Storj: Decentralized Autonomous File Storage

BitCumulus + Storjcoin

Authors: Shawn Wilkinson + Jim Lowry
me@super3.org + lowry.jim@gmail.com.

Abstract

Storj is an open source software project seeking to prove conceptually that cloud storage services can be made decentralized, more secure, and more efficient while providing hard asset backing for sound money. In furtherance of this goal, we propose developing two distinct, interoperable, and interdependent software applications: 1) BitCumulus, a file hosting web application that provides an interface to non-technical users and a development platform for the storage network, and 2) Storjcoin, a cryptocurrency that serves as both an incentive payment mechanism and a datastore for file information and location. Each will seek to operate autonomously as a peer to peer network of nodes running open source code. Modularity is a key design principle for Storj. New cryptocurrency technologies such as Mastercoin, Ethereum, and Bitshares are all extending and improving the body of work introduced through Bitcoin. Storj's modular design will allow for integration with these technologies. Storj's higher aspiration is to provide a platform for decentralized storage, computing, and autonomous agents.

Introduction

"Cloud storage" is a marketing term that takes advantage of people's craving for novelty. It caught on because to ordinary users, "the cloud" sounded like a newer technology than "the internet", "client-server", or "as a service" but this rebranding of existing technology is simple misdirection, not magic. When data is stored "in the cloud" it is transferred over TCP/IP from the client's computer to the host's server in a data center. It is the same old client-server model that has existed since the days of the mainframe and dumb terminal. That server then copies it to other servers to comply with industry standard redundancy policies so that three (3) copies are

made. The current model of cloud storage through centralized institutions that are entrusted with private information is inherently insecure in many ways. Information thieves, spies, and censors can seek to copy or destroy data stored on the host servers through political strategy, legal tactics, and technological means.

The distinction between these three categories has become increasingly blurry over time, but it is now clear that personal privacy and enterprise information security can only be achieved if off-site data stores can be protected from attacks originating in and operating through each category. Having easily identifiable central points of attack built into the model is a problem solvable through decentralization and automation. Other inherent security flaws in the current cloud storage model are the types of payment mechanism currently in widespread use by cloud storage services, which are not private or information secure because most online payment technology stores and leaks information about payer and payee.

Storj is going to change what people mean when they say “cloud storage”. We need a cloud storage model that is not based on trust between client and host. All potentially private data, including filename, date and other metadata, must be encrypted before any transfer takes place from a client’s computer to the cloud. There will be no centralized point of attack using political or legal attack vectors. All incentive payments for both resource providers and consumers will automated and made in Storjcoin, a pseudonymous cryptocurrency. It is time for the cloud to truly become a cloud, made up of a vast multitude of resource droplets that are added and subtracted as the cloud forms, moves, and changes shape.

The overarching design principles enabling a decentralized storage network have been known for several years. Projects like MaidSafe [1] and Tornet [2] have outlined possible solutions. Unfortunately, achieving the security, scalability, and cost efficiency of a truly decentralized storage system will require software of immense technical complexity. We must design the nodes and network in an extremely secure manner as we can trust neither the lines of communication nor the nodes themselves. Nodes on the network must collaborate to achieve the level of redundancy and performance of current centralized networks. Furthermore, the software must run in an unmanaged environment, very different from the current cloud networks.

All of these are solvable problems using a variety of existing technologies such as Bittorrent Sync[3], Bitcoin[4], public key encryption, and cryptographic hash functions. Storj aims to simplify the development of a decentralized storage network by allowing integration with other existing open source projects in a modular fashion. Such a network must be built incrementally, with software that consists of interoperable, easily replaceable modules that support a wide variety of hardware, including the hardware as a service model that existing cloud storage service providers offer.

