Sia: Simple Decentralized Storage - Draft

David Vorick
Nebulous Inc.
david@nebulouslabs.com

Luke Champine
Nebulous Inc.
luke@nebulouslabs.com

October 29, 2014

Abstract

Sia is a platform for decentralized storage. Storage providers agree produce regular proofs of storage in return for compensation. Failure to provide storage proofs results in financial penalty.

Sia is a Bitcoin-based altcoin with added support for decentralized storage. With the primary goal of security, few changes have been made to the Bitcoin protocol.

1 Introduction

We wish to create a decentralized cloud storage platform that has the ability to compete with existing solutions cloud storage, both at the p2p level and the enterprise level. We wish to achieve the following goals:

- Anyone can join the network as a storage provider and be compensated.
- Clients can upload files to the network, but do not need to remain online to submit proof of storage challenges to storage providers.
- Storage providers will suffer a financial penalty for losing a file, even if the client is not online to submit storage challenges.
- Storage providers will be compensated for successfully storing a file, regardless of client behavior.
- Clients can achieve high file reliability, even when the average storage provider is not reliable.

We have chosen to use a blockchain similar to Bitcoin [1, 11] as the foundation for our platform. This is accomplished by adding a smart contract transaction type to the blockchain. With this contract, a storage provider agrees to store a file for a period of time, submitting publicly auditable proofs of storage at some frequency. Failure to submit a proof of storage results in a penalty for the storage provider, but succeeding results in compensation. A blockchain is necessary to enforce that storage providers are paid or penalized appropriately regardless of client behavior.

The initial implementation of our decentralized storage platform will be a blockchain with an altcoin. Future support for a two way peg with Bitcoin is planned, once there is a well established method for creating Bitcoin sidechains. With security in mind as the highest priority, our blockchain mimics Bitcoin except for the changes noted below.

2 General Structure

The general rules for mining blocks, handling orphan blocks, etc. on Sia will be the same as Bitcoin. Sia's primary departure from Bitcoin lies in its transactions. Bitcoin uses a scripting system to enable a range of transaction types, such as pay-to-public-key-hash and pay-to-script-hash. Sia opts instead to use an M of N multi-signature hash scheme for all transactions. This reduces complexity and attack surface.

Sia further extends transactions to enable file storage on the network. Two extensions are needed to accomplish this: file contracts and storage proofs. File contracts declare the intention of a storage provider to store a file with a certain size and hash. They specify that a provider must periodically prove that they are still storing the file. Providers accomplish this by submitting storage proofs to the network. The specifics of this arrangement are defined in sections 4 and 5, respectively.

3 Transactions

A transaction contains the following fields:

Field	**Description**
Version	Protocol version number
Arbitrary Data	Used for metadata or otherwise
Miner Fee	A fee for the miner
Input List	Incoming funds
Output List	Outgoing funds (optional)
File Contract	See: File Contracts (optional)
Storage Proof	See: Proof of Storage (optional)
Signatures	Signatures from each input

A more detailed breakdown of these fields follows.

3.1 Inputs

Each input comprises a previous output ID and the spend conditions associated with that output.

3.2 Outputs

Each output holds an amount to be sent, along with a hash of the output spend conditions. Each output has an associated identifier, which is derived from the transaction that the output appeared in. The ID of output i is defined as $H(\text{input}_0||i)$, where input_0 is the ID of the first transaction input and H is a cryptographic hashing function.

The miner subsidy and fees each block are given output ID = $H(H(Block)||\text{"}blockReward\text{"})$.

3.3 Spend Conditions

Both inputs and outputs make use of spend conditions, which dictate the circumstances under which an output can be spent. The two allowed conditions are a time lock (which locks the output until a certain time) and an M of N signature scheme. In such a scheme, there are N public keys on the "whitelist". Some subset $M <= N$ must sign the transaction in order to unlock the output.

3.4 Signatures

Each cryptographic signature is paired with an input ID and a time lock. The input ID indicates which input the signature is being applied to. It must match one of the inputs in the transaction. The time lock prevents the signature from being used before a certain time. A bit array is also provided with the signature, which marks which fields of the transaction were signed. Any field in the transaction can be marked as signed, including any subset of inputs and outputs—except for the signatures themselves. This allows for more nuanced transaction schemes.

The actual data being signed, then, is a concatenation of the time lock, input ID, bit array, and every field specified by the bit array. Every such signature in the transaction must be valid for the transaction to be accepted.

4 File Contracts

A file contract is an agreement between a storage provider and a client requiring the provider to store a certain file. At the core of a file contract is the file's Merkle root hash [2]. This hash, along with the total size of the file, can be used to verify storage proofs.

