Secure Decentralized File Sharing (SDFS) Network

Janine Terrano (jg@topiatechnology.com)

Dan Joslin (djoslin@topiatechnology.com)

John Haager (jhaager@topiatechnology.com)

Cody Sandwith (csandwith@topiatechnology.com)

Jeff Pack (jpack@topiatechnology.com)

December 16, 2017

V6.0

Executive Summary

The rise of blockchains has created an inflection point that is driving the creation of a new "decentralized Internet". With the rise of decentralized blockchain-based cryptocurrencies, such as Bitcoin and Ethereum, momentum for the creation of a decentralized Internet has surged. Numerous projects have appeared that offer pieces of this decentralized Internet, from name lookup services to a world-wide shared storage system. But missing from amongst all of these pieces is a decentralized data layer for the decentralized applications being built on top of the blockchain.

The SDFS network is designed to provide decentralized applications with a secure, point-to-point data layer that enables the exchange of data between instances of an application across the network. Data and digital assets can be securely sent to other application instances and will be automatically shredded and encrypted using Topia's world-class encryption technology to ensure that the data is available only to the application instances for which it is intended.

By using the SDFS network, developers and manufacturers of IoT devices will be able to build security into their products from the ground up using ready-made building blocks provided by the SDFS libraries. These libraries will simplify the process of securing devices and software. Using the SDFS network, movement of data between devices, applications, and servers can be secured without requiring each developer or manufacturer to create their own security solutions. Instead, they can build on top of the tried and tested technology of SDFS and be assured that their users, devices, and data will be protected.

By combining blockchain, file sharing protocols, and libraries, the SDFS network fills the gap in the decentralized Internet and enables developers to leverage the decentralized Internet to power their decentralized applications. The SDFS

network will create the secure data layer that will allow the decentralized Internet of the future to achieve its full potential.

Introduction

With their ability to securely record transactions and activity in a network, blockchains have the ability to revolutionize the Internet and power the decentralized networks of the future. However, blockchains by themselves cannot solve all the issues inherent in a truly decentralized network. Many gaps exist between blockchains and the full stack necessary to power the decentralized Internet. Technology has arisen to fill some of these gaps, but a critical capability is still required: the ability to securely exchange data in a manner that prevents unwanted disclosure.

Sharing of data between systems is a solved problem in the current Internet world. Clients contact servers to request information; servers send the requested information back to the clients. When security is required, clients will establish SSL/TLS protected connections that encrypt the data flowing between the client and server.

Using SSL/TLS to protect data connections is necessarily dependent on the centralized certificate authority system underpinning the secure Internet of today. These organizations act as gatekeepers that control access to the digital certificates necessary to establish secure connections. In a decentralized Internet, such gatekeepers are unwanted and undesirable. This presents a challenge when establishing trust.

While peer-to-peer systems have arisen that allow clients to communicate directly with one another, they represent a fraction of the traffic relative to their traditional client-server based counterparts. Adapting the client-server model to

decentralized applications running in a decentralized Internet presents multiple challenges. Users of these applications don't want to be reliant on centralized systems for accessing data, and blockchains are not designed to store data of any significant size. This leaves a gap when it comes time to securely transfer data between instances of a decentralized application.

The blockchain market has fostered the development of several decentralized file storage solutions, such as FileCoin, Storj, and MaidSafe. Each of these products promises users access to large swaths of storage distributed across the Internet and the opportunity to rent out their excess storage capacity to other users. But these solutions only focus on allowing a user to store and access their content. None of the current solutions are focused on application level data sharing and do not provide any way for an application to send data to other instances of itself across the network.

Solution

The SDFS network is designed to provide decentralized applications with a secure, point-to-point data layer that allows the exchange of data between instances of the application across the decentralized network. Data and digital assets can be securely sent to other application instances and will be automatically shredded and encrypted using Topia's world-class encryption technology to ensure that the data is available only to the application instances for which it is intended.

The SDFS network adopts a number of techniques from the peer-to-peer and blockchain worlds to create a system for securely transferring digital assets between application instances in a manner that eliminates the need for a centralized server. By combining blockchain protocols and peer-to-peer data transfers with the security of Secrata[i], developers will be able to easily and

securely transfer digital assets between users of their decentralized applications with the blockchain providing a cryptographically secure log of all data transfers and ensuring only authorized users have access to the assets being sent across the network.

The SDFS data layer will allow applications to establish containers for digital assets and invite other application instances/users to access the container. In this way, applications will be able to securely exchange data between their instances.

Secure Collaboration Application

Topia Technology will use the data layer libraries and SDFS network to develop a Secure Collaboration suite. This suite will provide users with the ability to create secure collaboration workspaces. Files and digital assets uploaded to these collaboration spaces will be shredded and encrypted to ensure security. Other users can be invited to the collaboration space to view and modify the existing assets, or add new ones. Users will be able to send secure messages to one another in the collaboration space allowing them to coordinate on projects.

The Secure Collaboration suite will provide users with an intuitive UI that presents them with a list of all of the collaboration workspaces of which they are members. Entering a workspace will show them all of the files and digital assets contained in the workspace along with the messages left by the members. To address the challenge of user discovery, the application will include features such as address books and auto-discovery to allow users to invite members using email addresses.

