Factom

Business Processes Secured by Immutable Audit Trails on the Blockchain

Contributors: Paul Snow, Brian Deery, Jack Lu, David Johnston, Peter Kirby

Advisors: Adam Stradling, Shawn Wilkinson, Jeremy Kandah, Dexx, Marv Schneider, Steven Sprague, Andrew Yashchuk

Reviewers: Vitalik Buterin, Luke Dashjr, Ed Eykholt, Ryan Singer, Ron Gross, J.R. Willett, Dustin Byington

Version 1.0
November 17, 2014
www.Factom.org

Abstract

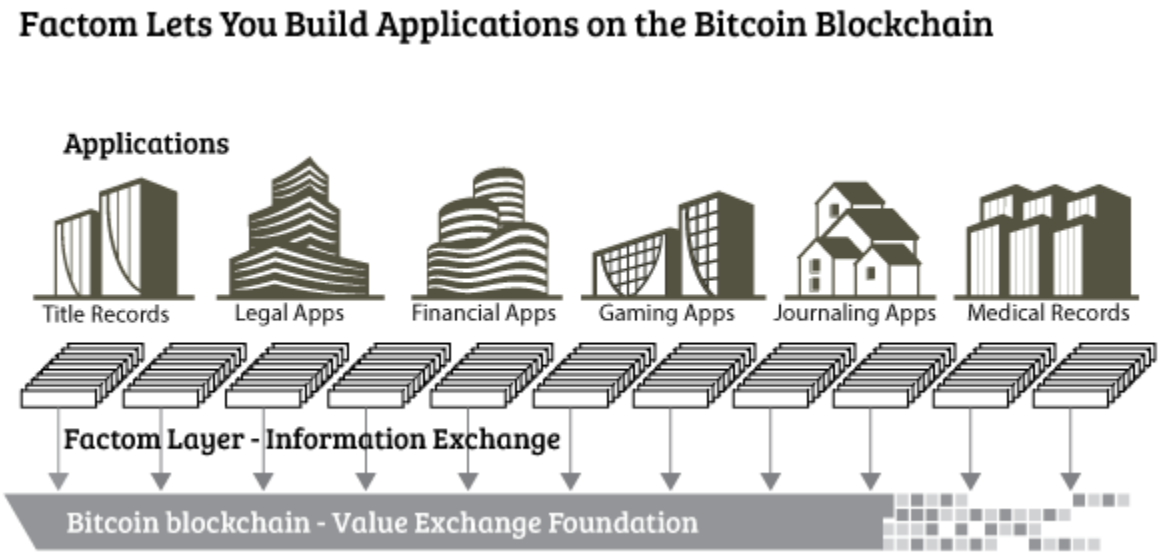
“Honesty is subversive” - Paul Snow

In today’s global economy trust is in rare supply. This lack of trust requires the devotion of a tremendous amount of resources to audit and verify records - reducing global efficiency, return on investment, and prosperity. Moreover, incidents such as the 2010 United States foreclosure crisis demonstrate that in addition to being inefficient, the current processes are also terribly inaccurate and prone to failure. Factom removes the need for blind trust by providing the world with the very first precise, verifiable, and immutable audit trail.

In the past, records have been difficult to protect, challenging to synchronize, and impossible to truly verify because of the manual effort involved. Computers automated some of these tasks, but they are even harder to protect, synchronize, and verify because computer records are so easy to change. Authority is fragmented across innumerable independent systems.

Blockchains provide a distributed mechanism to lock in data, making data verifiable and independently auditable. Bitcoin’s blockchain is the most trusted immutable data store in existence; however, it is not very useful for non-Bitcoin transactions. Factom gives businesses access to blockchain technology without getting bogged down in currencies.

In this paper, we describe how Factom creates a distributed, autonomous protocol to cost effectively separate the Bitcoin blockchain from the Bitcoin cryptocurrency. We discuss client-defined Chains of Entries, client-side validation of Entries, a distributed consensus algorithm for recording Entries, and a blockchain anchoring approach for security.



Design Goals

Factom Creates a Faster, Cheaper, Bloat-free Way to Develop Blockchain Based Applications

When Satoshi Nakamoto launched the Bitcoin blockchain he revolutionized the way transactions were recorded. There had never before existed a permanent, decentralized, and trustless ledger of records. Developers have rushed to create applications built on top of this ledger. Unfortunately, they have been running into a few core constraints intrinsic to the original design tradeoffs of Bitcoin.

1) Speed – because of the design of the decentralized, proof-of-work consensus method used by Bitcoin, difficulty requirements are adjusted to maintain roughly 10 minute confirmation times. For applications that wish greater security, multiple confirmations may be required. A common requirement is to wait for 6 confirmations, which can lead to wait times over an hour.

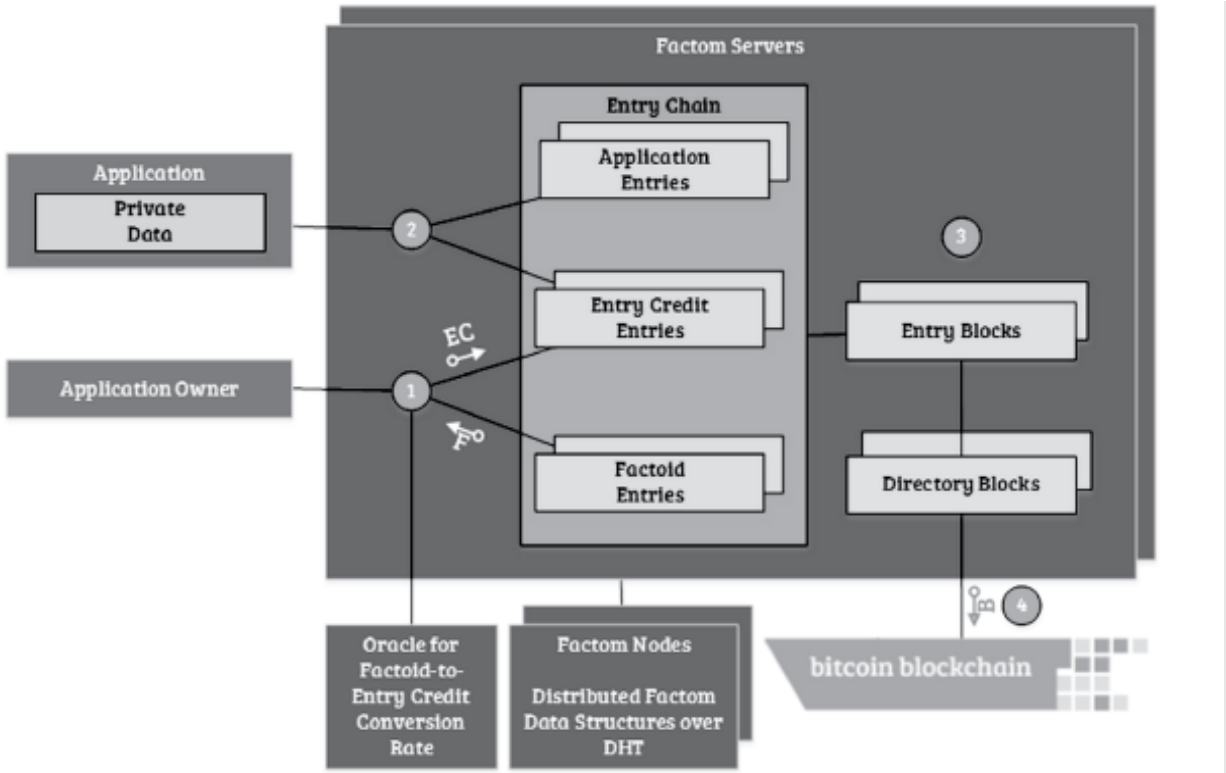
2) Cost – the default transaction cost is around .01 mBTC (roughly $0.003 USD in November 2014). The exchange price of BTC has been volatile throughout its history. If the price of BTC rises, then the cost of transactions can go up. This can prove to be a serious cost barrier to applications that need to manage very large numbers of transactions. Additionally, many factors including constraints on block size and reward halving could act to increase transaction fees.

3) Bloat – with the Bitcoin blockchain size limit of 1 MB per block, transaction throughput is capped at 7 transactions per second. Any application that wants to write and store information using the blockchain will add to the traffic. This problem has become politically charged as various parties seek to increase the block size limit.

Factom is a protocol designed to address these three core constraints. Factom creates a protocol for Applications that provide functions and features beyond currency transactions. Factom constructs a standard, effective, and secure foundation for these Applications to run faster, cheaper, and without bloating Bitcoin.

The Factom Ecosystem

There are several primary components in the Factom ecosystem, as depicted below:



Once the system is set up, including issuance of Factoids (i.e., the cryptocurrency of Factom) and user accounts, token value is transferred among users, Factom, and Bitcoin with the following primary interactions:

1. Application Owner purchases Entry Credits with Factoid
2. Application records an Entry
3. Factom Servers create Entry Blocks and Directory Blocks
4. Factom secures an anchor (hash of the Directory Block) onto the blockchain

Details of these and other interactions are in the upcoming sections.

Security and Proofs

How Factom Secures Entries

Factom extends Bitcoin's feature set to record events outside of monetary transfers. Factom has a minimal ruleset for adding permanent Entries. Factom pushes most data validation tasks to the client side. The only validation Factom enforces are those required by the protocol to trade Factoids, convert Factoids to Entry Credits, and to ensure Entries are properly paid for and recorded.

