.

We also analyzed the memory consumption for the Multichain nodes when their core daemons got initiated. We carried out 20 successive observations this time to see the memory consumption, which changed from the initial memory usage of 938 to 970 MB after the multichain daemon began. So, it can be concluded that the memory usage was around 28 MB and is not so huge to operate the model with the Multichain blockchain. Moreover, it is also based on the number of unspent transactions.

There are also around 300 bytes of memory already kept for each block in the chain. Therefore, if the node is subscribed to millions of streams, then that would increase memory usage. However, our model has focused on storing the user profile data, and even 1 million of those data will have a size of just around 100 MB. So, this model is very effective in terms of a quick start, quick response and fewer memory consumptions.

Furthermore, we analyzed the transaction cost while deploying smart contracts and executing the associated functions. As can be seen from **Figure 12**, for instance, the cost to execute the function for payment settlement (transferring ether from deployed contract to customer and consumer) was 0.001247 ether when the higher gas fee for a faster transaction was chosen. This cost is 0.18 USD and is considered as acceptable as per the standard gas fee for Ethereum[17] All the functions of the contract were tested successfully with the respective role confinement. **Table 3** presents the transaction fees and time used for different functions (including constructors) to change the contract's states. The total cost associated with the contract was 0.022979 ether, which is the sum of the cost for executing constructor and other functions. The cost to execute the function to change the contract state from "Locked" to "Inactive" is higher than that to change the state from "Created" to "Locked." This increase is justified because the former method caused the contract to transfer ether to both data consumer and customer, but the later made the data consumer transfer ether to the smart contract.

Thus, we evaluated the performance of our new platform with implementation under the data-sharing model and implementation under the user incentive model. The result shows that our data sharing model has very low latency for an enterprise Multichain node to get connected to the network and to respond to the actions like starting stream, viewing a stream item, loading or publishing the items into the stream. It also consumes less memory when blockchain daemon gets started. The user incentive model has an acceptable transaction cost for executing smart contracts.

Therefore, this new platform based on blockchains and smart contracts technologies allows building automatic verification of the conditions for access or modification of each data entity. Smart contracts can be deployed to encode allowed purposes of

