Use of blockchain as a software component to send messages anonymously for a data trading platform

By

Shrenik Shah

Abstract:

The problem of maintaining complete control over and security while content/ data sharing digitally identity is growing more urgent as our lives become more dependent on online and digital services. What once was rightfully ours and under our control can now be easily stolen and sent to uncountable entities across many locations. Digital Security is a massive problem in the world, especially with recent leaks of Yahoo. According to reports released from Distil Networks, every year data breaches cost the industry around $18.5 billion. Be it banking, healthcare, national security, citizenship documentation or online retailing, identity authentication and authorization is a process intricately woven into commerce and culture worldwide. Identity thefts, hacked databases and breached accounts are shining light on the growing problems of a technologically advanced society, without outpaced identity-based security innovations. Current methods use problematic password-based systems of shared secrets exchanged and stored on insecure systems.

Blockchain based authentication systems are based on irrefutable identity verification using digital signatures based on public key cryptography. In this paper an attempt is made built a platform that can address the need of sending messages securely and anonymously over blockchain.

Introduction :

Companies face new challenges in data management and security, it's blockchain technology emerging as a way to let companies make and verify transactions on a network instantaneously without a central authority. Known by many as the technology underpinning the bitcoin digital currency, blockchain has acquired a new identity in the enterprises. Today, many top financial institutions and a growing number of firms across industries are experimenting with distributed ledger technology as a secure and transparent way to digitally track the ownership of assets, a move that could speed up transactions and cut costs while lowering the risk of fraud.

A blockchain is a data structure that makes it possible to create a digital ledger of transactions and share it among a distributed network of computers. It uses cryptography to allow each participant on the network to manipulate the ledger in a secure way without the need for a central authority. Once a block of data is recorded on the blockchain ledger, it's extremely difficult to change or remove. When someone wants to add to it, participants in the network all of which have copies of the existing blockchain run algorithms to evaluate and verify the proposed transaction. If a majority of nodes agree that the transaction looks valid that is, identifying information matches the blockchain's history then the new transaction will be approved and a new block added to the chain.

The term blockchain today usually describes a version of this distributed ledger structure and distributed consensus process. There are different blockchain configurations that use different consensus mechanisms, depending on the type and size of the network and the use case of a particular company. The bitcoin blockchain, for example, is public and "permissionless", meaning anyone can participate and contribute to the ledger. Many firms also are exploring private or "permissioned" blockchains whose network is made up only of known participants. Following example explains blockchain. Assume an organization has 10 transactions per second. Each of those transactions receives its own digital signature. Using a tree structure, those signatures are combined and given a single digital fingerprint a unique representation of those transactions at a specific time. That fingerprint is sent up the tree to the next layer of infrastructure, such as a service provider or telecom company. This process happens for every organization in the network until there is a single digital fingerprint that encompasses all the transactions as they existed during that particular second. Once validated, that fingerprint is stored in a blockchain that all the participants can see. A copy of that ledger is also sent back to each organization to store locally. Those signatures can be continuously verified against what is in the blockchain, giving companies a way to monitor the state and integrity of a particular asset or transaction.

Anytime a change to data or an asset is proposed, a new, unique digital fingerprint is created. That fingerprint is sent to each client node for validation. If the fingerprints don't match, or if the change to the data doesn't fit with the network's agreed-upon rules, the transaction may not be validated. This setup means the entire network, rather than a central authority, is responsible for ensuring the validity of each transaction.

The consensus mechanism is a set of rules the network uses to verify each transaction and agree on the current state of the blockchain. For the bitcoin blockchain, the consensus mechanism is called proof of work, in which participants on the network run algorithms to confirm the digital signatures attached to blocks verify each transaction. In private or "permissioned" blockchain networks, the consensus mechanism may be less stringent since each participant is known. In those cases, "you don't need the blockchain to establish trust, it already exists," At this time there is no universally agreed-upon consensus mechanism. A transaction manipulates ledger data based on rules described by business logic. After a transaction is executed on a node, the result is a proposed modification of the ledger's data. Before committing the answers to a node's ledger, the answer is validated locally with other nodes in the network. Approved transactions are packaged into a block and re-distributed to all the nodes in the network, which re-validate to ensure their records match.

Other than transactions, blocks also contain the state of the whole system after applying those transactions, In Bitcoin, the state is the collection of coins of all the accounts that have not been spent yet. In Ethereum, the state of the system is the changes of the whole contract storage. In Ethereum, every contract has its own storage where only the contract can write to. The contract storage can be viewed as a flexible key-value data store. The data stored in contract storage can be updated through sending transactions to the corresponding contract with new value. The contract has an address, which is used to query the contract storage.

