The FreitX Network Technical Whitepaper

June 23, 2018

Version 0.9

Authors

Makho Kituashvili, Elene Evsia, Nikoloz Jaghiashvili.

***Abstract* - Technology has been playing a significant role in our lives. One definition for technology is "all the knowledge, products, processes, tools, methods and systems employed in the creation of goods or in providing services." This makes technological innovations raise the competitiveness between organizations that depend on supply chain and logistics in the global market. With increasing competitiveness, new challenges arise due to lack of information and assets tractability. Supply Chain Management has a multi-dimensional approach which manages the flow of raw materials and works in progress within the organization and the end product outside the organization till it reaches the hands of the final consumer with a complete emphasis on the customer requirement. This paper introduces three scenarios for solving these challenges using the Blockchain technology. In this work, FreitX Network targets main issues within the supply chain, data, transparency, track & trace, market growth, and resource sharing.**

Vision

The FreitX Network vision is to develop an innovative network for supply chain management. Today, the supply chain environment is changing. Globalisation and changes demanded by stakeholders have influenced the attitudes of supply chain entities. These entities are becoming more professional, showing ever-increasing levels of complexity, and adopting philosophies and management practices with the objective of increasing their levels of competitiveness.

The recent adoption of Internet of Things and Blockchain technologies give promise at better supply chain provenance. Supply chain processes could be effectively transformed to meet the Blockchain architecture. Blockchain facilitates valid and valid measurement of outcomes and performance of fundamental supply chain processes.

The FreitX Network allows us to have a decentralized system with opportunities to start evolving from old, traditional development ways. Within Network, the need for human involvement in the process is minimized, presenting previously vulnerable players with more safety and efficiency. It connects market participants and automates their relationship. Through FreitX Network, the once complicated process can be reduced to a single click on a customer's side, enabling businesses to concentrate on their core practices, and rendering the need for blind trust obsolete.

Executive Summary

Since the rise of the 20th century, the dependency of businesses on supply chains has increased. Supply chains have become a vital business process that can encourage the competitiveness of a business in the market. It has been recently emphasized that the success of a company is based on the interaction between flows of information, materials, workers and capital equipment. Logistics have been playing a significant role in achieving a useful value proposition for the customers. The author mentioned that retail business logistics are essential for creating business value as he defined the logistics as "The integrated management of purchasing, transportation, and storage on the functional basis."

Supply chain management was used for decades to deliver goods. Later, the supply chain management concept changed from focusing just on logistics to become mandatory to track and manage other flows within the supply chain. Such flows include a flow of money and flow of information. There are billions of products being manufactured every day through elaborate systems of supply chains that extend to all parts of the world. However, there are several questions regarding these products. For example, how, when and where these products were originated, manufactured, and used through their life cycle. Even before reaching the end consumer, goods pass through a wide network of retailers, distributors, transporters, inventories, and suppliers that participate in every process in the supply chain (i.e., design, production, delivery, sales, etc...).

Most of these processes need quick actions. Unfortunately, little information is available during the supply chain lifecycle. This is one of the significant challenges in the supply chain domain. Blockchain – a kind of distributed ledger technology—has been described as the next industrial revolution. The blockchain is a data structure format that makes it possible to create a stack of existence with a specific time stamp for each digital asset. This could be helpful to sign each transaction in a distributed digital ledger. Due to the above challenges, the primary purpose of this paper is to introduce the merge between Blockchain and supply chain management.

Supply Chain Challenges

Market Growth - Increasing the customer base can be a hard challenge to tackle to expand distribution both at home and abroad. There are many factors to take into consideration when doing so, including trading policies, fees, and government policies, therefore is not always easy to get to grips with.

One of the factors that present a challenge is the pursuit of new customers. The cost of a developing a product is significant. Therefore, companies are trying to expand their distribution to emerging markets to grow revenues and increase market share. Companies all around the world are expected to expand in their home and foreign markets. The introduction of new markets is challenging due to trading policies, fees, and government policies.

Data - Managing the information in the supply chain is very critical in the supply chain processes since it is the main component that affects supply chain planning. Supply chain operations include a massive amount of data processing. Every day it gets harder to manage and correctly use data. The right use of data techniques introduces excellent opportunities in the economy, but also shows up problems which consist many other difficulties such as visualization, data capture, searching, storage, capacity, analysis, and statistics. These challenges need to be solved to exploit the capabilities of data.

Raw information coming from suppliers, partners, and even customers are also often composed of both structured and unstructured data which makes it even more difficult for enterprises to consume, analyze, and generate insights from these disjointed pieces of information. Proper data management and integration transform this raw information into compatible formats required by different supply chain management systems to ensure their seamless flow.

Data management and integration address supply chain management challenges at the most basic level of the value chain and in every activity. Furthermore, providing visibility not only to manufacturers but also to suppliers and partners can potentially improve trust and long-term relationships.

Trust & Transparency - Trust is vital to Supply chain management, Government, Consumers, Business, etc. In freight transportation one of the most important element is trust. If company aim to create any long-term relationships trust moves to the top of the list. We can call it the issue which increases the success and makes external and internal partnerships stronger. While you have a situation of risk, trust creates the bond called positive expectation between partners. FreitX wants to build varieties of methodologies for evaluating and increasing the trust.

The transparency in the supply chain is a problem. Transportation control methods can help to create a detailed overview of these freight conditions. Empty trips and lead times can be minimized while resource capacity will be maximized. The solution should become stronger in line with international standards, and there should be sufficient resources and training to assure implementation and respect for customers. There should be greater decision-making, transparency so that everybody can find out who is responsible for the decisions in the transportation chain. There is an urgent need to improve the transparency.

Track & Trace - The Transportation tracking for a delivery network is an essential issue for providing customer service in the supply chain business. There are requirements in the industry for monitoring and managing the shipment from the A point to the B position. Today's services are looking for the appropriate and sufficient tools to track their delivery shipment with real-time tracking. The main problem is that these mediums for transferring valuable information are not based on real-time environment. In supply chain management, there are many requirements for tracking the delivery shipment. Some companies provide tracking systems for the transportation chain. This kind of solutions requires a centralized server where the tracking data is gathered. The server is maintained by the provider, which usually is not any of the suppliers or the customer of the product. The introduction of a third party in the transportation chain could result in, delays or failure in data integrity during the delivery process.

A solution of FreitX Network

More than ever before, excellence in supply chain management, especially in logistics functions, is about driving revenue growth, capturing market share, and enhancing customer satisfaction and loyalty. Rather than dive into the details of every end-to-end supply chain process which are continuously evolving in response to changes in customer requirements, the competitive landscape, technologies, regulations, and other factors let us focus on market growth, data, trust & transparency, track & trace which are fundamental supply chain disciplines that enable or influence virtually all supply chain functions and processes.

FreitX Network aims to develop new ways based on the blockchain, which gives the ability to be universal solution to all following problems. FreitX enables to increase on-time

sufficient services at efficient prices, for everyone involved in the transportation chain. Increased technology means that any disruptions within the delivery process can be spotted right away and therefore can be passed onto customers immediately to avoid disappointment. It may be worth finding technical advice from professionals in the know. Technology should develop quickly over time due to the advancements that are already taking place.