> *Factom has a few rules regarding token incentives for running the network and for internal consistency, but it cannot check the validity of statements recorded in the chains used by its users.*

Bitcoin limits transactions to those moving value from a set of inputs to a set of outputs. Satisfying the script required of the inputs (generally requiring certain signatures) is enough for the system to ensure validity. This is a validation process that can be automated, so the auditing process is easy. If Factom were used, for instance, to record a deed transfer of real estate, Factom would be used to simply record the process occurred. The rules for real estate transfers are very complex. For example, a local jurisdiction may have special requirements for property if the buyer is a foreigner, farmer, or part time resident. A property might also fall into a number of categories based on location, price, or architecture. Each category could have its own rules reflecting the validation process for smart contracts. In this example, a cryptographic signature alone is insufficient to fully verify the validity of a transfer of ownership. Factom then is used to record the process occurred rather than validate transfers.

Bitcoin miners perform two primary tasks. First, they resolve double spends. Seeing two conflicting transactions that spend the same funds twice, they resolve which one is admissible. The second job miners perform (along with the other full nodes) is auditing. Since Bitcoin miners only include valid transactions, one that is included in the blockchain can be assumed to have been audited. A thin client does not need to know the full history of Bitcoin to see if value they receive has already been spent. (See SPV.)

How Factom Servers and Auditing Servers Validate Entries

Factom splits the two roles that Bitcoin miners do into two tasks: 1 - recording Entries in a final order and 2 - auditing Entries for validity.

1 - The Factom servers accept Entries, assemble them into blocks, and fix their order. After 10 minutes, the Entry ordering is made irreversible by inserting an anchor into the Bitcoin blockchain. Factom does this by creating a hash of the data collected over the 10 minutes, then recording the hash into the blockchain.

2 - The auditing of Entries is a separate process which can be done either with or without trust. Auditing is critical, since Factom is not able to validate Entries before they are included in the Factom dataset.

With trust-based auditing, a thin client could trust a competent auditor they choose. After an Entry was entered into the system, an auditor would verify the Entry was valid. Auditors would submit their own cryptographically signed Entry. The signature would show that the Entry passed all the checks the auditor deemed was required. The audit requirements could in fact be part of a Factom Chain as well. In the real estate example from earlier, the auditor would double check the transfer conformed to local standards. The auditor would publicly attest that the transfer was valid.

Trustless auditing would be similar to Bitcoin. If a system is internally consistent with a mathematical definition of validity like Bitcoin, it can be audited programmatically. If the rules for transfer were able to be audited by a computer, then an Application could download the relevant data and run the audit itself. The application would build an awareness of the system state as it downloaded, verified, and decided which Entries were valid or not.

Mastercoin, Counterparty, and Colored Coins have a similar trust model. These are all client-side validated protocols, meaning transactions are embedded into the Bitcoin blockchain. Bitcoin miners do not audit them for validity; therefore, invalid transactions designed to look like transactions on these protocols can be inserted into the blockchain. Clients that support one of these protocols scan through the blockchain and find potential transactions, check them for validity, and build an interpretation of where the control of these assets lie (usually a Bitcoin address). It is up to the clients to do their own auditing under these protocols.

Moving any of these client-side validated protocols under Factom would be a matter of defining a transaction per the protocol and establishing a Chain to hold the transactions. The transaction protocols wouldn’t be much different under Factom than under Bitcoin, except where Factom allows an easy expression of the information needed instead of having to encode it in some special way into a Bitcoin transaction.

Proving a Negative

Bitcoin, land registries, and many other systems need to solve a fundamental problem: proving a negative. They prove some “thing” has been transferred to one person, and prove that thing *hasn't been transferred to someone else*. While proof of the negative is impossible in an unbounded system, it is quite possible in a bounded system. Cryptocurrencies solve this problem by limiting the places where transactions can be found. Bitcoin transactions can only be found in the Bitcoin blockchain. If a relevant transaction is not found in the blockchain, it is defined from the Bitcoin protocol perspective not to exist and thus the BTC hasn't been sent twice (double spent).

Certain land ownership recording systems are similar. Assume a system where land transfer is recorded in a governmental registry and where the legal system is set up so that unrecorded transfers are assumed invalid (sans litigation). If an individual wanted to check if a title is clear (i.e., that no one else claims the land) the answer would be in the governmental registry. The individual using the government records could prove the negative; the land *wasn't* owned by a third party. Where registration of title is not required, the governmental registry could only attest to what has been registered. A private transfer might very well exist that invalidates the understanding of the registry.

In both of the above cases, the negative can be proven within a context. With Mastercoin the case is very strong. With a land registry, it is limited to the context of the Registry, which may be open to challenge. The real world is messy, and Factom is designed to accommodate not just the precision of digital assets, but the real world's sometimes messy reality.

In Factom, there is a hierarchy of data categorization. Factom only records Entries in Chains; the various user-defined Chains have no dependencies that Factom enforces at the protocol level. This differs from Bitcoin, where every transaction is potentially a double-spend, and so it must be validated. By organizing Entries into Chains, Factom allows Applications to have smaller search spaces than if all Factom data were combined together into one ledger.

If Factom were to be used to manage land transfers, an Application using a Chain to record such registries could safely ignore Entries in the other Chains, such as those used to maintain security camera logs. Were a governmental court ruling to change a land registration, the relevant Chain would be updated to reflect the ruling. The history would not be lost, and where such changes are actually invalid from a legal or other perspective, the record cannot be altered to hide the order of events in Factom.

Nick Szabo has written about Property Clubs, which have many overlaps with this system. Here is a nugget from his paper "Secure Property Titles with Owner Authority":

> *While thugs can still take physical property by force, the continued existence of correct ownership records will remain a thorn in the side of usurping claimants.*

How Applications Validate Factom Chains

Factom doesn't validate Entries; Entries are instead validated client-side by users and Applications. As long as an Application understands and knows the rules a Chain should follow, then the existence of invalid Entries doesn't cause unreasonable disruption. Entries in a Chain that do not follow the rules can be disregarded by the Application.

Users can use any set of rules for their Chains, and any convention to communicate their rules to the users of their Chains. The first Entry in a Chain can hold a set of rules, a hash of an audit program, etc. These rules then can be understood by Applications running against Factom to ignore invalid Entries client-side.

An enforced sequence can be specified. Entries that do not meet the requirements of the specified enforced sequence will be rejected. However, Entries that might be rejected by the rules or the audit program will still be recorded. Users of such chains will need to run the audit program to validate a chain sequence of this type. The Factom servers will not validate rules using the audit program.

Validation in the Applications (in combination with user-defined Chains) provides a number of advantages for Applications written on top of Factom:

1. Applications can put into Factom whatever Entries make sense for their application. So, a list of hashes to validate a list of account statements can be recorded as easily as exchanges of an asset.

2. Rule execution is very efficient. Where the distributed network must execute your validation rules, then validation requires all nodes to do all validation. Client-side validation only requires the systems that care about those rules to run them. Factom allows a Chain to define its rules in whatever language the designers choose, to run on whatever platform they choose, and to use any external data. None of these decisions on the part of one Application has any impact on another Application.

3. Factom Servers have little knowledge about the Entries being recorded. We use a commitment scheme to limit knowledge, where the commitment to record an Entry is made prior to revealing what the Entry is. This makes Factom's role in recording Entries very simple, and makes individual server processes public. Factom servers accept information from the network of full nodes, and their decisions and behavior are always in view. Failure to perform can be audited both from the network outside Factom, and within Factom. It is easy to independently verify that a Factom server is fulfilling its Entry-recording responsibility; Factom can't hide potentially errant behavior.

4. Recording speeds can be very fast, since the number of checks made by the Factom servers are minimal.

5. Proofs against any particular Chain in Factom do not require knowledge of any other Chains. Users then only need the sections of Factom they are using and can ignore the rest.

How Factom Federated Servers Manage Chains

At its heart, Factom is a decentralized way to collect, package, and secure data into the Bitcoin blockchain. Factom accomplishes this with a network of Federated servers. These servers rotate responsibility for different aspects of the system. No single server is ever in control of the whole system, but only a part of the system. And no server is permanently in control of any part of the system; the responsibility for each part of Factom cycles among the servers each minute.

The Federated servers each take responsibility for a subsection of the user Chains at the beginning of the creation of a Directory Block. The process works like this:

1. All servers reset their process lists to empty.
2. The user submits an Entry Payment using a public key associated with Entry Credits
3. Based on the public key used to pay for the Entry, one of the servers accepts the payment.
4. That server broadcasts the acceptance of the payment.
5. The user sees the acceptance and submits the Entry.
6. Based on the ChainID of the Entry, one of the servers adds the Entry to its process list, and adds the Entry to the appropriate Entry Block for that ChainID (creating one if this is the first Entry for that Entry Block).
7. The server broadcasts an Entry confirmation, containing the process list index of the Entry, the hash of the Entry (linked to the payment), and the serial hash so far of the server's process list.
8. All the other servers update their view of the server's process list, validate the list, and update their view of the Entry Block for that ChainID.
9. As long as the user can validate the relevant process list holds their Entry, then they have a fair level of assurance it will be successfully entered into Factom.
10. At the end of the minute, all servers confirm the process list height, reveal a deterministic secret number (Reverse Hash, publishing successive preimages of a long hash chain), and the serial hash of the process block (that will match the last item in process list).
11. The Directory Block for that minute is constructed from all the Entry Blocks defined by all the servers. So, each server has all Entry Blocks, all Directory Blocks, and all Entries.
12. The collection of Reverse Hashes are combined to create a seed to reallocate ChainIDs among the servers for the next round.

13. At the completion of the 10th Directory Block, do the following:
    a. Create the Merkle roots for the last minute's Entry Blocks, sorted by ChainID.
    b. Create the last minute's Directory Block, and calculate its Merkle root.
    c. Create an anchor out of the Merkle root of the 10 Directory Blocks' Merkle roots.
    d. The server's Reverse Hashes are combined to create a seed that selects the server to write the anchor into Bitcoin.
14. Repeat. (Go back to 1)

The Federated servers for their minute are constructing a process list for the Chains for which they are responsible, as well as constructing the Entry Blocks that will be used to create the Directory Block at the end of the minute. The process list is important for broadcasting decisions made by a server to the rest of the network.

