Secure Decentralized File Sharing (SDFS) Network

Janine Terrano (j9@topiatechnology.com)

John Haager (jhaager@topiatechnology.com)

Cody Sandwith (csandwith@topiatechnology.com)

Jeff Pack (jpack@topiatechnology.com)

August 8, 2018

V9.0

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 like Bitcoin and Ethereum, momentum for the creation of a decentralized Internet has surged. Numerous projects offer pieces of this decentralized Internet, from name lookup services to a world-wide shared storage system. But a decentralized data layer for the decentralized applications (dApps) being built on top of the blockchain is missing.

The Secure Decentralized File Sharing (SDFS) network is designed to provide decentralized applications with a secure, point-to-point data layer that enables data exchange between instances of an application using secure micro-networks. Data and digital assets can be securely sent to other application instances and will be automatically shredded and encrypted using Topia Technology's world-class encryption technology; this ensures that the data is available only to intended application instances and users.

Using the SDFS network and its secure micro-networks in the SDFS libraries, developers of dApps and manufacturers of IoT devices can build security into their products from the ground up. These libraries will simplify the process of securing devices and software. Using the SDFS network, movement of data between devices, applications, and servers is secured by the micro-networks without requiring each developer or manufacturer to create their own security solutions. Instead, developers build on top of SDFS' tried and tested technology, and their users, devices, and data will be protected.

By combining blockchain, file-sharing protocols, and proven data security methods, the SDFS network fills the gap 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 future decentralized Internet to achieve its full potential.

Introduction

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

Blockchains can revolutionize the Internet and power the decentralized networks of the future because they can securely record transactions and activity in a network. However, blockchains by themselves will not solve every issue inherent in a truly decentralized network. Many gaps exist between blockchains and the full stack necessary to power the decentralized Internet. There are solutions to bridge some of these gaps, but the ability to securely exchange data in a manner that prevents unwanted disclosure is still needed.

Currently, data is shared between systems in this way: clients contact servers to request information and servers send the requested information back to the clients. When security is required, clients establish SSL/TLS protected connections to encrypt data flowing between the client and server.

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

While existing peer-to-peer systems allow clients to communicate directly with one another, they represent a fraction of the traffic compared to traditional client-server-based counterparts. There are many challenges to adapting the client-server model to decentralized applications running in a decentralized Internet. Application users don't want to rely on centralized systems for accessing data, and blockchains are not designed to store high-volume data. This leaves a gap when securely transferring data between instances of a decentralized application.

The blockchain market has fostered the development of several decentralized file *storage* solutions, including FileCoin, Storj, and MaidSafe. Each promises users access to high-volume storage distributed across the Internet and the opportunity to rent out their unused storage capacity to other users. But these solutions are limited to allowing users to store and access their content. Currently, no solution allows application-level data *sharing* or enables an application to send data across the network to other instances of itself.

Solution

The SDFS network will provide decentralized applications with a secure, point-to-point, micro-network-based data layer that allows the exchange of data between instances of an application across a decentralized network. The micro-networks allow data and digital assets to be securely sent to other application instances and will automatically shred and encrypt the assets using Topia Technology's world-class encryption technology; this ensures that the data is available only to the application instances for which it is intended.

The SDFS network acts as a “support network” for individual micro-networks, providing blockchain hosting and data replication services. Since blockchains require an active node running to maintain continuity of the chain, the support network hosts micro-network blockchains on high-availability systems. These systems provide reliable blockchain hosting while maintaining the security of the micro-network’s encrypted content.

The SDFS network adopts techniques from the peer-to-peer and blockchain worlds to create a system to transfer digital assets securely between application instances while eliminating the need for a centralized server. By combining blockchain protocols, peer-to-peer data transfers, and the data security technology of Secrata[i], developers can easily and securely transfer digital assets between users of their decentralized applications; the blockchain provides a cryptographically secure log of all data transfers and ensures that only authorized users can access the assets being sent across the network.

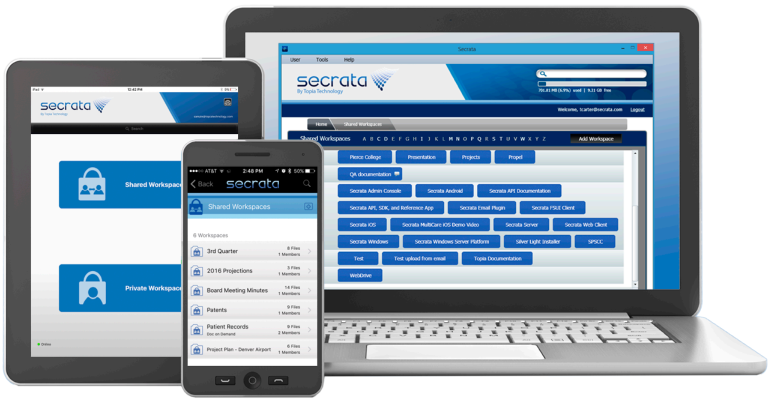
The SDFS data layer will allow applications to: 1. establish containers for digital assets and 2. invite other application instances/users to access the container. This way, decentralized applications can securely exchange data between instances and retain their ability to interact with public blockchains to accomplish their primary function.

Addressing Performance of Large Blockchains

Public blockchains may be hampered by their sheer size; bootstrapping a new node onto a public blockchain may require transfer of tens or hundreds of Gigabytes of data. For example, the Bitcoin blockchain is currently over 170 GB[ii], requiring hours or days to synchronize a new node. SDFS addresses this issue by allowing the developer to run outside the public blockchain in most cases. These much smaller chains allow faster operation.

Secure Collaboration Application

To demonstrate the SDFS network and its capabilities, Topia Technology will use the data layer libraries and SDFS network to develop a Secure Collaboration suite. This suite will allow users to create secure collaboration spaces via SDFS containers. Files and digital assets uploaded to these collaboration spaces will be shredded and encrypted to ensure security. When invited to the collaboration space, other users may view and modify existing assets or add new ones. Users can send secure messages to each other in the collaboration space, which allows them to coordinate on projects.