File contracts also specify a duration, challenge frequency, and payout parameters, including the reward for a valid proof, the reward for an invalid or missing proof, and the maximum number of proofs that can be missed. If the file contract is still valid at the end of the specified duration, it "successfully terminates." Conversely, if the contract funds are exhausted before the duration elapses, or if the maximum number of

missed proofs is exceeded, the contract "unsuccessfully terminates." Each of these potential outcomes (valid proof, invalid proof, successful termination, unsuccessful termination) has an associated recipient. The funds for these various payouts are provided by the inputs to the transaction.

The rewards for completed proofs and missed proofs create a new transaction output belonging to the recipient specified in the contract. The recipient will be a hash of the output spend conditions for the new output. The output ID is defined as $H(\text{input}_0||\text{outcome}||\text{index})$. The outcome has four possible values, corresponding to the four potential outcomes listed above. The index is the number of times this outcome has occurred during the contract. Terminations will always have an index of 0, as the contract can only terminate once.

It is noted that during standard use, the intended recipient for valid proofs and successful terminations is the storage provider, and the intended recipient for invalid proofs and unsuccesful terminations is the client. It is expected that both the client and the storage provider provide funds to the contract, which enables providers to be penalized for losing files.

The ID of contract i is defined by $H(\text{input}_0||\text{"}contract\text{"}||i)$, where input_0 is the first input of the transaction and "contract" is the string "contract".

5 Proof of Storage

Storage proof transactions are periodically submitted in order to fulfill file contracts. Each storage proof targets a specific file. A storage proof does not need to have any inputs or outputs; only a contract ID and the proof data are required.

5.1 Algorithm

Each contract has a challenge frequency, which requires the storage provider to submit a storage proof every N blocks. Each interval of N blocks then constitutes a "window" in which to submit the proof. The first such window, W_0, begins at a time which is specified in the contract. Contracts can be submitted to the blockchain before the first window begins, which will lock down the funds and guarantee enforcement of the contract.

The storage challenge requires a random number as a seed. The random number for window W_i is generated by concatenating contract ID to the hash of the block prior to W_i (i.e. $W_i + i \times N - 1$) and hashing the result. This seed is used to select one of the file segments used to produce the Merkle tree. The provider supplies this segment data, along with the set of hashes that can be used to reconstruct the root of the tree. The verifier can use these hashes to verify that the segment provided is indeed part of the original file. Because the proofs are submitted to the blockchain, everyone can verify that the provider has provided the correct random segment.

If the providre is consistently able to demonstrate possession of a random segment, then the provider is very likely storing the whole file. A provider storing only 50% of the file will be unable to complete approximately 50% of the proofs.

5.2 Random Number Generation

The random number generator is subject to manipulation via block withholding attacks. Because the random number is derived from a specific block, an attacker has only one chance to manipulate the random number for a particular challenge. Furthermore, withholding a block to manipulate the random number will cost the attacker the block reward.

If an attacker is able to mine 50% of the blocks, then 50% of the challenges can be manipulated. However, the remaining 50% are still random, so even in this scenario, an attacker storing only 50% of a file will still fail 25% of its storage proofs. Clients can therefore protect themselves against random number attacks by having very large penalties for missing proofs. Assuming the attacker is financially motivated, large windows and severe penalties should be sufficient to deter any attacker that controls less than 50% of the network's hashing power. Clients are nonetheless advised to plan around potential Byzantine attacks which may not be financially motivated.

5.3 Closed Window Attacks

Storage providers can only complete a proof of storage if they can get a transaction with the storage proof into the blockchain. Miners could maliciously exclude storage proofs from blocks, depriving themselves of transactions fees but forcing a penalty on storage providers. We call this a 'closed window' attack, because though the storage provider has created a storage proof within the required window of time, they are unable to get the proof onto the blockchain.

The defense for this is having large windows, and assuming that at least some percentage of miners will be content submitting the storage proof in a block in return for a transaction fee. Because providers consent to all file contracts, the storage provider is able to reject any contract that they feel leaves them vulnerable to closed window attacks.

6 Arbitrary Transaction Data

The arbitrary data provides storage providers and clients a decentralized way to organize themselves. Standards can be agreed upon that are used to announce providers, announce files, or create an entire decentralized file tracker. No part of the protocol enforces that arbitrary data be stored - nodes will accept blocks without having the arbitrary data. Full nodes are nonetheless incentivized to store the arbitrary data, as clients may be willing to pay to download the data.

7 Storage Ecosystem

Sia relies on an ecosystem that facilitates decentralized storage. Storage providers can use the arbitrary data field to announce themselves as providers to the network. This can be done using standardized template that clients will be able to read. Clients can read these announcements and create a database of providers. They can then determine a set of providers that they trust (as a set) and create file contracts with those providers. Storage providers and clients can then create micropayment channels [10] which can be used to negotiate downloading the file.