With FreitX Network we can reach greater transparency and visibility by establishing one shared view of supply chain data and intelligence across all suppliers, partners, systems, and processes. Such visibility should include both private and public sources of data both structured and unstructured. The primary purpose is to provide the supply chain organization with contextual intelligence that shows the impact of business decisions.

Our solution stands on analysis of real facts that can help to predict risks such as reputational, logistical, geopolitical, etc. FreitX Network allows us to react quickly, be risk-free, choose radical ways of development and avoid disruptions. Real-time monitoring helps to create a modern supply chain that is transparent, intelligent and predictive. By establishing greater visibility into supply chain data and processes and leveraging technologies, supply chain organizations can both predict and mitigate disruptions and risks and deliver more value to the business. A proactive view can play a significant role to become guided by well-known robust economic model.

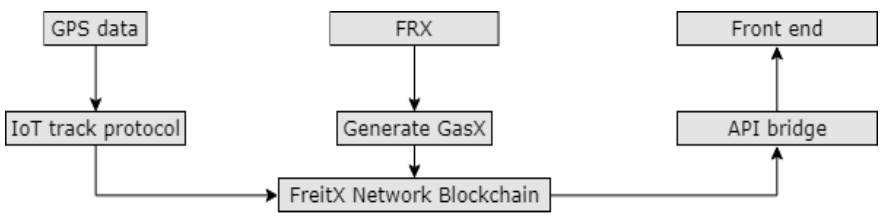
Technical overview of FreitX Network

Blockchain for supply chain - Throughout this technical paragraph, we address the main technical details to develop blockchain for supply chain management. Primary purposes to be useful blockchain for a supply chain management:

- Applicability and use cases for blockchain technology.
- Data privacy, ownership and transparency.
- Accountability.
- Access control.
- Scalability.
- System availability.
- Data compress, archival, and retention.
- Ease integrate to businesse applications.

Dual Token - The FreitX Network produces FRX token which itself includes energy token that is called GasX. FreitX Token (FTX) and GasX are the cryptographic currencies that drive the FreitX Network. FTX token is representative of shareholder and will be used by consumers as the financial, governance and payment currency. GasX represents as a fuel of FreitX Network. FTX holders get voting priority for governance in the FreitX Network and have the ability to run nodes to generate GasX.

To clarify how FRX and GasX tokens work together we can discuss the actual working process. As an example, we can take track protocol which is the primary source for asset tracking on FreitX Network. While creating transparency and trust between customers and network with the help of tracking we face challenges such as permanent data updates. Tracking requires the GasX as fuel to have non-stop current location updates on FreitX blockchain. While we have several operations in the process and we are token holders we never face this problem because tokens itself remains to be the solution. This is a simple example of how our tokens work and how they can be used to achieve your goals without obstacles.
GasX spent in the tracking flow:



Design Principle

Security - We understand that security is a significant point of concern for everyone, cryptographic and security principals are incorporated throughout the entire development lifecycle to guarantee a secure core meeting the requirements and expectations of our users. Through continuous risk analysis and internal recurring penetration testing, we provide a system that fulfills the high standards required by this type of environment.

Privacy - We are developing Blockchain where public and private data will be in one space. The primary hurdle in integrating companies with blockchain is ensuring the privacy and encryption of private information in such a way that it is only accessible to the parties intended to see it. An impressive recent development in cryptography has been the invention of ZK-SNARKs, which is the primary solution for this case. We will explain more details about ZK-SNARKs in later sections.

Flexibility - FreitX Network provides development tools and API bridges for customers to develop and secure integration services on FreitX Network. We allow flexibility in any FreitX product, which can easily customize all settings for implementation of an upcoming project. For example, a developer can change parameters of FreitX protocols and unified their services.

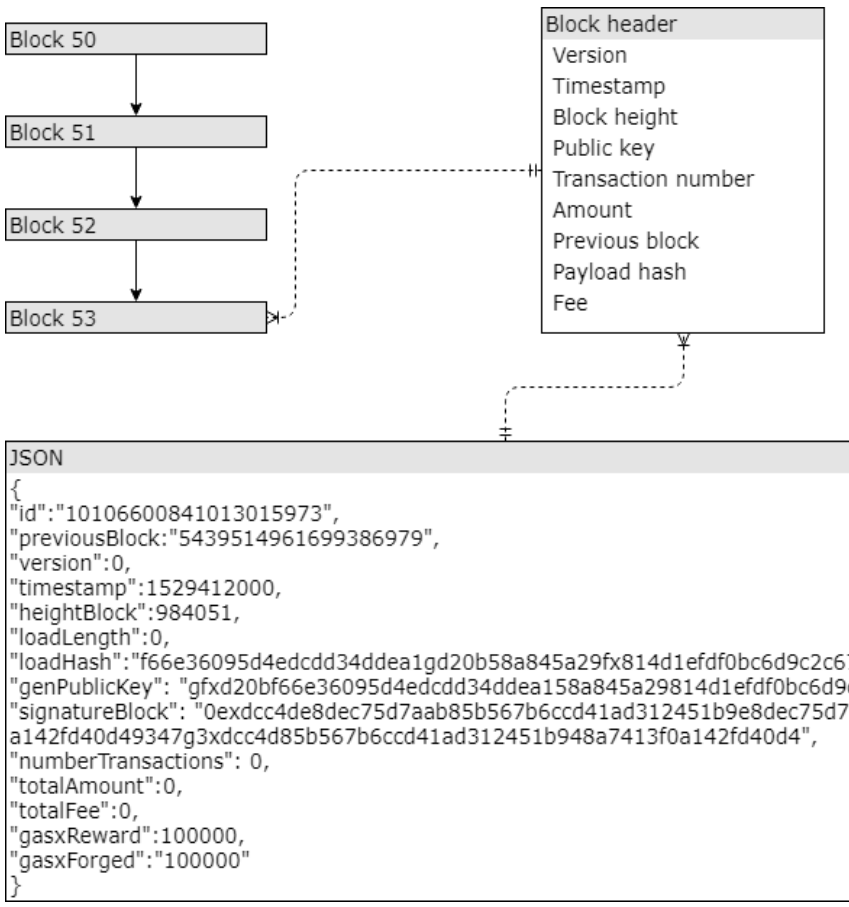
Simplicity - The FreitX Network should be as simple as possible. Developers can use JavaScript and numerous npm libraries to integrate on FreitX Network. Unlike C++ and transaction script for Bitcoin and new Solidity for Ethereum, JavaScript is more popular, has more followers, and is easier to master.