The Secure Collaboration suite will provide users with an intuitive UI that includes a list of all of the collaboration spaces where they are members. Each collaboration space will show all files and digital assets contained therein, along with any messages left by members. The application will include address books and auto-discovery that allow members to invite new users using email addresses.

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

SDFS Use Cases

The SDFS network enables the development of applications that allow users to work together securely. Selected application use cases are described below.

SDFS Secure Messaging Application

A company requiring secure communication between employees can create a Secure Messaging application on top of the SDFS network. Initiating a secure communication with another user would automatically create a secure container. Messages between 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 and then recorded in the container blockchain. The application could allow file editing of using SDFS to do a secure and encrypted download of the asset to a software editing program, with the result being 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 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, with 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 securely deliver both the purchased digital assets and a sales receipt. 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 is largely served by products like FileCoin, StorJ, and MaidSafe. Each of these products focuses on allowing a user to store and access their content. None of these products focuses on the problem of securely *sharing* information and digital assets. SDFS focuses on this important aspect.

SDFS addresses the need for sharing digital assets by providing a secure data layer to reliably and securely transfer 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 asset integrity and security. Using the support network, these secure micro-networks can provide high-availability of both blockchains and digital assets even when individual users are not online. As part of its data transfer capabilities, the SDFS data layer can distribute digital assets to other members as well as provide secure replication and high availability through the support network.

To meet rigorous security requirements, SDFS uses Topia's hardened security methods to shred and encrypt digital assets before they are stored on the decentralized network. This protects digital assets and ensures that only authorized users can access necessary decryption keys.

Token Economy

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

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

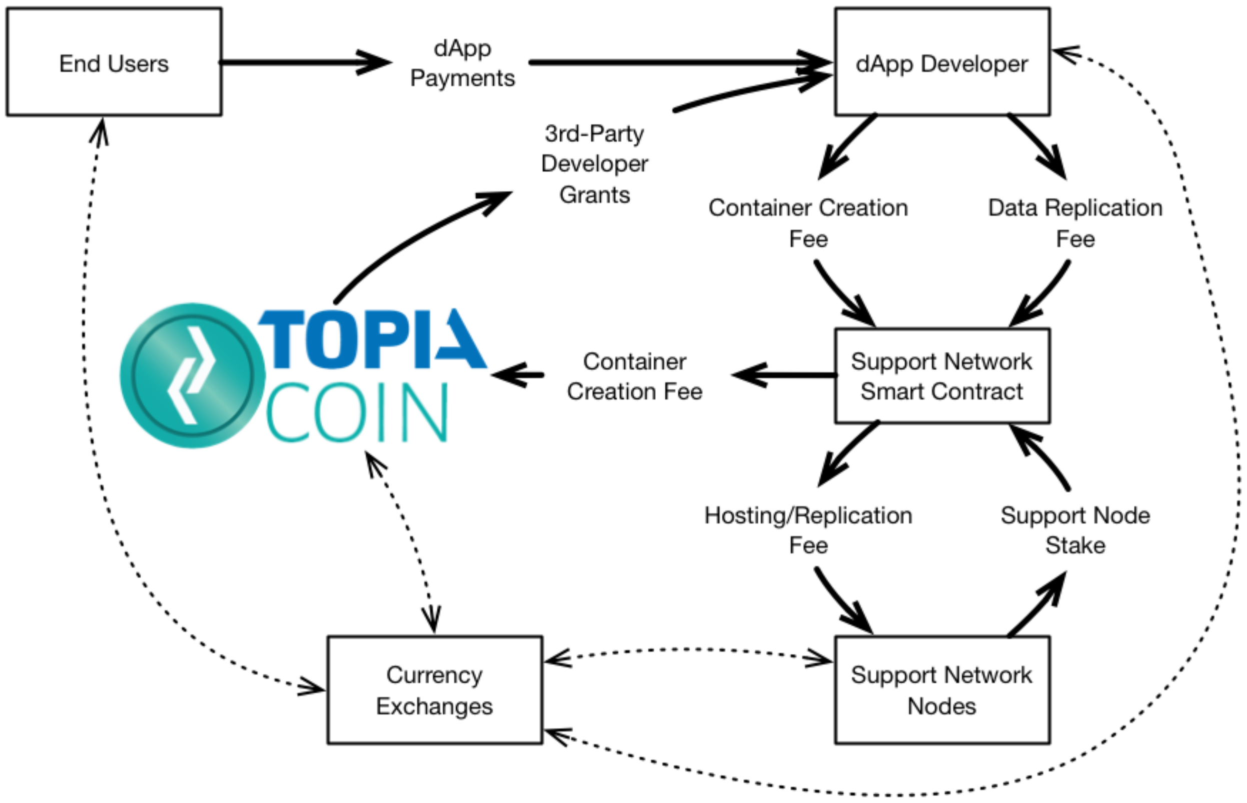


Figure 1 - Currency flow through the SDFS ecosystem.

TopiaCoin will be used to pay container creation fees. These fees will be kept low 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 container creation cost by covering the creation fee.

Within a container, TopiaCoin is used to power the transfer and replication of digital assets amongst the container's members. Applications that want 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, applications that provide the replication service will provide proof of replication, which will allow them to earn a small amount of TopiaCoin paid by the asset owner. This payment will be deducted from the owner's account and credited to the replicator's account.

3rd-party developers will be able to accept TopiaCoin as a first-class currency for payments within their applications. Since the SDFS network can already handle transactions, 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 offer a bug bounty on defects discovered in SDFS libraries. Payments on these bounties will be made in TopiaCoin.