Federated servers are re-ranked every four hours. The ranking is a function of a vote by the users, who must create a profile Chain in Factom. The profile contains any number of signed public address Entries. The weight of a user's vote is determined by the public addresses in their profile. The function computing the weight of a public address is the sum of:

- Weighted Number of Entry Credits purchased in the last six months (Sum of purchases of each month, weighted by 6 in the current month, 5 in the previous, etc.)
- Weighted Number of Entries used in the last six months (Sum of Entries used in each month, weighted by 6 in the current month, 5 in the previous, etc.)

When running with **n** servers, the top ranking **n** servers are the Federated servers, and another **n** are Auditing servers. All the servers are maintained by rank based on the quantity of votes for them. The number **n** is initially specified as 16, but this number is up for community debate, and could be a floating value based on transaction volume.

All servers must broadcast a heartbeat every heartbeat period. (An Entry confirmation broadcast serves in place of a heartbeat.) If a server does not receive a heartbeat or a verified Entry within the timeout period, the server broadcasts a Server Fault Message (SFM). If a majority broadcast a SFM on a server, the Federated server is considered "Failed" and relegated to being an Audit server, and the next ranking Audit server takes over. The promoted server will serve to the end of the current 4 hour term. In the subsequent re-ranking of servers, the Failed server must sit out for another full term.

The heartbeat period and timeout period will be modifiable by the majority of Federated servers documented in a configuration Chain. Looking at Bitcoin propagation times, the heartbeat should be 4 seconds, and the timeout period 8 seconds.

More detail about Factom consensus, the algorithm we are developing, can be found in the working paper, "Factom Consensus".

Factom System Overview

Factom is constructed from a set of layered data structures

Factom is constructed of a hierarchical set of blocks, with the highest being Directory Blocks. They constitute a micro-chain, consisting primarily of compact references. To keep the size small, each reference in the Directory Block is just a hash of the Entry Block plus its ChainID. These Entry Blocks have references which point to all the Entries with a particular ChainID which arrived during a time period. The Entry Block for a Chain ID is also part of a micro-chain. The bulk of the data in Factom is at the leaves, the Entries themselves. These hierarchical data structures are rendered unchangeable by Bitcoin's hashpower. They can be conceptualized as different layers.

The layers and concepts in the Factom system are:

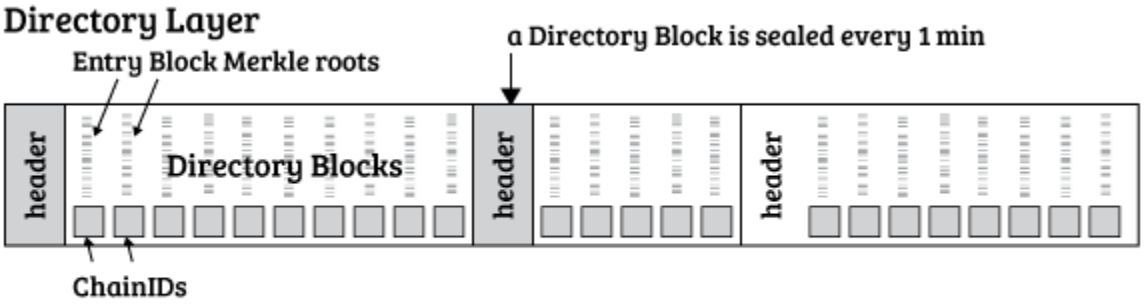
1) **Directory Layer** -- Organizes the Merkle Roots of Entry Blocks

2) **Entry Block Layer** -- Organizes references to Entries

3) **Entries** -- Contains an Application's raw data or a hash of its private data

4) **Chains** -- Grouping of Entries specific to an Application

Directory Layer: How the Directory Layer Organizes Merkle Roots



The Directory layer is the first level of hierarchy in the Factom system. It defines which Entry ChainIDs have been updated during the time period covered by a Directory Block. (ChainIDs identify the user's Chain of Entries; the generation of the ChainID is discussed later.) It mainly consists of a list pairing a ChainID and the Merkle root of the Entry Block containing data for that ChainID.

Each Entry Block referenced in the Directory Block takes up 64 bytes (two 32 byte hashes, the ChainID and the Merkle root of the Entry Block). A million such Entries would result in a set of Directory Blocks roughly 64 MB in size. If the average Entry Block had 5 Entries, 64 MB of Directory Blocks would provide the high level management of 5 million distinct Entries.

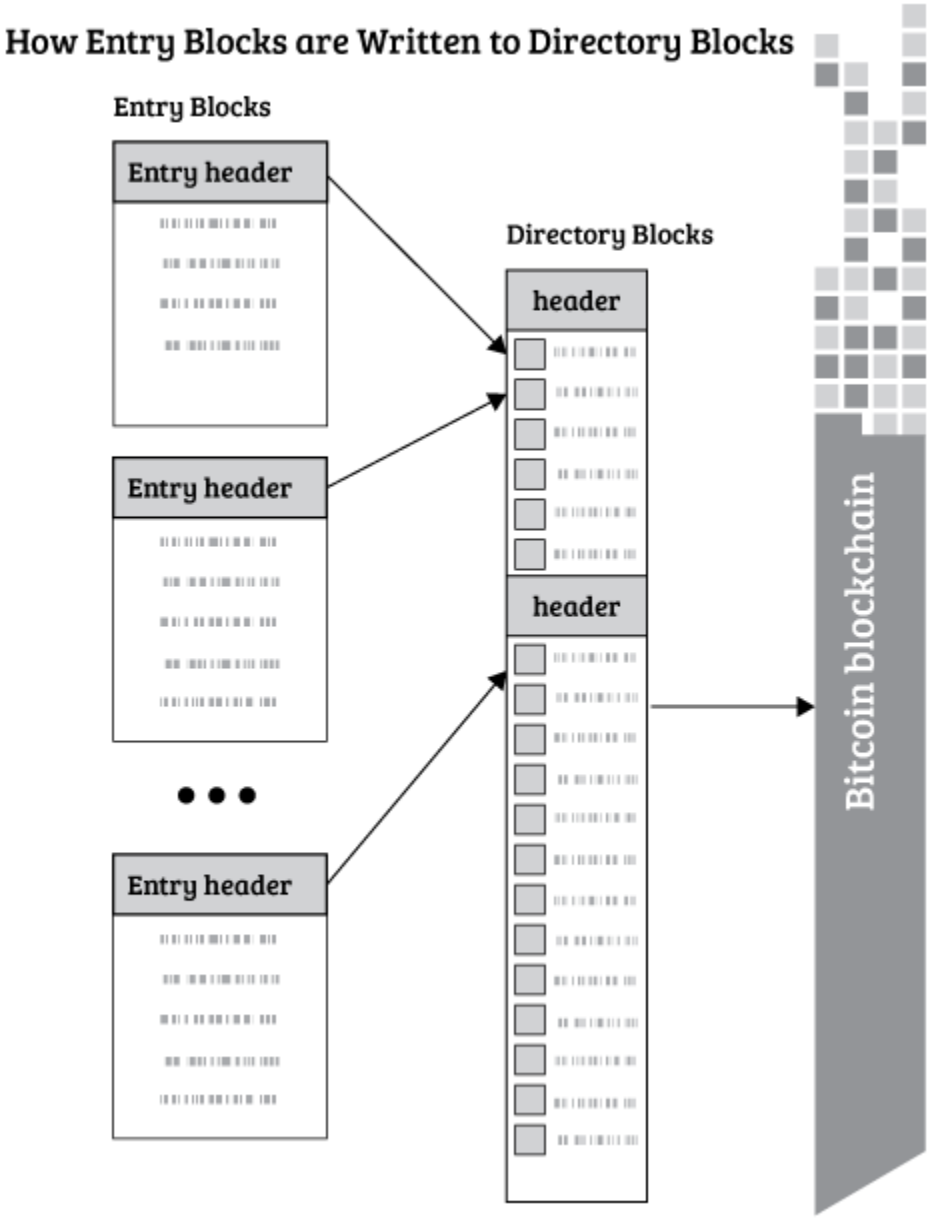
If an Application only has the Directory Blocks, it can find Entry Blocks it is interested in without downloading every Entry Block. An individual Application would only be interested in a small subset of ChainIDs being tracked by Factom. This greatly limits the amount of bandwidth an individual client would need to use with Factom as their system of record. For example, an Application monitoring real estate transfers could safely ignore video camera security logs.

Factom servers collect Merkle roots of Entry Blocks and package them into a Directory Block. Ten successive Directory Blocks are hashed via a Merkle tree, and the Merkle root is recorded into the Bitcoin blockchain. This allows the most minimum expansion of the blockchain, and still allows the ledger to be secured by the Bitcoin hash power. The process of adding the Merkle root into the Bitcoin blockchain we referred to as “anchoring”. See the section “Appendix: Timestamping into Bitcoin” for further details.

Data entered into Directory Blocks is the most expensive, from a bandwidth and storage perspective. All users of Factom wishing to find data in their Chains need the full set of Directory Blocks starting from when their Chain began.

Activities that increase the Directory Block size include the creation and first update of individual Chains. These activities externalize costs of Applications attempting finer-grained organization. The Applications must be required to expend more Entry Credits than a simple Entry would necessitate to discourage bloating the Directory Blocks.

Entry Block Layer: How the Entry Block Layer Organizes Hashes and Data



Entry Blocks are the second level of hierarchy in the system. Individual Applications will pay attention to various ChainIDs. Entry Blocks are the place where an Application looking for Entries can expand its search from a ChainID to discover all possibly relevant Entries.

