ATLASCHAIN

A reputation based blockchain for building digital marketplaces

Version 0.3

Note: We recommend reading both the Ethereum whitepaper[1] and the Quorum whitepaper[2] as prerequisites to this paper. Active research is under way, and new versions of this paper may change. Project progress can be tracked at https://github.com/exfidabona/atlaschain. For comments and suggestions, please email us at research@atlaschain.io.

[1] https://github.com/ethereum/wiki/wiki/White-Paper
[2] https://github.com/jpmorganchase/quorum-docs/blob/master/Quorum%20Whitepaper%20v0.1.pdf

Introduction

The combination of the internet and e-commerce has made it possible to buy almost anything imaginable and have it delivered to your doorstep. However, the same open marketplaces do not exist for information. You cannot directly sell your personal information about your credit to your landlord or bank. Despite the lack of a direct market, large information bureaus make billions of dollars every year collecting this information and reselling it, so we know the information itself holds value.

The reason for the disparity between physical goods and information markets is that the accuracy of information is hard to trust. Information is easy to manipulate, so we have to rely on the reputation of the source for an accurate verification that at least resembles the truth.

With new emerging technologies, we are now on the brink of decentralized applications, and we predict that information marketplaces will be one of the chief market segments to emerge.

The key innovations enabling an explosion in growth of direct information commerce are the advent of the blockchain and the advancement of zero-knowledge proof protocols.

This project combines these ideas into a new framework built specifically to enable powerful information marketplaces to be built and deployed quickly and easily. This paper addresses proposed solutions to the new features required and ties it all together into a product design.

Blockchain for Data Storage

A decentralized storage model is ideal for a middleman-free marketplace because access and the free flow of data cannot be easily controlled.

Our framework will be built using a permissioned blockchain project JPMorgan Chase & Co. developed called Quorum[3]. Quorum is based on the Ethereum[4] protocol, which is a production-hardened open-source peer-to-peer protocol that stores data in a blockchain.

[3] https://github.com/jpmorganchase/quorum
[4] https://github.com/ethereum/go-ethereum

A blockchain is a decentralized datastore that functions as a ledger shared among a network of peers. The communication protocol between peers to determine how to make updates to the ledger only allows for new entries to be written to the end of the ledger. This common standard among participants makes it really hard to trick other hosts into accepting malicious changes. This protocol doubles as a security mechanism because it makes the blocks functionally read-only once there is consensus about the validity of their creation.

The upside of immutability is that data can not be easily corrupted or tampered with so you can trust its integrity. The downside is that incorrect information remains in place and cannot be changed without a complete network reset.

Data Privacy

Quorum added an encrypted message exchange to each peer-to-peer node in order to enable private transactions and restricted contract visibility.

These privacy features allow us to grant control of the information solely to the owner of that information and then allow the owner to control who can gain or lose access to that data after it is stored in the blockchain.

For example, a piece of personal private information can be added by an external account to create a new private storage smart contract. Access to that data can then be granted to other public key holders besides the block creator, by sending a new encrypted transaction granting access to that key. While we require open access to the information in order to prevent centralized control, we also require that the information be encrypted such that only the user that created the information can access it.

Authenticating Information

The goal of the framework is to allow users to put encrypted data onto the blockchain such that they control their own data assets and those that get access to view it.

The challenge in a marketplace is how can we get buyers on the demand side for the information to trust that the information is accurate without granting a middleman access. The answer is by adding an authenticating proof engine to our nodes. All information requires different means and methods for authentication, so our engine will require the proofing

mechanisms to be exchangeable plugins. All of the proof protocol plugins for the engine will be based on a common zero-knowledge proof[5] system.

The basic idea behind zero-knowledge proof algorithms is to create a verification mechanism which allows a separate party to verify underlying information without actually having to view the information explicitly. A common approach to solving privacy access problems with a zero-knowledge proof is to disclose indiscernible pieces of information one at a time such that their confidence in the underlying information improves with every new iteration.