Finally, TopiaCoin will be directly exchangeable between users. This may allow a user to "tip" another user in exchange for a designated 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 issued 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 3 years of network launch.

3rd Party Development Libraries

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

These libraries will encapsulate all blockchain activity and all peer-to-peer interactions required for SDFS; the libraries will also provide developers a straightforward, fully functional API for developing end user applications. This includes: the APIs for the creation of new secure containers; the addition and replication of digital assets; and the transfer of TopiaCoin to other users or accounts. To user SDFS, a 3rd party dApp developer loads the SDFS library into their dApp and uses its API to create a container, invite members, and share files.

These 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 maintaining 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.

SDFS Support Network

The SDFS Support Network is made up of systems that offer their CPU and storage to other SDFS users and host micro-network blockchains. Thus, end user systems do not need to run individual blockchains; rather, this requirement offloaded onto higher-power systems whose purpose is to host these blockchains and provide data replication. Support Network nodes are compensated in TopiaCoin for the use of their systems. These fees are paid by users who host their blockchains or replicate data on the node.

To ensure that Support Network nodes function properly, SDFS requires that all systems wishing to participate in the Support Network stake TopiaCoin as a guarantee. Nodes that do not fulfill their obligations or that fail to provide the agreed upon service lose some of their staked tokens. If a node's staked tokens falls below a certain threshold, the node can no longer participate in the Support Network.

Support Network nodes must stake TopiaCoin with the SDFS Master Smart Contract (SMSC) to be registered with the Support Network. dApps using SDFS communicate with the SMSC to locate a Support Network node that is available to host a new micro-network. Once a node has been identified, the SDFS library will contact the node, request the creation of a new blockchain, and use that new blockchain to set up a new container. If the user or dApp needs to provide redundancy and high-availability, the SMSC will be further queried to identify additional Support Network nodes. The nodes can be contacted and asked to join an existing micro-network or to provide data replication for micro-network content. These nodes will then join the blockchain, start providing consensus within the micro-network, and replicate secured data on behalf of users.

To ensure that Support Network nodes meet expected service levels for high availability, they must regularly check in with the SMSC to report the status of the blockchains they are hosting and the data they are replicating. Nodes that fail to check in forfeit a portion of their staked tokens. If a node loses too many staked tokens, it is downgraded in the smart contract, and fewer blockchains are directed to it for hosting. Eventually, the SMSC will stop sending any blockchains to them. This provides incentive for Support Network nodes to maintain high availability and reliability for hosted blockchains.

Solution Design

SDFS starts by defining a container, which is a place where digital assets can be securely shared among 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 using blockchains. The blockchains provide a cryptographically secure digital ledger where 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 these assets is accomplished using peer-to-peer data transfers.

Defining and Syncing Shared Containers

Secure SDFS containers are defined on the blockchain using a Smart Contract. This Smart Contract stores the container’s metadata and membership information. Creating a new container involves:

- creation of the micro-network blockchain that will host the container,

- setting the name and description of the container, and
- recording the creator as a member and saving their encrypted copy of the container's key.

Once these steps are complete, the container can be used.

As new members are invited to the container, their clients connect to the blockchain to retrieve the container's current state. Whenever the state of the container changes, all members will receive updates via the blockchain and can query the smart contract to retrieve updated container information.

To ensure that no data stored in the blockchain is unintentionally disclosed, the information stored in the smart contract must be protected by encrypting the information using the container key. Because each member of the container receives a copy of the key encrypted using their unique public key, the key itself is secured and the data in the blockchain is protected from inadvertent disclosure. Only invited members can read the state of the container.

Members can share digital assets and exchange data using peer-to-peer data sharing techniques as described later in this document. Members will have a reliable, secured record that describes the state of the container and can access the digital assets stored in the container without needing a central server.

Micro-Networks

SDFS is built to maintain privacy and security both for users and for the contents of their containers. This is accomplished by encrypting data and metadata, as well as through the use of micro-networks. These micro-networks involve using private blockchains for storing and exchanging container state, as well as using peer-to-peer transfer of encrypted data chunks. Basically, each of the member node of the container runs the blockchain and exchanges transactions, blocks, and state with one another; member nodes also host and share the encrypted data chunks with each other. In this mode, all nodes can maintain a shared state and can access all of the encrypted data stored in the container without having to reveal the existence of the container, or the identity of container members, to any outside party.

While completely private micro-networks provide excellent security, issues arise when the network is made of transient or unreliable nodes. For the network to maintain a consistent, coherent state, nodes must remain online. While the network can work in a somewhat degraded mode when some of the nodes are offline, at a certain point, the network will become difficult or impossible to use.

Should all nodes ever go offline, the network will become extremely difficult to recover when the nodes come back online: returning nodes have no way of knowing the blockchain's current state because there are no other nodes from which to obtain that state. If a node simply resumes operation from its last known state, it risks creation of a fork in the blockchain. When other nodes with longer blockchains come online, network nodes are forced to resolve the fork, and may need to discard transactions and blocks that contained user operations, which now must be re-executed on the newly reconciled blockchain. In a worst-case scenario, there is a high possibility of data loss; this could occur multiple times, as each node that comes up has a different, possibly longer, fork of the blockchain.

To mitigate this risk, at least one node of the blockchain must remain online and operational at all times. This way, when a second node comes online, it can pull the current blockchain state from the running node and catch up to the blockchain head without causing a fork. While this mitigation theoretically works, in practice it is challenging to ensure. In today's mobile-device dominated world, most, if not all, members of a container will operate from mobile devices that cannot run blockchain software. One or more users must have an always-on desktop system running the blockchain software for the container to: 1. provide a contact point for mobile devices, and 2. to act as a central clearinghouse for distribution and storage of the encrypted data chunks.

Such a system may work for power users, but it is unwieldy for novice and non-technical users. This leads to the solution of using external systems to host the micro-network on behalf of the container’s members. This system is known as the SDFS Support Network.