Technical Components

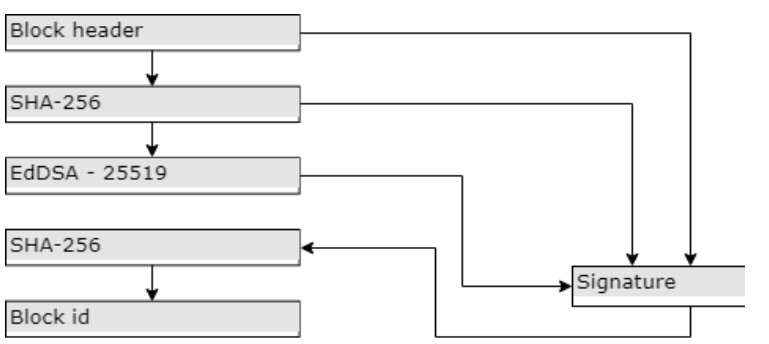
Blocks structure - A block is a container data structure. A blockchain represents the linked list of blocks. A blockchain is immutable, which means once the block is added to the blockchain it cannot be altered. Block header - The header contains metadata about a block. Block header contains following fields:

- Version (32 bits): Identifying the version of the block.
- Timestamp (64 bits): Unix time() of the created block.
- Block height (64 bits): Size of the block.
- Public key (256 bits): The public key of the delegate who generated the block.
- Previous block (64 bits): Id of the previous block.
- Number (32 bits): The number of transactions in the block.
- Reward (64 bits): Delegate reward for forging block
- Amount (64 bits): Amount of transferred FRX token or GasX.
- Payload hash (256 bits): Hash payload.
- Fee 64 (bits): Total GasX fee of transactions in the block.

The following figure is the representation of the complete structure of a block header:



Block generation - Block generation will be every 10 seconds in the FreitX Network using DPoS+PBFT consensus together which is described next topic.
Block id generation flow:



Block Rewards - Each final Block comes with a block reward that generates new GasX. The block reward will be spread over a period of 10 years decreasing over time. In FreitX Network, all active delegates that successfully participate in securing network are rewarded. Exactly number of rewarder per block will be revealed on Testnet.

Consensus Algorithms

Delegated Proof of Stake (DPOS) - is approved a method of securing a network. DPOS attempts to solve the problems of both traditional; Proof of Work system, and the Proof of Stake system. DPOS performs a layer of technological democracy to offset the harmful effects of centralization. By using a decentralized voting process, DPOS is by design more democratic than comparable systems. Rather than eliminating the need for trust, DPOS has safeguards in place the ensure that those trusted with signing blocks on behalf of the network are doing so correctly and without bias. Additionally, each block signed must have verification that a trusted node signed the block before it. DPOS eliminates the need to wait until a certain number of untrusted nodes have verified a transaction before it can be confirmed.

Practical Byzantine Fault Tolerance (PBFT) - The Byzantine Generals' Problem is a classic problem presented within any distributed computer network. Mainly, a distributed network must come to a verifiable consensus despite the potential percentage of faulty nodes or forged identities. PBFT is a replication algorithm that can tolerate Byzantine faults and reach consensus in a distributed network. Ripple, Stellar and Hyperledger are some of the popular distributed networks using PBFT. It is a multi-stage verification process where verification is done by a select number of nodes at the beginning of the process and continues to need more confirmation as it goes through the process. A practical algorithm for state machine replication that tolerates Byzantine faults. The algorithm offers both liveness and safety provided at most (n-1)/3 out of a total of n replicas are simultaneously faulty. This means that clients eventually receive replies to their requests and those replies are correct according to linearizability. The algorithm works in asynchronous systems like the Internet, and it incorporates important optimizations that enable it to perform efficiently.

Network Delegates & Delegates Voting - A delegate is merely a node (stakeholder) tasked to broadcast and verify network transaction. Any shareholder in FreitX Network ecosystem can become a delegate. A delegate is trusted accounts voted by a network, and 101 delegates who have the highest number of votes are allowed to running nodes which control the network, generate or forge new blocks.

Transactions - FreitX network has developed a conceptual transaction layer, on which almost all the functionalities of system core are established. The main difference between each transaction is their types and assets. These transaction types include:

- Tp 0: Transfer funds to specified address.
- Tp 1: Register a delegate.
- Tp 2: Voting for delegate.
- Tp 3: ZK-Snark transaction.
- Tp 4: Authorithy node registration.
- Tp 5: Transfer FreitX in and out of a sidechain.
- Tp 6: Transfer GasX in and out of a sidechain.
- Tp 7: Protocol transactions.
- Tp 8: DaPP registration.
- Tp 9: Multisignature registration.

The data structure of a fundamental transaction is as follow, and the extension of the transaction will be saved in different asset table according to its type.

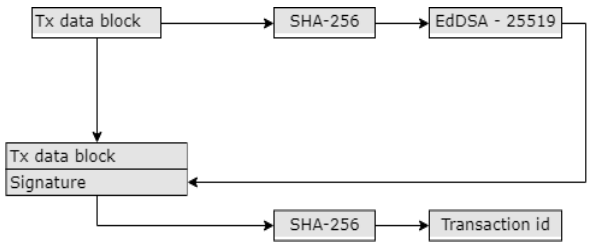
Let us take one more a vote transaction as an example:

Transaction Signing - Every transaction, regardless of the type, must be signed by the sender prior to being accepted by the network. The process of signing the transaction is identical for every transaction. First, a data block representing the transaction must be generated. Each data block contains a specific set of standardized information. Additional information contained in the data block will differ depending on the type of the transaction. The following fields must be present in all types of transactions:

- An 8-bit integer identifying the type of the transaction
- A 32-bit epoch timestamp of when the transaction has been created
- The 256 bit public key of the issuer of the transaction
- A 64-bit integer representing the amount of transfer.

The other fields will be added to this schema depending on the transaction type. Once the data block has been generated, it is hashed using the SHA-256 algorithm, and this hash is signed using the key pair of the issuer. If the issuer has enabled a second passphrase, the first signature is appended at the end of the data block, and the process is repeated, generating a second signature. The same concept applies to multisignature accounts. The transaction id is generated from the data block. In order to compute the transaction id system takes the data block with the completed signature information and hashes this block using SHA-256 and the first 8 bytes of the hash are reversed and which is then used as the transaction id.

A signed transaction uses the following flow:



Sidechain - Sidechains are technically still fully-independent chains, they are able to change features such as block structure, transaction chaining, consensus mechanism and account architecture. Sidechain can be hosted in the clusters of nodes on delegates, that hierarchy can be naturally generated hence avoid the rapid exapnsion of the core blockchain.

Account - FreitX Network is an account based system. System permis all accounts to be references unique readable name of up to 16 characters in lenght. The name is created by owner of account. Each account in FreitX Network consists of one passphrase a pair of public, private key and address. Users can set their own additional recovery password.

The passphrase is a mnemonics used to produce real wallet in line with the BIP39 standard. The mnemonics are much easier to remember for people than other binary or hexadecimal characteristics. The way in which a passphrase is generated is to convert entropy on a multiple of 32bit into several words, particular in Asch system, the length of entropy is 128bit and it will be converted into 18 words. The passphrase, as a top-level password, is maintained by users instead of published in public. Once users lost their passphrase, they would lose the ownership of their account. A passphrase is generally like follows:

isolate sentence clarify account gossip grow salon midnight motor feed defense expand begin strong stereo foster identify subway

A pair of keys includes a public key and private key, which uses a SHA256 hash of passphrase as seed and is generated through ed25519 Edwards-curve Digital Signature Algorithm (EdDSA). Public Key:

27a99989e73320347f92a13465a28ba08e687a049f52df5b388bb1e107dc2f9a

Private key:

```
08dc2f9a0347f9679f52df3ed220a13465e4625d733250d48593fc0fb570c5bdb25c5e4456
9a0232a909989388ba287a1079d8c7de2e78891f653eeb1e2b1eba
```

The account address is a big number generated by reversely converting the first 8 bits of a SHA256 hash of the public key, shown as follows:

```
8042187504902894678
```

FreitX Protocols

Data Privacy Protocol - The system itself is designed as follows. The blockchain accepts two new types of transactions: Taccess, used for access control management; and Tdata, for data storage and retrieval. These network operations could be easily integrated into a mobile software development kit (SDK) that services can use in their development process.