In our framework the proofing engine that will function as a black box within each node that encrypts the information asset provided to it. The engine will also split the information into chunks and send it to other parties in order to achieve a minimum disclosure proof, which is a subset of zero-knowledge proofs.

An overly simple example would be a social marketplace to buy access to users' birthdates. In the real world this example would not be incredibly valuable because it is still easy to create a fake birthday, as well as the fact that date formats are trivial enough to validate without involving peer oversight. Nevertheless it exhibits how a multi-party protocol can collectively verify the information without anyone getting full knowledge of the underlying information.

In this scenario a user would submit a private transaction with their birthdate as the asset. The proof engine inside the node would receive the raw information from the request and cryptographically hash it such that only a encrypted digest of the information would be stored. This proof protocol would also chunk the birthday into separate day, month and year values and send them to other peers on the network for approval. Each peer would receive one of the separate pieces in the form of a question, such as "Is 1976 a valid year for a birth date". The information includes no identifying information about the user. If all the peers respond yes than the information is deemed correct and the transaction gets written into a block. If any of the independent verifiers respond with no, than the validation process fails and the transaction does not get stored.

A more realistic application of the protocol would be to validate users' birth certificates. In order to achieve this, the user will upload a scanned image of the birth certificate. The proof engine would encrypt the image and simultaneously carve it up into smaller images such that the entire birth certificate is not discernable from one slice alone. The proof engine would then send out these individual slices to separate peers asking them if this looks like it could be a birth certificate for the public identifiable name provided by the user. The protocol would ask the validator, "Could this image slice be a birth certificate for a user named John Smith?". If any peers respond no, then the asset gets rejected. If all peers respond yes than the asset gets approved.

[5] https://en.wikipedia.org/wiki/Zero-knowledge_proof

The first protocol used for birth dates would be a multi-party minimum disclosure protocol for ascii character determination. The second would be a partial image slicing protocol for total image verification. Ideally these proof protocols are installable as docker containers and interchangeable within the proof engine.

A Reputation System

The peers who verify in the zero-knowledge proof handshake are called Validators. They earn a reputation rating based on the reaffirmation of accuracy of the information over time that they approve/disapprove as well as the frequency in which they participate in the marketplace as consensus voters.

An accuracy score and voting record for every Validator will be stored in a smart contract on the blockchain. Smart contracts help function as the application layer for the marketplace. All the marketplace logic smart contracts have hardcoded addresses assigned in the genesis block, which is another name for the very first block in a blockchain.

The accuracy score is continually compounding as buyers of information confirm that the underlying information is valid after purchase.

These confirmation binary grades are distributed to all the Validators who participate in a verification proof. If they help successfully verify a piece of information, they continue to earn positive accuracy points every time it is reaffirmed. If a Validator verifies a piece of information and that asset is graded as inaccurate by the buyer after disclosure, then the accuracy score is reduced. Ideally an asset with more negative grades than positive grades will be blacklisted such that new buyers do not continue to purchase that asset.

Also if a Validator rejects an asset that is subsequently verified in another attempt then the rejecting validator gets a compounding negative grade for every positive grade given. This is called an inverse negative grade.

Full Reputation Equation

$$Accuracy\ Coefficient = max\left((sum\ of\ true\ proof\ evaluations\ -\ sum\ of\ false\ proof\ evaluations\ -\ inverse\ negatives), 0\right)$$
$$Attendance\ Record\ =\ max((times\ voted\ -\ missed\ votes), 0)$$
$$Reputation\ =\ Accuracy\ Coefficient\ *\ (Attendance\ Record)/Total\ System\ Blocks$$

Reputation in Distributed Consensus

All peer-to-peer networks require a consensus algorithm to accurately establish and communicate the current state of the network's data. In our framework we will use raft

consensus with a couple of variants. Raft is used as a consensus algorithm by Quorum. The Raft White Paper[6] describes the algorithm in detail. In our version, we have incorporated reputation and incentivization into leader elections.