In the system that we propose in this paper, blockchain in private and permissioned and thus both participants and miners must take necessary permissions for transacting on the proposed system.

Proposed solution:

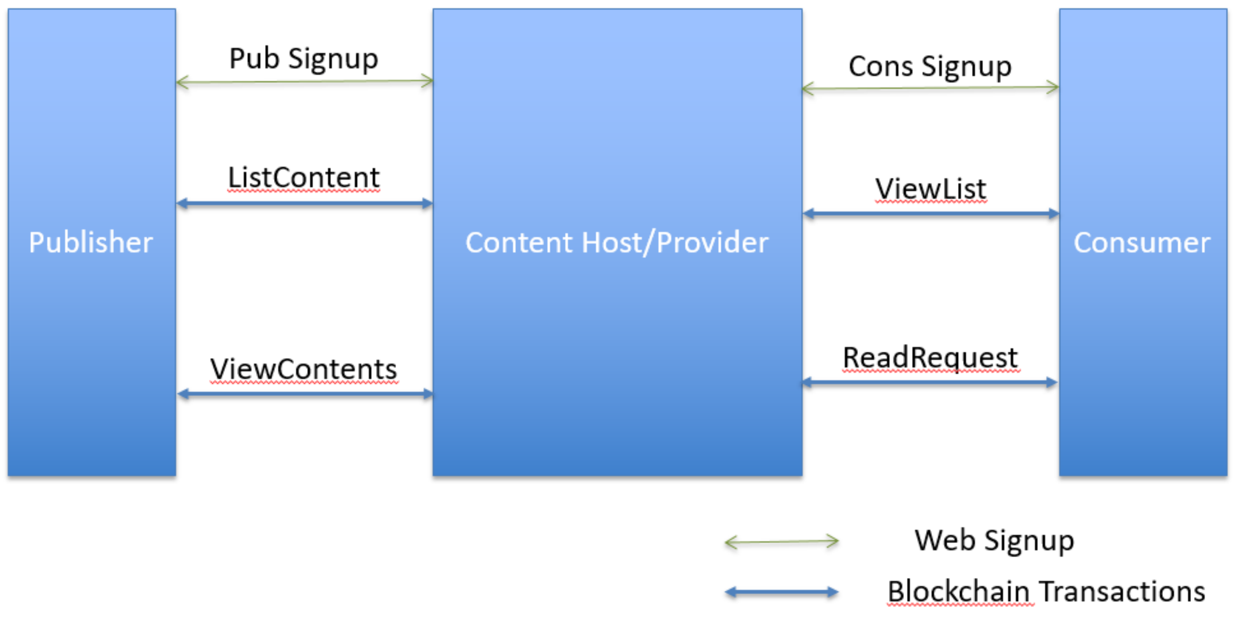


Fig.Simplistic representation of system proposed

Since our system requires a private permissioned blockchain with features for both privacy and confidentiality; one of fundamental requirement is that the identity and patterns of behavior of any party on a network must be impossible for unauthorized parties to ascertain by inspecting the ledger. We also anticipate a requirement to allow blockchain users to make certain business logics and/or other parameters of a transaction confidential, rendering them inaccessible to anyone other than the stakeholders for the contract or the asset being transferred.