Using a model similar to that employed by Bitcoin, where miners are paid block rewards for contributing cryptographic hashing power resources to the network, the Storj network can pay Storjcoin to network nodes that provide storage space and bandwidth to the network. This

model harnesses the powerful free market force of individual pursuit of self interested profit to drive the network to grow and become more efficient while remaining decentralized. For example, if another faster method of file transfer is found the network will gravitate toward that method, until someone produces an even faster method. Nodes must be able to work in a constantly changing and improving environment.

BitCumulus

BitCumulus serves as the non-technical user interface for Storj as well as a development platform for the storage network. Using the BitCumulus web interface, users may securely upload their files to the Storj network and download any file stored on the Storj network. Files are encrypted, preferably through the client side of the interface during the upload process using the private key provided by the user. If a user has not used BitCumulus before, the web interface may offer to assist the user in generating a private key. BitCumulus communicates with the Storj network to find available storage resources, then transfers the preferably-encrypted file to three (3) separate locations to maintain the three times (3x) redundancy considered the industry standard for cloud storage. When a user uploads a file, a record of the upload is made in the Storjcoin blockchain.

We have devised a recordkeeping method that offers efficient functionality for the blockchain as a data store. After the file is encrypted during the upload process, a 256 bit Secure Hashing Algorithm is applied to the file post-encryption, which produces a hash of the file that serves as both a unique identifier and a way to detect file tampering. If any alteration of the file occurs after it is uploaded, the hash will be different, so the network can spot check files to verify their integrity without ever having access to the plaintext of the file. The hash will be stored in a blockchain entry along with the 3x storage locations of the file used to generate the hash. Currently the locations are stored as full URLs pending the emergence of a decentralized solution for URL shortening, which is a source of potential collaboration with other cryptocurrency projects.

All files are secured from unauthorized reading, copying, or theft by the use of public key encryption. Because all data entering the network is encrypted, bad actors and malicious nodes are powerless to spy on, copy, or modify the data because they will not have the decryption keys.

Security and Anonymity

BitCumulus relies on client side javascript encryption to secure the file in the client's browser. Only the user has access to the decryption keys for a particular file. Furthermore, the hash of the file used to identify the file while in storage is taken after the file is encrypted. Therefore, an attacker cannot verify the contents of a file on the network, even if the file may be known. As the file is encrypted before transmission, BitCumulus is also resistant from man-in-the-middle attacks and may be used on compromised data lines.

Security - Client Side Fail Case

Due to current browser limits, it may not be easy to encrypt files larger than 1MB with client side javascript [5][6]. This is acceptable within the bounds of the early BitCumulus project. In the worst case, files can be encrypted on the server side, as long as its made clear to the user that this method of transmission is not guaranteed to be secure. In the case where users do want full file security, a small program can be used to encrypt the file before uploading to the node.

Cost

To simplify initial implementation, BitCumulus is currently using dozens of free and public filehosts as storage providers. We are also actively working to implement Maidsafe [1] support. Using plowshare [7], a command-line (CLI) download/upload tool for popular file sharing websites we can simplify the task of uploading to these services. Implementation plans also include allowing users to connect their own storage hardware from personal or enterprise computing devices. Because all three of these storage providers are zero cost to the BitCumulus node, the only marginal operating cost is that of the bandwidth and upkeep for the operation of the node as it shuttles data back and forth between the user and the storage providing hosts.

As an example test case using the VPS provider Digital Ocean, which could easily host a BitCumulus node, we can do some cost estimates. For $5 per month we are provided with a maximum of 1 TB of transfer [8]. This works out to approximately $0.0049 per GB at full utilization. To put this in context, 100 GB would cost us a total of $1.47 to store, with 3x redundancy, and $0.49 to fully retrieve, while Dropbox charges $99/year for that same 100GB of storage. Unlike a service like Dropbox, BitCumulus is pay per usage. This means that the user only pays for storage fee bandwidth that they actually use, at the time of use.

