COVER PAGE

Overview: Subspace is a decentralized database which allows developers to easily manage application state within a decentralized app, or *dApp*. *dApps* provide services over the Internet without a central server. Current constructions rely on blockchain based smart contract platforms such as Ethereum for application code and Decentralized Storage Networks (DSNs) such as the Interplanetary File System (IPFS) for file hosting. Mutable application state management, in the form of a decentralized database, remains an unsolved problem. In Subspace, data is stored across a peer-to-peer (P2P) network of end user devices who act as hosts by pledging free space in return for subspace credits. The initial target audience includes developers building decentralized applications such as a decentralized social networks, or any app that requires regulatory compliance for the storage of end user data.

Terms: Peer-to-peer (P2P) networks, decentralized applications (dApps), Distributed Ledger Technologies (DLT), blockchain, Decentralized Storage Network (DSN), database, gossip protocol, Distributed Hash Table (DHT), Web real-time communication (Webrtc), Proof of Space (PoS), Proof of Replication (PoR), General Data Protection Regulation (GDPR)

Subtopic: IT6 Networking Technology

Intellectual Merit: This Small Business Innovation Research Phase I project addresses the key problem of developing a new network protocol that allows for a low-latency, secure, persistent and highly-available, decentralized No-SQL database which may operate at Internet scale. Research will be conducted over a 12 month period with an experienced team of three decentralized app and protocol developers through a $225k Phase I award. Key objectives include developing a WebRTC overlay network that combines a Kademlia DHT and an anti-entropic gossip protocol, such as Scuttlebut; constructing a proof of space blockchain; creating a secure Decentralized Storage Network utilizing a proof of replication based on verifiable delay functions; and implementing and testing the protocol across common host devices and client runtimes. The end goal of the project is to validate the technical feasibility, scalability, and market interest for the subspace protocol.

Broader Commercial Impact: Once the protocol has been validated, subspace will be ready for commercialization. Subspace will first bootstrap the network, generating revenue through the sale of space through it's initial monopoly on credits, and then taking a small transaction fee on all storage payments between clients and hosts. In time, Subspace will build and sell custom consumer hardware optimized for the hosting and farming on the subspace network and eventually operate a data marketplace that will allow users to self-monetize their data. Broader impacts include hastening the spread of dApps by providing a scalable and affordable state management solution, reducing the dependence of developers on centralized cloud services and allowing for the widespread deployment of applications that allow users to have full ownership and control over their data.

ELEVATOR PITCH

Customer: Ownership and control over user generated data on the Internet is one of the defining social and technical issues of our time. These concerns have led to a renewed interest in decentralized web technologies among **application developers** who want to build services which allow users to own their data. Blockchains and other Distributed Ledger Technologies (DLTs) have been proposed as a solution to this problem. Despite their potential, blockchains still present challenges around scalability, privacy, and usability. Moreover, blockchains only allow for one pattern of data management, an immutable append-only log, which is not suitable for all application use cases. The Subspace protocol presents **developers** with a simple and familiar put, get database API that allows them to easily manage state in a decentralized application (dApp). Data hosted on the Subspace network is sharded across a network of end-user devices. These hosts pledge local disk space to the Subspace network to earn Subspace credits, which they receive for hosting and serving data for Subspace enabled apps. **Developers** pay Subspace credits to reserve space on the network. These payments are encoded as smart contracts on a proof of space ledger. Importantly, the data itself is not stored on the ledger, only the protocol token used to incentivize participation in the network.

Value Proposition: Subspace abstracts away many of the challenges inherent in building decentralized protocols that developers face such as: designing a cryptographic storage schema; creating a peer-to-peer (P2P) overlay network; managing distributed consensus; creating a ledger and protocol token; and bootstrapping a network of nodes. This **allows developers to easily build dApps**, which are more secure through the use of public key cryptography, digital signatures and native encryption; are more reliable, since data is replicated across many devices and self-healing; and **that allow end users to fully own and control their data**.

Innovation: The primary innovation and key differentiator in Subspace is its ability to work natively across all application platforms including browser, mobile, and desktop environments, while providing fast reads and writes (less than 300 ms) at Internet scale. This is a two part problem. First, Subspace must be built around WebRTC, the only transport protocol that allows browsers to communicate directly with other devices on the network, creating a device and environment agonistic transport layer. This involves rewriting and optimizing existing primitives such as the Distributed Hash Table (DHT) and gossip protocol for WebRTC. Second, the entire protocol must be written in JavaScript so that it can run in the browser by default and be ported to mobile (via Cordova and React Native), desktop (via Electron), and the server (via Node JS). This allows for portability of host and client implementation across all devices and runtimes, solving the critical problem of usability that exists in this space today. Furthermore, Subspace implements a dual-purpose proof of space, that allows for verification of data storage between hosts and clients, and serves as the consensus mechanism for the ledger. This serves to incentivize both large and small nodes in a way that other networks fail to do.