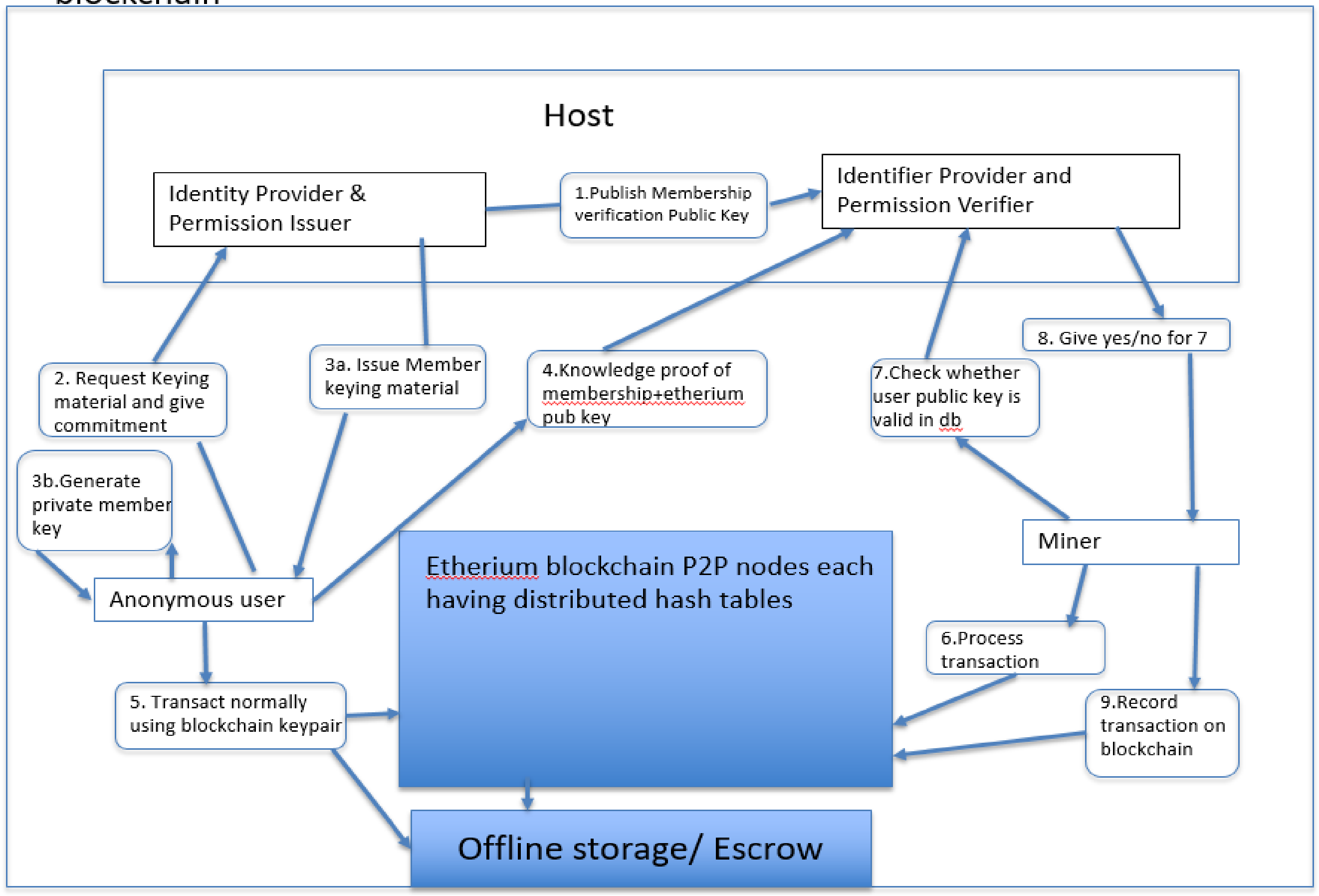
Description of owner Entities in the system

Identity Provider and Permissions Issuer (IdP-PI):
Permissions Issuer function is merged with the Identity Provider function to reflect the need, among others, for addressability of the anonymous User who holds the self-asserted transaction public-key. That is, the Identity Provider function will need the ability to communicate out-of-band with the anonymous User outside the blockchain system in order to engage with the User.

Permissions Verifier (IdP-PV): The Permissions Verifier is the entity that performs the anonymous identity verification of a given a User by running the zero knowledge proof protocol with that User. The Permissions Verifier maintains the Verified Identities Database containing all the transaction public-keys belonging to the anonymous Users who have successfully undergone the anonymous identity verification. Together both these entities form host.

In our blockchain only hashes of msg, msg description, permission and signatures are stored. Actual message is stored in offline storage.

Architecture of proposed system using private permissioned etherium blockchain


Host
Identity Provider & Permission Issuer
1.Publish Membership verification Public Key
Identifier Provider and Permission Verifier
2. Request Keying material and give commitment
3a. Issue Member keying material
4.Knowledge proof of membership+etherium pub key
7.Check whether user public key is valid in db
8. Give yes/no for 7
3b.Generate private member key
Anonymous user
Etherium blockchain P2P nodes each having distributed hash tables
Miner
6.Process transaction
5. Transact normally using blockchain keypair
9.Record transaction on blockchain
Offline storage/ Escrow

Data Flow functionality process

Host(IDP-PI) Establishes Permissioned Group: Depending on the implementation required, model of the IdP-PI and IdP-PV, the IdP-PI can establish permissioned-group as fee-paying service to customers (e.g. Enterprises). As part of the creation of a permissioned-group, the IdPPI generates a number parameters that are unique to the permissioned-group and are used to create two important keys related to the function of the IdP-PI as the Permissions Issuer viz. Membership Verification Public Key, Membership Issuing Private Key

Step 1: IdP-PI Shares Verification Public Key with IdP-PV: In this step, the IdP-PI makes known the Membership Verification Public Key (KMV PK) to the Permissions Verifier (IdPPV). A secure channel with shared verification is utilized between the IdP-PI and IdP-PV elements.