To illustrate, consider the following example: a user installs an application that uses our platform for preserving her privacy. As the user signs up for the first time, a new shared (user, service) identity is generated and sent, along with the associated permissions, to the blockchain in a Taccess transaction. Data collected on the phone (e.g., sensor data such as location) is encrypted using a shared encryption key and sent to the blockchain in a Tdata transaction, which subsequently routes it to an off-blockchain key-value store, while retaining only a pointer to the data on the public ledger (the pointer is the SHA-256 hash of the data).

Both the service and the user can now query the data using a Tdata transaction with the pointer (key) associated to it. The blockchain then verifies that the digital signature belongs to either the user or the service. For the service, its permissions to access the data are checked as well. Finally, the user can change the permissions granted to a service at any time by issuing a Taccess transaction with a new set of permissions, including revoking access to previously stored data. Developing a web-based (or mobile) dashboard that allows an overview of one's data and the ability to change permissions is fairly trivial and is similar to developing centralized-wallets, such as Coinbase for Bitcoin1.

The off-blockchain key-value store is an implementation of Kademilia [16], a distributed hash table (or DHT), with added persistence using LevelDB2 and an interface to the blockchain. The DHT is maintained by a network of nodes (possibly disjoint from the blockchain network), who fulfill approved read/write transactions. Dataaresufficiently randomized across the nodes and replicated to ensure high availability. It is instructive to note that alternative off-blockchain solutions could be considered for storage. For example, a centralized cloud might be used to store the data. While this requires some amount of trust in a third party, it has some advantages concerning scalability and ease of deployment.

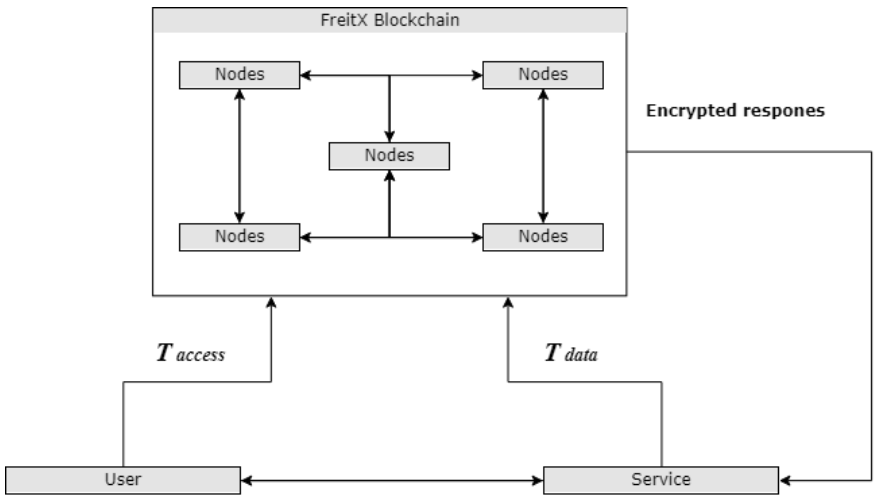


Figure 1 - Decentrilized data privacy distribution

We now describe in detail the underlying protocol used in the system. We utilize standard cryptographic building blocksinourplatform:asymmetricencryptionschemedefined by the 3-tuple (Genc,Eenc,Denc) – the generator, encryption and decryption algorithms respectively; a digital signature scheme (DSS) described by the 3-tuple (Gsig,Ssig,Vsig) – the generator, signature and verification algorithms respectively, implemented using ECDSA with secp256k1 curve [12]; and a cryptographic hash function H, instantiated by aSHA-256 implementation.

We now briefly introduce relevant building blocks that are used throughout the rest of this paper. We assume familiarity with Bitcoin and blockchains.

Identities - Blockchains utilize a pseudo-identity mechanism. Essentially a public-key, every user can generate as many such pseudo-identities as she desires in order to increase privacy. We now introduce compound identities, an extension of this model used in our system. A compound identity is a shared identity for two or more parties, where some parties (at least one) own the identity (owners), and the rest have restricted access to it (guests). Protocol 1 illustrates the implementation for a single owner (the user) and a single guest (the service). As illustrated, the density is comprised of signing key-pairs for the owner and guest, as well as a symmetric key used to encrypt (and decrypt) the data, so that the data is protected from all other players in the system. Formally, a compound identity is externally (as seen by the network) observed by the 2-tuple:

$$Compound_{u,s}^{(public)} = (pk_{sig}^{u,s}, pk_{sig}^{s,u})$$

Similarly, the entire identity (including the private keys) is the following 5-tuple:

$$Compound_{u,s} = (pk_{sig}^{u,s}, sk_{sig}^{u,s}, pk_{sig}^{s,u}, sk_{sig}^{s,u}, sk_{enc}^{u,s)})$$

Protocol 1 Generating a compound identity

1: **procedure** CompoundIdentity(u, s)
2: u *and* s *form a secure channel*
3: u executes:
4: $(pk_{sig}^{u,s}, sk_{sig}^{u,s}) \leftarrow \mathcal{G}_{sig}()$
5: $sk_{enc}^{u,s} \leftarrow \mathcal{G}_{enc}()$
6: u *shares* $sk_{enc}^{u,s}, pk_{sig}^{u,s}$ *with* s
7: s executes:
8: $(pk_{sig}^{s,u}, sk_{sig}^{s,u}) \leftarrow \mathcal{G}_{sig}()$
9: s *shares* $pk_{sig}^{s,u}$ *with* s
10: // Both u and s have $sk_{enc}^{u,s}, pk_{sig}^{u,s}, pk_{sig}^{s,u}$
11: **return** $pk_{sig}^{u,s}, pk_{sig}^{s,u}, sk_{enc}^{u,s}$
12: **end procedure**

Blockchain Memory - We let L be the blockchain memory space, represented as the hashtable L: {0, 1}256 → {0, 1}N, where N >> 256 and can store sufficiently large documents. We assume this memory to be tamperproof under the same adversarial model used in Bitcoin and other blockchains. To intuitively explain why such a trusted data-store can be implemented on any blockchain (including Bitcoin), consider the following simplified, albeit inefficient, implementation: A blockchain is a sequence of timestamped transactions, where each transaction includes a variable number of output addresses (each address is a 160-bit number). L could then be implemented as follows – the first two outputs in a transaction encode the 256-bit memory address pointer, as well as some auxiliary meta-data. The rest of the outputs construct the serialized document. When looking up L[k], only the most recent transaction is returned, which allows update and delete operations in addition to inserts.