COMMERCIAL OPPORTUNITY

Broader Societal Need: Today, the average consumer has come to rely on the Internet as their primary means of communication and exchange. Services that provide these functions are dominated by a handful of large technology companies who offer free access in return for ownership rights to user generated data on their platforms. There are two inherent problems with this model. First, the user becomes the product, in the form of the data they generate. Companies sell this data to third parties for pennies on the dollar compared to service and hosting costs. Second, users must trust these companies to be good stewards of their data. The data must be kept safe from hackers, it must be kept private from government agencies, and the company must exercise discretion with regards to what purposes the data is used for. This trust has been and continues to be breached at an alarming rate. Subspace addresses these broader societal needs by providing a new framework for application developers and entrepreneurs to provide services over the Internet, where the user does not have to trust anything but the protocol itself. Services built on top of Subspace clearly separate the data model from the application logic, allowing users to maintain full ownership and control over the data they generate through their Subspace private key. Users can then decide who they wish to share their data with, if and how much to sell their data for, and may easily move their data between applications.

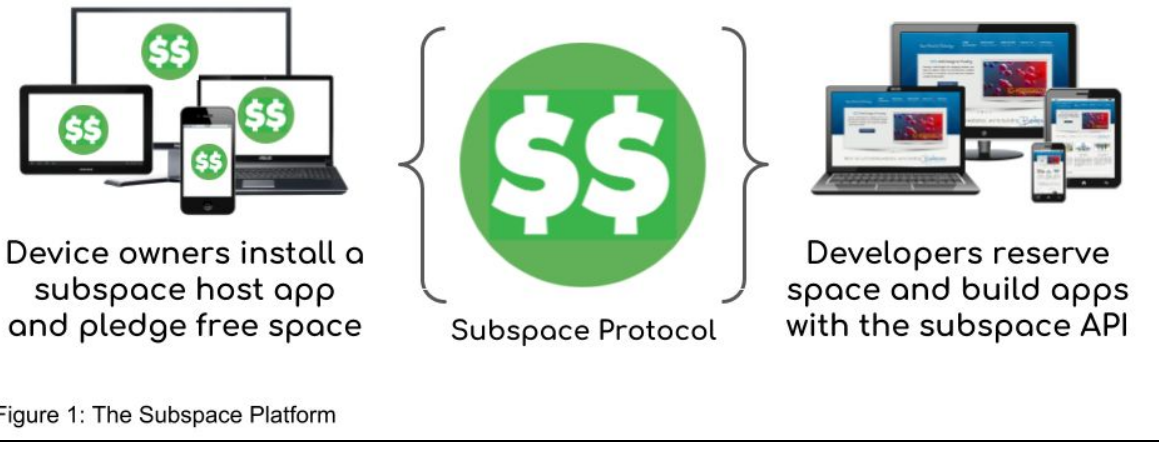

Figure 1: The Subspace Platform

Total Addressable Market: The Subspace protocol facilitates a two-sided marketplace (Figure 1) which connects device owners (hosts), who wish to monetize free space, with developers, who wish to reserve space for generated data by their applications (clients). On the supply side, the total addressable market consists of all end user compute devices connected to the Internet, including smartphones, tablets, laptops, and desktop computers. While this market is enormous, it may only be obtained in proportion to demand for the service. The demand side can be modeled as a decentralized form of Infrastructure as a Service (IaaS). In 2017 this was a $30B market, comprising the fastest growing sector of the cloud computing market with a 35.9% annual growth rate [1]. While the market size for cryptographic protocols can be estimated,

measuring developer adoption is more difficult. One indicator is the number blockchain related jobs postings on Upwork, which has seen 6,000% increase over the last year [2].

Market Economics: The fundamental economic driver of the platform is the growing demand for applications which allow users to own their data, from both the users themselves and regulations such as the General Data Protection Regulation (GDPR). Application developers and entrepreneurs sensitive to these demands will seek out new frameworks like Subspace that allow them to provide these services back to the end user. As developers reserve more space on the network, the rewards for hosting will increase in accordance with the basic laws of supply and demand. As hosting rewards increase, more hosts will be incentivized to join the network. This will create a virtuous cycle whereby more users lead to more developers, more developers lead to more hosts, and more hosts lead to more users.