Topia Technology plans to develop clients for Windows desktops, iOS mobile devices, and Android devices. All the software versions will include full SDFS network capabilities and provide full end-to-end encryption.

SDFS Use Cases

The SDFS network enables the development of numerous applications that enable users to work together securely. Several possible application use cases are listed below.

Secure Messaging dApp using SDFS

A company requiring the need to for secure communication between employees, can create a Secure Message application on top of the SDFS network. Initiating a secure communication with another user would automatically create a secure container. Messages between the users would be encrypted and the action stored in the container blockchain. Files attached to messages would be automatically and securely uploaded to the container using SDFS's secure digital asset sharing capability, as well as having the action recorded in the container blockchain. The application could allow the editing of these files by using SDFS to do a secure and encrypted download of the asset to a software editing program, with the result encrypted and uploaded back to the container.

Lawyers, CPAs, and other Professional Services Providers

Lawyers, CPAs, and other professional service providers need to be able to exchange confidential documents with their customers and clients. An application can be created using the SDFS network and API to allow such exchanges. Using an SDFS-based application, they could track the documents they shared with their clients, ensure that they

were delivered securely, receive sensitive documents from their clients, and securely communicate without fear of eavesdropping or disclosure of sensitive digital assets. SDFS would handle the creation of a container for the digital asset, as well as the secure transfer and messaging between Service Provider and client. A payment system could also be added to the SDFS based application, and the container blockchain recording the transaction.

Delivery of Digital Assets for Online Sales

A company that needs to securely process the sale and delivery of digital assets (such as movies, music, or electronic tickets) could use the SDFS network and data layer libraries to streamline the delivery process. Using the SDFS network, the company would be able to deliver purchased digital assets securely along with a receipt of the sale. SDFS ensures the security of the delivered assets, preventing theft or loss, and creating an immutable log of the delivery process. The company's application would, upon completion of a sale, create a container for delivery of the purchased assets. It would then place copies of the purchased assets, along with the purchase receipt, into the container. These actions, along with the invitation of the purchaser to the container, would be recorded in the container's blockchain. When the purchaser accesses the container to retrieve their purchased assets, the blockchain would be updated to record their acceptance of the digital assets, providing a record to the company of the successful delivery. Once the transaction is complete, the company's application could keep the container and its contents for as long as needed, discarding it after an appropriate period of time.

SDFS Differentiators

The market for blockchain-based decentralized file storage applications has grown crowded with products such as FileCoin, StorJ, and MaidSafe launching to huge fanfare. Each of these products focuses on allowing a user to store and access their content.

SDFS addresses the other side of the decentralized storage problem, a secure data layer to provide secure transfer of data and digital assets between application instances. Once digital assets have been created and stored, the next step in a typical application involves collaborating with other applications using the digital assets. The SDFS network provides a secure data layer that allows applications to distribute digital assets to other instances in a manner that maintains the integrity and security of the assets. As part of its data transfer capabilities, the SDFS data layer can distribute digital assets to other systems for replication and high availability. SDFS limits where the digital assets are replicated to ensure that the applications don't lose control of them.

To maintain the high levels of security that customers need, and that Topia Technology is known for, SDFS uses Topia's hardened security methods to shred and encrypt the digital assets before they are stored on the decentralized network. This ensures the highest protection for the digital assets and that only authorized users have access to the keys necessary to decrypt them.

Token Economy

As part of the launch of the SDFS network, Topia Technology will release a new cryptocurrency token known as TopiaCoin. This token will be used within the SDFS network to pay for services as well as reward users who contribute to the healthy functioning of the network.

In the TopiaCoin economy, token uses range from paying fees associated with the creation and replication of digital assets, to functioning as a value exchange medium in applications built on top of the SDFS network.

TopiaCoin will be used to pay container creation fees. These fees will be kept small and will allow users to securely transfer digital assets to other SDFS users. This fee may be divided between Topia Technology and 3rd party developers based on a negotiated split. In certain cases, Topia Technology, or 3rd party application developers, may choose to underwrite the cost of container creation by covering the creation fee.

Within a container, TopiaCoin is used to power the transfer and replication of digital assets amongst the members. Applications that wish to upload digital assets will deposit a small amount of TopiaCoin into the container to cover the cost of paying the other members to replicate the digital assets for high availability. Periodically, the applications providing replication service will provide proof of replication, allowing them to earn a small amount of TopiaCoin from the asset owner. This payment will be deducted from the owner's account and credited to the replicators account.

3rd party developers will be able to accept TopiaCoin as a first-class currency for payments within their applications. Since the SDFS network is already setup to handle transactions in TopiaCoin, 3rd party developers can leverage this capability to handle payment transfers on behalf of users for products other than SDFS containers and replication services.

Topia Technology will be offering a bug bounty on defects discovered in the SDFS libraries. Payments on these bounties will be made in TopiaCoin.

Finally, TopiaCoin will be exchangeable between users directly. This may include the ability for a user to "tip" another user in exchange for some service.