As storage media capacity increases at an exponential rate, doubling every 12 months, it has become industry standard practice for cloud storage providers to store files for long periods of time and continually lower their prices per GB to consumers. Therefore under our full 100 GB usage example, the cost of $1.47 per 100 GB for the first year could easily be $0.74 and approaching zero for the second year and ongoing years. This is competitive with centralized file hosts because even if their cost for storage media halves each year, their ongoing operating costs in data center rents, employee salaries, accounting costs, regulatory burden, legal fees, etc. will remain fixed or increase year over year, limiting their ability to compete with a decentralized model that has no such costs.

Reducing Cost and Increasing Profit

Using unmetered nodes may further reduce the cost for bandwidth, but it is unclear how a hosting provider would respond to the large amounts of data that a BitCumulus node could generate. If a node is deleted or destroyed it has no effect on the network or file availability, so nodes are inherently disposable. This disposability is a cost reducer because there is no need to expend overhead costs insuring against specific node destruction, only continual creation.

Another example use case would be a node running on Dreamhost [9], which offers an unmetered VPS node for $15 a month at 9.8 MB/s of transfer. Let's take a conservative estimate and limit our transfer rate to 5 MB/s. Under full utilization our node could then transfer 12,840 GB per month. At 3x redundancy per file, selling storage space at the Dropbox prices mentioned above creates a potential gross profit of $357 per month for a node operator.

To be more fair to the hosting provider and avoid abusing shared hosting providers, another example use case is an unmetered dedicated server provider like Hivelocity [10]. Their fee for a 1 Gbps unmetered connection allows an estimated 330,000 GB transfer for a total of $638 per month. Again using 3x redundancy and Dropbox prices we achieve a gross profit of $36,666 per month for the node operator.

Ordinary Users and Sunk Costs

We can also consider the profits of an average residential high speed bandwidth user running their own public BitCumulus web node from their personal computer. In this case, the average data speed is 2.0975 MB/s [11]. Using Dropbox prices and a 3x redundancy, this user would see a gross profit of $149 per month. In this case the user has already chosen to pay for the internet connection and personal computer as sunk costs. Marginal costs are quite minimal to such a user considering that extra storage space, electricity, and bandwidth usage are just a small portion of the total cost of their entire home computing budget. This profit potential should drive BitCumulus and Storj adoption to create a powerful network effect that benefits all users of the network by bringing cloud storage costs down significantly.

Many potential residential users might be technically uncomfortable using BitCumulus. There will be advanced users that will be able to understand cryptocurrency, but others may not wish to climb that learning curve yet. To allow for rapid network growth and adoption while we wait for mass adoption, we propose desiging BitCumulus and the distributed storage network so that it can be a lower level service upon which user-friendly higher level abstractions can operate without confusing ordinary users with the messy low level details.

An example high level service, which we can choose to call Filebox, would offer similar functionality to a service like Dropbox. Filebox could have its own simplistic cloud storage software, and charge a competitive rate for file storage. On the backend, Filebox could run entirely through BitCumulus nodes, eliminating most startup and fixed costs that such a service would otherwise incur. It could purchase credits for its network usage while keeping its own caches of some of the most used files. Filebox would offer much needed abstraction and simplicity to the end user who may not even realize that they are using the Storj network of BitCumulus nodes. In return Filebox would enjoy low market entry costs with an ongoing profit margin over other centralized cloud storage provides. It would only need to maintain the top level software and not the underlying storage hardware.

Storjcoin

Storjcoin is a proposed cryptocurrency whose blockchain contains a distributed hash database for the Storj network. The design goal for Storjcoin is for it to act as a kind of sound money. To do this, it takes all the code and functionality that has already been developed for Bitcoin that makes bitcoin such an excellent form of money and extends it to provide backing by something of fungible value, cloud storage, to lend that money a soundness beyond mere social acceptance. Storjcoin will be a deflationary currency by design, meaning that over time the amount of cloud storage increments that can be purchased for each storjcoin increment will grow as more resources are added to the storj network and its value will increase.