Step 2: User requests Authentication and Membership : A User gets authorization to execute on the permissioned blockchain by asking for enrollment to the permissionedgroup that actualizes the permissioned blockchain. The User should first confirm itself to the IdP-PI and get approval to join the permissioned-chain. The technique used to verify is user to the blockchain can be left on how host system admin wants implementation to verify external users to blockchain. Now in the Host system of permissioned blockchain User is definitely not mysterious to the IdP-PI, and the IdP-PI knows the client user. To join the permissioned-aggregate the User sends the request to the IdP-PI that deals with the permissioned-gathering of system.
The User must perform out various steps to be a user of the system as follows. User acquires the Membership Verification Public Key(KMV PK) from the IdP-IP utilizing a safe channel, with shared confirmation (e.g.

utilizing their individual open keys KP I and Kuser). User then approves and confirms the Membership Verification Public Key. After that User creates his own dedicated keying parameters. User client sends keying parameters to the IdP-Pi who should confirm that these parameters are framed accurately.

Step 3A IdP-PI conveys Group-Member Keying Parameters : In this progression, the IdP-PI creates various group member keying parameters that are particular to the requesting user, in light of the dedication parameters that the User had submitted in the past step.

Step 3B: User Generates User-Member Private Key : After accepting the client particular gathering part keying parameters, the User utilizes these parameters to create his/her possess User-Member Private Key, meant as KUMPK. It is advantageous to note at his point that there is a Many to-1 unbalanced relationship between numerous User-Member Private Keys and the single Membership Verification Public Key (KMV PK) that was conveyed from the IdP-PI to the IdPPV substance in Step 1. That one check open key KMV PK permits the IdP-PV check all permissioned-chain individuals (i.e. Clients) who employ their own particular individual User-Member Private Key KUMPK. All the more particularly, if two Users U1 and U2 freely presents a message with a sign made utilizing keys KUMPKu1 also, KUMPKu2. individually, at that point the Permissions Verifier IdP-PV can confirm both
signature utilizing the one confirmation open key KMV PK yet it won't have the capacity to recognize Users U1 and U2 for privacy enabling purpose

Step 4:User Client Anonymously Proves Membership to IdP-PV : The anonymous membership check convention comprises of various sub-steps taking after the test reaction display. The User sends a request to the Permissions Verifier (IdP-PV), and then the IdP-PV challenges the User with a few parameters(challenge m and random nonce n) that the User must react to. For this, the user must compute sign of knowledge as Ω. The following are inputs to Ω. The User's Membership Verification Public Key ($K_{MV\ PK}$) which the User obtained from the Permissions Issuer IdP-PI in Step 2. The User's own User-Member Private Key K_{UMPK} which the User computed in Step 3. The challenge m and the nonce n_{pv} obtained from the Permissions Verifier IdP-PV. The user must sign Ω using User's transaction private key (Kt-1)to form SIGΩ. The User sends the following three values to the IdP-PV:(Ω; SIG_;Kt-1). The IdP-PV approves sign of-knowledge, and gives back an affirmation of a successful check procedure to the User together with a few parameters to set up a couple shared key (PSK) between the Client and the IdP-PV. The IdP-PV then includes the User's exchange public key Kt to the Verified Identities Database. The User and the IdP-PV engage in a key agreement subprotocol that results in a pair-wise shared key (PSK) denoted as $k_{U;PV}$. This PSK is shared
between the User (who is anonymous throughout Step 4) and the IdP-PV.