There is no planned currency inflation in TopiaCoin. However, Topia Technology reserves the right to issue additional tokens in the future. These tokens would be issue in a manner that ensures the functionality of the SDFS ecosystem, generates additional benefits for users of the system, and meets market demand. Topia Technology will not engage in any new token issuance within the first 3 years after network launch.

3rd Party Development Libraries

As part of building the SDFS network, Topia will develop and release a set of open-source libraries that will enable applications to be developed on top of the secure decentralized network. The SDFS libraries will provide a turnkey infrastructure improving time to market for solution providers. The

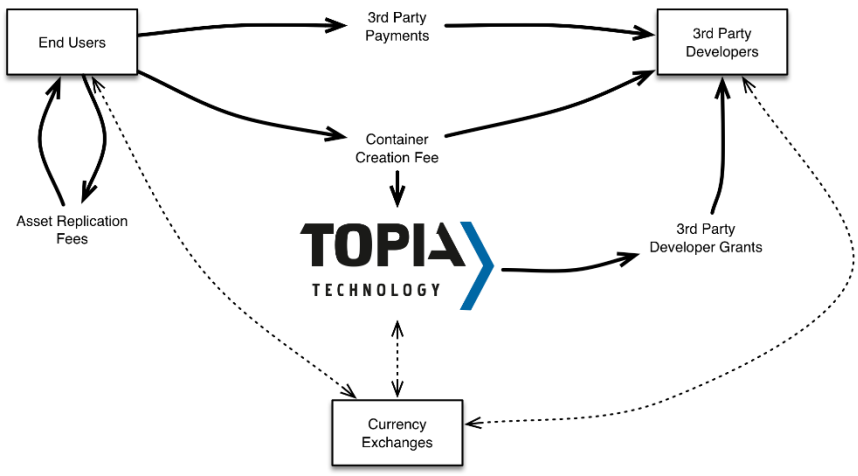

Figure 1 - Currency flow through the SDFS ecosystem.

decentralized, secure, non-repudiation and data transfer infrastructure enabled by the SDFS libraries will allow developers to focus on their particular applications and the digital assets they need to move across the network.

These libraries will encapsulate all of the blockchain activity and peer-to-peer interactions required for SDFS and provide developers a straightforward, fully functional API for developing end user applications that leverage SDFS. This includes the APIs for the creation of new secure containers, the addition and replication of digital assets, the configuration of the library to control whether it will participate in digital asset replication operations, and the transfer of TopiaCoin to other users or accounts.

The libraries will be developed in the open as the product matures toward launch and will be available on a public source repository system, such as GitHub. As part of the maintenance of the libraries after network launch, Topia Technology will offer a bug bounty program that will reward users and developers who report issues in the libraries.

Solution Design

SDFS starts by defining a container, a place where digital assets can be securely shared amongst a known set of application instances. Unlike traditional peer-to-peer systems, access to a container is by invitation only. The existence and management of these containers is accomplished through the use of blockchains. The blockchains provide a cryptographically secure digital ledger in which all transactions that occur within a container are recorded.

Once the container is established, digital assets can be added to it. The transfer of the actual data that make up the assets is accomplished through the use of

peer-to-peer data transfers using technology first developed as part of peer-to-peer file sharing systems.

How Shared containers are Defined and Synced

A container is represented by a blockchain that is shared amongst all the members of a container. This blockchain contains a sequence of transactions that describe the operations that have been performed in the container. Creating a container, then, requires creating a new blockchain and adding a container creation transaction that establishes the container, followed immediately by transactions that add the container creator and give them the cryptographic key that will be used to encrypt subsequent transactions in the blockchain. From this point forward, anytime a member wants to perform an action in the container, they create a transaction describing that action, encrypt it (except in certain cases), digitally sign it, and then submit it to the other members of the container, if any, for validation and inclusion in a new block. Once validated, the new block is added to the blockchain and sent out to all members of the container so that they can update their local copy of the blockchain and use the new transaction(s) to update their model of the container.

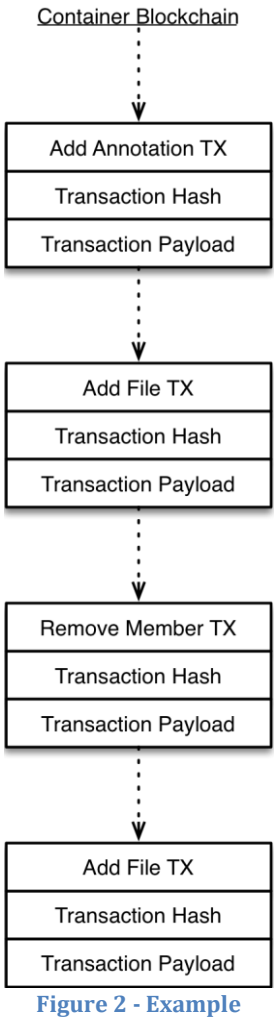

Figure 2 - Example container Blockchain

As other members are added to the container, they will request the blockchain from the current member(s). Once obtained, they can reconstruct the current state and operate in the container as a full peer.

When the time comes to share digital assets, the members can exchange the data using peer-to-peer data sharing techniques as described later in this document. In this way, members can get a reliable,

secured record describing the state of the container and get access to the digital assets stored in the container without the need for a central server.