Storjcoin will allow users to pay for bandwidth and storage on the Storj distributed storage network through BitCumulus portals. The Storj software will act as an autonomous agent market maker setting the price, in storjcoin increments, for each storage and bandwidth increment. The Storj software will then pay the providers of storage and bandwidth storjcoin increments periodically for continuing to provide these resources. We propose testing and difficulty functions to accomplish this design goal in the section entitled Proof of Resource, below.

Through these programmed automatic behaviours, Storjcoin acts as an autonomous agent providing value and incentives for users and hosting providers to maintain the network. As a cryptocurrency, we can also use the Storjcoin blockchain not just to store currency balances, but as a data store.

This was previously attempted by the Datacoin [12] cryptocurrency. With Datacoin, “data is stored in the blockchain forever and can be retrieved using a transaction hash as an identifier [12].” Unfortunately, this lead to a problem known as block chain bloat. Every full node must store a copy of the blockchain. That means with users only storing a few megabytes of data each, the blockchain will quickly scale to an unmanageable several gigabytes. Just storing a single movie file would cost the uploading user thousands of dollars worth of Datacoin and flood the network. To its credit, once a file is stored on the Datacoin network it is near uncensorable, which may be important for data that must survive any potential collapse of global civilization, such as an encyclopedia of mathematical principles and scientific knowledge, but is not necessary for data of a more personal nature.

Metadata

We address blockchain bloat issues by only storing a small amount of metadata information about each file. We store its hash, file location, and any other information that we deem essential. A sample of this metadata might look something like this:

```
{
  version: "0.1",
  datetime: "1391212800",
  filesize: "23124",
  file_hash: "6e163442e29ec8d7538bc86fe2c4a48778e8ae2254632f0889da753b1c357b1b",
```

```
  "uploads": [
  { "host_name":"mediafire" , "url":"http://www.mediafire.com/?qorncpzfe74s9" },
  { "host_name":"rapidshare" , "url":"http://rapidshare.com/files/130403982" }
  ]
}
```

This sample metadata, regardless of uploaded file size, takes up about 330 bytes of information. In that case, we could store 3.2 million files before the metadata in our blockchain reached 1GB in size. By further refinements that minimize the amount of necessary metadata information, using multiple chains, blockchain pruning, and compression, we can greatly increase those numbers.

Proof of Resource - Bandwidth and Storage

Bandwidth is the foreseeably more scarce resource than Storage looking forward when we apply Nielsen's Law, which predicts doubling every twenty one (21) months of high speed residential internet and compare to Kryder's Law, predicting storage capacity will double every twelve (12) months. In order to maintain our development goal of Storjcoin acting as a new form of sound money, we must write the proof of resource function with a separate difficulty adjustment for each required resource to account for differing rates of increase and methods of measurement. As more resource providers connect to the network, the amount of bandwidth-connected storage that a storjcoin increment can buy will increase. In concert with this adjustment of difficulty, the amount of bandwidth-connected storage that a storage provider must connect over a period of time to obtain a storjcoin increment will also increase.

This will enable a free market to set the price of storjcoin based on the difficulty of obtaining storjcoin by providing bandwidth-connected storage hardware to the network in the same way that bitcoin prices are in part determined by the cost of mining bitcoin. Because Storj will be designed to work with existing file hosting services, the Maidsafe network, and custom hardware storage solutions from miners, we expect the proof of resource function to be one of the most complex and technically difficult for our developers to implement. Development in this area is ongoing in coordination with the Maidsafe project.

The authors believe that the largest supercomputer in the world in terms of electric power consumption and processing power, which is currently being used to compute arbitrary numbers in a guessing game to prove its own existence, could be better used to provide a useful service in its own right, like storing data. The scope of the vision is commensurate with the difficulty of the problem, but it is a problem we expect to be able to solve within a matter of several months, not several years.