Policy - A set of permissions a user u grants service s, denoted by P OLICYu,s. For example,if u installs a mobile application requiring access to the user's location and contacts, then P OLICYu,s = {location, contacts}. It is instructive to note that any type of data could be stored safely this way, assuming the service will not subvert the protocol and label the data incorrectly. Safeguards to partially prevent this could be introduced to the mobile SDK, but in any case, the user could easily detect a service that cheats, as all changes are visible to her.

Auxiliary Functions - Parse(x) de-seralizes the message sent to a transaction, which contains the arguments; CheckP olicy(pkk sig, xp), illustrated in Protocol 2, verifies that the originator has the appropriate permissions.

Protocol 2 Permissions check against the blockchain

1: **procedure** CheckPolicy(pk_{sig}^{k}, x_p)
2: $s \leftarrow 0$
3: $a_{policy} = \mathcal{H}(pk_{sig}^{k})$
4: **if** $L[a_{policy}] \neq \emptyset$ **then**
5: $pk_{sig}^{u,s}, pk_{sig}^{s,u}, POLICY_{u,s} \leftarrow Parse(L[a_{policy}])$
6: **if** $pk_{sig}^{k} = pk_{sig}^{u,s}$ **or**
7: $(pk_{sig}^{k} = pk_{sig}^{s,u}$ **and** $x_p \in POLICY_{u,s})$ **then**
8: $s \leftarrow 1$
9: **end if**
10: **end if**
11: **return** s
12: **end procedure**

Here we provide a detailed description of the core protocols executed on the blockchain. Protocol 3 is executed by nodes in the network when a Taccess transaction is received, and similarly, Protocol 4 is executed for Tdata transactions.

As mentioned earlier in the paper, Taccess transactions allow users to change the set of permissions granted to a service, by sending a POLICYu,s set. Sending the empty set revokes all access-rights previously granted. Sending a Taccess transaction with a new compound identity for the first time is interpreted as a user signing up to a service.

Similarly, Tdata transactions govern read/write operations. With the help of check policy, only the user (always) or the service (if allowed) can access the data. Note that in lines 9 and 16 of Protocol 4 we used shorthand notation for accessing the DHT like a normal hashtable. In practice, these instructions result in an off-blockchain network message (either read or write) that is sent to the DHT.

We rely on the blockchain being tamper-free, an assumption that requires a sufficiently large network of untrusted peers. In addition, we assume that the user manages her keys in a secure manner, for example using a secure-centralized wallet service. We now show how our system protects against adversaries compromising nodes in the system. Currently, we are less

Protocol 3 Access Control Protocol

procedure HANDLEACCESSTX(pk^k_{sig}, m)
  $s \leftarrow 0$
  $pk^{u,s}_{sig}, pk^{s,u}_{sig}, POLICY_{u,s} = Parse(m)$
  if $pk^k_{sig} = pk^{u,s}_{sig}$ **then**
    $L[\mathcal{H}(pk^k_{sig})] = m$
    $s \leftarrow 1$
  end if
  return s
end procedure

Protocol 4 Storing or Loading Data

procedure HANDLEDATATX(pk^k_{sig}, m)
  $c, x_p, rw = Parse(m)$
  if $CheckPolicy(pk^k_{sig}, x_p) =$ `True` **then**
    $pk^{u,s}_{sig}, pk^{s,u}_{sig}, POLICY_{u,s} \leftarrow Parse(L[\mathcal{H}(pk^{u,s}_{sig})])$
    $a_{x_p} = \mathcal{H}(pk^{u,s}_{sig} \parallel x_p)$
    if $rw = 0$ **then** ▷ rw=0 for write, 1 for read
      $h_c = \mathcal{H}(c)$
      $L[a_{x_p}] \leftarrow L[a_{x_p}] \cup h_c$
      (DHT) $ds[h_c] \leftarrow c$
      return h_c
    else if $c \in L[a_{x_p}]$ **then**
      (DHT) **return** $ds[h_c]$
    end if
  end if
  return $\emptyset$
end procedure

concerned about malicious services that change the protocol or record previously read data, as they are likely to be reputable, but we provide a possible solution for such behavior in section.

Given this model, only the user has control over her data. The decentralized nature of the blockchain combined with digitally-signed transactions ensures that an adversary cannot pose as the user, or corrupt the network, as that would imply the adversary forged a digital-signature or gained control over the majority of the network's resources. Similarly, an adversary cannot learn anything from the public ledger, as only hashed pointers are stored in it.

An adversary controlling one or more DHT nodes cannot learn anything about the raw data, as it is encrypted with keys that none of the nodes posses. Note that while data integrity is not ensured in each node, since a single node can tamper with its local copy or act in a byzantine way, we can still in practice minimize the risk with sufficient distribution and replication of the data.

Finally, generating a new compound identity for each userservice pair guarantees that only a small fraction of the data is compromised in the event of an adversary obtaining both the signing and encryption keys. If the adversary obtains only one of the keys, then the data is still safe. Note that in practice we could further split the identities to limit the exposure of a single compromised compound identity. For example, we can generate new keys for every hundred records stored.

Track & Trace Protocol - In this section we explain tracking protocol based on IoT with Kalman filter. Kalman filter is reliable and efficient way of noise removal process for especially signal and image processing fields. The position data obtained by the proposed device consists of the latitude and longitude data. Accordingly, faults occurring in the location data are the result of faults occurring in the values of latitude and longitude data. Kalman filter is applied separately on the latitude and longitude values. The new latitude and longitude values, enhanced by the help of filtering process, are reassembled to determine the new location, which is far more accurate than the previous measurements as it is expected. The Kalman filter offers complicated equations for different problems and systems. When the filter is applied, the appropriate equations for each problem should be taken into account so as to reduce overall complexity of the system. Essentially, the matrices that are not needed in equations can be omitted. In this study, the following formula is obtained by subtracting the unused condition matrices from the main formulas when the Kalman filter is required to apply for the signal processing problem

$$X_k = X_k.Z_k + (1 - K_k).X_{k-1}$$

According to the previously given equation, k, is used as a sub-index in the form, denoting the operating states of the system. The purpose of the formula is to compute the predicted Xk values of the signal. Zk value is the original value obtained from the receiver continuously which encompasses a certain amount of error. Besides, Xk-1 denotes the estimated value of the signal of the previous state, whereas The Kk value is called the Kalman gain. This is the only unknown value in the equation and for each case it is recalculated with the following formula (2) based on the values of the previous error covariance Pk and the standard deviation of the measurement R values.

$$K_k = P_k / (P_k + R)$$

In cases by recalculating the Kalman gain in each step, the optimum average value can be calculated and also the capabilities of the Kalman filter will be used. In order to calculate the previously mentioned values preferred in this algorithm, some initial values and parameters need to be determined. The standard deviation of measurement R-value to be used as recalculating the Kalman gain at each step is set to 1 in practice. For Xk, the first estimated value X0 to be used at time k=0 is set as the first position data received from the device. That is when the Kalman filter is applied on latitude values, while the first position received from the device is assigned to the latitude value X0, while the filter is applied on the longitude values, the longitude value of the first position received from the device is assigned to the value X0. The first value to be used at time k=0 of the Pk value used as error covariance P0 value is set to 4. This value can be set to a nonzero value. Setting P0 to zero means that there is no noise in the environment. The Pk value for each case will be recalculated using the Kalman gain value, Kk, and the previous error covariance value Pk using the following formula.