Addressing Performance of Large Blockchains

One downside of public blockchains is their sheer size. Bootstrapping a new node onto a public blockchain can require tens or hundreds of Gigabytes of data be transferred. For example, the Bitcoin blockchain is currently over 125 GB in size[ii] requiring hours or days to synchronize a new node. To address the performance issues that arise when blockchains get large, SDFS will use a separate blockchain for each of the containers that are created. Because these blockchains will only contain transactions associated with their particular container, the size of the blockchains will remain manageable, and adding a new node (i.e. member) to a container will take seconds or minutes, not hours or days. Further, since the number of members in a container is typically small (e.g. less than 10), the number of actions performed on a container over its lifetime typically remains fairly small as well (e.g. hundreds or thousands). This will also ensure that container blockchains remain manageable in size.

TopiaCoin Usage

Depending on the specific application being developed, transactions in a container may require a payment in TopiaCoin. When an application wishes to execute one of these paid transactions, it will transfer an appropriate amount of TopiaCoin to a Smart Contract. This payment transaction will include the transaction hash of the Container Transaction that is being paid along with the transaction type.

Once the cryptocurrency network validates the payment transaction, the payment transaction hash will be inserted into a Payment Receipt transaction in the container chain along with the transaction hash of the transaction to which it pertains. SDFS will automatically verify payment receipts encountered in the blockchain to verify that they are legitimate. In addition, the SDFS may be

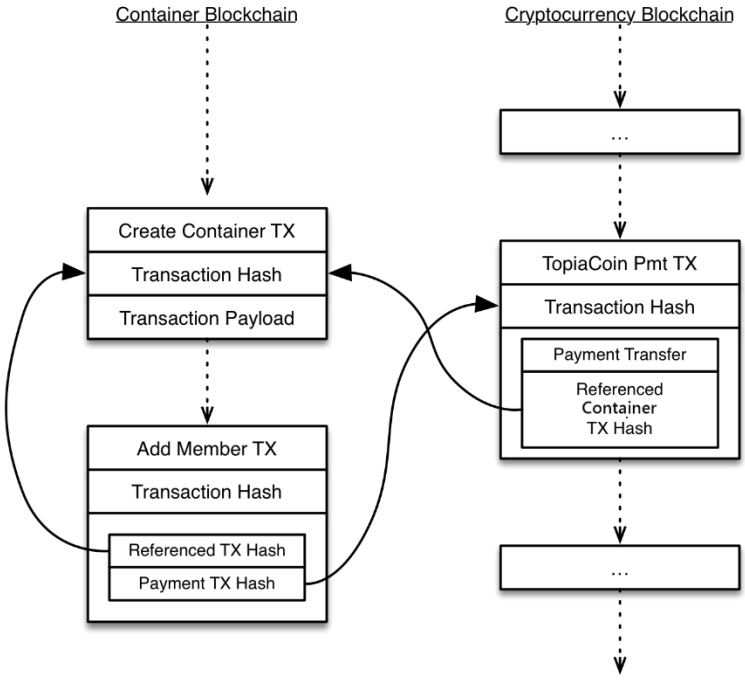

Figure 3 - Relationships between container, payment, and receipt transactions between container and cryptocurrency blockchains.

configured to require payment for other transactions on a container. It will verify that all such transactions have a corresponding payment receipt transaction in the blockchain that references a valid cryptocurrency transaction.

These payments are handled through the use of Smart Contracts. Any application written using SDFS will reference a Smart Contract that is used to pay for any transactions for which the application developer wishes to charge. The smart contract will handle the transfer of TopiaCoin from the user to the

application developer as well as recording the transaction that verifies payment. The pricing of the transactions is handled by the smart contract.

Technical Details

This section discusses the various technologies and components that make up the SDFS as well as the flow of data between the components and participants in a container.

Components

SDFS will be built on top of a number of existing technologies, including Distributed Hash Tables, peer-to-peer data transfer, and digital ledgers. Some examples of these technologies are discussed below, including how each of the technologies is used in SDFS.

Kademlia

Kademlia[iii] is a Distributed Hash Table (DHT) designed for use in decentralized peer-to-peer networks. SDFS uses Kademlia as a discovery mechanism for finding information on users, containers, and nodes. Because of its design, Kademlia allows efficient lookups in large-scale networks. This allows SDFS clients to quickly retrieve information from the DHT when needed.

S/Kademlia

S/Kademlia[iv] (i.e. "Secure Kademlia") is an adaptation of the Kademlia system that attempts to secure it by defending against its most common attack vectors. Node forging is repelled using cryptographic signatures to sign the nodes. Taking control of a dominant percentage of node IDs (known as a Sybil attack) is made significantly more difficult by forcing node creators to perform cryptographic puzzles to slow the maximum

creation rate of node IDs. An attack in which an adversary that understands Kademlia's internal routing structure attempts to take control of a node and attack communications being routed through it (known as an Eclipse attack) is defended by creating a strong sibling consensus network, such that a single adversary will be outnumbered by uncompromised nodes that constantly disagree with it.