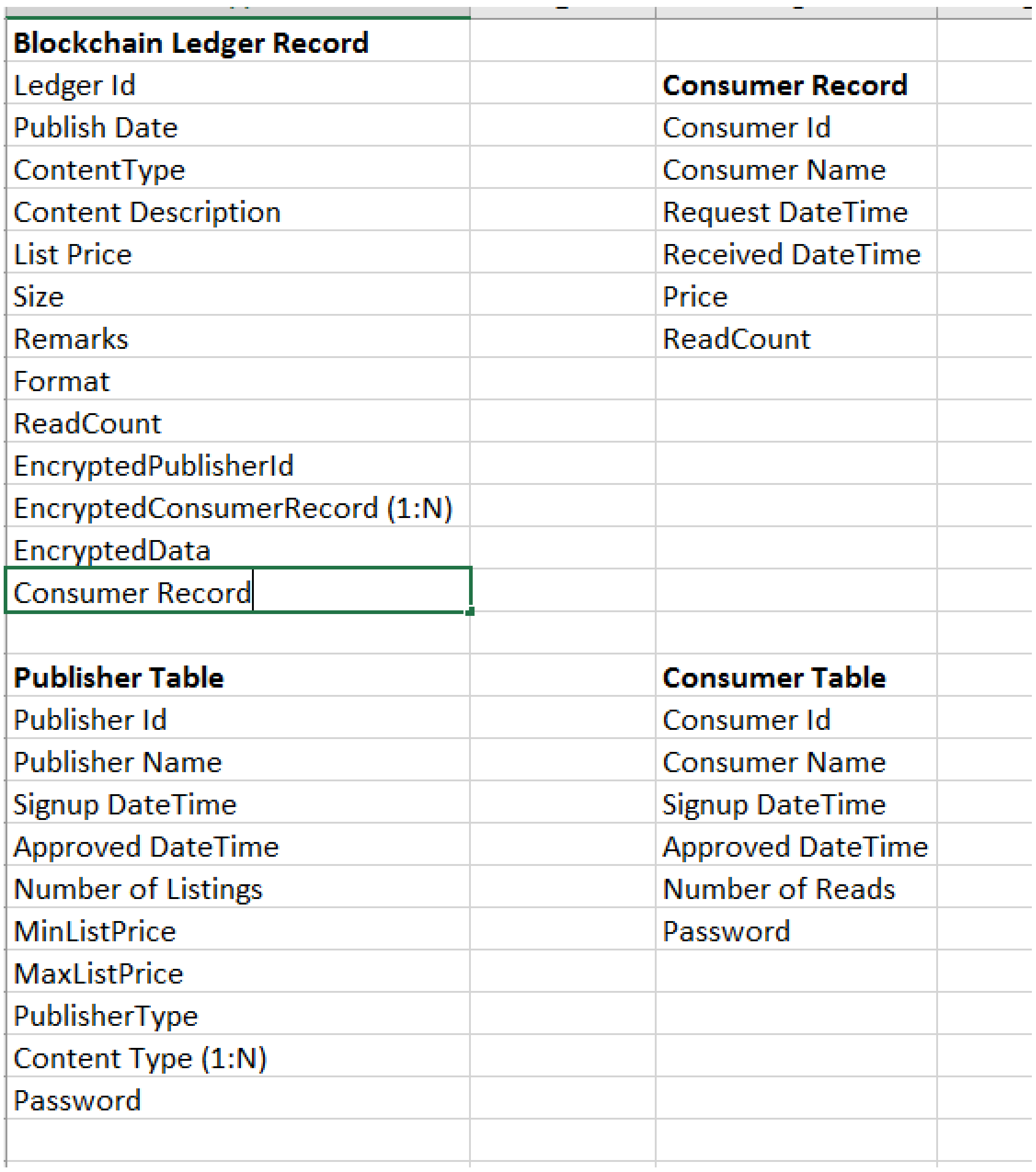
Step 5:User Transacts on Permissioned Blockchain: In this phase the User transacts on the permissioned
blockchain in the usual manner (as in Bitcoin), using the transaction private-key K_{t-1} to sign transactions.

Step 6: Miner Processes Transaction:Only permissioned miners from host can participate in the mining process. Taking after the typical Bitcoin exchange handling in the shared system, a Miner brings an transaction(i.e. from the pool of unprocessed transaction) and gets ready to handle that transaction.

Step 7 :Miner Validates User's Public Key : Before handling a transaction for incorporation into a block, Miner whether the public key found in the transaction has the approval to participate in the permissioned blockchain.
That is, the Miner should first look-into the Verified Identities Databases at the IdP-PV to guarantee the public key is in the database.

Step 8: Miner Records Transaction: If the public key Kt used in the User's transaction exists in the Verified Identities Database, Miner proceeds with processing the transaction (i.e. add to block, perform proof of work, etc). Otherwise the Miner ignores the transaction.

Following snapshot shows which attributes can be part of blockchain ledger table and which other attributes should be outside blockchain network in the offline storage as blockchain can have only limited data