Support Network

The SDFS Support Network provides long-running, reliable nodes that host blockchains and data storage and distribution services on behalf of other SDFS network users.

The SDFS Support Network comprises highly-available systems that offer their CPU and storage capacity to other SDFS users in exchange for a fee. These nodes can host a number of micro-network blockchains and can store and distribute encrypted data chunks to blockchain members for which it replicates data.

To coordinate the SDFS Support Network nodes and match micro-network users with an appropriate Support Network node, SDFS uses a Smart Contract hosted on a shared blockchain, which is accessible to all of the micro-network users and Support Network nodes. This SDFS Master Smart Contract (SMSC) tracks available nodes, matches a micro-network with a Support Network node, and acts as the escrow and arbiter of the relationship to ensure that the micro-network is receiving the requested service, and that the Support Network node is paid according to the agreed-upon terms.

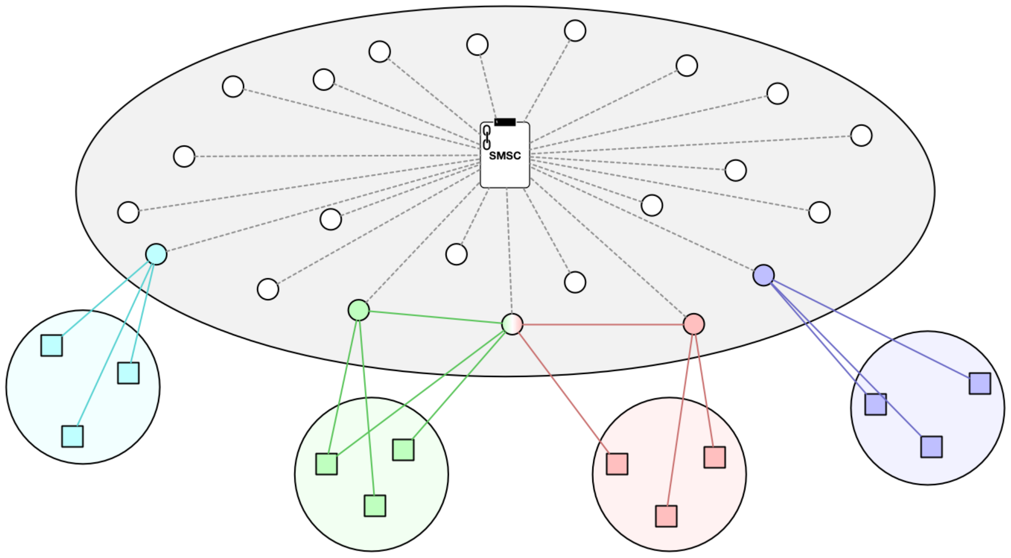


Figure 2 - The Support Network in use by several Micro-Networks

To help ensure that Support Network nodes function properly for the micro-networks they host, all Support Network nodes must stake TopiaCoin with the SMSC to become an available node. As part of this staking process, the node will also declare its capacity in terms of the number of blockchains it can support, as well as the amount of storage it has available for hosting encrypted. This information is stored inside the Smart Contract and used to match micro-networks with appropriate nodes.

In operation, when a dApp using the SDFS library decides to make use of a Support Network node for high availability, the library will contact the

SMSC and request a node to host the blockchain. Based on this request, the SMSC Smart Contract will select an available Support Network Node and assign it to the new container. This node's info is provided to the SDFS library which then contacts the node directly to request that it create a blockchain for the new container. Once the container's blockchain is created, initialized, and started, the node returns the RPC URL of the new blockchain to the SDFS library. The SDFS library connects to the blockchain, loads the container Smart Contract, and initializes the container information inside the blockchain.

A dApp may want to use more than one Support Network node for redundancy and high-availability. The SDFS library will repeat the process described above, except that instead of asking the assigned node to create a new blockchain, it will ask the node to attach to an existing blockchain, providing the address and port of an existing node that is already running the blockchain. The node will then initialize and start the blockchain software, instructing it to connect to the existing node and synchronize the blockchain. Once this is complete, the node will return the RPC URL of the blockchain node to the SDFS library, which can then use like any other blockchain node.

For data replication, the SDFS Library again contacts the SMSC, this time requesting a node that can replicate encrypted content. The request will include an estimate of how much storage space the container requires. The SMSC will select a node with enough available storage to support the expected load of the container.

The SMSC is responsible for assigning nodes to containers when requested. To ensure the best performance and fairness for all Support Network nodes, the SMSC assigns nodes to containers based on either the total number of containers that are currently being serviced by each node, or by the available replication space remaining on the node. The SMSC will select the node with the fewest containers currently assigned, up to the limit specified by the node on registration/staking.

Technical Details

Technologies and components that make up the SDFS, as well as the flow of data between the components and participants in a container, are discussed in this section.

Components

SDFS will be built on top of existing technologies, including Distributed Hash Tables, peer-to-peer data transfer, and digital ledgers. Examples of these technologies are discussed below, including how each technology 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 to discover information on users, containers, and nodes. Kademlia's design ensures efficient lookups in large-scale networks. SDFS clients can quickly retrieve information from the DHT.

S/Kademlia

S/Kademlia[iv] (i.e. "Secure Kademlia") is an adaptation of Kademlia that attempts to secure the system by defending against its most common attack vectors. Node forging is repelled using cryptographic signatures to sign 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, which slows the maximum creation rate of node IDs. An Eclipse 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, is defended by creating a strong sibling consensus network. This means that a single adversary will be outnumbered by uncompromised nodes that 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 data transfer between peer-to-peer clients that conserves bandwidth for other operations. SDFS will use this protocol to transfer data between nodes. Asset chunks and user information are transferred via this method.