µTP

Micro Transport Protocol[v] or µTP is a UDP-based variant of the BitTorrent peer-to-peer file sharing protocol. It provides low-priority transfer of data between peer-to-peer clients in a way that ensures that bandwidth remains available for other operations. SDFS will use this protocol as a primary method for transferring data between nodes. The data transferred via this method includes asset chunks and user information.

Blockchain

A Blockchain is a growing list of transaction records that are cryptographically linked to one another. In a decentralized system, a blockchain is used to create a secure, non-modifiable record of transactions. As nodes in the decentralized system verify the blocks in a blockchain, the preceding blocks become more permanent as the effort required to forge blocks or change previous blocks becomes greater and greater. SDFS will use blockchains as the basis for the container. They will form a distributed ledger that represents the container and all actions taken within it.

Transactions

Blocks in a blockchain contain transactions. SDFS defines specific types of transactions that represent all the actions that can be performed within a container. By combining these into blocks and validating them, the transactions form an immutable record of all that has happened in a container and serve as a functional audit log as well as a transaction log that allows a client to reconstruct the current state of the container from only the contents of the transactions in the blockchain.

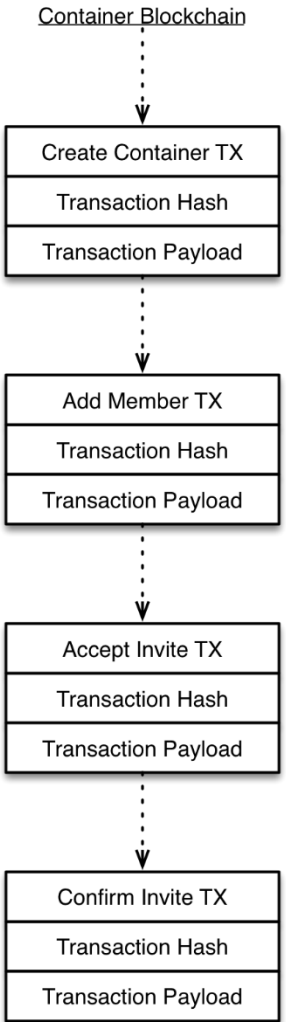

Figure 4 - Transactions necessary to create a new container.

Data Encryption

SDFS will leverage the secure data transfer techniques developed by Topia Technology for its Secrata Enterprise File Sync and Share product. These techniques involve the shredding of digital assets into separate chunks and the application of multiple layers of encryption to ensure that the chunks are protected and can only be accessed by the members of the container.

Access Control

SDFS will utilize the information in the container blockchain to enforce data security on a container and the chunks that make up the digital assets it contains. Requests to access chunks in a container are restricted to only those users who are current members of the container.

Data Flow

The following section describes in greater detail the processes by which standard Secrata operations are performed. These data flows make use of Blockchains as well as the Distributed Hash Table.

Creating a Container

Creating a container requires several steps. First, a client must create a new blockchain to represent the container. The client must then generate a Create Container transaction which describes the basic information about the container, an Add Member transaction that invites the creator to the container, an Accept Invite transaction that accepts the aforementioned invitation, and a Confirm Invite transaction that sets up the cryptographic keys for the creator of the container. These transactions will then be added to a block along with any required Payment Receipt transactions, the creator will sign the block, and the block will be added as the first block in the new container's blockchain. Finally, the creator will add appropriate information to the DHT recording the existence of the container and the creator's membership in it.

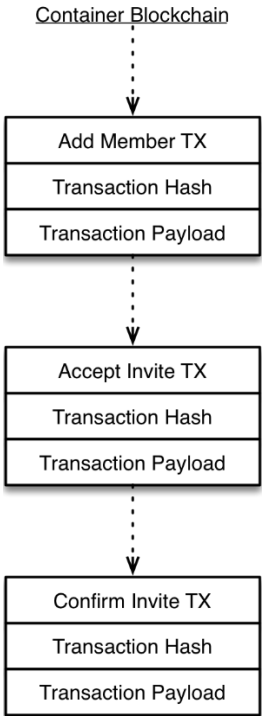

Figure 5 - Transactions necessary to add a new Member to a container

Adding Members to a Container

New members are added to a container in a manner very similar to the one used to add the creator. First, the inviter adds an Add Member transaction to a container. This transaction will contain the public key hash of the member being invited. The process of obtaining the public key hash of a user is handled by the application. If the application is attempting to invite someone who is already in another container, the client may use the information from the other container to determine the public key hash of the invitee. If, however, the user is attempting to invite someone who is not currently in any container, the client may choose to query other known collaborators to find out if they know about the invitee. Failing that, the application may require the end user to provide the public key hash manually.

Once the Add Member transaction has been added to the blockchain, the inviter will notify the invitee via the DHT. The invitee would then request the blockchain from one of the existing members, and notify the invited application of the invitation. When the invitation is accepted, an Accept Invite transaction is placed on the blockchain and the application waits for the inviter to respond with a Confirm Invite transaction containing the container key for the newly added member. Once the invited application sees that transaction, it will be able to decrypt the transaction bodies in the blockchain and reconstruct the current state of the container.