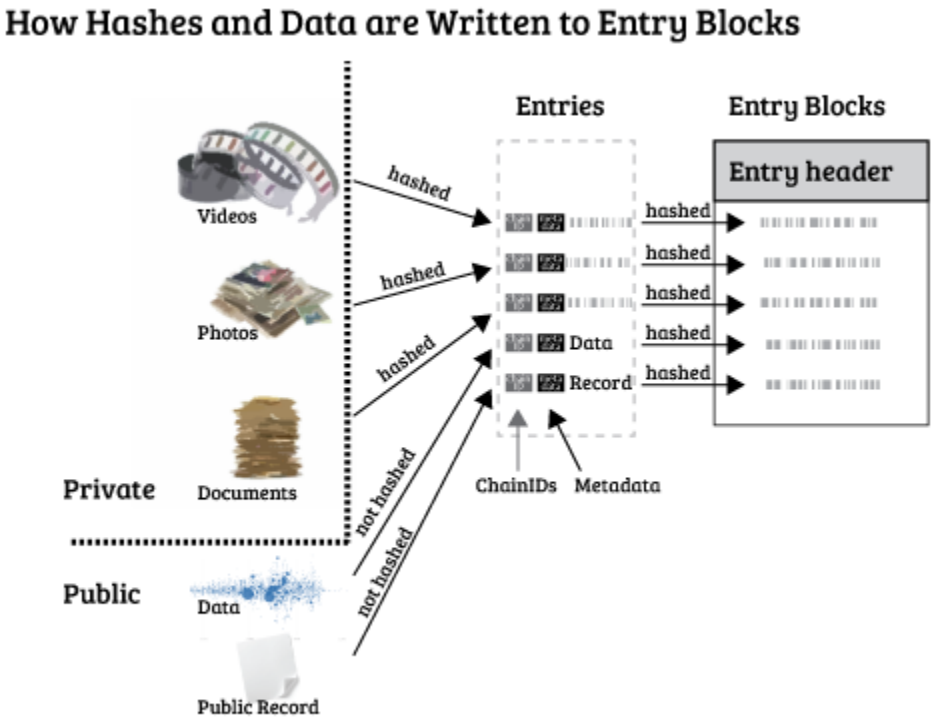
There is one Entry Block for each updated ChainID per Directory Block. The Entry Blocks contain hashes of individual Entries. The hashes of Entries both prove the existence of the data and give a key to find the Entries in the Distributed Hash Table (DHT) network. (See the section “The Factom Peer-to-Peer Network” for more detail.)

The Entry Blocks encompass the full extent of possible Entries related to a ChainID. If an Entry is not referred to in an Entry Block, it can be assumed not to exist. This allows an Application to prove a negative, as described in the section Security and Proofs.

The Entry Block intentionally does not contain the Entries themselves. This allows the Entry Blocks to be much smaller than if all the data was grouped together. Separating the Entries from the Entry Blocks also allows for easier auditing of auditors. An auditor can post Entries in a separate chain that approves or rejects Entries in a common chain. The audit can add reasons for rejection in its Entry. If an Application trusts the auditor, they can cross reference that the auditor has approved or rejected every Entry, without knowing what the Entry is. The Application would then only attempt to download the Entries which passed the audit. Multiple auditors could reference the same Entries, and the Entries would only exist once on the Distributed Hash Table (DHT). Entries are expected to be significantly larger than the mere 32 bytes a hash takes up. Lists of things to ignore do not have to have the full object being ignored for an Application to know to ignore it.

An Entry detailing the specifics of a land transfer would be entered into a Chain where land transfers of that type are expected to be found. One or more auditors could then reference the hashes of land transfer in their own Chains, adding cryptographic signatures indicating a pass or fail. The land transfer document would only need to be stored once, and it would be referenced by multiple different Chains.

Entries: How Entries are Created

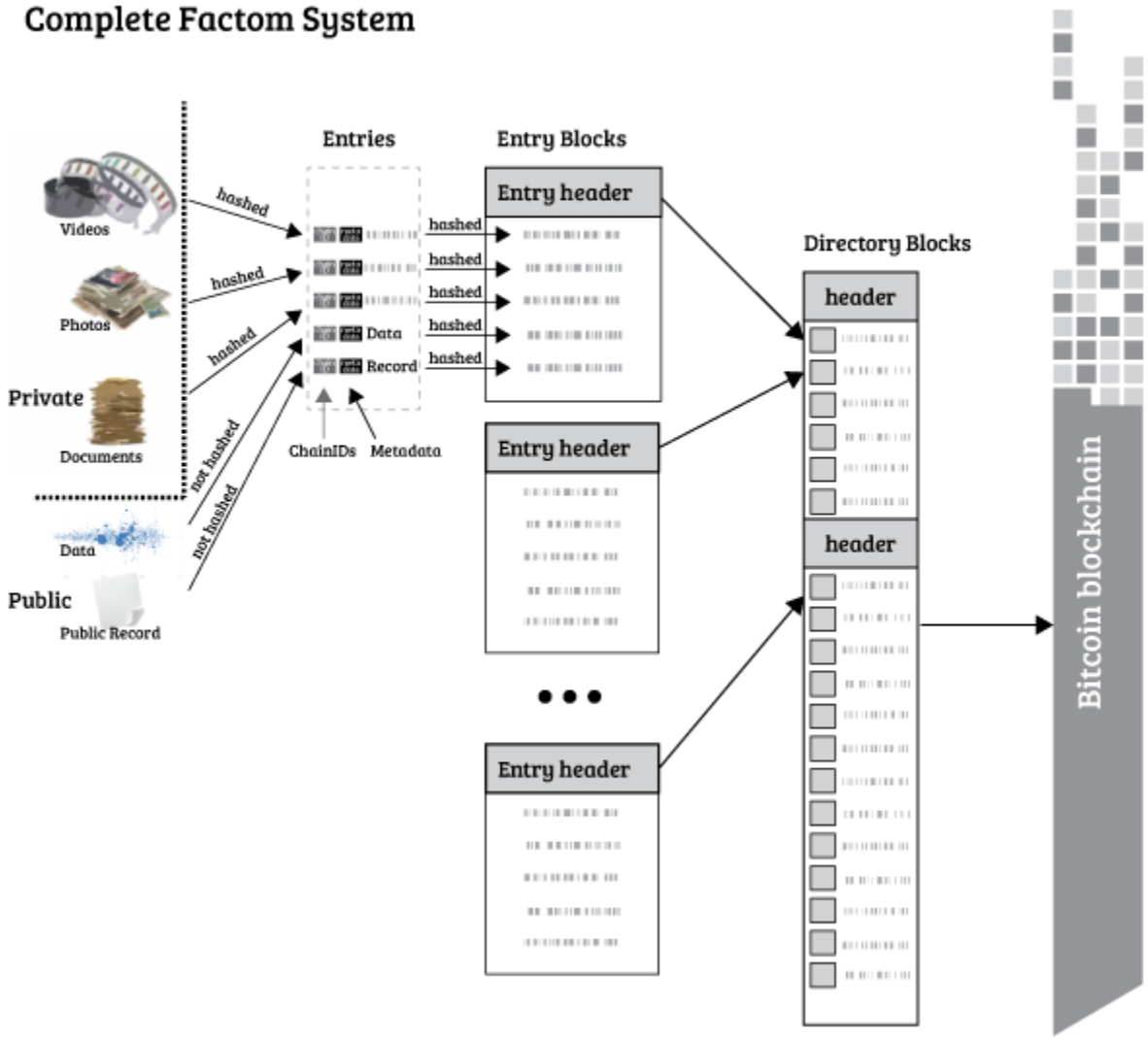


Entries are constructed by users and submitted to Factom. By hashing or encoding information, the user can ensure the privacy of Entries. The Entries can instead be plain text if encoding or obscuring the data isn't necessary. By recording a hash of a document, Factom can provide basic proof of publication. Presenting the document at a later time allows one to create its hash, and compare it to the hash recorded in the past.

There is lots of flexibility in the data that is accepted. It can be something short like a hyperlink. It could also be larger, but not too large, since fees limit the size of the data accepted. This is similar to Bitcoin. Large 100 kB+ Bitcoin transactions are possible, but would need to pay a proportionately greater transaction fee. This size, while gigantic in Bitcoin, would be moderately sized for Factom. Since every Bitcoin full node needs the entire blockchain to fully validate, it needs to stay small. In Factom, only the highest level Directory Blocks are required to fully validate a Chain. If someone is not specifically interested in a Chain's data, they would not download it.

Take a simple example of an uneditable Twitter-like system. A celebrity would craft an Entry as a piece of text. They would then sign it with a private key to show it came from them. Followers of the celebrity would find which Chain they publish in and would monitor it for updates. Any new signed Entries would be recognized by follower's Application software as a tweet. Others could tweet at the celebrity by adding Entries to the celebrity's Chain.

Chains: How Entries are Organized into Chains



Chains in Factom are sequences of Entries that reflect the events relevant to an Application. These sequences are at the heart of Bitcoin 2.0. Chains document these event sequences and provide an audit trail recording that an event sequence occurred. With the addition of cryptographic signatures, those events would be proof they originated from a known source.

Chains are logical interpretations of data placed inside Directory Blocks and Entry Blocks. The Directory Blocks indicate which Chains are updated, and the Entry Blocks indicate which Entries have been added to the Chain. This is somewhat analogous to how Bitcoin full clients maintain a local idea of the UTXO (Unspent Transaction Output) set. The UTXO set is not (currently) in the blockchain itself, but is interpreted by the full client.