$$\begin{aligned}
&for\ k = 0 \text{ to } 30\\
&Z_k \leftarrow receiver_Values[k]\\
&X_k' \leftarrow X_k\\
&P_k' \leftarrow P_k\\
&\qquad K_k' \leftarrow P_k' / (P_k' + R)\\
&\qquad X_k \leftarrow K_k * Z_k + (1 + K_k) * X_k\\
&\qquad P_k \leftarrow (1 - K_k) * P_k'\\
&\qquad Kalman_Values[k] \leftarrow X_k\\
&end - for
\end{aligned}$$

The steps of applying the Kalman filter to an example latitude values from obtained from the receiver device is illustrated in Table 1, at times k=0 and k=1. The position data obtained from the receiver consists of the latitude and longitude data as previously mentioned. In order to defeat faults, the Average filter can also be applied separately to latitude and longitude values. The new latitude and longitude values obtained from this filter based enhancement addresses more accurate position data. The Average filter applied for the given problem is defined as follows. First, 30 latitude data of 30 position data obtained from the GPS receiver are recorded in an array.

R	1	1
k	0	1
Z_k	39,953250	39,953200
X_k	39,953250	39,953250
P_k	4	0,8
X_{k-1}	39,953250	39,953250
P_{k-1}	4	0,8
K_k	$K_k = P_{k-1} / (P_{k-1} + R)$, $K_0 = 4 / (4 + 1)$ $K_0 = 0,8$	$K_1 = 0,8 / (0,8 + 1)$, $K_1 = 0,44$
X_k *(new)*	$X_k = K_k.Z_k + (1 - K_k).X_{k-1}$ $X_0 = 0,8.39,953250 + (1 - 0,8).39,953250$ $X_0 = 39,953250$	$X_1 = 0,44.39,953200 + (1 - 0,44).39,953250$ $X_1 = 39,953228$
P_k *(new)*	$P_k = (1 - K_k).P_{k-1}$ $P_0 = (1 - 0,8).4$ $P_0 = 0,8$	$P_1 = (1 - 0,44).0,8$ $P_1 = 0,448$

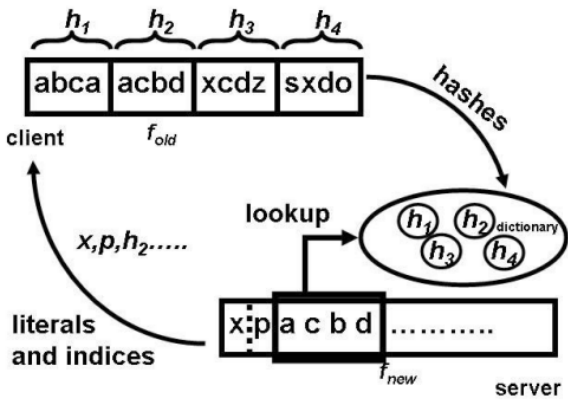
An example scenario using based on Kalman Filter.

This section introduces post-processing techniques to enhance GPS data obtained from a variety of satellites. Position data is obtained from a device which was previously designed by authors. The device can connect satellites and obtain location data using Latitude and Longitude values. However, due to the unexpected noise of the signals, there may occur big position

errors regarding approaching the exact location data. To reduce this critical error, the corresponding noise must be reduced to a tolerable level. Accordingly, Kalman filter is compared to enhance the overall position estimation performance. According to the results, filtering approaches have improved the overall performance of the systems successfully. However, Kalman Filter has superiority over the Average filter on both clear and cloudy kinds of weather as it is expected.

Doc Protocol - In this sections, we will explain document exchange protocol based on blockchain with rsync. Rsync, which stands for "remote sync," is a remote and local file synchronization tool. It uses an algorithm that minimizes the amount of data copied by only moving the portions of files that have changed. Rsync consumes less bandwidth as it uses compression and decompression method while sending and receiving data both ends. Rsync synthesis on blockchain technology gives us an opportunity to send files trough blockchain. Documentation is one of the most critical aspects of supply chain management. For enterprises, it is essential to take control of each operation, and the first source of this is to have easy access to all documents. FreitX Network technology empowers us to create a system which is the guarantee of safe and secure documentation process between partners. FreitX Network introduces a new way for enterprises to digitize documents and exchange it with a blockchain based protocol.

In this section, we describe the rsync algorithm and then discuss and evaluate a few ideas for tuning the performance of the approach. The steps in rsync are as follows:



The rsync algorithm on a small example. The client sends a set of hashes while the server replies with a stream of literals and indices identifying hashes.

1. At the client

(a) Partition fold into blocks $B_i = fold[ib,(i+1)b-1]$ of some block size b.

(b) For each block B_i, compute two hashes, $u_i = h_u(B_i)$ and $r_i = h_r(B_i)$, and communicate them to the server. Here, h_u is a heuristic but fast hash function, and h_r is a reliable but expensive hash.

2. At the server

(a) For each pair of received hashes (u_i, r_i) insert an entry (u_i, r_i, i) into a dictionary, using u_i as key.

(b) Perform a pass through f_{new}, starting at position $j = 0$, and involving the following steps:

(i) Compute the unreliable hash h_u $(f_{new}[j, j+b-1])$ on the block starting at j.

(ii) Check the dictionary for any block with matching unreliable hash.

(iii) If found, and if the reliable hashes match, transmit the index i of the matching block in f_{old} to the client, advance j by b positions, and continue.

(iv) If none found, or if the reliable hash did not match, transmit symbol $f_{new}[j]$ to the client, advance j by one position, and continue.

3. At the clinet

(a) Use the incoming stream of symbols and indices of hashes in f_{old} to reconstruct f_{new}.

The process is illustrated in Figure II.1. All symbols and indices sent from server to client in steps (iii) and (iv) are also compressed using an algorithm similar to gzip. A checksum on the entire file is used to detect the (fairly unlikely) failure of both checksums, in which case the algorithm could be repeated with different hashes, or we simply transfer the entire file in compressed form. The reliable checksum is implemented using MD4 (128 bits), but only two bytes of the MD4 hash are used since this provides sufficient power for most file sizes. The unreliable checksum is implemented as a 32-bit "rolling checksum" that allows efficient sliding of the block boundaries by one character, i.e., the checksum for f[j + 1, j + b] can be computed in constant time from f[j, j + b − 1]. Thus, 6 bytes per block are transmitted from client to server.

Perfomance - Clearly, the choice of block size is critical to the performance of the algorithm, but the best choice depends on the degree of similarity between the two files. Moreover, the location of changes in the file is also important. If a single character is changed in each block of fold, then no match will be found by the server and rsync will be completely ineffective; on the other hand, if all changes are clustered in a few areas of the file, rsync will do well even with a large block size. Given these observations, some basic performance bounds based on block size and number and size of file modifications can be shown. However, rsync does not have any good performance bounds with respect to common file distance measures.