Adding Digital Assets to a Container

Digital assets are protected using Topia Technology's patented shredding and encryption methodology. This process takes a digital asset, breaks it up into a number of chunks, and encrypts each individual chunk using a unique encryption key. These keys, along with the chunk metadata (e.g. size, hash, etc.) are then encrypted using the container key. Finally, an Asset Add transaction containing the metadata of the newly added asset is added to the blockchain. The encrypted chunks can then be shared with other members via peer-to-peer communications as described in the Data Transfer section below.

Retrieving Digital Assets from a Container

To retrieve a digital asset from a container, that asset's metadata must be read out of the blockchain and decrypted using the container key. This decrypted information (i.e. asset entry) will tell the client which chunks are needed to reassemble the digital asset. The client will then check its local storage to see which chunks it has and which it needs to retrieve. For each chunk the client needs to fetch, it retrieves it from

another container member using the algorithm described in the Data Transfer section.

Once all of the chunks described in the Asset Entry are locally available, the client will decrypt each of them using their respective keys specified in the Asset Entry, decompress them (if necessary), and concatenate the data together in cardinal order as described in the Asset Entry to recreate the digital asset.

Data Transfer

An asset's chunk data is transferred between nodes via well-used peer-to-peer data transfer protocols, such as µTP. When a node needs to get data it doesn't have, it must determine which nodes to ask for that data. First, it will ask nodes that are, according to a standard algorithm, supposed to have the data. If the data is not available from any of those nodes, either because they haven't replicated yet, or because they aren't online, it will ask the node that created the data in question (e.g. the node from which the digital asset was uploaded). If the data isn't available from that node, the final step is to ask all remaining nodes for the data in question.

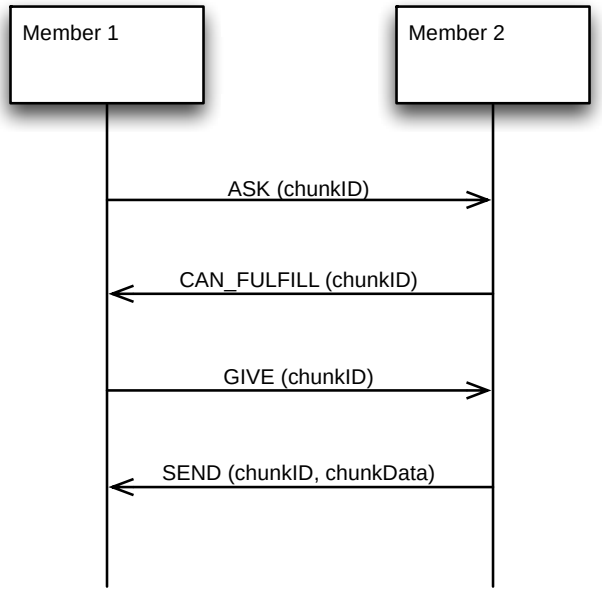

Figure 6 - Chunk data transfer protocol.

The basic workflow is that a Node will make an ASK request of multiple other nodes for each chunk it needs to

download. The queried nodes will respond with a CAN_FULFILL message if they have the requested data, or a CANNOT_FULFILL if they do not have it. Once a node responds with a CAN_FULFILL message, the requester will send a GIVE request to that single node. That node should then respond with a SEND message containing the requested data.

To ensure the integrity of the data being transferred, and that unauthorized interlopers in the network cannot gain access to data they are not authorized to have, all of the data transfer messages and responses are digitally signed by the sender. Recipients will validate the digital signature before responding to requests or processing responses. If a message's digital signature does not validate, the message is discarded and the recipient treats the message as if it had never arrived.

Digital Asset Replication

In SDFS, there is no central server that acts as the repository for digital assets shared within a container. This means that the members of a container must share the digital assets amongst themselves to ensure that each of the members can get access to them when required. Since the asset data is shredded and encrypted, it is possible to distribute the parts of an asset to multiple nodes participating in the container. In this way, all of the members will store some of the asset data. Any data not currently stored locally can be readily obtained from the other container members.

High Availability

High availability is maintained by having multiple nodes maintain copies of the data in the container. To ensure optimum performance of the

container cluster, the number of copies of the data in the cluster is generally determined by the formula $/n/2/ + 1$; that is, take the number of members in the cluster, divide by 2, and add 1. This value is known as the Replication Factor. This ensures that for any given cluster, over half of the members in the cluster should be replicating each piece of data. The replication factor may be adjusted as the number of container members grows to provide more efficient use of storage and to avoid unnecessary or excessive replication of data.