Blockchain

A blockchain is a growing list of transaction records that are cryptographically linked. In a decentralized system, a blockchain creates a secure, non-modifiable transaction record. 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 increases. SDFS will use blockchains as the basis for the container; it will be the distributed ledger that represents the container and contains all actions taken within it.

Data Encryption

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

Access Control

SDFS will utilize the information in the container blockchain to enforce data security on that container and the chunks comprising the digital assets it contains. Access to chunks in a specific container is restricted to current container members.

Data Flow

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

Creating a Container

Creating a new secure SDFS container involves multiple steps, which vary depending on the mode for which the SDFS library has been configured. These steps include: creating a blockchain to host the container; initializing the container; recording the container metadata; and setting up the creator as a container member.

The container's blockchain is created by contacting the SMSC and requesting a node to host the blockchain. Once a node is assigned, the library will contact the node and request creation of the blockchain. Upon successful creation of the blockchain, the node will return the blockchain's RPC URL to the library, which allows the library to connect to the blockchain and load the container Smart Contract.

Once the library is connected to the blockchain, it can invoke the Smart Contract to create the container and set up its metadata. This will record the existence of the container, set its name and description, and insure that a unique identifier has been assigned to it.

Next, the library must insure that the container creator has been added as a container member. The smart contract adds a record to the container that contains the creator's account name, as well as a copy of the container key, encrypted with the creator's public key. The container's creator can access the container from any device or application they choose.

Finally, the library will update the DHT with the list of nodes that are hosting blockchains and replicating data for this container. Other member's applications will use this information to successfully connect to the blockchain and access the encrypted data that form the assets stored in the container.

Adding Members to a Container

New members are added to a container in much the same as the creator is added. The inviter's dApp asks the SDFS library to add a new member to the container, optionally specifying a welcome message to be displayed to the new user. The library will encrypt the container's key using the invited user's public key. Then, the library will call the Smart Contract to create the necessary record for the new user, including their account name and their encrypted container key. Finally, the library will post the invitation notification, along with the invitation message, to the DHT so that the invitee receives the new invitation.

The invitee's dApp will be notified by the SDFS library of the new invitation. The SDFS library then connects to the appropriate blockchain and retrieves the container information. Once the dApp or user has decided to accept the invitation, the library will invoke the Smart Contract to mark the invitation as accepted, allowing it to retrieve the container information from the Smart Contract and access the digital assets and messages contained within.

The process for finding a particular user’s account is introduced in a later section.

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 chunks, and encrypts each 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, the add file method of the smart contract is invoked to store the metadata of the newly added asset in 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 them using their respective keys specified in the Asset Entry, decompress them (if necessary), and combine the data in cardinal order to recreate the digital asset.

Data Transfer

An asset's chunks are transferred between nodes via robust, proven peer-to-peer data transfer protocols, such as µTP. When a node requires data it doesn't have, it must determine which nodes to ask for that data. The first step is to determine where the required data might be stored. The library will query the DHT for the list of all Support Network nodes associated with the container. This list is combined with the list of all the member-owned nodes to form the query set.

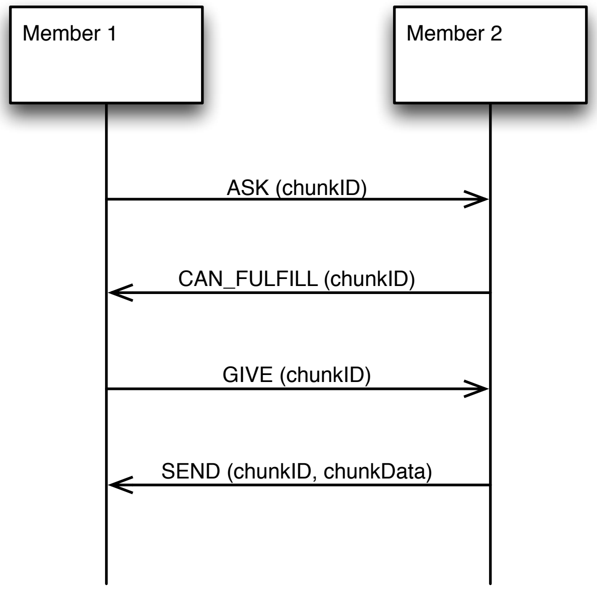


Figure 3 - Chunk data transfer protocol.

The basic workflow is that a client will make an ASK request of each of the 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.