In practice, rsync uses a default block size of 700 bytes except for very large files where a block size of √n is used. Decreasing the block size to 100 bytes or less is usually not practical: if one out of three hashes finds a match, this means that 18 bytes of hashes are transmitted for each discovered match, while simply applying gzip to the unmatched blocks might result in a reduction to about 25 bytes on average. We note that it is not difficult to find settings for the block size that perform significantly better than the rsync default size on particular data sets, but this by itself cannot be claimed as an improvement over rsync unless we get gains over a significant range.

Optimizations

We now explore a few possible optimizations of the rsync approach. We start with two fairly obvious ones: (1) use of a better compressor for literals, and (2) a better choice of the number of bits per hash.

In the first optimization, we replace the gzip algorithm used for the transmission of the unmatched literals and the match tokens in rsync with an optimized delta compressor. A delta compressor is a tool that compresses one file called target file with respect to another, usually similar, file called reference file. The resulting delta is essentially a description of the differences between target and reference file, with the property that the target file can be reconstructed from the delta and the reference file. In our modification of rsync, the server creates a reference file from the contents of all matched blocks, then compresses the current file with respect to this reference file, and transmits the resulting delta to the client. In addition, the server sends a (possibly compressed) bit vector telling the client which of its hash values has found a match, allowing the client to create the same reference file and then decode the current file.

Use of a delta compressor has two advantages: First, it exploits redundancies between unmatched and matched parts of the current file; we note that the idea of exploiting this redundancy was already discussed by Tridgell. Second, an optimized delta compressor may provide a more efficient way to encode offsets and indices than the tokens in rsync. (This also simplifies implementation and evaluation of our various methods by allowing us to sidestep the issue of how to optimize the representation and compression of these tokens.) We used the zdelta delta compressor, available at http://cis.poly.edu/zdelta/, which is highly efficient and achieves particularly good compression for small to medium size files. For large files beyond a few megabytes, a compressor such as vcdiff, which can capture global reorderings of substrings, would be preferable.

The second optimization chooses the number of bits in the hashes as a function of the file size (for the moment, assume both files are of similar size). In particular, we assume some upper bound on the probability of a collision, say 1/2d for some d, and then use lg(n) + lg(n/b) + d bits per hash. Of those bits, up to 32 are chosen from the weak but fast hash, and the rest from the slow hash. We now compare basic rsync with a version using zdelta and with a version using both zdelta and shorter hash values for d = 10. For the experiments, we used the gcc and emacs data sets also used in, consisting of versions 2.7.0 and 2.7.1 of gcc and 19.28 and 19.29 of emacs.

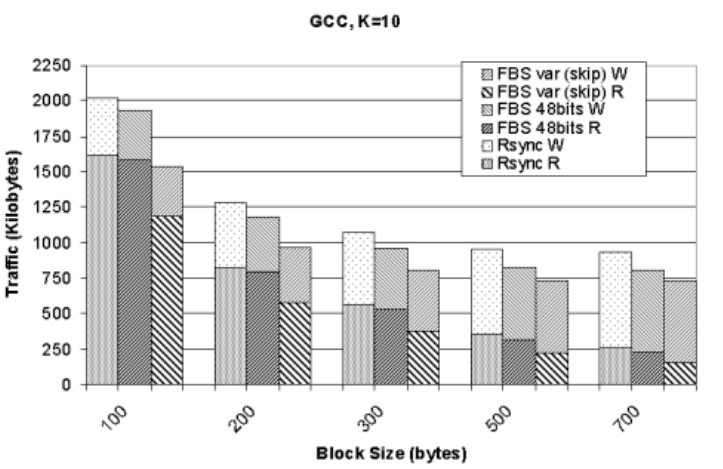


Figure II.2. Results of the first two optimizations on the gcc collection, for various block sizes.

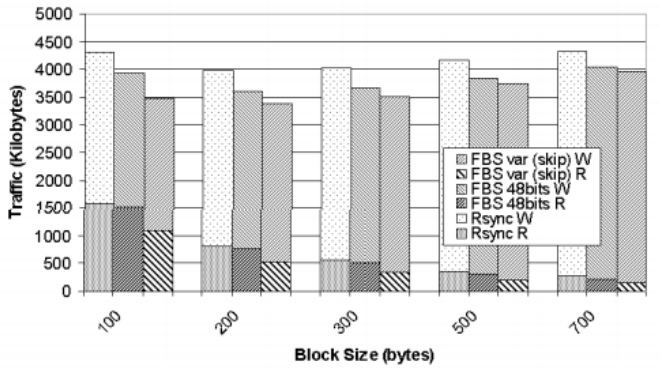


Figure II.3. Results of the first two optimizations on the emacs collection, for various block sizes.

The results are shown in Figures II.2 and II.3. We note that gcc has a much larger degree of similarity between the different versions than emacs. As a result, there are fairly few unmatched literals in gcc even with fairly large block sizes, and it is not profitable to spend extra bits on sending hashes for a smaller block size. The best block size for gcc is close to the default size of 700 bytes in rsync. For emacs, the best block size is fairly small, between 100 and 200, as this results in many additional matches that are not caught with larger block sizes. In general, the results show that there is no one optimal block size, and that the choice depends on the data. However, we see consistent improvements due to the two optimizations. We see improvements of 10% to 15% across the various block sizes, with improvements due to shorter hashes more prominent for small block sizes, since in this case the cost of the hashes is relatively higher.

Next, we look at the actual rate of collisions that we encounter with the shorter hash values, and their impact on performance. As in rsync, we assume that a 16-byte hash of the entire current file is transmitted to the client in order to detect any corruption due to false matches; in case of corruption the entire file is retransmitted encoded by gzip. We look at two different

d	% match	file coll	coll size	gzip(coll)	total
5	92.15	3.00	3.79	283909	997797
6	92.15	1.40	1.66	126869	846825
7	92.15	0.40	0.17	14845	740626
8	92.15	0.30	0.11	8998	740536
9	92.15	0.10	0.01	1627	738821
10	92.15	0.00	0.00	0	742971
5	92.14	0.20	0.24	19274	734323
6	92.13	0.00	0.00	0	720828
7	92.13	0.00	0.00	0	726476
8	92.13	0.00	0.00	0	732233
9	92.13	0.00	0.00	0	737889
10	92.13	0.00	0.00	0	743666

Table II.1

implementations of the match discovery process at the server: “skip” is the implementation currently used in rsync where after each matched block we move our window to the end of the block, thus disallowing overlapping matches in the current file, while “no-skip” also looks for overlapping matches. The results are shown in Table II.1. We see that as expected “skip” has significantly fewer collisions and file corruptions than “no-skip” and also fewer than predicted by our choice of k since it does not compare to all blocks in the current file. Thus, “skip” is the better option also in terms of bandwidth as it allows use of shorter hashes.

Variable Block Sizes - Our next idea for improving performance is to use variablesize instead of fixed-size blocks. In particular, we evaluate the use of Karp-Rabin fingerprints to determine the block boundaries, inspired by recent work, that uses these techniques in other scenarios. This is done by moving a small window (e.g., of size 20 bytes) over each file. For each byte position of the window, we hash the content using a simple random hash function (not identical to the block hash). If the hash value is 0 mod b (say, b = 256), then we introduce a block boundary at the end of the current window.