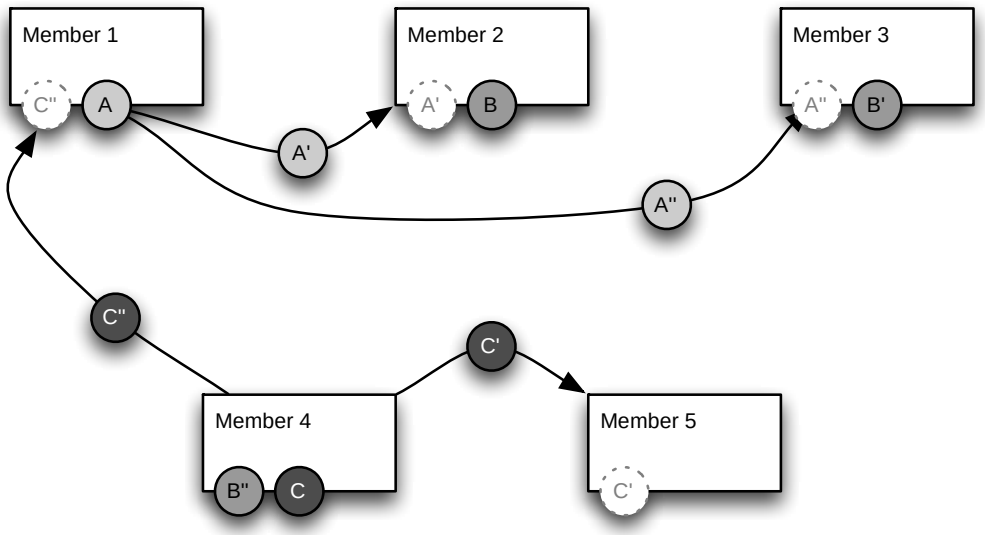

Figure 7 - Data Replication amongst container members. (N=5, R=3)

Replication Process

Replication is handled via a pull mechanism wherein a node will learn of the existence of a new asset from the blockchain and fetch the data that it is supposed to keep copies of. The basic process is:

1. The node receives a blockchain update containing a new transaction announcing that a digital asset has been added to the container.

2. The node will look at the ID of each of the asset chunks in the asset metadata to determine if a particular chunk of data should be replicated to this node.
3. If the chunk data should be replicated, the node will start a transfer of this chunk data using the Data Transfer process described in the previous section.

Proof of Replication

In order to facilitate payment for data replication, SDFS will use a Proof of Replication and Proof of Storage process. These processes will allow a replicator to prove that they have replicated asset data and still have access to the stored data. Presentation of these proofs will allow the replicator to receive payment for the data they have replicated.

Attacks

The use of blockchains to manage containers in SDFS highlights several possible vectors by which an attacker may attempt to intercept, subvert, and deny access to data. This section discusses several possible avenues of attack and how each attack vector is mitigated.

Data Acquisition Attacks

The attacks listed here relate to attempts by an attacker to gain unauthorized access to data in a container.

Intercepting the Blockchain

An attacker interested in exfiltrating information from a container might attempt to intercept the blockchain and extract the information contained within it in order to steal digital assets and other sensitive information. However, SDFS operates by encrypting nearly all of the

transactions in the blockchain using a container key. This key is present in the blockchain in an encrypted form that is only recoverable by the individual container members. Thus, acquiring the blockchain would not allow an attacker to access the digital assets contained in the container, or the messages exchanged between the members. The only information that could be extracted from the blockchain without a member's private key is the name of the container, and the IDs of its members.

Acquiring Digital Asset Chunks

An attacker may attempt to acquire asset data by directly requesting chunks from container members. In order to perform an attack like this, an attacker would need to know or obtain the following: the IDs of the chunks they wish to obtain, and the unique encryption key used to encrypt each chunk. The IDs and encryption keys are only available in encrypted form to the container members and can only be decrypted using the container key. The container key is protected in the blockchain using public key cryptography. Therefore, in order to access the IDs and encryption keys of the chunks, the attacker would have to either compromise a member's account and obtain their private key, or brute force their private key. In addition, all of the container members validate data transfer requests to ensure that a valid member has signed the request. Thus, an attacker would again need to obtain a member's private key in order to convince any of the members' systems to transfer a chunk in the first place.

Denial of Service Attacks

The attacks listed below are those attacks that might be performed in an attempt to prevent authorized members of a container from gaining access to the digital assets shared in the container.

Corrupting the Blockchain

An attacker may attempt to deny authorized members access to a container by corrupting the blockchain. This would require the attacker to obtain a copy of the blockchain, corrupt the data contained within the blockchain, and send it on to the container members in an attempt to corrupt their individual copies of the blockchain. Obtaining the blockchain would require either convincing a member node to send a copy of the blockchain or obtaining a copy out of band. Once a copy of the blockchain was obtained, the attacker could attempt to corrupt it in one of two ways, either by destroying blocks within it, or by attempting to rewrite or insert transactions into the chain. Either attempt would be thwarted by the inherent validation checking of the blockchain by the members that would detect invalid blocks in the blockchain and reject the chain. Likewise, rewriting or inserting fraudulent transactions would be caught either by a failed digital signature, by detecting that a transaction was signed by a user that isn't a member of the container, or by the transaction verification process detecting that the transaction is not legal in the container according to the business rules of the container.

Denying Chunk Access

An attacker may attempt to deny authorized access to the content of a container by preventing the members' systems from acquiring chunks from the other members. In order to do so, the attacker would have to

convince a member's system that none of the other container members' systems are available to obtain the chunks. This could be accomplished by the attacker placing themselves in a strategic network position, such as at a routing point, and dropping all the packets that request data from the other members. This attack only works if the attacker is between the victim and all other members of the container. If there is another route for the victim to reach a container member, they will be able to obtain the data they need through that pathway.