Business Model: The initial direct customers of Subspace. are application developers who wish to store data for their users on the network. The indirect customers of Subspace are device owners who pledge free space to the network in return for Subspace credits. Initially, Subspace Inc. will own the entire supply of Subspace credits, placed in the Subspace credit reserve, through a private pre-farm of the Subspace credit ledger. Initially, developers (later, users) will pay Subspace, Inc. for API keys, created by locking credits from the reserve into a client data contract on the ledger. These credits will eventually be distributed to hosts, who may then use them to create their own private data contracts or trade them to developers or other users through the Subspace exchange. Every time a host receives a disbursement of Subspace credits, Subspace will apply a 10% transaction fee, replenishing the Subspace reserve.

Market Validation: Subspace Inc. has validated demand through a set of marketing experiments conducted in the Spring of 2018. Developers were found at a series of cryptocurrency and decentralized web meetups and events in the San Francisco Bay Area. 15 individual developers working on dApps expressed strong interest in using the Subspace protocol, joining the developer wait list from the Subspace website. All of these developers were also interested in adding their spare devices as hosts on the network, demonstrating that the two-sided marketplace can initially be collapsed into a single group. In addition, Subspace has obtained letters of support from three enterprise software development companies all working in the larger decentralized space. Subspace has also identified 35 cryptocurrency early adopters who have joined the early host wait list.

Competitive Analysis: Existing and planned offerings for managed No-SQL database services can be broken down along two axes: centralized vs decentralized and mutable vs immutable (Figure 2). Centralized mutable databases such Amazon DynamoDB and Google Firebase currently dominate the market [3]. At the other extreme are decentralized immutable databases (blockchains) like Bitcoin [4] and Ethereum [5], where the high cost of storage makes it impractical for most use cases. Distributed immutable database like BigchainDB [6] and FlureeDB [7] offer the same guarantees of immutability at a much lower price, but at the cost of more centralization. Finally there are decentralized mutable databases like Subspace [8] that

rely on blockchains for incentives but store data off-chain. Wolk [9] and Bluzelle [10], both still in development, are tightly coupled to the Ethereum Swarm storage network [11, 12], making them most appropriate for Ethereum dApps. Xuma [13], also in development, is based on Proof of Stake, which incurs more centralization and higher entry costs for participation through deposits.