The Factom Peer-to-Peer Network

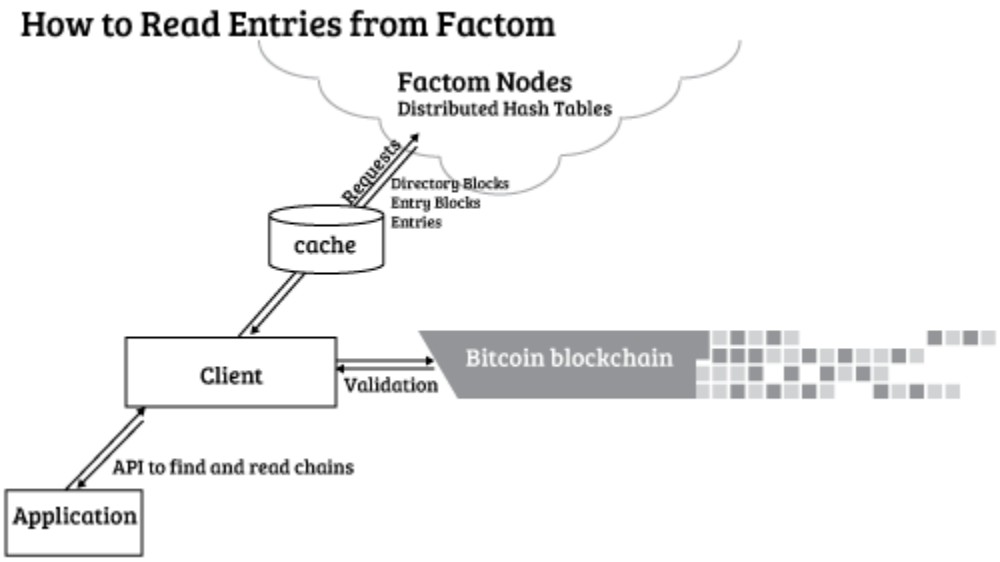
Factom will have a peer-to-peer (P2P) network which accomplishes two goals: communication and data preservation.

Factom Peer-to-Peer Communications

Factom will have a P2P network very similar to Bitcoin's. It will consist of full nodes which have all the Factom data. The full nodes create a mesh network which will flood fill valid data throughout the network. The Federated servers would be full nodes, but not all full nodes are Federated or Audit servers. This is very much like Bitcoin, where miners are full nodes, but not all full nodes are miners. This will limit the ability to DDOS the Federated servers individually. They can connect anywhere inside the network to acquire the data needed to build the data structures. As the servers are coming to consensus and disseminate their signed data, they would publish the data over the P2P network. The P2P flood filling also limits the ability of Federated servers to censor based on IP addresses, since valid traffic is mixed together by the nodes they connect to. It also helps to prevent censorship, since the Audit servers can see the Entries which should be included in the Entry Blocks. The Audit servers have an incentive to bring bad behavior to light, so they can be voted up to a Federated server.

Data Preservation and Dissemination

Factom data structures (Directory Blocks, Entry Blocks, Entries) are needed for Factom to be useful. They are public and will be preserved in two places. The Federated and Audit servers need to maintain this data to make correct decisions about adding new Entries. Since they have this data, they can provide it as a service, as part of being a full node. There will also be partial nodes, which share only part of the Factom dataset. The partial nodes could share only the data which is relevant to their specific application. Peer discovery for the partial nodes is handled by a Distributed Hash Table (DHT).



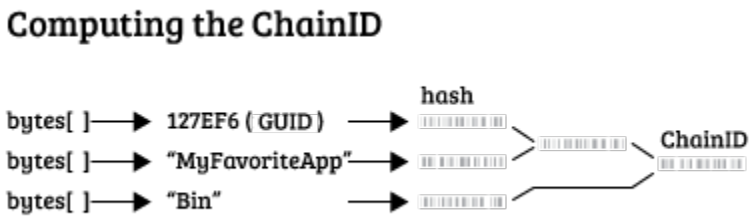
This setup allows for efficient peer distribution of data even if the entire Factom dataset grows to unwieldy sizes. The DHT also allows the data to be preserved independent of any Federated servers or full nodes. Even if all the full nodes were removed from the internet, the data would still be shared by a more numerous set of parties interested in specific subsets of the data.

A Deeper Discussion of Factom

How to Name Factom Chains

Factom groups all Entries under a ChainID. The ChainID is computed from a Chain Name. The ChainID is a hash of the Chain Name. The Chain Name is a byte array arbitrarily long in length. See figure below. Since the conversion from Chain Name to ChainID is a hash operation, it is a simple process. Deriving a Chain Name from a ChainID is not simple, so a lookup table would be needed.

The user must provide a Chain Name, so that the ChainID can be shown to be a hash of something. This prevents unhashed data from being a ChainID, which is stored all the way up to the Directory Blocks. This convention eliminates insertion of obscene plaintext in the block structure.



The Chain Name is fairly arbitrary. It could be a random number, a string of text, or a public key. An individual Application could derive meaning from different Chain Names.

One possible convention would be to use human readable text for the Chain Name. This would allow for the structuring of Chains in a logical hierarchy, even though Chains are not hierarchical by nature. Users can even use the same naming conventions, but by making simple modifications, ensure that there are no accidental intersections between their Chains and other Chains. Consider the following path:

```
MyFavoriteApp/bin
```

Where the slash is a convention for another level of hierarchy. The slash separating ASCII strings "`MyFavoriteApp`" and "`bin`" represents transitioning to a deeper level. These two strings must be converted to bytes, and there are many options for doing so. The strings could be encoded in UTF-16, UTF-32, ASCII, or even something like IBM's EPCDIC. Each of these encodings would result in entirely different ChainIDs for the same string, since the computation of the ChainID is done from the bytes. Furthermore, the application could utilize a Globally Unique IDentifier (GUID) number as the first byte array in their naming convention. This would eliminate overlap of one Application's ChainID "space" with another, at the expense of just a few more bytes in the Chain creation.

Using Factoids to Purchase Entry Credits

Factoids are the main internal scarcity token used to moderate and reward the system actors. The right to put Entries into Factom is represented by Entry Credits. Factom separates the two value-holding mechanisms, as they serve different purposes. Factoids can be converted into Entry Credits, but not vice versa.

Factoids are implemented in much the same way Bitcoin is implemented, allowing for multi-signature security, multiple inputs, multiple outputs, etc. Factoid transactions are managed on a special Factoid Chain. This Factoid Chain is handled more restrictively than other Chains. Entries in the Factoid Chain must be valid Factoid transactions, or the Factom Servers will reject the Entries.

Factoids are included into the protocol to completely decentralize Factom, and to reduce bloat and spam in both Factom and Bitcoin. Factoids can be converted to Entry Credits in the protocol, and payed out to Factom servers from the protocol. They also help to bind consensus. If consensus is lost, then the Factoids will fall in value.

The conversion of a Factoid to Entry Credits will be done via a special purchase transaction on the Factoid Chain. This purchase transaction will include:

- An Output directing a Factoid amount to be converted
- The public key that is to receive the Entry Credits

The Entry Credits, once purchased, cannot be transferred to another public key. They can only be used to pay for Entries or vote for Federated servers. This greatly reduces their value to thieves, since they cannot be resold. Entry Credit private keys can be held in low security areas with minimal risk.

Using Entry Credits to Purchase Entries

Adding Entries into Factom requires giving up a scarce resource. That resource is Entry Credits, which are derived from Factoids. Adding Entries to Factom is a two step process. First the Entry is paid for (committed). The payment accomplishes two things. It decrements the Entry Credits associated with a user's public key. In the same operation, the hash of the Entry is specified. After the Entry is paid for, the server will wait for the unhashed Entry and include it once seen (revealed).

1. Pay for Entry
    - Decrement Entry Credits owned by a user
    - User specifies hash of Entry in payment
2. Insert Entry
    - User publishes Entry for inclusion in Entry Block

There are many benefits of this two step process. One benefit is to separate the payment overhead from the recorded data. Future users will not be forced to download the data generated by payment minutia. They only need to download the minimum data to validate their system. It allows users to safely and easily ignore the payment information.

The other benefit is for censorship resistance. By committing to accept an Entry before knowing the content makes censorship by the Factom servers obvious. Adam Back has advocated for a similar mechanism for Bitcoin in a post titled "blind symmetric commitment for stronger byzantine voting resilience". If a user or Audit server can show an Entry which has been properly been paid for, but none of the Federated servers are accepting it, then the censorship is provable.

The transactions deducting Entry Credits will be recorded in a special Chain, similar to the Factoid Chain. The Federated servers will only fill the Chain with valid Entry Credit transactions.

Setting the Cost of Entries with a Central Server Oracle

The conversion rate of Factoids to Entry Credits will be determined by an oracle. In computer systems, oracles are processes that provide information to a system that cannot be verified or validated by the system. The oracle in this case maintains the conversion rate of Factoids to Factom Entry Credits at a cost 1/10 to 1/100 the cost of a comparably sized Bitcoin transaction.

Initially the conversion rate oracle will be implemented centrally. When we have recruited enough exchanges to record their Factoid exchange rates and trading volumes into a Factom Chain, we will establish a decentralized computation of the exchange rate. The qualifying source Chains (as maintained by exchanges) will be determined by the conversion rate of Factoids at various exchanges, weighted by volume.

After purchase, the Entry Credit Chain(s) will allocate the Entry Credits to the appropriate public key. That Entry will look like:

- Public Key
- Number of Entry Credits purchased

Using Factom without Factoids

Many users of Factom may not want a wallet, and will not want to hold any cryptocurrency asset. But they will want to create their Chains (ledgers) and add their Entries. Factom's two step recording process allows for the separation of Factoids, Factom's tradeable token, from the right to post Entries to Factom, represented by Entry Credits. Servers and other recipients of Factom Tokens can sell Entry Credits to customers for payment via Bitcoin, conventional credit card payments, etc. The user would provide a public key to hold the Entry Credits. The seller would convert the appropriate amount of Factoids to Entry Credits and assign those rights to the user's public key. Users could thus buy Entries Credits for Factom without ever owning the Factoids that drive the Factom servers.