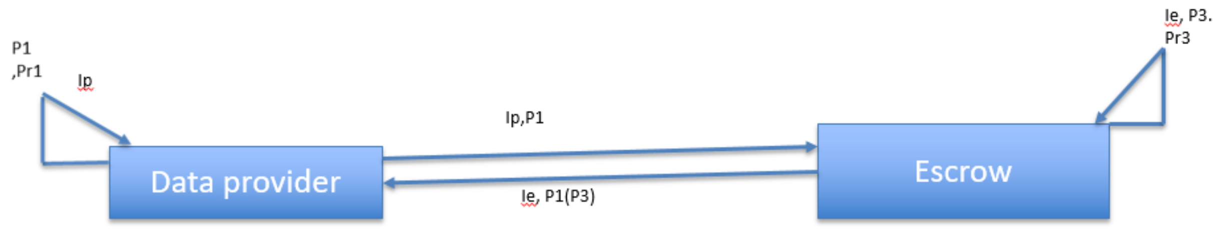


Blockchain Ledger Record		
Ledger Id		**Consumer Record**
Publish Date		Consumer Id
ContentType		Consumer Name
Content Description		Request DateTime
List Price		Received DateTime
Size		Price
Remarks		ReadCount
Format		
ReadCount		
EncryptedPublisherId		
EncryptedConsumerRecord (1:N)		
EncryptedData		
Consumer Record		
Publisher Table		**Consumer Table**
Publisher Id		Consumer Id
Publisher Name		Consumer Name
Signup DateTime		Signup DateTime
Approved DateTime		Approved DateTime
Number of Listings		Number of Reads
MinListPrice		Password
MaxListPrice		
PublisherType		
Content Type (1:N)		
Password		

The blockchain should contain data related to attributes that is listed under the blockchain ledger record of which consumer record is a part, while other data attributes may be stored outside blockchain.

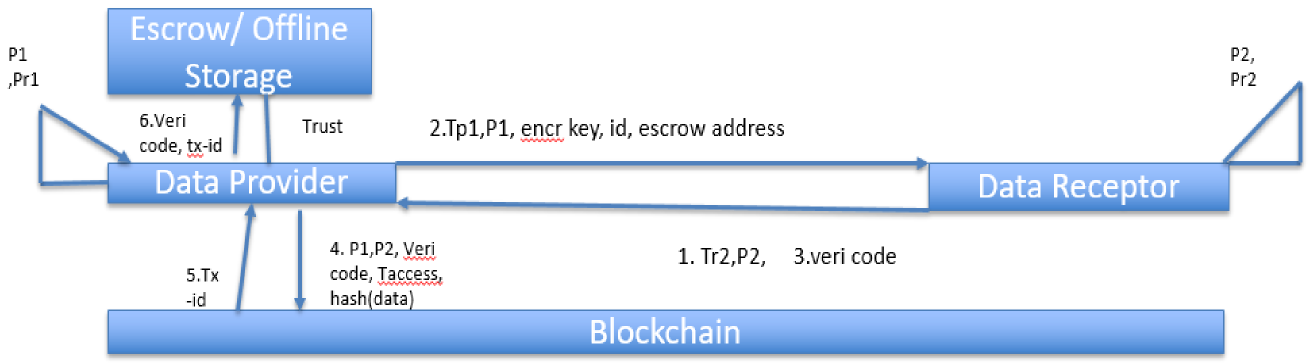
Implementation protocols high level overview

Escrow Key exchange



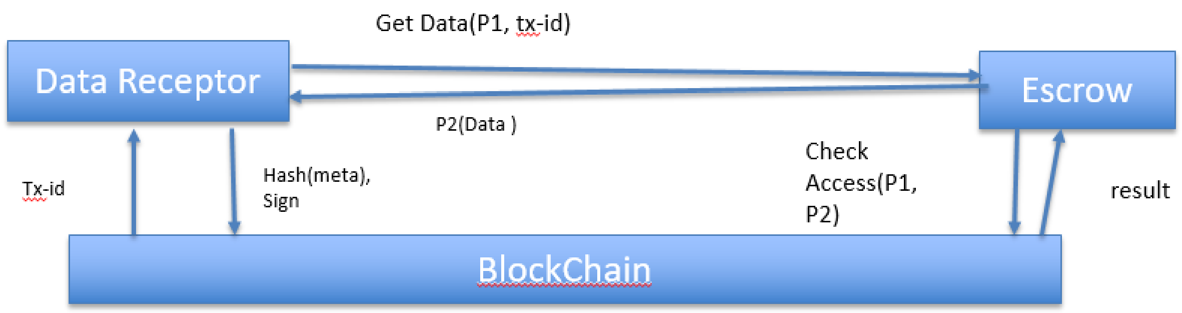
User Escrow Key Exchange:
While creating user account, the user additionally picks a escrow service exhibits an state outline of how a data provider client at first interfaces with its escrow. It is a comparable procedure to a Pretty Good Privacy (PGP) key exchange. Initially the data provider client creates an arbitrary 32 byte string that will be utilized to recognize the data provider client. At that point it creates asymmetrical keys utilizing an elliptic prime bend of 384-piece. Alongside its ID, the user data provider client sends its public key to permit the escrow to safely communicate with it. The escrow does a similar procedure and reacts with encrypted value of its public key.