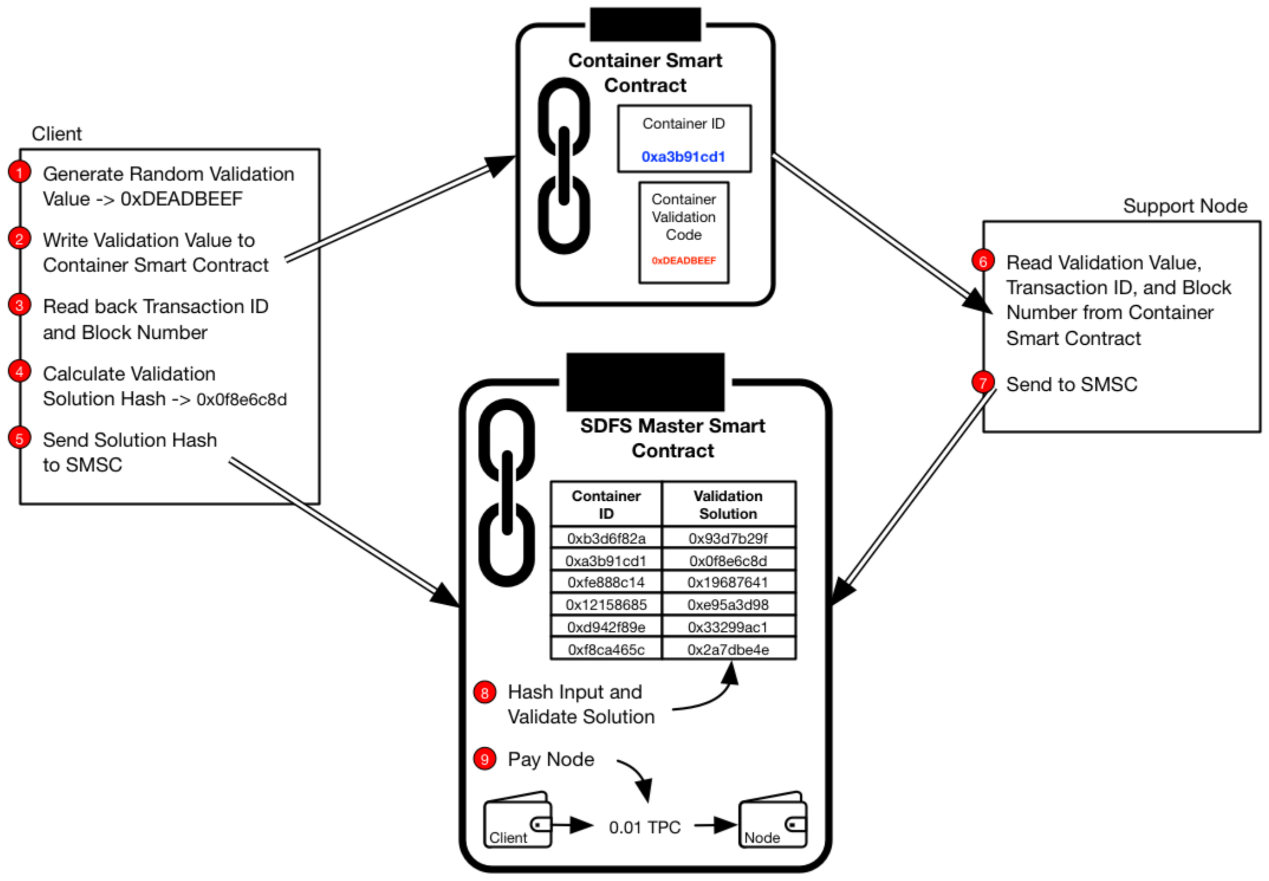
To ensure the integrity of the data being transferred, and that unauthorized interlopers cannot gain access to data without authorization, all of the data transfer messages and responses are digitally signed by the sender. Recipients will validate the digital signature before responding to requests or processing responses. If a message's digital signature does not validate or is not from an authorized container user, the message is discarded and the recipient ignores it.

Micro-Network High Availability Hosting

The SDFS Support Network is composed of highly-available systems that want to offer their CPU and storage capacity to other SDFS users in exchange for a fee. These nodes can host a number different micro-network blockchains as well as storing and distributing encrypted data chunks to the members of the blockchains it hosts. To incentivize Support Network nodes to provide reliable micro-network hosting services, they are paid a fee by the nodes for which they host micro-networks. To ensure that nodes are trustworthy and will continue to support the micro-network in question, they must regularly provide a Proof of Hosting to prove they still host the blockchain in question.

Proof of Hosting

The proof of blockchain hosting works as follows. The library will periodically post a container verification value into the container Smart Contract on the blockchain hosted by the node. This container verification value consists of a random value that is periodically updated by the client. The library will then create the solution by hashing the container verification value together with the transaction ID where the value was set, and the block number that contains the transaction. The library will then send this solution value to the for later validation against the Support Network node's solution.


Client
1 Generate Random Validation Value -> 0xDEADBEEF
2 Write Validation Value to Container Smart Contract
3 Read back Transaction ID and Block Number
4 Calculate Validation Solution Hash -> 0x0f8e6c8d
5 Send Solution Hash to SMSC
Container Smart Contract
Container ID
0xa3b91cd1
Container Validation Code
0xDEADBEEF
Support Node
6 Read Validation Value, Transaction ID, and Block Number from Container Smart Contract
7 Send to SMSC
SDFS Master Smart Contract
8 Hash Input and Validate Solution
9 Pay Node
Client
0.01 TPC
Node

Container ID	Validation Solution
0xb3d6f82a	0x93d7b29f
0xa3b91cd1	0x0f8e6c8d
0xfe888c14	0x19687641
0x12158685	0xe95a3d98
0xd942f89e	0x33299ac1
0xf8ca465c	0x2a7dbe4e

Figure 4 - Proof of Hosting Validation Process

The support node will periodically find the latest transaction in which the verification value has been set. The support node presents this value to the SMSC, along with the corresponding block number and transaction ID, as proof of ongoing blockchain support. Once the value has been validated by the SMSC, periodic payment to the node will be made. To prevent over burdening the SMSC, nodes are limited to invoking this method at a certain maximum rate (e.g. every 2 hours). If the client isn't online to update the container verification value, the hosting node can continue to provide proof of verification and get paid for hosting services. Once the client comes back online, the verification value will update.

Digital Asset Replication

In SDFS, there is no central server serving as the repository for digital assets shared within a container. This means that the container members must share the digital assets to ensure that each member can access 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