From a regulation point of view, this is powerful. The servers earn Factoids from the protocol. The only parties to that transaction are the server and the protocol. Then the server sells Entry Credits to users, who eventually return Factoids to the rest of the system. Entry Credits are non transferable, so the user cannot assign them to another user's public key, and selling private keys isn't practical or useful. In neither transaction is a tradable token (the Factoid) transferred between two parties.

First Entries on Entry Chains: Support for Homesteading

Factom preserves the first Entry of every Chain to the user who first claims it with a payment. A convention can be adopted where the first Entry documents the auditing rules for the chain, including a URL to documentation, or the rules in Text, or a hash of an audit program for the chain. This convention can be understood as homesteading.

Through a set of revealed secrets, Factom can ensure that the user who creates the Chain determines the contents of the first Entry. (See Appendix 2 for the Man in the Middle Attack)

The call to commit a new Chain requires a payment of 10 Entry Credits + 1 Credit per 1,024 bytes. The higher price is to reduce creation of new Chains, which is an externalized cost. The submission to start a Chain includes three parameters:

- The hash of the ChainID
- The hash of (the ChainID + the Entry Hash)
- The Entry Hash

These three hashes are placed in the Entry Credit Chain.

Once the Chain commit has been accepted, the reveal can occur, and the ChainID is now public. For the Chain creation reveal to be valid, the reveal must supply the Chain Name that hashes to produce the ChainID. The Entry must hash to produce the Entry Hash. Lastly, the concatenation of the ChainID with the Entry Hash must match the result in the commit.

If any of these conditions are not met, Factom servers do not record the first Entry and the Chain is not created.

Conclusion

Factom is a distributed, autonomous layer residing on top of the Bitcoin blockchain. The goal of Factom is to provide the power of Bitcoin's blockchain to a nearly unlimited range of Applications and uses. Further, Factom is architected such that its users do not need any cryptocurrency whatsoever.

A distributed, immutable ledger is the radical, foundational, and unprecedented technology represented by the Bitcoin blockchain. The dream of many is to extend the honesty inherent to an immutable ledger validated by math to chaotic, real-world interactions. By allowing the construction of unbounded ledgers backed by the blockchain, Factom extends the benefits of the blockchain to the real world.

Bibliography

“Bitcoin: A Peer-to-Peer Electronic Cash System” Nakamoto, Satoshi. Web. 16 Nov. 2014 https://bitcoin.org/bitcoin.pdf

"Can Blocks Remain Capped to 1MB Forever?" Transactions. Web. 15 Nov. 2014. http://bitcoin.stackexchange.com/questions/18101/can-blocks-remain-capped-to-1mb-forever

"Thin Client Security." - *Bitcoin*. Web. 15 Nov. 2014. https://en.bitcoin.it/wiki/Thin_Client_Security#Simplified_Payment_Verification_.28SPV.29

"Evidence of Absence." Wikipedia. Wikimedia Foundation, 11 July 2014. Web. 15 Nov. 2014. http://en.wikipedia.org/wiki/Evidence_of_absence

"Recording (real Estate)." Wikipedia. Wikimedia Foundation, 14 Nov. 2014. Web. 15 Nov. 2014. http://en.wikipedia.org/wiki/Recording_(real_estate)

"Secure Property Titles with Owner Authority." Secure Property Titles with Owner Authority. Web. 15 Nov. 2014. http://szabo.best.vwh.net/securetitle.html

"Patent US4309569 - Method of Providing Digital Signatures." Google Books. Web. 15 Nov. 2014. http://www.google.com/patents/US4309569

"Block Timestamp." - Bitcoin. Web. 15 Nov. 2014. https://en.bitcoin.it/wiki/Block_timestamp

"OP_RETURN and the Future of Bitcoin." - Bitzuma. Web. 15 Nov. 2014. http://bitzuma.com/posts/op-return-and-the-future-of-bitcoin/

"Goblin/chronobit." GitHub. Web. 15 Nov. 2014. https://github.com/goblin/chronobit

"How Can One Embed Custom Data in Block Headers?" Mining. Web. 15 Nov. 2014. http://bitcoin.stackexchange.com/questions/18/how-can-one-embed-custom-data-in-block-headers

"Headers-First Synchronization Coming Soon to Bitcoin Core - CryptoCoinsNews." CryptoCoinsNews. Web. 15 Nov. 2014. https://www.cryptocoinsnews.com/headers-first-synchronization-coming-soon-bitcoin-core/

“Enabling Blockchain Innovations with Pegged Sidechains - Block Stream ” Web. 15 Nov. 2014. http://www.blockstream.com/sidechains.pdf

"[Bitcoin-development] 2-way pegging (Re: is there a way to do bitcoin-staging?)" / Mailing Lists. Web. 27 May. 2014. http://sourceforge.net/p/bitcoin/mailman/message/32108143/.

"Could the Bitcoin Network Be Used as an Ultrasecure Notary Service?" Computerworld. Accessed 27 May. 2014. http://www.computerworld.com/s/article/9239513/Could_the_Bitcoin_network_be_used_as_an_ultrasecure_Notary_service_.

"Proof of Existence." Proof of Existence. Web. 27 May. 2014. http://www.proofofexistence.com/.

"Virtual-Notary." Virtual-Notary. Web. May 27. 2014. http://virtual-notary.org/.

“Commitment Scheme” Web. 16 November. 2014. http://en.wikipedia.org/wiki/Commitment_scheme

“Foundations of Cryptography: Volume 1, Basic Tools, (draft available from author’s site).” Cambridge University Press. ISBN 0-521-79172-3. 16 November. 2014. (see also http://www.wisdom.weizmann.ac.il/~oded/foc-book.html) :224

“Real-World Sybil Attacks in BitTorrent Mainline DHT Wang Liang. Jussi Kangasharju. University of Helsinki. Web. 17 Nov. 2014. http://www.cs.helsinki.fi/u/lxwang/publications/security.pdf

“Sybil-resident DHT routing” University of Cambridge. Danezis George. Chris Lesniewski-Laas. Kaashoek M. Frans. Anderson Ross. Web. 17 Nov. 2014. https://www.cl.cam.ac.uk/~rja14/Papers/sybildht.pdf

“A Sybil-proof one-drop DHT” Lesniewski-Laas Chris. Web. 17 Nov. 2014. http://pdos.csail.mit.edu/papers/sybil-dht-socialnets08.pdf

“Art Provenance: What It Is and How to Verify It” Web. 17 Nov. 2014. http://www.artbusiness.com/provwarn.html

“Equine Appraisal: The Value of our Horses” Web. 17 Nov. 2014. http://www.hgexperts.com/article.asp?id=7366

“Proof of work” Web. 17 Nov. 2014. https://en.bitcoin.it/wiki/Proof_of_work

“Why one time passwords using nested hash chain are not used” Web. 17 Nov. 2014. http://security.stackexchange.com/questions/35135/why-one-time-passwords-using-nested-hash-chain-are-not-used

”Proving Your Bitcoin Reserves” Web. 17 Nov. 2014.
https://iwilcox.me.uk/2014/proving-bitcoin-reserves

“Distributed Consensus from Proof of Stake is Impossible” Web. 17 Nov. 2014.
https://download.wpsoftware.net/bitcoin/pos.pdf

Appendix 1: Audit Application Examples: What Could Be Useful Today?

How to Create Useful Applications Today Using the Factom Protocol

‘Application’ is a generic term for user-side software that reads from and/or writes to the Factom system. It could be software with a human interface, or could be completely automated. The Application is interested in the data organized by the Chains it needs.

Applications are possibly Distributed Applications (DApps) interacting with Factom to provide additional services. For example, one might imagine a trading engine that processes transactions very fast, with very accurate timestamping. Such an Application may nonetheless stream transactions out into Factom chains to document and secure the ledger for the engine. Such a mechanism could provide real-time cryptographic proof of process, of reserves, and of communications.

Let us explore two separate applications that could have immediate demand in the current Bitcoin ecosystem.

Let us see how to implement a secure and distributed log platform. Log analysis is a complex task. Additionally, logs tend to be easily forgeable and also heterogeneous as they are produced by each system independently and stored in a variety of media (files, databases, cloud services etc.). With Factom and a few uniquely designed crypto-audit tools an entities log analysis can become safer, simpler, and much more powerful. Let’s see this with an example. Suppose a Bank (B), a Payment Provider (PP), and a Bitcoin company (BC) are interacting together as follows:

1 - The User goes to the BC website and wants to buy some bitcoins
2 - He asks for a quote, which is valid for 5 minutes
3 - Then he is redirected to the PP website
4 - Then the PP connects with the B platform so that the money of the user account is debited
5 - B notifies PP that the user account has been debited
6 - PP notifies BC
7 - BC sends the bitcoins to the user

This is the normal scenario for many fixed-rate Bitcoin exchanges globally. But assume now that for some reason the BC receives the payment notification 4 hours after the user transfers via the PP. Who is faulty? The User? The Bank? The Payment Provider? What if a similar payment problem happened for hundreds or thousands of payments over a period of days or weeks before the issue was identified and resolved? Who is “provably” liable for those loses/damages?

With current techniques a manual auditing of logs would be necessary and would probably require legal authorizations. With Factom and the right audit applications, it would be trivial to detect where the problem came from, and also make the changing of records impossible post-issue. Basically, every system (BB, PP, BC) will publish their relevant traces in the secure broadcast channel (Factom) in real time.

Here's another example of how Factom will be useful for Bitcoin exchanges audits. The so-called "Proof of Solvency" method for conducting Bitcoin exchange audits is a growing and important trend. However, there are significant weaknesses to this approach only solved by having the Factom secure broadcast channel functioning properly.