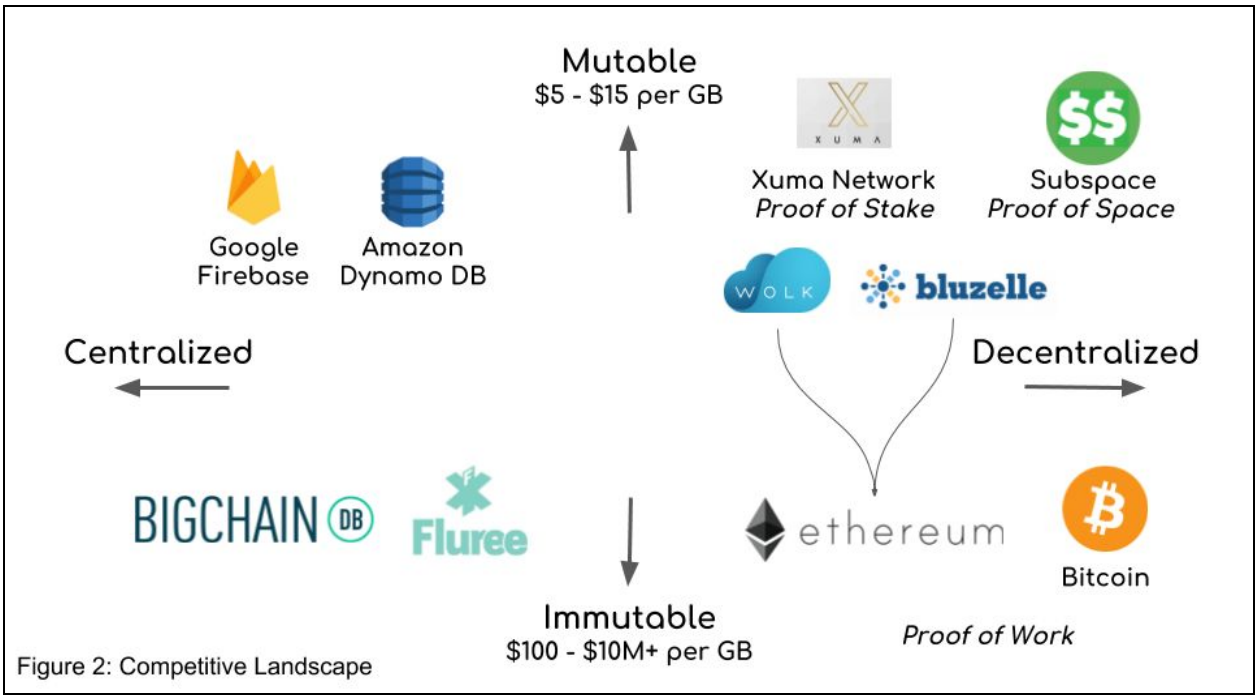

Figure 2: Competitive Landscape

Subspace aims to provide performance similar to a Firebase or Dynamo at close to the same price, while still offering high levels of decentralization (through Proof of Space) that appeal to users and dApp developers. It is important to note that Subspace will not compete with existing DSNs such as Sia [14] and IPFS [15], or those in development such as Storj [16] and Filecoin [17]. While they both expose a key-value (put, get) API, these DSNs are optimized for large files and incur a high network overhead that makes them impractical for the frequent, smaller reads and writes associated with a transactional database. The competitive landscape is expected to change dramatically over the next 12 to 18 months as many of these protocols come online and are ready for developers.

Key Risks: The key risks associated with bringing Subspace to the market include: a) Technical challenges associated with building a scalable DHT that works over WebRTC and securing the network with a Proof-of-Space blockchain. b) The willingness of developers to build apps on the platform and their ability to gain traction and long term support from end users. c) The ability of incumbent platforms such as the Google Play and Apple App stores to block app distribution or develop competing frameworks like Subspace d) Launching too late, after a competing decentralized storage network or Proof-of-Space blockchain has aggregated slack supply of spare space.

Estimated Revenue: Estimated revenue (Figure 3) is based on three assumptions: a) developers will pay $10 per GB - Month for a decentralized No-SQL database service. b) the growth in space reserved (demand) will follow a logistics curve. c) Subspace Inc. will be able to charge a 5% transaction fee on all storage payments processed on the network. Annual storage payments are calculated as the product of the average storage over the year, the number of months in the year and the price of storage. For example, year one revenue equals 5,000 GB * 12 months * $10 per GB - Month. Transaction fees (revenue) are simply 5% of storage payments in any given year.

Year (after launch)	**1**	**2**	**3**	**4**	**5**
Space Reserved	10 TB	100 TB	1 PB	10 PB	100 PB
Storage Payments	$600k	$6M	$60M	$600M	$6B
Tx Fees (revenue)	$30k	$300k	$3M	$30M	$300M

Figure 3: Projected Revenue

Commercialization Plan: Subspace, Inc. will commercialize its technology by raising venture capital investment with the long term goal of holding an Initial Public Offering (IPO) (Figure 4). The company was founded in November 2017 and raised a small pre-seed equity financing round through a private placement of $25k in January 2018. Subspace is seeking seed funding through NSF-SBIR Phase I to develop its core overlay network technology for the protocol and will next seek Phase II funding to help commercialize this technology and launch the protocol. After the protocol has gone live with demonstrated traction and revenue, Subspace will raise a Series A equity financing round through a private placement to quickly scale operations. Subsequent rounds will follow until the company has demonstrated sufficient revenue and traction to proceed with an IPO.