The current state of development simply gives storjcoin when a storage node is used to the owner of the node and charges storjcoin for uploading data, but further development modules could include the following: 1) A speed test mechanism to send spikes of encrypted garbage data through the network periodically to determine actual bandwidth. 2) A storage test mechanism using a stream of garbage data flow generated autonomously by the network and

testing their provision of storage space. 3) Metering autonomous data flow generated by the housekeeping functions that maintain file storage redundancy as nodes join and leave the network. In all three these instances, this garbage and housekeeping data is indistinguishable from real data due to encryption and gives nodes plausible deniability because they cannot know whether the data stored on their machine is auto-created garbage or user-uploaded data. This indistinguishable nature of the autonomous and testing data also prevents nodes from gaming the system to receive tokens without actually providing resources.

Decentralized Development Bootstrapping

A persistent problem for open source software projects that disavow the traditional model of profiting from centralized ownership of intellectual property is inventing a way to pay for development costs without ceding executive control over project goals to multinational corporations or government agencies. From the authors' paradigm of desiring rapid technological and scientific progress from which all humans can potentially benefit through sharing innovative solutions and knowledge, 19th century concepts of property rights in copyrights, patents, and trademarks are outdated, of no use, and potentially destructive. By definition the idea of a decentralized storage network and the code that executes that idea and allows the network to operate autonomously must be not be owned by anyone ever. Therefore Storj must be resistant to any claims of ownership by the developers that write its code and the community that uses the network.

The authors intend to have a crowd sale event for Storj software. All those who wish to have user access to the alpha phase development builds along with community participation and voting rights may make an advance purchase during the crowd sale event. The authors intend to use the pre-sale to recoup costs of development thus far, but mostly to establish a Developer Pool and a Community Pool, with the respective groups of developers and community members proposing and voting on the best use of pool resources. The authors intend to establish a non-profit legal entity to act as the custodian of the pre-sale, which entity will have by-laws that oblige it to act in accordance with the respective Pools' wishes in allocating resources to further project development progress. The goal is for the non-profit legal entity to act in the public interest and seek to phase out its own existence over time as its functions cease to be necessary due to further developments within Storj and the field of decentralized autonomous organization research.

The crowd sale customers will receive a special code that unlocks their share of a cryptocurrency commensurate with the amount of their purchase. This pre-alpha prototype of Storjcoin, which will work with the initial implementations of BitCumulus and the Storj network, but lacks implementation of some Storjcoin's proposed features, will be launched at the conclusion of the crowd sale event. As much as is practicable, all relations between developers, the community, and the project code will someday be governed by smart contracts. However, as smart contract research is still in its seminal stages, many functions of project development oversight will still need to be carried out through more traditional means of human interaction for

the time being. The goal is for all early customers to be rewarded for their trust and loyalty in all later iterations of coin development, with additional pre-order codes made available for free or at preferential rates in the event that a transition to a new blockchain is ever required in moving from this prototype cryptocurrency to beta phase Storjcoin itself.

References

[1] “Distributed Platform, MaidSafe,” http://maidsafe.net/, 2014
[2] “Tornet - Generic P2P Tools,” https://github.com/bytemaster/tornet, 2013
[3] “BitTorrent Sync,” http://www.bittorrent.com/sync, 2014
[4] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” https://bitcoin.org/bitcoin.pdf, 2009
[5] M. Angelov, “Creating a File Encryption App with JavaScript,” http://tutorialzine.com/2013/11/javascript-file-encrypter/, 2013
[6] D. Pogorzelski, “Client-side File Encryption,” http://haanto.com/client-side-file-encryption/, 2013
[7] “Plowshare - Download and upload files from file sharing websites,” https://code.google.com/p/plowshare/, 2014
[8] “Pay-As-You-Grow Pricing,” https://www.digitalocean.com/pricing, 2014
[9] “Virtual Private Server,” http://www.dreamhost.com/servers/vps/, 2014
[10] “Unmetered Dedicated Servers,” https://hivelocity.net/dedicated-servers/unmetered-dedicated-servers/, 2014
[11] “Download Speed by Country,” http://www.netindex.com/download/allcountries/, 2014
[12] “Datacoin,” http://datacoin.info/index.php?id=index, 2013