Leader- Election Changes

In merit-based raft, each participating Validator node has a reputation. This reputation is earned over time based on the graded accuracy from the proof process and participation in elections.

A lottery is held among the Validators to select 3 candidates for leader. These 3 candidates then perform the standard raft leader election process until consensus on a leader is reached. The last chosen leader is required to sit out the next candidate lottery.

The distribution of lottery tickets is based on the reputation scores of the Validators. The higher a Validator reputation, the more likely they are to become a candidate. For example, let us say there are five Validators and each has a respective Reputation of 50, 100 and 150, 200 and 250. The total number of lottery tickets will be 750, and each Validator gets the exact same number of tickets as their reputation. If the reputation is a decimal, we will always round down to the nearest whole number. The odds for each Validator to become the first candidate selected in the block lottery would be 6.67%, 13.33%, 20%, 26.67%, and 33.33% respectively.

After the first candidate is selected, the lottery tickets of the chosen candidate are removed and the new odds become updated to reflect the updated pool. If no more candidates are available, the candidate selection stops and the election begins with fewer than 3 participants.

Once the election is completed, all non-leader Validator nodes become followers and pass all transactions to the leader node. The leader fills up a block with entries until either block size capacity is reached or it receives an asset transaction. Once either happens a block is created, the leader receives the cumulative transactions' gas costs, the mining-block reward or both depending on how the network is configured. Leadership is then rescinded after block creation, and a new candidate-selection process begins without including the most recent leader in the lottery.

This consensus algorithm is designed specifically for permissioned information-based blockchains.The goal is to tie information authentication together with peer-to-peer consensus tracked through proof validity in a reputation score. Reputation compounds over time to increase the probability of the frequency for earned incentives in the future. Similar to other blockchain consensus algorithms, the earned incentive combined with the append-only nature of the ledger protocol adds an extra layer of security assurance to the integrity of the data being

[6] https://raft.github.io/raft.pdf

stored. As long the incentive to earn a positive reward on the network outweighs malicious behavior, the data in the network shall remain secure.

Governance

The coming challenge blockchains will face is how to adapt the rules and upgrade the technology over time to make the network better. The early flavors of blockchain did not account for the need to make fast system-level changes, so they rely on the leadership of the core developers and the slow propagation of ideas through word of mouth. Later flavors added governance rules into the on-chain consensus process, that has led to faster adaptability.

The concept of system-level governance in a blockchain is emerging along with the maturation and growth of the technology. Public and private blockchains face different governance obstacles.

These new public blockchain governance models are loosely modeled after democracies. For this reason, public blockchains face the classic democracy problem of slow adaptation of change. Private blockchains on the other hand are by default authoritarian and can adapt and change almost immediately. However, private blockchains are presented with a public perception problem. The public needs to know that the authority in charge of the blockchain will always make decisions that are in the best interest of the ecosystem and not just themselves.

Adaptability is incredibly important for the future sustainability of blockchain-based applications, therefore we aim to to make the governance model configurable within our framework.

The default configuration on our framework is to use a benign authority model. The creator of the network, labeled a Steward is charged with sustaining a healthy network through incentive manipulation, while unable to directly exert influence over their own incentive. Extra protections to offset the Steward’s power can be customized in the network defining smart contracts. Governance rules should be customizable with out of the box plugin options that are easily installable.

Product Design

The ambition behind our project is to build a framework that makes it simple to design and easily deploy a fully functional information marketplace where the assets being traded are completely customizable. A useful analogy would be to call it a wordpress-like system for blockchain-based marketplaces.

Similar to centralized web applications, our products will need three layers: the data layer, the application layer, and the web-based graphical interface.

The Data Layer

The Node

Quorum Blockchain

All of the asset transaction data will be stored on a private blockchain. The blockchain protocol will be adapted from Quorum to fit the needs of a standard marketplace.

Private Transactions with Constellation

Constellation is a peer-to-peer encrypted message exchange designed to restrict read access to only a select few users who are given explicit access. Constellation comes working out of the box as part of Quorum.