In the Merkle tree approach for Solvency Proofs suggested by the Maxwell-Todd proposal, users must manually report that their balances (user's leaf) have been correctly incorporated in the liability declaration of the Financial Institution (FI) (the Merkle hash of the FI's database of user balances). The proposed solution works if enough users verify that their account was included in the tree, and in a case where their account is not included, it's assumed that this instance would be reported. One potential risk with this process is that an exchange database owner could produce a hash that is not the true representation of the database at all; the exchange hashes an incomplete database which would reduce its apparent liabilities to customers, thereby making them appear solvent to a verifying party. Here are some scenarios where a fraudulent exchange could easily exclude accounts:

- "Colluding Whales" Attack: There is evidence that large Bitcoin traders are operating on various exchanges and moving markets significantly. Such traders need to have capital reserves at the largest exchanges to quickly execute orders. Often, traders choose exchanges that they "trust". In this way they can be assured that should a hack or liquidity issue arise, they have priority to get their money out first. In this case, the exchange and trader could collude to remove the whales account balance from the database before it's hashed. An exchange's top 10 whales could easily represent 5 to 20% of an exchanges liabilities, so colluding with just a few of them could have a significant impact.

- "Site Manipulation" Attack: To date, each Proof of Solvency audit has reported (the hash tree) on the institution's website. This gives no guarantee at all to users, since a malicious exchange could publish different states/balances to different groups of users, or retroactively change the state. Thus it is fundamental to publish this data through Factom's secure broadcast channel, and publish it frequently.

The second attack is obviously solved by using Factom, while the first is not so obvious. As this paper doesn't focus on the mechanics of exchanges audits, we won't delve in the nitty-gritty details. However, the basic concept is that by having frequent time-stamped copies of the exchanges database Merkle hash, one could detect the inclusion or exclusive of large balances before or after audits. Then, the auditor could simply look into those large inclusions or exclusions, manually. Remember, the trader will ultimately need to get his money on or off the exchange at some point, and that'll show up in either the bank history or the Bitcoin transfer history.

There are established process for detecting such fraudulent tactics in the traditional audit industry; however, it all starts with having accurate, verifiable, immutable time-series of the information in question.

Appendix 2: Attacks on Factom

Denial of Service from Spam

Since Factom is an open system, any user can put Entries into almost any Chain. Bitcoin has a similar phenomenon. In order for an Application to reject those transactions, the Application would first need to download and process them. A large number of bogus Entries could slow down the initial processing of the Application's transactions. This threat is mitigated by an attacker needing to spend money (resources) to carry it out. This is similar to Adam Back's Hashcash solution to email spam.

Audits are another useful tool against spam, if the application is willing to trade off security versus convenience. Auditors could post "ignore" lists on the same chain, or create their own audit chains with those lists. An auditor could use a profile chain to develop their reputation, which would also allow review by other auditors. If any auditor made a bad call, it would be easily verifiable and the record of it would be permanent. Some validity processing is gray, in the sense that opinions may vary. Solving that problem would be implementation specific.

Sybil Attack of the DHT

Distributed Hash Tables in general are particularly susceptible to sybil attacks. An attacker could create many peers which make it difficult for honest nodes to communicate. In a simplistic DHT architecture, attackers can isolate a required piece of data from honest nodes. Sybil attacks have been observed on the BitTorrent network routing table. The paper "Real-World Sybil Attacks in BitTorrent Mainline DHT" detail these attacks. Fighting this type of attack is an active topic in academic research. One mitigation technique uses complex lookup techniques to find honest nodes among the sybils, studied in "Sybil-resistant DHT routing". Some sybil mitigation techniques rely on a web-of-trust by adding a social network to the routing table, as explored in "A Sybil-proof one-hop DHT". Factom will rely on the latest academic and open-source research in this topic to secure its DHT.

Man in the Middle

There is a convention where the first Entry in a Chain can carry more significance than later Entries. An attacker with a privileged position in the network can front run the original committer by intercepting and getting an attack Chain creation request to the server prior to the authentic request. This could be done with merely a faster connection to the relevant Factom Server.

Factom uses a delay in claiming the Chain to prevent this attack, similar to Namecoin's solution along with a set of revealed secrets to ensure the first one to pay for a ChainID has the right to create the first Entry in that Chain. The payment includes these three pieces of data:

- Hash of the ChainID
- Hash of the (ChainID+Hash of the Entry)
- Hash of the Entry

An attacker could see the attempt to purchase, but will not know the ChainID. Because they do not know the ChainID, they cannot create a valid second parameter.

Once this payment is recorded, the user must wait for the Directory Block holding the new Chain payment to finish, and the next Directory Block, a delay of one to two minutes.

Now the user reveals:

- The Chain Name (series of byte arrays) that hash to the ChainID
- The Entry that matches the Entry in the payment

The attacker cannot create a false payment, and insert their first entry without revealing the ChainID, which will match the earlier payment and invalidate their transactions.

Dictionary Attack

In this case, the attacker runs through all the Chain Names deemed to be possible or desirable and creates their ChainIDs, and the hashes of those ChainIDs. Then they watch for someone trying to create those Chains.

Now the attacker can front run on a match. Because on a match, they know the ChainID, so they can construct a proper, but malicious Entry of their own, create the proper Chain payment and submit it rather than the users payment.

If the attacker gets ahead of the user, then they will win. The defense against a dictionary attack is to avoid common name spaces and to submit your payment to multiple, long standing nodes in the network.

In Factom, the flexibility of defining the Chain namespace makes efforts to hog the namespace ineffective.

Denial of Chain

In this case, an attacker predicts the Chains an Application wants based on the patterns of Chain usage by an Application. Then they pay for those Chains, but never submit the Entry. Because Factom will wait for the Entry submission before allowing any other Entries in the Chain, the Chain is effectively locked until the Entry that never comes is revealed.

Factom converts Denial to Delay. After 10 minutes, if no Entry for a Chain creation is revealed, the payment expires. Now the attacker must submit payment after payment to keep the Chain locked. In the mean time, the Application can use a naming process that skips over the locked Chain to use a Name with a slight modification.

Fraudulent Servers

All Entries in Factom require signatures from the users, or must match a hash that has been signed by the users. This means that fraudulent Federated servers in the Federation pool have very limited attacks they can make on the protocol. Invalid Entries do not validate, and upon broadcasting an invalid Entry, the honest Federated Servers will immediately broadcast a Server Fault Message (SFM) on the fraudulent server. If a majority detect a fault, the faulty server is removed. As long as the majority do not collude, then the protocol will remain honest. Any Federated server that failed detect the fault likewise risks losing its support from Factom users, and dropping from the Federated server pool.

Federated servers can delay recording of Entry payments. But because Entry payments are submitted via a distributed set of Factom Nodes, delaying of Entry payments will be noted. Users may withdraw support from servers without reasonable performance compared to the rest of the network.

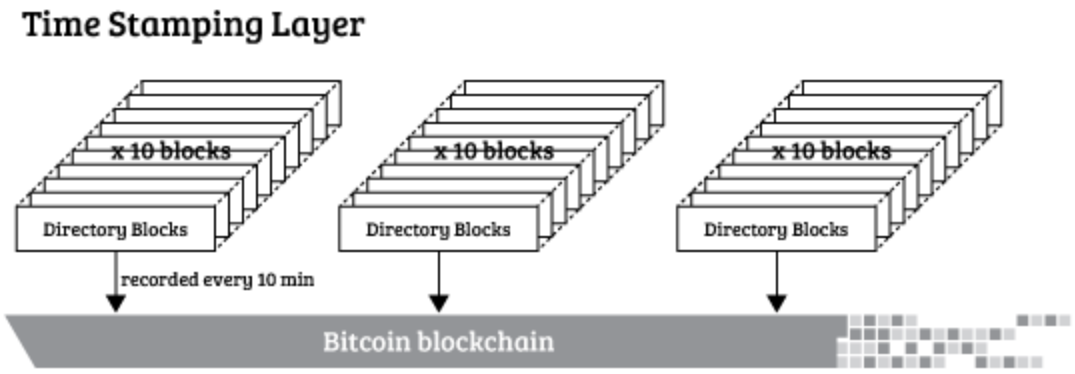
Federated servers can delay the recording of Entries. Here the payment is accepted (generally by another server) fairly quickly. But for one reason or another, a Federated server refuses to record the Entry. In the next minute, responsibility for that Chain will shift to another server. As long as most servers are honest, the Entry will be recorded. Then the data over time will show that a server is delaying Entries. This will cause them potentially to lose support.

Federated servers can at any point send false messages. The other Federated servers then would issue a SFR on the on the rogue server when those messages didn’t make sense. A majority of the servers issuing an SFR would boot the rogue server, then the network would ignore their messages and not forward them on.

Federated servers can refuse to accept valid Entry payment messages based on the public address, under the assumption that the public address is associated with some party. Again, assuming a majority of servers are honest, the payment will be accepted when the control shifts to an honest server. Furthermore, nodes watching will see the delay, and perhaps a pattern of delays, and support will be lost for the misbehaving servers.

Appendix 3: Timestamping into Bitcoin

How the Factom Timestamping Mechanism Secures Transactions in the Blockchain



Factom data is timestamped and made irreversible by the Bitcoin network. A user's data is as secure as any other Bitcoin transaction, once published to the Bitcoin blockchain. A compact proof of publication is possible for any data entered into the Factom system.

Data is organized into block structures, the highest level being Directory Blocks, which are combined via Merkle trees. Every 10 minutes, the data set is frozen and submitted to the Bitcoin network. Since Bitcoin has an unpredictable block time, there may be more or fewer than one Factom timestamp per Bitcoin block.

Bitcoin internal header block times themselves have a fluid idea of time. They have a 2 hour possible drift from reality. Factom will provide its own internal timestamps, adhering with standard time systems.