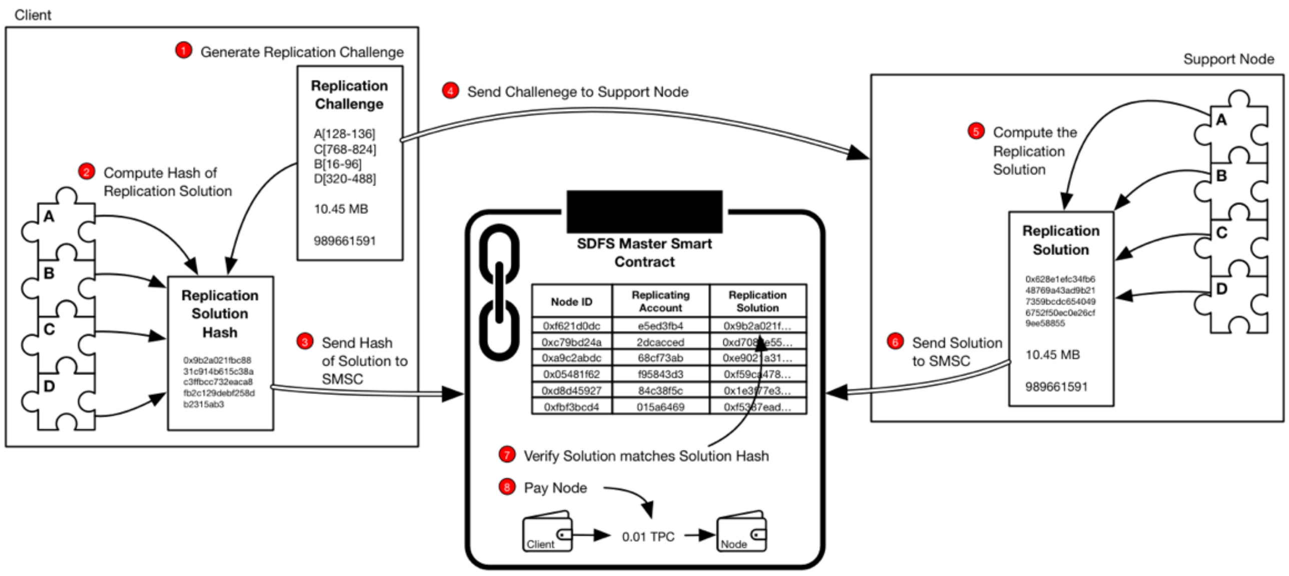
Containers can attain high availability status by using multiple Support Network nodes. These nodes would each be asked to host the container's blockchain and will work together to process transactions and maintain the blockchain. In addition, the container can use the nodes to provide redundant replication of data chunks, ensuring that other member nodes can access the necessary chunks to reassemble a digital asset even when the user who uploaded the asset isn't online.

Proof of Replication

To facilitate payment for data replication, SDFS will use a Proof of Replication process. This process will allow a replicator to prove that it still has the data it has agreed to replicate. Presentation of these proofs will allow the replicator to receive payment for data they have replicated.

Only the client that replicated the data to the node can trigger proof of data replication. Proof of data replication involves the client presenting the node with a replication challenge and giving the SMSC the solution to that challenge. The node must then present the SMSC with the correct solution to the challenge to receive payment for data replication.

Specifically, the client will generate a cryptographic hash from a portion of one or more of the chunks that it has replicated to the node along with the total amount of data replicated by the node and a nonce value. The client sends this hash to the SMSC before sending the challenge to the node. The challenge consists of details on what portions of the chunks must be hashed to generate the correct hash value along with the nonce that will be combined with them. The node will calculate the hash value of the chunk regions and present this hash, along with the total amount of data it is replicating and the nonce, to the SMSC. The SMSC will then validate that these values correspond to the hash value submitted by the client. If the values match, the node will be paid for the data replication based on the reported amount of data being replicated.



Node ID	Replicating Account	Replication Solution
0xf621d0dc	e5ed3fb4	0x9b2a021f…
0xc79bd24a	2dcacced	0xd708[illegible]e55…
0xa9c2abdc	68cf73ab	0xe902[illegible]a31…
0x05481f62	f95843d3	0xf59c[illegible]478…
0xd8d45927	84c38f5c	0x1e3f77e3…
0xfbf3bcd4	015a6469	0xf53[illegible]7ead…

Figure 5 - Proof of Replication Process

This process is repeated for each client/node replication pair. If a client is using multiple nodes to replicate chunks, then it must produce a challenge and solution for each node. If multiple clients in a container are replicating data on a particular node, each client must separately produce a challenge and solution for that node for the data they have asked the node to replicate. If multiple clients have asked a node to replicate the same data, each client will pay for data replication, as if they were the only one who had asked. This ensures that the data continues to be replicated even if others later decide they no longer want to replicate that data on the node.

This proof of replication technique makes it possible for the client and the node to agree on the amount of data being replicated; as well, the node can prove that it is still replicating the data in question, without the client revealing information to the SMSC that would enable the node to cheat the proof process. Since the SMSC does not contain the amount of data replicated, or the information on which chunk regions are involved in the hash, the node cannot use information stored in the SMSC to generate the solution without having the actual data available.

In some cases, the client may be unavailable to initiate a data replication proof. In these cases, the node can periodically present the SMSC with the solution to the data replication proof again. Assuming the proof solution in the SMSC hasn't changed, the SMSC will again validate the solution and pay the node for replicating the data for the intervening time period.

Table 1 - Proof of Replication Message Contents

Challenge:	[Chunk portion list, replication id, nonce]
Solution:	Hash(Hash(Chunk Portions, replication id, nonce), Total data replicated by node)
Node Solution to SMSC:	[Hash(Chunk Portions, replication id, nonce), Total data replicated by node]

To avoid excessive processing, the SMSC will only allow proof of replication solutions to be presented by the node at a specific interval (e.g. once every 24 hours). This interval allows time for the client to provide an updated proof of replication challenge and solution to the node and SMSC before further payment is issued.

If the SMSC rejects a solution provided by the node, the node can cease replicating the data for the client, and notify the SMSC of this decision, or wait until the next challenge is issued to reestablish the necessary proof. The client, likewise, can summarily terminate the replication agreement with the node by: 1. instructing the node to remove all replicated data, and 2. submitting a solution to the SMSC that indicates that no data is being replicated. The node can then only succeed by proving it doesn't have any data, and thus isn't owed payment.

To ensure that replicating nodes are always paid, anytime a client sends new data to a node for replication, it must initiate a replication challenge with the new data. If a client does not initiate the replication challenge after sending the data to the node, the node can delete the newly replicated data and notify the SMSC of the client's failure.