7.1 Storage Provider Protections

A contract requires consent from both the storage provider and the client, allowing the provider to reject unfavorable terms and illegal files. The provider additionally does not need to sign a contract until the full file has been uploaded. The contract terms give storage providers flexibility. They can advertise themselves as minimally reliable, offering a low price and a agreeing to minimal penalties for losing files. Or, they can advertise themselves as highly reliable, offering a higher price and agreeing to high penalties for losing files. An efficient market will optimize storage strategies.

7.2 Client Protections

Clients can use erasure coding such as regenerating codes [4] codes to split files up into many pieces, of which only some need to be recovered. Each piece is then encrypted and stored on many storage providers. By encrypting the pieces after encoding them, the client ensures that collaborating providers are unable to reduce the redundancy.

Even if the average network reliability is very low, the client can maximize the reliability of its files by putting the files on many providers. By only needing 10% out of 100 providers, the client is relying on the most reliable 10% of the 100, instead of the average reliability or least reliable subset. If the client is able to restore lost pieces on occasion, reliability goes up even more.

By putting a file in many places, the client also benefits from an increased number of sources from which it can download data. By only needing to connect to the closet 10%, the client can reduce latency. By only needing to connect to the fastest 10%, the client can increase throughput.

Finally, this strategy also protects the client from malicious storage providers. A storage provider that is demanding ransom for a file is the same to the client as a provider that is offline. As long as 10% of the providers are not acting maliciously, the client can retrieve the file.

7.3 Uptime Incentives

The storage proofs contain no provisions that require constant uptime. There are also no provisions that require storage providers to upload files to clients upon request.

However, providers and clients can create micropayment channels to facilitate downloads. If the client is offering a nontrivial fee for downloading a file, providers will be incentivized to collect the fee. If one provider is unavailable or expensive, the client will pay the fee to a provider that is available and reasonably priced. Providers are then incentivized to be online all the time, so that they can collect the bandwidth fees from clients. If clients behaviorally pay providers well, then providers are more heavily incentivized to be online all the time. Clients can also more heavily reward greater throughput and lower latency. Clients could even do random "checkups" that pay providers some nontrivial reward simply for being online, even if they do not wish to download anything. This further incentivizes providers to be online and available. However, we reiterate that uptime incentives are not part of the Sia protocol; they are entirely dependent on client behavior.

7.4 Basic Reputation System

Clients need a reliable method for picking storage providers. Hosts could potentially Sybil attack the network, and use false files to build up a fake reputation. One defense for a Sybil attack would be for clients to only consider hosts that have X coin-days locked in an unspendable output.

Storage providers declare themselves using the arbitrary data in a transaction that creates a large timelocked output. The timelocked output represents a commitment on the part of the storage provider to be reliable and inhibits the ability of malicious providers to perform Sybil attacks, as they will need large timelocked outputs. When clients are scanning the blockchain, they select providers at random, but weight them according to the number of coins that are timelocked for more than a certain period of time. Clients can also weight providers according to the size of the refund for losing files, and the price of the files. Choosing a careful scheme for weighting providers during random selection will protect clients and ensure that malicious providers need to make significant investment to maintain a majority weight during the selection process.

This scheme does not leave clients any way to leave reviews for storage providers. Additionally complexity could potentially allow clients to rate providers based on throughput, latency, and availability. Such a review system would need careful design to prevent malicious providers from creating fake clients and files to boost their ratings. A centralized storage tracker could provide a more reasonable environment for a reputation system, and could manage the whole system out of band.

8 Under Consideration

The primary foundation of Sia has been established above. Other considerations, such as mining algorithm, block time, etc., can be assumed to mirror the settings found in Bitcoin.

Giving careful attention to "A Treatise on Altcoins" [5], we are considering the following changes to Sia for the overall improvement of the cryptocurrency. We caution that these propositions have not yet been rigorously examined from a security standpoint.

8.1 Flexible Contracts

Contracts are currently strict. There is a set penalty for each missed storage proof, and a termination upon N total missed storage proofs. Increased flexibility in the penalty schedule may be desirable.

Contracts are also currently permanent, creating what is essentially an uneditable file on the network. There may be value in adding flexibility that allows clients and hosts to negotiate an updated file hash or other updated contract terms. Updating the terms of the contract will require consent from all parties.

8.2 Different Storage Proofs

Solutions such as "Compact Proofs of Retrievability" [3] offer a potentially superior way to verify that hosts are storing files.

8.3 Proof of Existence Windows

In an attempt to partially resolve the closed window attacks, we could use a proof of existence strategy. A host can create a hash of the storage proof which they submit to the blockchain within the window. The host then has a greatly extended window in which they can demonstrate that the proof of storage was created during the required window.