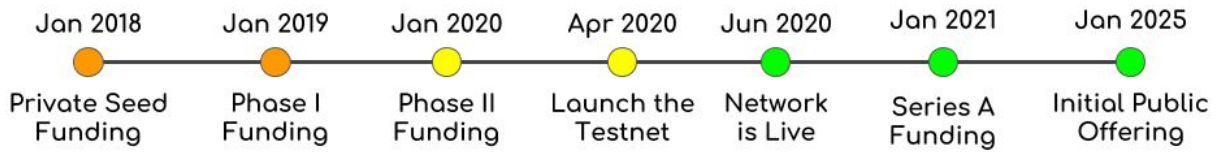

Figure 4: Commercialization Plan

THE INNOVATION

Overview: The subspace protocol creates a decentralized storage network that allows users to have full control over data they generate on the Internet without going through a remote server. Users may host their data directly on their own devices, while replicating it across other devices on the network in a secure and persistent manner. To developers, Subspace presents a decentralized key-value store with a familiar JavaScript API. Devices running the Subspace protocol form a peer-to-peer network connected over WebRTC that is accessible via any browser, mobile, or desktop app. Data is replicated across multiple devices where it is encrypted at rest under a cryptographically secure schema, using a verifiable proof of replication. Hosts pledge free space to the network in return for regular distributions of Subspace credits from client application and users. These agreements are encoded as smart storage contracts on a distributed ledger secured by Proof-of-Space consensus. Subspace aims to be a simpler construction of a decentralized storage network focused on developer usability which seeks mass adoption by end users.

Background: Subspace began as an effort to adapt a traditional Software as a Service (SaaS) web application, *OmniBuilds* (www.omnibuilds.com) into a decentralized application which separated the data and application layers, allowing the users to own their data. This came at the request of several existing and prospective customers. After initially examining blockchains and experimenting with platforms such as Ethereum and IPFS, OmniBuilds was rebuilt from scratch as a purely decentralized application, *Hex*, modeled after the software revision control protocol *Git* [18]. After creating a POC for *Hex*, and getting further customer feedback, it was realized that simple state management, in the form of a persistent and secure mutable database, remained an unsolved problem in the decentralized application space. At this point the state management layer for Hex was abstracted in to a separate protocol, leading to Subspace.

Current Stage: A Proof of Concept (POC) has been written in Node JS which combines several open-source frameworks and libraries into a novel construction [8]. While this POC achieves the basic database API and meets the usability aims of Subspace, it has several shortcomings which prevent it from being used in a production environment. Since there is not a working open-source implementation of a Kademlia DHT over WebRTC, two different frameworks were combined to create one, at the cost of doubling the O(log N) latency already present in Kademlia [19, 20]. This prevents the POC from scaling beyond a few hundred nodes while maintaining sub-second lookups (get). The POC is still susceptible to a variety of attacks that must be accounted for in any DSN, discussed later. Finally the POC only includes the storage and retrieval protocol. It does not include the proof-of-space ledger and smart contract layer needed to incentivize participation by hosts.

Key Technical Challenges

1) Create a WebRTC Overlay Network: Subspace relies on a variety of cryptographic data structures to ensure that the protocol can work in a decentralized and trustless manner. A Kademlia inspired sloppy Distributed Hash Table (DHT) allows nodes to announce (put) new records to the network and lookup (get) existing records from the network [21, 22]. A Scuttlebut inspired gossip protocol [23, 24] is used to ensure that all nodes have an accurate picture of the current membership pool of all nodes and their uptime, necessary for the replication and load balancing of data across hosts and the accurate disbursement of hosting rewards. Both of these protocols were designed before the advent of WebRTC and have almost universally been implemented using TCP or UDP. WebRTC presents a unique set of challenges and opportunities for decentralized networks. It allows for a universal transport layer across devices and runtimes, but at the cost of persistent connections which add complexity and communications overhead. For the Subspace protocol to work as envisioned, a new class of overlay network needs to be designed from scratch that combines elements of a DHT and gossip protocol in a manner that is optimized for WebRTC and can work at Internet scale. For example, the BitTorrent file sharing protocol adapted Kademlia in a way that once accounted for up to 40% of all Internet traffic [25]. Creating this new overlay network is the primary technical hurdle that this proposal seeks to address during Phase I.

2) Develop a Proof of Space Ledger: Subspace also relies on a proof of space distributed ledger to properly incentivize hosts and farmers with a consensus mechanism that is best suited for a decentralized storage network. A proof of space is a special algorithm that can only be computed given sufficient disk space, first described using hard to pebble graphs [26]. Proof of space was first implemented for blockchain consensus in 2014 in BurstCoin, which has subsequently been shown to be susceptible to grinding and history rewriting attacks [27, 28]. Two other projects, SpaceMint and the Chia Network are each developing different proposals for a secure proof-of-space blockchain, with expected release dates in early 2019 [28, 29, 30]. Subspace will be closely monitoring and implementing developments in proof of space based consensus.

3) **Ensure the Network is Secure:** Decentralized Storage Networks (DSNs) have a large attack surface that must be properly accounted for, especially if financial incentives are present in the network. Some of these attacks can be thwarted easily with existing cryptographic primitives, some may be prevented with theoretical or recently developed primitives, while others remain open questions. Importantly, solving them in a manner that maintains the simplicity of the protocol and does not incur additional communications overhead is critical to developing a protocol that scales. This proposal addresses three key attack during Phase I, the generation attack, outsourcing attack, and hostage attack. Other attack vectors that will likely be pushed to Phase II include node eclipse attacks, broader denial of service attacks against the network, and variants of the Sybil attack that may be used to engage in parallel farming and hosting.