User Account Resolution

Adding a member to a container will require converting a common user identifier (e.g. an email address) into a blockchain wallet ID or account name. For users that the application hasn't encountered before, two different approaches to this issue are available: querying existing systems for information; and engaging an email loop to resolve an address to a wallet ID.

Querying Existing Systems

By querying existing systems, the application developer can take advantage of the existing infrastructure investment and quickly resolve user identifiers to their associated wallet IDs or account names. To make the process less onerous for developers, SDFS will provide a standardized mechanism for querying existing systems, such as Active Directory. This standardized mechanism involves the installation of an SDFS gateway application that receives queries from SDFS-based applications and, in turn, queries the existing system for the corresponding wallet ID. This system can be expanded to any service or organization that wants to provide name-to-wallet resolution services.

It is understood that such a system may lead to a central point of failure. The decision to use such a system is left to the dApp developer.

Email Loop Resolution

In applications that prefer not to use existing, possibly centralized, systems, an email loop approach can be taken. The SDFS-based libraries will generate and send an email that contains a specially constructed URL to the user being invited via the existing email infrastructure.

Once the email is received, invitees can click on the link to access the decentralized application and forward their wallet ID to the inviter. When the application launches, it will ask the user to either login or create a new wallet. Once the user has successfully logged in to their wallet, the wallet ID will be sent back to the inviter using the SDFS's DHT network.

The inviter's application will see the information on the DHT and be able to complete the invite process and notify the invitee of the new container.

This process can be generalized for communication mechanisms other than email. Similar instructions and links could be sent via social media networks or cellular text messages. In each case, the link would cause the SDFS application to launch and complete the process described above.

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 to mitigate risks associated with each attack vector.

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 contained information to steal digital assets and other sensitive information. However, SDFS operates by encrypting nearly all of the container state information 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. Only the container's name and its members' IDs can be extracted from the blockchain without a member's private key.

Acquiring Digital Asset Chunks

An attacker may attempt to acquire asset data by directly requesting chunks from container members or support nodes. An attack like this requires the attacker to know or obtain the following: the IDs of the chunks they want to obtain and the unique chunk encryption key. The IDs and encryption keys are only available to container members in encrypted form and can only be decrypted using the container key. The container key is protected in the blockchain using public key cryptography. Therefore, to access the IDs and encryption keys, the attacker must either compromise a member's account and obtain their private key or use brute force to obtain their private key. In addition, all container members validate data transfer requests to ensure that a valid member has signed the request. Again, an attacker must obtain a member's private key to convince the system to transfer a chunk.

Denial of Service Attacks

The attacks listed below may attempt to prevent authorized container members 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. To corrupt individual copies of the blockchain would require the attacker to: obtain a copy of the blockchain; corrupt the data contained within the blockchain; and send it to the container members and support nodes. To obtain the blockchain, the attacker must either convince a member system or support node to send a copy of the blockchain or obtain a copy out of band. Once a copy of the blockchain is obtained, the attacker could corrupt it either by destroying blocks within it or by attempting to rewrite or insert transactions into the chain. Either attempt would be thwarted by the blockchain's inherent validation checking whereby members 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 nonmember; or by the Smart Contract detecting that the transaction is not legal in the container.

Denying Chunk Access

An attacker may attempt to deny authorized access to a container's content by preventing members' systems from acquiring chunks from other members or support nodes. The attacker would have to convince a member's system that no other container members' systems or support nodes were available to obtain the chunks. The attacker would place itself in a strategic network position like at a routing point and would then drop all the packets that request data from the other members. This attack can only work when the attacker is between the victim and all other container members. If the victim can reach a container member by another route, they will be able to bypass the attacker and retrieve data via 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. To accomplish this, the attacker must convince a member's system that it is a legitimate source for blockchain updates and chunk data. The member system could 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 without a matching hash would be discarded and acquired from a different member. Likewise, invalid blockchain updates would be detected by a failed digital signature on the block, or by noting the block was signed by a public key that isn't a member of the container.

Future Research

SDFS is an evolving technology designed to interoperate with other systems and technology. New blockchain and peer-to-peer technologies are being developed that complement SDFS' capabilities. As they arise, Topia will investigate them for application and usefulness within SDFS. This section highlights several technologies and systems that improve SDFS capabilities.

StorJ/FileCoin

StorJ[vi] and FileCoin[vii] are peer-to-peer cloud storage networks that allow files to be stored without relying on traditional 3rd-party storage providers. In certain situations, a developer might use StorJ in conjunction with SDFS as the mechanism for storing encrypted chunks. The security, integrity, and availability of data stored in these systems must be investigated.

Keybase

Keybase[viii] provides a public directory of individuals, including their public keys. An SDFS-based application could use such a system to look 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. Each of these systems is advantageous when developing decentralized applications on SDFS. Topia will continue to monitor the development of Blockstack and identify potential 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, Topia will continue to investigate its application for enterprises that want to leverage the SDFS technology.

IPFS

IPFS[xi] is a peer-to-peer distributed file system that seeks to create a global file system accessible by all computing devices. As it grows and matures, Topia will continue to evaluate IPFS's usefulness as an alternative chunk storage mechanism. Since IPFS is a globally shared file system, the security, integrity, and availability of data stored in it must be investigated to ensure SDFS maintains its high levels of security.

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 attempting to establish trust with an unknown entity. SDFS requires trust between container members. How this trust is established and verified is an area of continuing research. Topia will continue to investigate trust establishment mechanisms and to determine how those trust relationships might affect the actions that can be taken by container members.

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, the Federal Aviation Administration, the US Air Force and the Transportation Security Administration. Each of these customers required security complemented by strong performance metrics – these challenges were 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 environments. It is the only triple layer enterprise security platform providing encryption and separation end-to-end and protecting against both 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://www.blockchain.com/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.spovnet.de/files/publications/SKademlia2007.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.