High level Permission Grant Diagram



High Level Permission Grant
To make these sorts of communications possible, first the data provider user needs to give the msg access to handle its information. Figure above presents a state review for offering authorizations to data receptor user. It begins with the data provider creating asymmetric keys P1, Pr1 that are utilized to approve exchange marks inside the blockchain, an ID, a symmetric key to encode its valuable data, an email address (or any other approval strategy) and the address of the escrow. The data receptor user gets the data provider transaction ID TP1, alongside the common encryption key and email address, then it creates its own asymmetric keys P2, Pr2, and an unique ID TP2 and sends back to data provider its Tr2, public key P2 and an email with a confirmation code. The data provider communicates an exchange with the authorization, two open keys P1 , P2 and check code, hash data on blockchain and updates its escrow with the got check code and Tx-id it received from blockchain after it broadcasted permission credentials. These public keys will relate future related transactions by their signs.

Reading data



Reading Data: The data/msg put away in escrows is accessible on-request to data requestors without requiring explicit action given that the data provider gave get to rights and recorded them inside the blockchain. Figure demonstrates state outline for an on-request data/msg read req issued by an data requestor. Initially, the data requestor is required to review its expectations inside the blockchain, for instance "reading client address". At that point it sends a demand to recover wanted information from the data provider's escrow. The escrow watches that the administration for sure has entry rights and reacts with the put away encoded information. The data provider and data requestor who share the symmetric encryption keys to data and only they can access that data.

Revoke Permissions

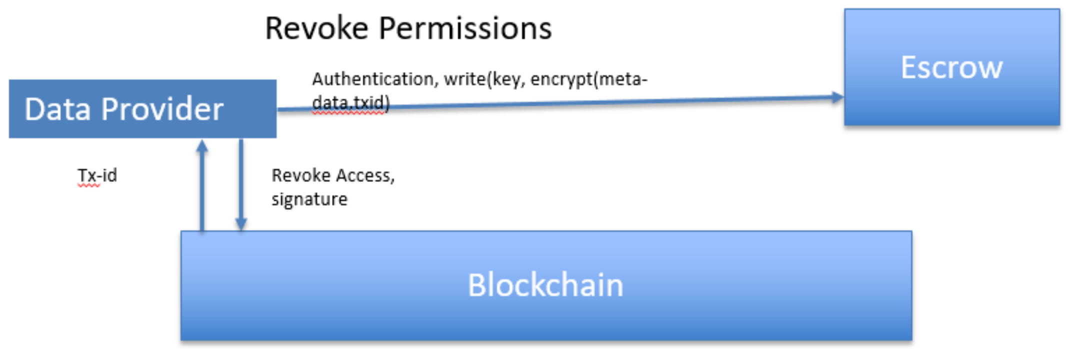


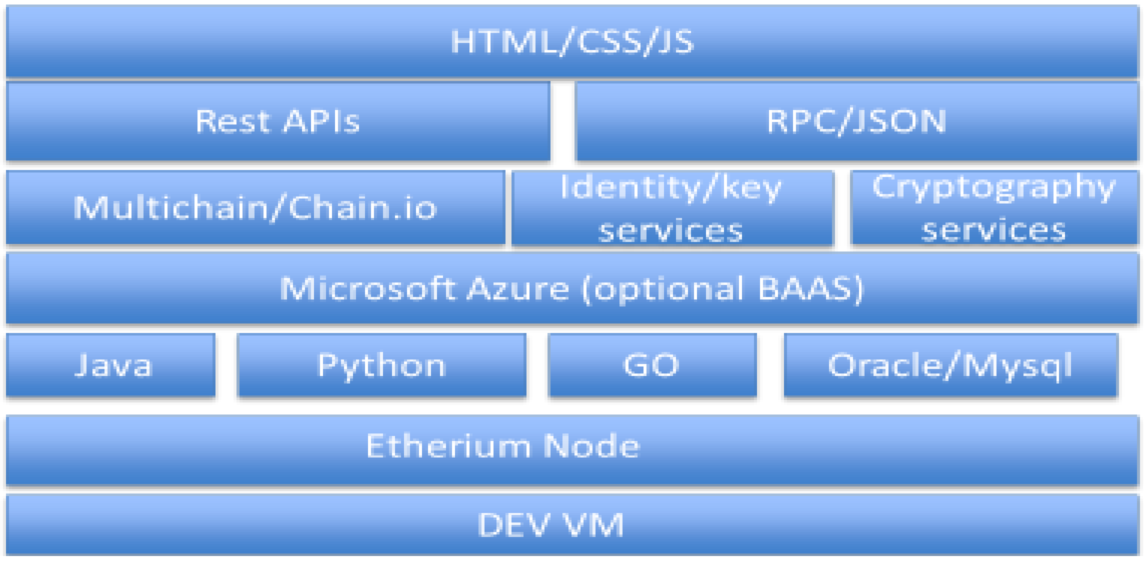
Figure above presents a diagram of how a data provider revokes authorizations from a data requestor to get to their valuable information. The key part in this outline is the transaction

(revoke access, signature) put away inside the blockchain(tx-id) that demonstrates the data provider's expectations to deny access to its information for that particular administration. Later, the client's escrow will discover this transaction (authentication, write (key, encrypt(meta-data, txid) and deny access to valuable information for the data requestor. The data provider additionally triggers a call to compose a review log to keep it for its record.