Proof Engine

The proof engine will be a docker container added to the geth node. The docker container holds a PGP encryption tool as well as a service that adds in the multi-party zero-knowledge proof protocol plugin that is to be used by that unique marketplace.

Centralized Data Storage

NoSQL Database

The NoSQL database will store all of the user and administration data for the marketplace since these entities are controlled outside the blockchain. The database is structured in a very similar manner to the wordpress mysql database layer except that the data will only be accessible through the web based api layer.

The Application Layer

The Genesis Block Installation

The genesis block is the first block created for a blockchain and must be created by the marketplace Steward. This block defines any initial asset allocation, the Steward accounts, as well as the application layer smart contracts that drive the system. The genesis block is defined in a json file named genesis.json. The contracts will define validator reputation tracking, the rules of the information bidding system, and the incentivization model and network-governance rules. Each contract type will have a default contract that is installed if not overwritten. The steward will be defined by three accounts which will all get an equal distribution of the initial token supply created in the genesis block. These three account keys will be required to create and update all smart contracts. All smart contract creation attempts not signed by the Stewards three private keys will be rejected.

```
{
  "alloc": {
    "0x0000000000000000000000000000000000000020": {
      "code": "[Code String Redacted]",
      "storage": {...}
    },
"0x0000000000000000000000000000000000000040": {
      "code": "[Code String Redacted]",
      "storage": {...}
    },
"0x0000000000000000000000000000000000000060": {
      "code": "[Code String Redacted]",
      "storage": {...}
  },
    "0xed9d02e382b34818e88b88a309c7fe71e65f419d": {
      "balance": "10000000"
    },
    "0xca843569e3427144cead5e4d5999a3d0ccf92b8e": {
      "balance": "10000000"
    },
    "0x0fbdc686b912d7722dc86510934589e0aaf3b55a": {
      "balance": "10000000"
    }
  },
  "coinbase": "0x3333333333333333333333333333333333333333",
```

```
  "config": {
    "homesteadBlock": 0,
    "isAtlasChain":true,
    "currencyName":"Atlas",

"stewards":["0xed9d02e382b34818e88b88a309c7fe71e65f419d","0xca843569e342714
4cead5e4d5999a3d0ccf92b8e","0x0fbdc686b912d7722dc86510934589e0aaf3b55a"]
  },
  "difficulty": "0",
  "gasLimit": "2100000",
  "mixhash":
"0x0000000000000000000000000000000000000000000000000000000000000000",
  "nonce": "0",
  "parentHash":
"0x0000000000000000000000000000000000000000000000000000000000000000",
  "timestamp": "0x00"
}
```

Figure 1: Example genesis.json

Application Smart Contracts

Direct ether transactions are turned off in Quorum and therefore ether transactions are not be available on our blockchains. The majority of the transactions on the system will be transacting with smart contracts. There will be three main smart contracts required for the marketplace to function, plus an additional smart contract to customize business logic if necessary. These smart contracts will be configured using an administration interface and hardcoded into the peer-to-peer client.

Marketplace and Reputation Smart Contract

This contract is defined such that Validators can be added or removed by the Steward. Further it tracks each Validators accuracy from the zero-knowledge proofs over time. Each Validator has a map of variables that go into calculating the ever evolving reputation including sums of positive proofs, negative proofs, inverse negatives, times voted and missed votes. The functions in these smart contracts will update the Validators reputation and the underlying variables as actions happen on the network.

This smart contract also defines the rules for the buy and sell bidding system within the marketplace.

Incentivization and Currency Smart Contract

Incentives can be attached to any fiat currency or cryptocurrency and be charged as transaction fees. Alternatively a new cryptocurrency can be defined to be used as a specific utility token within that blockchain network both to reward block creation as well as be the medium exchange for the buy and sell of assets. If you want to define a cryptocurrency for an ICO, this is the place to do it.