[17]https://ethgasstation.info/

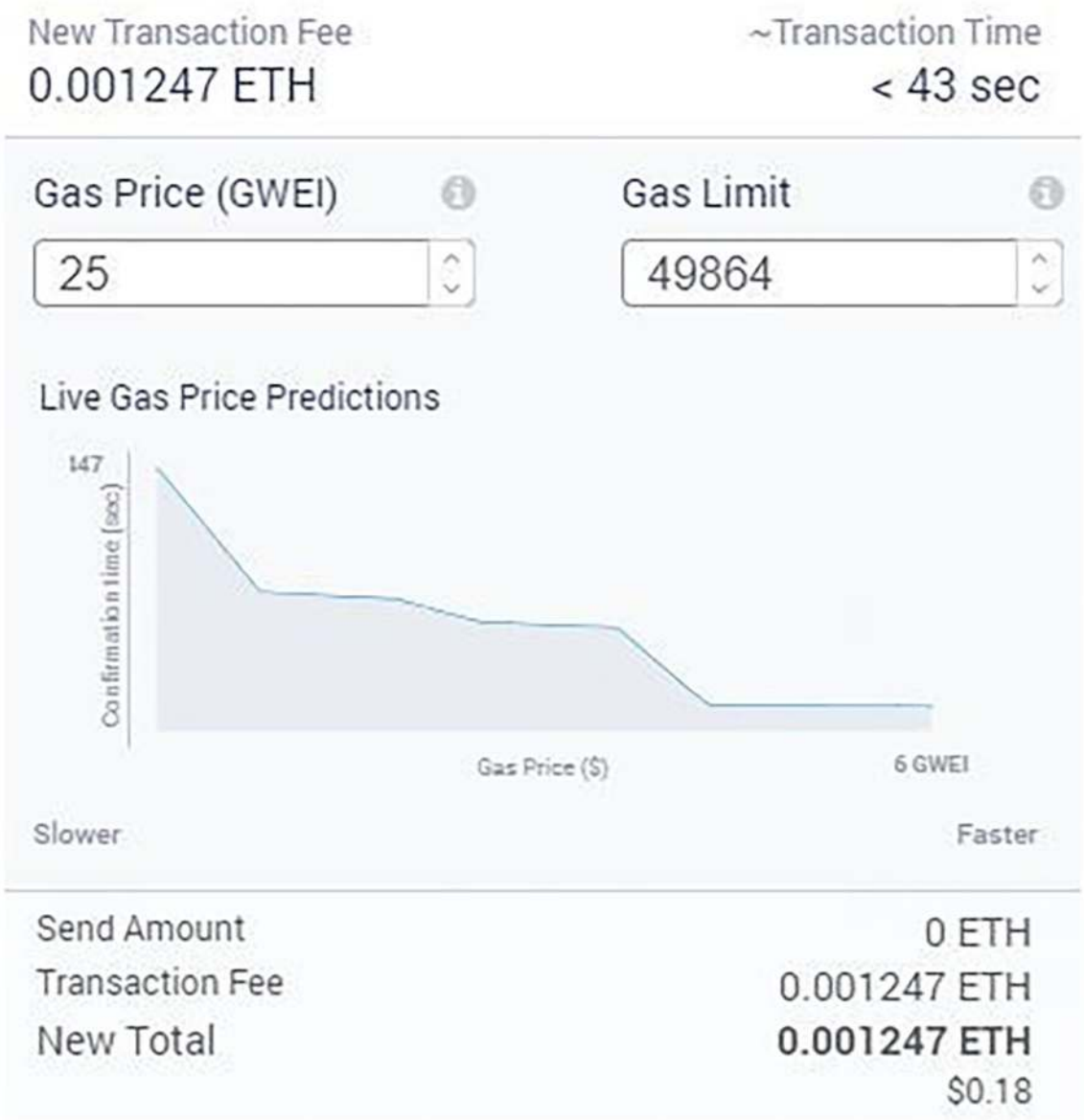

FIGURE 12 | Tx cost for executing a smart contract function.

Table 3 | Transaction cost for a smart contract execution.

Contract state	Methods	Gas price (GWEI)	Gas used	Tx fee (ether)	Tx time (s)
None to created	contract_deploy	25	834,625	0.020866	<46
Created to locked	consumer_buy	25	34,639	0.000866	<43
Locked to inactive	payment_settle	25	47,611	0.001247	<43

data use, allowed software apps, people or businesses who can access the data, time limitations, price for access, etc. Therefore, this new decentralized data-sharing platform is useful for sharing user data of different kinds (user models and user-contributed data), by providing solutions to the privacy, user control and incentive problems. This usable blockchain-based platform can allow users to create a proof of ownership and provenance of their data, share data without losing control and ownership of it, provide/ receive incentives for sharing and give users full transparency and control over who accesses their data, when and for what purpose. However, the most important criticisms to blockchain-based approaches to date relate to their performance

and scalability for the public Ethereum blockchain; yet the rapid development of the technology allows, through thoughtful combinations of blockchains to achieve acceptable performance.

CONCLUSION

In summary, we did an experimental study of our new blockchain-based platform for sharing user profile data that allows users to retain control of the sharing and earn rewards. It is based on user-controlled privacy and data-sharing policies encoded in smart contracts. It also naturally supports building up incentives for users to share their profile data, in terms of rewards (micro-payments or credits). In this way, users become owners of their data and can decide how their data is collected and used, as well as shared. To share user profile data in a distributed fashion, the concept of streams from the MultiChain has been successfully interpreted by taking the case of the travel booking domain. We presented a hotel reservation system as one of the enterprise nodes of Multichain which collects users' profile data and allows users to receive rewards while sharing their profile data with other travel industries according to their privacy preferences expressed in smart contracts. The user data from the repository is converted into an open data format and shared via stream in the blockchain so that other nodes can efficiently process and use the data. The smart contract verifies and executes the agreed terms of use of the data and transfers digital tokens as an incentive to the data owner. The smart contract imposes double deposit collateral to ensure that all participants act honestly. The paper has provided a basic use of smart contracts on privacy-preserving data sharing and management. We have combined blockchains and off-blockchain repository to create a data sharing and management model focused on security and privacy. This blockchain coupled user data sharing model is not just limited to the travel domain but also applicable to other similar domains such as eCommerce, education, health. The paper also evaluates the performance of our new platform, and it met our expectations in terms of the latency, memory consumption and transaction cost for smart contracts deployment. The node responded quickly in all our cases with a befitting transaction cost. Our future work will evaluate the usability and usefulness of the approach, and the trust users can have in the system. We will improve the model by studying users' attitudes to data sharing and the incentives they would find attractive for sharing their data.