Technology Stack

Below is the proposed technical stack that can be changed according to system feasibility as there are many new technologies supporting blockchain which are rapidly making new sets of features.

Technology Stack for using etherium blockchain



Dev VM would be individual nodes on distributed peer to peer Etherium blockchain network. Nodes on Etherium are flexible as smart contracts can be developed on them which can be used in sync with the distributed blockchain network to implement a wide variety of transactions. The etherium blockchain implementation of smart contracts can be written in either Java/ Python / Go languages which gives it robust functionality as there are many libraries already developed in these languages for smart contracts. Oracle/ Mysql would be used for interacting with offline storage. Microsoft Azure provides Blockchain as a Service for Etherium Blockchains which makes development work easier but it is expensive hence decision should be left open whether to use it or not. Multichain.com and Chain.io has many services to develop private blockchain and provide Rest APIs for interacting with the blockchain using JSON objects. Identity key/ Services and cryptography services will be used to interact securely by users to participate in the system and interact securely over the network.

Challenges to implementation

The challenges that I see in implementing such a system would be as follows:

Scalability is one of the main criticisms of public blockchains. Currently, the public blockchains, like Bitcoin and Ethereum, can only handle on average 3-20 transactions per second, while the mainstream payment service, like VISA, can handle on average 2000 transactions per second. Although private blockchains can process more transactions than public blockchains, it would be much lesser than that of current market players like Visa.

Another concern of using blockchain is that all the information on the blockchain is publicly available to all the participants within the network, especially the information on the public blockchain, which is publicly accessible by everyone. Cryptography is the only way to preserve data privacy it makes the system very complex. Besides, if a public blockchain is used, running computations on the blockchain costs actual money. Thus, it is not feasible for applications to deploy all computations and store all data on the blockchain.

Further integrating offline/escrow storage with blockchain ledger will also be challenging because of complexity of the system.

Conclusion:

In this paper, architecture for a system is designed so that blockchain as a software component can be used to send messages anonymously and securely. The implementation of proposed system block chain infrastructure for data transfer would enable a new class of business methods that enables the maintenance of privacy of personal information while giving access to actionable data and implementing a fair and transparent market for data producers and data buyers. Such a system if implemented can be used for wide variety of applications apart from data trading platform. It can give the content creators/generators ownership right to their data such that no one else can use their data without their permission. This enables content generators to give conditional permission to other people/application to securely access their data; such that these permissions can be revoked at the data providers/content generators will. For example this system can be used by user to give permission to applications which he installs on his mobile laptop for his personal data and revoke the permissions when he feels particular application should not access his data.

References:

[1] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System.”

[2] T. Hardjono and N. Smith (Eds), “TCG Infrastructure Reference Architecture for Interoperability (Part 1) – Specification Version 1.0 Rev 1.0,”
June 2005, http://www.trustedcomputinggroup.org/ resources.

[3] Wikipedia. Distributed Hash Table. http://en.wikipedia.org/wiki/ Distributed_hash_ table.

[4] Ethereum. A platform for decentralized applications. https://www. ethereum. org/, 2014.

[5] Factom. A Scalable Data Layer for the Blockchain. 2014.

[6] BitTorrent. BitTorrent Labs - BitTorrent Bleep. 2014.

[7] Coinbase. Bitcoin wallet, for merchants and an exchange. *https://www.coinbase.com/,* 2012.

[8] CounterParty. A platform for free and open financial tools on the Bitcoin network. *http://counterparty.io/,* 2014.

[9] Yves-Alexandre de Montjoye, Erez Shmueli, Samuel S Wang, and Alex Sandy Pentland. openPDS: protecting the privacy of metadata through SafeAnswers. *PloS one,* 9(7):e98790, January 2014.

[10] Cypherpunks. Mailing List. https://www.cypherpunks.to/list/

[11] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa). International Journal of Information Security, 1(1):36–63, 2001.

[12]Latanya Sweeney. k-anonymity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 10(05):557–570, 2002.

[13] Michael Lesk. How much information is there in the world?

[14] Microsoft Corp, “Trusted Platform Module and Bitlocker Drive Encryption,” https://msdn.microsoft.com/enus/library/windows/hardware/dn653315