The Governance Smart Contract

This smart contract sets the rules for how new versions are upgraded and deployed on the blockchain. For instance, if the encryption algorithm needs to be upgraded across the board, this smart contract will enforce Validator node holders to the update to the latest version in order to participate in future consensus votes. By default, Steward signatures will be required to establish changes, but a new democratic voting system could put in place as well to override the default.

The Optional Business Rules Smart Contract

This smart contract is completely optional, but when provided it allows extra logic to be applied to assets after data verification.

Blockchain APIs

Quorum has a subset of APIs that are accessible to extract data from the blockchain. The blockchain is accessible via JSON RPC API calls.

Web based APIs

OAuth 2.0 based APIs will be created to access all of the data objects stored outside of the blockchain. Each object will be read, edited and deleted through the get, edit and delete api calls. Access to these APIs will be controlled through user permissions.

Interface Layer

Web Based Administration

This interface is designed to be a modern web application interface similar to the admin section of a wordpress site. It allows for customization and control as well as the deploy of system mandated upgrades and governance changes by the Steward.

A Client for Validator Nodes

Validators will be required to host a copy of the blockchain and therefore they will need to download a full copy of the blockchain on a hosted server or mobile device depending upon the required criteria and any necessary manual steps in the zero-knowledge proof process. These nodes will make up the peer-to-peer network.

Web Based Marketplace Interface

The web -based interface will be built and designed to optimize marketplace interaction. The default will have a similar feel to an ebay-like interface. However this will be fully customizable by the Steward.

Use Case: Selling Email Addresses To Marketers

This use case defines a marketplace to sell valid email addresses directly to internet marketers.

The Proof Engine Plugin is an email verification handshake process that uses a Validator's own email address to facilitate sending the verification link. The link then points back to the proof engine such that the validation comes full circle and becomes verified.

Install Governance using the default benign authority model such that the Steward can manually adjust system incentives and initiate platform upgrades, but cannot adjust their own business model which is a flat 5% transaction fee.

Add Additional Business Logic for marketers to rate the value of each email address they purchase.

Override Consensus and Reputation Algorithm in order to focus the reputation calculation on Validator node's availability instead of the accuracy of asset grading since all that is required is for the Validator to be an email proxy.

Incentivize using a custom cryptocurrency named EmailCoin. The network creates 10 million coins in the initial supply and they sell 7 million of those to raise $3.5 million dollars in an Initial Coin Offering to help bootstrap demand and to pay initial validators to participate up front. Mining for the coin works such that, for every new block created, 12 new EmailCoins are created and assigned to the validator as a reward for their work on the network. These new EmailCoins are meant to capture the immediate value created by the expansion of this database. EmailCoin is also to be used as the exchange of value sent from the marketer to the email owner.

Use Case: Building Inspections

An alternative marketplace that is created and mandated by a municipality. The municipality would pass a variance mandating that building owners pay for their inspections through the marketplace. This would allow municipalities to scalably monitor all inspections within their own jurisdiction.

Validators are the physical inspection companies assigned by the municipality to inspect buildings for compliance of fire and/or electrical code regulations.

The Proof Engine only requires a valid signature from a pre-qualified inspector.

All Governance is controlled authoritatively by the municipality

All Incentivization is controlled and transacted in USD for inspection payments through an attached payment processor such as Stripe.

Other Potential Use Cases

- A decentralized credit score sold directly from consumers to lenders
- Vendor compliance monitoring that compiles business data into an actionable checklist
- A direct due diligence information sharing database
- A network of educators willing to sell their time to share specific knowledge

Conclusion

Blockchain is the perfect medium for decentralized marketplaces, and Quorum's modifications to the go-ethereum project have done most of the heavy lifting. We plan to extend it by creating an easily configurable marketplace framework for entrepreneurs who want more control over their own economies. In order to achieve this, we need to combine blockchain transaction privacy with an unique authentication proof engine and the power to build tokens that can capture the utility of each asset database being built in real-time.