This has two advantages. First, an attacker cannot selectively exclude proof of existence hashes, because there's no way to figure out who owns each hash. Either the attacker doesn't include any unknown proof of existence hashes, or the attacker risks including undesired proof of existence hashes. Second, this allows hosts to submit small transactions to the network during peak hours and then the larger transactions when the traffic has died down.

A further improvement would enable Merkle Tree proofs of existence. This would enable a host to submit multiple proofs of storage in a single proof of existence hash.

8.4 Miner Fee Adjustments

If a block has miner fees which are significantly higher than the fees in the current block, there is incentive for miners to re-mine the previous block and change who gets the miner fees. This can be mitigated by putting all of the fees into a pool which pays out 33% every block, making re-mining unprofitable for any party with less than 40% of the network hashing power.

Additionally, miners have incentives not to propagate high fee transactions, because this will prevent other miners from mining the transaction and collecting the fees. It may be possible to construct a system using fee deterioration that means a miner has the highest expected total reward when the transaction is mined as soon as possible - regardless of who mines the transaction.

8.5 File Contract Fees

Some percent, like 1% of all coins paid into a file contract would be given to miners. This guarantees that as long as Sia is being used for it's core purpose (storage), there will be mining incentive.

8.6 Reduced Block Time

We are comfortable reducing the block time to 5 minutes, with the primary goal being to reduce the variance with which a miner receives payment. A 5 minute block time makes the network more vulnerable to attack, but improvements in block technology such as invertible bloom filters [6] are enabling faster block propagation times. We believe that the risk induced by halving the block time is less than the benefit gained from the reduced incentive to join mining pools.

8.7 More Frequent Difficulty and Subsidy Adjustments

We would wish to preserve the maximum difficulty adjustment of 4x every 2 weeks, however we would like to have more frequent difficulty adjustments. We would wish to have more frequent subsidy adjustments, which gives people a more continuous sense of the economic state and prevents moments where many miners turn off simultaneously.

8.8 Max Block Size Adjustment

Max block size will be initially set at 256kb and increased by 256kb every 3 months for 2 years. Then it will increase by 25% anually.

8.9 Committing to State

One thing that could allow for substantially lighter weight clients is if the miners committed to the current state of the network, instead of just to the new transactions. This would mean creating a structure

for a database that represents the state of the network and hashing it. We could follow suggestions similar to those presented in "Ultimate blockchain compression" [8].

9 Conclusion

Sia takes the Bitcoin cryptocurrency and makes a few major changes. The first is removing the scripting system and instead enforcing a multisig solution, to reduce complexity and attack surface. The second is to introduce a file contract, which enables storage providers to claim that they will store a file in return for compensation, and that they will pay a penalty for losing the file. Finally, arbitrary data fields have been added to all transactions to enable coordination among decentralized storage providers, and among decentralized file sharing systems.

We have additionally proposed an array of potential changes that range from simple and probably safe to complex and probably unsafe. We do intend to explore these proposed changes further and decide concretely on which ones are worth integrating into Sia.

References

[1] Satoshi Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System.*

[2] R.C. Merkle, *Protocols for public key cryptosystems*, In Proc. 1980 Symposium on Security and Privacy, IEEE Computer Society, pages 122-133, April 1980.

[3] Hovav Shacham, Brent Waters, *Compact Proofs of Retrievability*, Proc. of Asiacrypt 2008, vol. 5350, Dec 2008, pp. 90-107.

[4] K. V. Rashmi, Nihar B. Shah, and P. Vijay Kumar, *Optimal Exact-Regenerating Codes for Distributed Storage at the MSR and MBR Points via a Product-Matrix Construction.*

[5] Andrew Poelstra, *A Treatise on Altcoins.*

[6] Gavin Andresen, *O(1) Block Propagation*, https://gist.github.com/gavinandresen/e20c3b5a1d4b97f79ac2

[7] Gregory Maxwell, *Deterministic Wallets*, https://bitcointalk.org/index.php?topic=19137.0

[8] etotheipi, Ultimate blockchain compression w/ trust-free lite nodes, https://bitcointalk.org/index.php?topic=88208.0

[9] Gregory Maxwell, *Proof of Storage to make distributed resource consumption costly.* https://bitcointalk.org/index.php?topic=310323.0

[10] Mike Hearn, *Rapidly-adjusted (micro)payments to a pre-determined party*, https://en.bitcoin.it/wiki/Contracts#Example_7:_Rapidly-adjusted_.28micro.29payments_to_a_pre-determined_party

[11] Bitcoin Developer Guide https://bitcoin.org/en/developer-guide