We use this technique to partition both fold and fnew. The purpose of the small window is to define block boundaries in a content-dependent manner. Thus, when a substring in one file contains a block boundary, then if the same substring also appears in another file, it will also contain the same block boundary. The advantage of this technique for file synchronization is that we do not have to compare each hash from fold to all alignments in fnew, but only to those corresponding to blocks in fnew. Thus, the number of bits per hash can be reduced by lg(b) to a total of 2 lg(n/b) + d for expected block size b. Since b is typically a few hundred bytes, this can result in nontrivial reductions in the cost of sending the hashes. this can result in nontrivial reductions in the cost of sending the hashes.

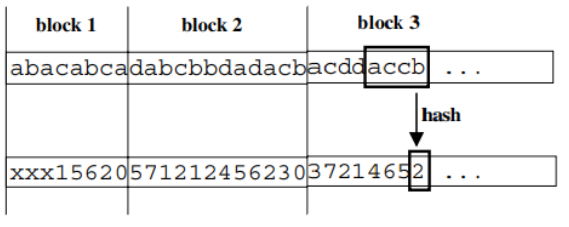


Figure II.4

We experimented with two implementations. In one, we defined block boundaries as above, by introducing a block boundary at the end of the current window if the window hashes to 0 mod b. In the second implementation, we use overlapping blocks by including both the boundary window to the right and to the left in each block. We also experimented with an alternative partitioning rule proposed in that guarantees a lower variation in block sizes, but this did not result in any improvements.

Half-Block Alignment - We studied one other optimization that goes a little beyond the standard rsync framework. The goal is to try to address two common shortcomings in rsync: (1) Suppose we have hashes for two consecutive blocks that do not find a match at the server, but if we had a hash for a block of the same size that goes from the middle of the first block to the middle of the second block, it might be possible to find a match. In general, we would like to be able to find matches of large enough size that go across the block boundaries, at least for a selected set of alignments. (2) Having identified a match of one block, we should be able to efficiently extend such a match, say, into one half of the neighboring block, even if the whole neighboring block does not find a match. This is basically the idea behind the continuation hashes proposed in, that far fewer bits are needed if a hash is only compared to one block position in the other file, in this case the position adjacent to a known match.

However, implementing these ideas in a single round is tricky, and we can not get everything we want. We take the following approach: We partition the client file into blocks of fairly small size b (say, half the size b that we would usually select under rsync), but send far fewer bits per hash (only about half as many). At the server, we look for matches in the other file, but we only accept matches that are part of a sequence of at least 2 consecutive matches. The reason is that the number of hash bits per single block is not large enough to identify isolated matches with confidence. To get a probability of less than 1/2d of a false match in the file, we need to satisfy two conditions on the number of hash bits h per block:

$$2h \geq \lg(n) + \lg(n/b) + d \text{ and } h \geq \lg(n/b) + d$$

The first condition assures that two consecutive block matches suffice to identify a valid match, while the second condition assures that we can extend a match by single blocks to the left and right without too much danger of a false match. We note that instead of choosing blocks of half the usual size, other settings are possible to recognize various other alignments not exploited by rsync, but we did not find significant benefits in this. The above algorithm, which we refer to as half-block alignment, is basically a fairly crude way to exploit the fact that matches in files are clustered and that adjacent blocks in one file are more likely to match with adjacent blocks in the other file than with blocks that are far away from each other. We implemented this method for d = 10 and also checked that the frequency of false matches is as expected.

Two Optimizations - In Figure II.5 and II.6, we compare the performance of halfblock alignment and of the approach using variable size blocks against rsync and the optimized version from the previous subsection, on a range of block sizes. For the methods with variable block size, we show the expected block size and for half-block alignment we show twice the size b of the small blocks in the figure (since the small blocks are half the “normal” size b to capture half-block alignments).

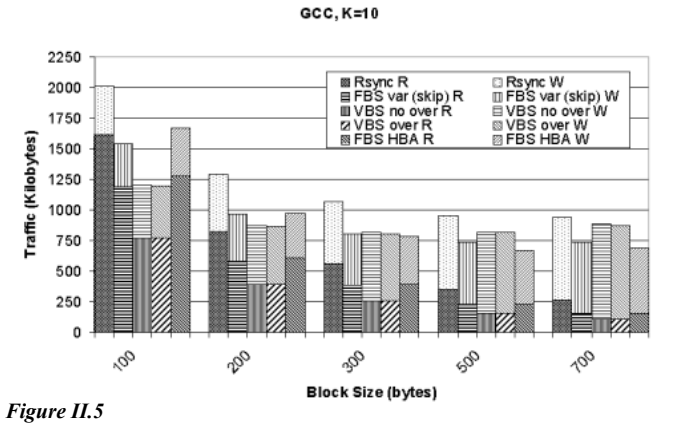


Figure II.5

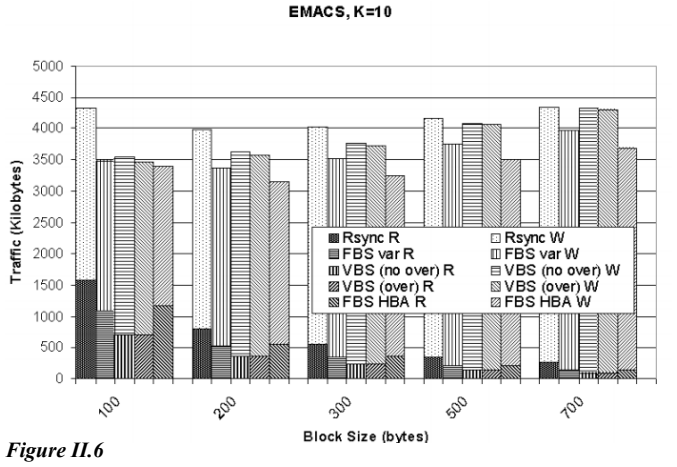


Figure II.6

We observe that the variable block-size methods do very well for small blocks. The reason is that these methods use shorter hashes, and thus benefit when the size of the hashes is significant compared to the total cost. On the other hand, methods with variable blocks tend to result in more unmatched literals, mainly due to variations in the block sizes, that dominate the savings in hash bits for larger block sizes. As suggested in, we enforce certain minimum and maximum block sizes to deal with regular patterns in the data, but even with optimum choice of but even with optimum choice of these parameters the block sizes are distributed over a certain range, and large blocks are more likely to not find a match. Moreover, we note that the savings in bits per hash for variable-size blocks are only compared to the “no-skip” version of the fixed-size method, and as seen in Table II.1 we could actually use fewer bits per hash when using the “skip” method. We observe that there is at most a very slight benefit in using overlapping instead of non-overlapping blocks.

The half-block alignment approach outperforms the other methods on most block sizes, with the notable exception of gcc for block size 100 where the second condition on the number of hash bits stated above results in a fairly high cost for the hashes. Note that this block size is not a good choice for gcc under any method, and is far away from the default size of 700 for rsync or any suitable default size. On emacs, we see an improvement of 5% to 10% over the next best method, and overall we see an improvement of 15% to 25% over the basic rsync method across the different blocks sizes, including the default size. Similar results were also obtained on several other data sets. Thus, some moderate improvements are possible through careful tuning of the rsync approach.