DATA AVAILABILITY STATEMENT

The datasets generated for this study are available on request to the corresponding author.

AUTHOR CONTRIBUTIONS

All authors listed have made a substantial, direct and intellectual contribution to the work, and approved it for publication.

FUNDING

AS has received the University of Saskatchewan CGPS Dean's Scholarship and Teacher Scholar Doctoral Fellowship. His research was also supported by the Natural Science and Engineering Research Council (NSERC) of Canada Discovery grant of JV.

ACKNOWLEDGMENTS

The authors acknowledge the University of Saskatchewan for providing the laboratory space to conduct this research.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fbloc.2020.497985/full#supplementary-material

REFERENCES

Abel, F., Herder, E., Houben, G.-J., Henze, N., and Krause, D. (2013). Cross-system user modeling and personalization on the social web. *User Model. User Adapt. Interac.* 23, 169–209. doi: 10.1007/s11257-012-9131-2

Alharby, M., Aldweesh, A., and van Moorsel, A. (2018). "Blockchain-based smart contracts: a systematic mapping study of academic research 2018," in *2018 International Conference on Cloud Computing, Big Data and Blockchain (ICCBB)* (Fuzhou: IEEE), 1–6.

Assad, M., Carmichael, D. J., Kay, J., and Kummerfeld, B. (2007). "PersonisAD: distributed, active, scrutable model framework for context-aware services," in *International Conference on Pervasive Computing* (Berlin; Heidelberg: Springer), 55–72.

Azaria, A., Ekblaw, A., Vieira, T., and Lippman, A. (2016). "MedRec: using blockchain for medical data access and permission management," in *2016 2nd International Conference on Open and Big Data (OBD)* (Vienna: IEEE), 25–30.

Benet, J. (2015). *IPFS-Content Addressed, Versioned, P2P File System (DRAFT 3)*. Available online at: https://ipfs.io/ipfs/QmV9tSDx9UiPeWExXEeH6aoDvmihvx6jD5eLb4jbTaKGps (accessed June 20, 2019).

Bierer, B. E., Crosas, M., and Pierce, H. H. (2017). Data authorship as an incentive to data sharing. *N. Engl. J. Med.* 376, 1684–1687. doi: 10.1056/NEJMsb1616595

Buterin, V. (2015). *A Next Generation Smart conTract & Decentralized Application Platform*. Available online at: http://www.the-blockchain.com/docs/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf (accessed June 20, 2019).

Carmagnola, F., Cena, F., and Gena, C. (2011). User model interoperability: a survey. *User Model. User Adapt. Interac.* 21, 285–331. doi: 10.1007/s11257-011-9097-5

Cassidy, C. M., and Chae, B. (2006). Consumer information use and misuse in electronic business: an alternative to privacy regulation. *Inform. Syst. Manag.* 22, 75–87. doi: 10.1201/1078.10580530/46108.23.3.20060601/93709.8

Castells, M. (2007). Communication, power and counter-power in the network society. *Int. J. Commun.* 1, 238–266. Available online at: https://ijoc.org/index.php/ijoc/article/view/46/35 (accessed March 1, 2019).

Chen, C.-S., Chang, S.-F., and Liu, C.-H. (2012). Understanding knowledge-sharing motivation, incentive mechanisms, and satisfaction in virtual communities. *Soc. Behav. Personal. Int. J.* 40, 639–647. doi: 10.2224/sbp.2012.40.4.639

Davoust, A. (2015). *Decentralized Social Data Sharing*. Carleton University. Available online at: https://curve.carleton.ca/295070d6-5ad8-4c55-953e-d10109745f19 (accessed March 2, 2019).

Dim, E., and Kuflik, T. (2012). "User models sharing and reusability: a component-based approach," in *AUM 2012: Augmented User Modeling* (Montreal, QC). Available online at: http://ceur-ws.org/Vol-872/aum2012_paper_2.pdf (accessed March 11, 2019).

Dolog, P., and Vassileva, J. (2005). "Decentralized, agent based, and social approaches to user modeling (DASUM)," in *Workshop DASUM-05, at the 9th International Conference on User Modeling (UM'05)* (Edinburgh). Available online at: http://people.cs.aau.dk/~dolog/pub/DASUM-proceedings.pdf

Fanning, S., Fanning, J., and Kessler, E. (1999). *Real-Time Search Engine*. Available online at: https://patents.google.com/patent/US6366907B1/en (accessed June 27, 2019).