The user data ordering will be assigned when received at the Federated servers. Factom organizes the submitted Entry references into sets of blocks. The block time for Factom is one minute. On closing, the Federated Server network generates consensus and the Entries that are part of that block structure are timestamped to that minute.

As a general note, the data could have existed long before it was timestamped. An Application running on top of Factom could provide finer and more accurate timestamping services prior to Entries being recorded in Factom. The Factom timestamp only proves the data did not originate after the Factom timestamp.

The Merkle root for 10 Directory Blocks is entered into the Bitcoin blockchain with a spending transaction. The spend includes an output with an OP_RETURN. We refer to this as “anchoring” the Directory Blocks to the Bitcoin blockchain. This method is the least damaging to the Bitcoin network of the various ways to timestamp data.

Two possible alternatives to the OP_RETURN data in the blockchain is anchored to the P2Pool headers (as in chronobit) or in the Bitcoin block header coinbase. The P2Pool headers would require several hours of mining to find a block which satisfies the P2Pool rules, and the added complexity to the Factom protocol would not be worth the benefits. Including the Merkle root into the coinbase of a block would require cooperation with miners, above and beyond the transaction processing they are already doing. The coinbase entry would still need to have a crypto signature from the Factom system, so would not save on much space relative to a signed transaction. Not only would it be un-prunable, it would also be included in headers-first downloads, affecting Bitcoin SPV clients.

The first two bytes of the available 40 in the anchor will be a designator tag (2 bytes with the value “Fa”). The Factom anchor (32 bytes) is concatenated onto the tag, then the block height is added (up to 6 bytes, allowing for ~500,000 years with 1 minute blocks). The designator tag indicates the transaction could be a Factom anchor. Other qualifiers are required, but the tag and Factom block height eliminates most of the OP_RETURN transactions that would otherwise need to be inspected.

The block height in the OP_RETURN helps to fix the order in those cases where the Bitcoin blockchain records the anchors out of order.

The anchored data is the Merkle root of list containing the 10 individual Directory Blocks’ Merkle roots. Querying the Factom DHT for the anchored data will return the list of 10 Directory Block Merkle roots, which are in turn DHT keys for the Directory Blocks themselves.

The Merkle root timestamp will be entered into the Bitcoin blockchain by one of the Federated servers. The server delegated to timestamp the federation’s collected data creates a Bitcoin transaction. The transaction will be broadcast to the Bitcoin network, and be included in a Bitcoin block. Bitcoin transactions that look like a Factom anchor, but are not spent from an address known as a Factom server would either be junk, or an attempt to fork Factom. Most users/applications would ignore such anchors.

Bitcoin blocks are generated with a statistical process, and as such their timing cannot be predicted. This means that the anchors are only roughly time-bound by the OP_RETURNs inserted into the Bitcoin blockchain, and its timestamping mechanism. The real value of anchoring Factom to Bitcoin is to prevent anyone from generating false Factom histories. Due to bad luck of Bitcoin miners, or delayed inclusion of Factom transactions, the time between when the Factom state is frozen for a particular 10 minute period and when the anchor appears in Bitcoin can vary, perhaps significantly.

Appendix 4: Comparing Factom with Other Blockchain Technologies

How Factom Differs from Bitcoin and Sidechains

Factom is very different from Bitcoin, and in fact very different from any current cryptocurrency project.

Cryptocurrencies like Bitcoin implement a strict, distributed method for the validation of transactions, where anyone can validate each transaction, and the validity of every input into a transaction can be verified. Because each transaction is authorized via cryptographic proof, no transaction can be forged. Each transaction can be checked for validity by verifying signatures of each transaction, and the miners hold each other accountable for only including valid transactions.

The Bitcoin protocol is transactionally complete. In other words, the creation and distribution of Bitcoins through transactions is completely defined within the Bitcoin protocol. Transactions (which specify movement of bitcoin) and block discovery (which move bitcoin via mining fees and provide block rewards) are the only inputs into the Bitcoin Protocol, and nothing leaves the Bitcoin Protocol. In other words, the 21 million bitcoins that will ultimately exist will always and forever exist within the protocol. Pegged sidechains, when implemented, will provide additional movement of bitcoin value outside the blockchain, while the pegged value is in stasis in the blockchain.

The sidechains proposal describes a solution to increase the scalability of Bitcoin by allowing value control to move off the blockchain and onto a sidechain. In the sidechain, many trades can occur. Later, a cryptographic proof (not all the transactions in between) can be recorded in the blockchain which moves the BTC out of stasis in Bitcoin. This proof would have to be available to the Bitcoin miners, but the bulk of the transaction data would be left behind in the sidechain.

Factom is in some sense attempting to increase scalability, but not by enabling more value transactions, but by moving non-BTC transactions off blockchain. This would be transactions that are not primarily intended to transfer Bitcoin value. For example transactions could manage domain name registrations, log security camera footage, track the provenance for art work, and even establish the value of show horses by documenting their history. Some of these do not move a value at all, like transactions establishing a proof of publication.

Sidechains and Factom are both trying to move transactions off the blockchain, but to achieve similar ends via completely different mechanisms. At some point, Factom will integrate with a Bitcoin sidechain in order to take advantage of the atomic swaps from BTC to Factoids.

How Factom is Different from Other Blockchain Technologies

Many different groups are looking to find ways to leverage the Bitcoin approach for managing other sorts of transactions besides tracking bitcoin balances. For example, the trading of assets such as houses or cars can be done digitally using Bitcoin extensions. Even the trading of commodities such as precious metals, futures, or securities might be done via clever encoding and inserting of information into the Bitcoin blockchain.

Efforts to expand Bitcoin to cover these kinds of trades include Colored Coins, Mastercoin, and Counterparty. Some developers choose to build their own cryptocurrency with a more flexible protocol that can handle trades beyond currency. These include Namecoin, Ripple, Etherium, BitShares, NXT, and others.

Open Transactions (OT) uses Cryptographic signatures, signed receipts and proof of balance for users (i.e., a user does not need the transaction history to prove their balance, just the last receipt). In this way, OT can provide the spend of centralized servers without the risk of a centralized server that can alter client balances. Factom is decentralized, and only records Entries. So Factom can record data that would not meet OT's rules. But Factom cannot execute at the rate OT can. Still, we expect that some, but not all users will employ cryptographic techniques similar to OT with their records.

The great advantage to an independent platform over trying to build upon Bitcoin is found in flexibility. The Bitcoin protocol isn't optimized to allow for recording of arbitrary pieces of data, so the "bookkeeping" necessary for non-Bitcoin type transactions isn't necessarily supported by Bitcoin. Furthermore, Bitcoin's Proof of Work (PoW) based consensus method is not a "one size fits all" solution, given that some transactions must resolve much faster than 10 minutes. Ripple and Open Transactions vastly speed up confirmation times by changing the consensus method.

An Application built upon Factom seeks to gain the ability to track assets and implement contracts, by leveraging the blockchain directly. Instead of inserting transactions into the blockchain (viewed as "blockchain bloat" by many), Factom records its Entries within its own structures. At the base level, Factom records what Chains have had Entries added to Factom within the Directory Block time. Scanning these records, Applications can pick out the Chains in which they are interested. Factom records each Chain independently, so Applications can then pull the Chain data they need.

Factom is organized in a way that minimizes connections between user Chains. A Chain in Factom can be validated without any of the information held in other, unrelated Chains. This minimizes the information a Factom user has to maintain to validate the Chains they are interested in.

Appendix 5: Proof of Stake Similarities

Factom Consensus Similarities and Differences from Proof of Stake

The policy and reward mechanism in Factom is similar to Proof of Stake (PoS). Factom differs from most PoS systems in that only a subset of users' stake is recognized. Only value which has been committed to the system has a voting share. The transferable Factoid value does not have a voting share. Only value which has been turned into Entry Credits, which is not transferable, has a say in choosing the Federated servers. This is an attempt to make the servers answerable to the users actively using the service, instead of users with mere potential to use the service. The individual users would delegate their votes to a server. The Federated servers with the top numbers of votes would be responsible for coming to consensus.

Some with a deep understand of Bitcoin have recognized that pure PoS consensus mechanisms are fundamentally flawed. There are two attacks that make pure PoS unworkable. The problems are referred to as "Stake Grinding" and "Nothing at Stake". Although Factom has PoS elements, it does not suffer from these problems.

Stake Grinding

Stake Grinding is a problem where an attacker with a sizable (say 10%), but not majority share can formulate false histories. From some point in history, they can costlessly fork the blockchain, choosing to reorder past transactions such that their stake is always selected to create the subsequent blocks. They would be able to present this alternate version of history as part of an attack to steal value by double spending. Bitcoin solves this problem by strongly linking the information domain, where computers make decisions, with the thermodynamic domain, where humans burn energy. Considerable resources are expended in the thermodynamic domain, and is provable in the information domain. Bitcoin makes forming false histories hugely expensive.

Factom is unable to create alternate histories after the fact, since it is unable to insert transactions into historical Bitcoin blocks. It is also unable to create parallel histories without being detected, since Factom is linked to Bitcoin with known Bitcoin private keys.

Nothing at Stake

The Nothing at Stake problem is more subtle. With a policy disagreement in Bitcoin, miners must choose either one policy or the other. If they choose against the majority, they will be burning lots of electricity without a chance of recouping costs. PoS miners do not face this dilemma. They can hedge their bets and costlessly create forks complying with each side of the policy. They would simultaneously agree with both sides of the disagreement. This would open up the economy to double spend attacks. One of two merchants following different forks will ultimately have that money becomes worthless.

Bitcoin solves this problem by having unintelligent unambiguous automatable rules for selecting the correct fork. In Bitcoin, the correct fork is the one with the most Proof of Work (PoW). Factom will also have unintelligent unambiguous automatable rules to select a correct fork, should one arise.