Providing Corrupt Chunk or Blockchain Update Data

An attacker may attempt to deny authorized users access to a container by distributing corrupt chunks or blockchain updates. In order to accomplish this, the attacker would have to convince a member's system that he was a legitimate source of blockchain updates and chunk data. The member system would be able to detect corrupted blocks by comparing the cryptographic hash of the chunk against the hash stored in the asset's metadata entry in the blockchain. Chunks whose hash doesn't match would be discarded and acquired from a different member. Likewise, invalid blockchain updates would be detected either due to a failed digital signature on the block, or because the block was signed by a public key that isn't a member of the container.

Future Areas of Research

SDFS is an evolving technology that is designed to interoperate with other systems and technology. New blockchain and peer-to-peer technologies are being developed that complement the capabilities of SDFS. As these technologies arise, we will investigate them for application and usefulness

within SDFS. This section highlights several technologies and systems that have the possibility of improving the capabilities of SDFS.

StorJ/FileCoin

StorJ[vi] and FileCoin[vii] are peer-to-peer cloud storage networks that allow files to be stored without relying on traditional third-party storage providers. In certain situations, it might be advantageous for a developer to use StorJ in conjunction with SDFS as the mechanism for storing encrypted chunks. The security, integrity, and availability of data stored in these systems needs to be investigated.

Keybase

Keybase[viii] provides a public directory of people, including their public keys. Such a system could be used by an SDFS-based application as a mechanism for looking up users before inviting them to a container.

Blockstack

Blockstack[ix] is a new network for decentralized applications. It aims to address the centralization of the Internet at the application-layer. Specifically, Blockstack has created an alternate DNS system, an alternate public-key infrastructure, and a distributed data storage system. All of these systems are advantageous when developing decentralized applications on top of SDFS. We will continue to monitor the development of Blockstack and identify synergies between the two networks.

Microsoft Azure Coco

Coco[x] is an open-source system that enables high-scale, confidential blockchain networks that meet all key enterprise requirements. As the project matures and

becomes generally available, we will continue to investigate its application for enterprises that wish to leverage the technology of SDFS.

IPFS

IPFS[xi] is a peer-to-peer distributed file system that seeks to create a global file system that all computing devices can access. As it grows and matures, we will continue to evaluate its usefulness as an alternative mechanism of storing chunks. Since IPFS is a globally shared file system, the security, integrity, and availability of data stored in it needs to be investigated to ensure that it is able to maintain the high levels of security that SDFS promises.

Trust

Trust in a decentralized system is an open area of research. In a system with no centralized authority, it is challenging to establish trust between entities. This is especially true when an attempt is being made to establish trust with an entity of which you don't have direct knowledge. SDFS requires trust between container members. How this trust is established and verified is an area of continuing research. We will continue to investigate trust establishment mechanisms and how those trust relationships might impact actions that can be taken by a container member.

About Topia Technology

Topia Technology was founded in 1999 and spent a decade securely moving and managing data in complex distributed environments for programs with the US Army, Federal Aviation Administration, US Air Force and Transportation Security Administration. Each of these customers required security coupled with strict performance metrics – challenges met by Topia's innovative solutions and seasoned engineering teams. With this experience in high security, high

performance environments, Topia developed its battle-tested security platform, Secrata, to provide unmatched security, flexibility, extensibility and performance.

Secrata is a patented technology that shreds and encrypts data end-to-end to harden security for cloud, Big Data and mobile. It is the only triple layer enterprise security platform providing encryption and separation end-to-end, and protecting against brute force attacks and more innovative security threats. Secrata ensures a new level of security, privacy and compliance for data regardless of where it is stored or how it is accessed.

---

[i] Secrata Security.

https://secrata.com/file-sync-share/security/.

[ii] Bitcoin.Info. Blockchain Size.
https://blockchain.info/charts/blocks-size.

[iii] P. Maymounkov, D Mazières. Kademlia: A Peer-to-peer Information System Based on the XOR Metric.
https://pdos.csail.mit.edu/~petar/papers/maymounkov-kademlia-lncs.pdf.

[iv] Ingmar Baumgart, Sebastian Mies. S/Kademlia: A Practicable Approach Towards Secure Key-Based Routing (2007).
http://www.tm.uka.de/doc/SKademlia_2007.pdf.

[v] A. Norberg. uTorrent transport protocol.
http://bittorrent.org/beps/bep_0029.html.

[vi] S. Wilkinson *et al*. Storj A Peer-to-Peer Cloud Storage Network (2016).
https://storj.io/storj.pdf.

[vii] Protocol Labs. Filecoin: A Decentralized Storage Network (2017).
https://filecoin.io/filecoin.pdf.

[viii] Keybase Inc. Keybase. https://keybase.io.

[ix] Blockstack PBC. Blockstack: A New Internet for Decentralized Applications https://blockstack.org/whitepaper.pdf

[x] M. Russonivich. Announcing the Coco Framework for enterprise blockchain networks. https://azure.microsoft.com/en-us/blog/announcing-microsoft-s-coco-framework-for-enterprise-blockchain-networks/.

[xi] J. Benet. IPFS - Content Addressed, Versioned, P2P File System (DRAFT 3) (2014). https://github.com/ipfs/ipfs/blob/master/papers/ipfs-cap2pfs/ipfs-p2p-file-system.pdf.