Ferrer-Sapena, A., and Sánchez-Pérez, E.-A. (2019). Applications of blockchain technology in scientific documentation: current situation and perspectives. *Profes. Inf.* 28:10. doi: 10.3145/epi.2019.mar.10

Fospha (2019). *The State of Marketing Measurement, Attribution & Data Management*. Available online at: https://www.clickz.com/resources/the-state-of-marketing-measurement-attribution-data-management/ (accessed August 28, 2019).

Friedlmaier, M., Tumasjan, A., and Welpe, I. M. (2016). "Disrupting industries with blockchain: the industry, venture capital funding, and regional distribution of blockchain ventures," in *Proceedings of the 51st Annual Hawaii International Conference on System Sciences (HICSS)* (Hawaii).

Graham, V., and Sacha, W.-V. (2007). *Participative Web and User-Created Content: Web 2.0 Wikis and Social Networking*. Paris: Organization for Economic Cooperation and Development (OECD). Available online at: https://dl.acm.org/citation.cfm?id=1554640

Greenspan, G. (2013). *MultiChain Private Blockchain - White Paper*, 1–17. Available online at: http://www.multichain.com/download/MultiChain-White-Paper.pdf (accessed June 20, 2019).

Iyilade, J., and Vassileva, J. (2013). "A decentralized architecture for sharing and reusing life-logs," in *LLUM 2013: LifeLong User Modelling*, Vol. 997 (Rome), 4–10. Available online at: https://pdfs.semanticscholar.org/7398/8175fc23e72af94cea85a3cfdaa3d6c5da55.pdf (accessed July 15, 2019).

Liu, H., Li, X., Xu, M., Mo, R., and Ma, J. (2017). A fair data access control towards rational users in cloud storage. *Inform. Sci.* 418–419, 258–271. doi: 10.1016/j.ins.2017.07.023

Lo, B., and DeMets, D. L. (2016). Incentives for clinical trialists to share data. *N. Engl. J. Med.* 375, 1112–1115. doi: 10.1056/NEJMp1608351

McMahan, B., and Ramage, D. (2017). Federated learning: collaborative machine learning without centralized training data. *Google AI Blog*. Retrieved from: https://ai.googleblog.com/2017/04/federated-learning-collaborative.html (accessed June 29, 2019).

Meadows, A. (2014). *To Share or Not to Share? That is the (Research Data) Question...* Available online at: https://scholarlykitchen.sspnet.org/2014/11/11/to-share-or-not-to-share-that-is-the-research-data-question/ (accessed June 28, 2019).

Molina-Jimenez, C., Solaiman, E., Sfyrakis, I., Ng, I., and Crowcroft, J. (2019). *On and Off-Blockchain Enforcement of Smart Contracts*. Cham: Springer, 342–354.

Nakamoto, S. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. Available online at: https://bitcoin.org/bitcoin.pdf (accessed March 20, 2019).

Ning, Z., Liao, J., Zhang, F., and Shi, W. (2018). "Preliminary study of trusted execution environments on heterogeneous edge platforms," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)* (Seattle, WA: IEEE), 421–426.

Niu, X., Mccalla, G., and Vassileva, J. (2004). Purpose-based expert finding in a portfolio management system. *Comput. Intell.* 20, 548–561. doi: 10.1111/j.0824-7935.2004.00253.x

Poslad, S. (2009). *Ubiquitous Computing*. Chichester: John Wiley & Sons, Ltd.

Protocol Labs, (2017). *Filecoin: A Decentralized Storage Network*. Available online at: https://filecoin.io/filecoin.pdf (accessed June 14, 2019).

Schilling, T., Mohanty, S., and Mohanty, S. (2012). *Secure Cloud Storage and Synchronization Systems and Methods*. Available online at: https://patents.google.com/patent/US8892866B2/en (accessed June 10, 2019).

Schmidt, D. C. (2018). *Google Data Collection*. Nashville, TN. Available online at: https://bloximages.newyork1.vip.townnews.com/wsmv.com/content/tncms/assets/v3/editorial/f/1b/f1bc6c94-a539-11e8-905a-136f4f930796/5b7bff66f1d7a.pdf.pdf (accessed March 12, 2019).

Shrestha, A. K. (2014). "Security of SIP-based infrastructure against malicious message attacks," in *The 8th International Conference on Software, Knowledge, Information Management and Applications* (SKIMA 2014) (Dhaka: IEEE), 1–8.

Shrestha, A. K., Deters, R., and Vassileva, J. (2017). "User-controlled privacy-preserving user profile data sharing based on blockchain," in *Future Technologies Conference (FTC)* (Vancouver, BC: The Science and Information (SAI) Organization), 31–40. Available online at: https://saiconference.com/Downloads/FTC2017/Proceedings/3_Paper_127-User-Controlled_Privacy-Preserving_User_Profile_Data_Sharing.pdf

Shrestha, A. K., and Vassileva, J. (2016). "Towards decentralized data storage in general cloud platform for meta-products,". in *Proceedings of the International Conference on Big Data and Advanced Wireless Technologies – BDAW* (Blagoevgrad), 1–7.

Shrestha, A. K., and Vassileva, J. (2018). "Blockchain-based research data sharing framework for incentivizing the data owners," in *Blockchain – ICBC 2018*, S. eds Chen, H. Wang, and L.-J. Zhang (Seattle, WA: Springer International Publishing), 259–266.

Shrestha, A. K., and Vassileva, J. (2019). "User acceptance of usable blockchain-based research data sharing system: an extended TAM-based study," in *Proceedings - 1st IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications, TPS-ISA 2019* (Los Angeles, CA: IEEE), 203–208.

Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H. (2003). Chord: a scalable peer-to-peer lookup service for internet applications. *Netw. IEEE ACM Trans.* 11, 17–32. doi: 10.1109/TNET.2002.808407

Szabo, N. (1997). Formalizing and securing relationships on public networks. *First Monday* 2. doi: 10.5210/fm.v2i9.548

Tenopir, C., Palmer, C. L., Metzer, L., van der Hoeven, J., and Malone, J. (2011). "Sharing data: practices, barriers, and incentives," in *Proceedings of the American Society for Information Science and Technology*, 48, 1–4. doi: 10.1002/meet.2011.14504801026

Thilakarathna, K., Petander, H., Mestre, J., and Seneviratne, A. (2014). MobiTribe: cost efficient distributed user generated content sharing on smartphones. *IEEE Trans. Mobile Comput.* 13, 2058–2070. doi: 10.1109/TMC.2013.89

Trainor, K. J., Andzulis, J., Rapp, A., and Agnihotri, R. (2014). Social media technology usage and customer relationship performance: a capabilities-based examination of social CRM. *J. Bus. Res.* 67, 1201–1208. doi: 10.1016/j.jbusres.2013.05.002

Vassileva, J., McCalla, G., and Greer, J. (2003). Multi-agent multi-user modeling in I-Help. *User Model. User Adapt. Interac.* 13, 179–210. doi: 10.1023/A:1024072706526

Vorick, D., and Champine, L. (2014). *Sia: Simple Decentralized Storage*. Available online at: https://sia.tech/sia.pdf (accessed March 16, 2019).

Vrandečić, D., and Krötzsch, M. (2014). Wikidata: a free collaborative knowledgebase. *Commun. ACM* 57, 78–85. doi: 10.1145/2629489

Wilkinson, S. (2014). *Storj A Peer-to-Peer Cloud Storage Network*. Available online at: https://storj.io/storj2014.pdf (accessed March 16, 2019).

Wood, G. (2016). *Blockchain What and Why*. Available online at: https://www.slideshare.net/gavofyork/blockchain-what-and-why (accessed July 19, 2017).

Yang, J., Onik, M., Lee, N.-Y., Ahmed, M., and Kim, C.-S. (2019). Proof-of-familiarity: a privacy-preserved blockchain scheme for collaborative medical decision-making. *Appl. Sci.* 9:1370. doi: 10.3390/app9071370

Zyskind, G., Nathan, O., and Pentland, A. (2015a). Enigma: decentralized computation platform with guaranteed privacy. *ArXiv:1506.03471*, 1–14.

Zyskind, G., Nathan, O., and Pentland, A. S. (2015b). "Decentralizing privacy: using blockchain to protect personal data," in *Proceedings - 2015 IEEE Security and Privacy Workshops, SPW 2015* (San Jose, CA), 180–184.

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.