Generation Attack: In the generation attack, a host may return false data to a client who has requested a record by ID. If the client has no prior knowledge of the data's contents, how can it know if the data returned is valid? Luckily a framework for handling this has been proposed for an extension of the BitTorrent protocol to include arbitrary data in the DHT [31]. Under this schema mutable records are associated with a unique public key, with the ID being the cryptographic hash of the key. The record owner controls the private key, which is used to sign the record value, allowing anyone with the public key (embedded in the record) to validate its authenticity. Immutable records are simply stored under the hash of the record value, allowing for simple verification by any client. This proposal has yet to be formally or empirically validated in a production DHT but will be included in the Phase I iteration of the subspace protocol.

***Outsourcing Attack*:** In the outsourcing attack, a host does not actually keep its assigned data, instead hoping another host is acting honestly, since the data is replicated amongst many hosts. When a client requests the data from the cheating host, the host quickly fetches the data from an honest host and returns it to the client as if it had been storing the data all along. Proof of Replication [32] using Verifiable Delay Functions (VDFs) [33, 34, 35] offers a solution through the use of a time-delay encoding, whereby each host must store a unique replica of the data that is encoded with some data which only they have access to, such as the private key associated with their network identity. The encoding function can be configured such that it takes an order of magnitude longer than a reasonable timeout for a network request. Now any host who tries to outsource the data must then encode the data into a unique replica, but will be unable without the client noticing, since they will have exceeded the timeout period for the request. VDFs are extremely new and have yet to be implemented in a production DSN. The Subspace protocol will implement a simple construction during Phase I.

Hostage Attack: In the hostage attack, hosts simply refuse to answer get requests, in effect holding the data hostage. Some proposed DSNs circumvent this attack by requiring clients to embed a payment in each get request which the host can only receive funds after sending the data [14, 16, 17]. This adds significant transactional overhead to the ledger and also presents a poor user experience, since it only allows clients with credits to access the network, as opposed to the pattern of free requests over HTTP users have come to expect today. Another alternative is to employ a traditional byzantine consensus algorithm around signature voting by a small quorum of neighboring hosts [36]. This remains a theoretical proposal that needs to be further explored during Phase I.

Intellectual Property: Since the goal of the Subspace protocol is to create an open network, in the original spirit of the Internet and the World Wide Web, with no trusted central authority or gatekeeper, Subspace Inc. will not seek intellectual property assignment on any algorithms or copyright on any code. Public protocols and blockchains are expected to be open-source and it would receive little support from the community of developers who must first adopt it if the project were closed-source. Instead the protocol and client implementations will be released under the MIT Software License, to allow for the most permissive use of the protocol with the hopes of creating a strong developer ecosystem and community of open-source contributors.

COMPANY & TEAM

OmniBuilds, Inc was founded as a Delaware C-Corp in November of 2017 and received a small pre-seed investment of $25k from SoFI Ventures in January of 2018. The company has not launched any products and has no revenue. OmniBuilds began operating as Subspace, Inc. in March of 2018. Subspace is raising a small seed round to sustain the core team until the Phase I Grant has been awarded. Subspace currently consists of three founding team members.

Jeremiah Wagstaff: Chief Executive Officer (CEO) and lead architect for the Subspace platform. Jeremiah is a self-taught full-stack application developer who has built web, mobile, and desktop apps in JavaScript. He has extensive leadership experience including four years in the Texas A&M Corps of Cadets, eight years as a US Army Infantry Officer, two years as the founder of a small business, and three years as a project manager at two IoT hardware startups. He spent the last year building a service for revision control of hardware engineering design files and documentation (www.omnibuilds.com), eventually developing a fully decentralized implementation as the hex protocol [8] which serves as the inspiration for the Subspace protocol]. His greatest strength lies in building and leading technical teams to bring new products and services to market.

Nazar Mokrynskyi: Chief Technology Officer (CTO) and lead protocol engineer for the subspace platform. Nazar studied computer science at the National Aviation University of the Ukraine, leaving early to cofound an IoT startup, Ecoisme (www.ecoisme.com), and served as the CTO over the next three years. Nazar has gained notoriety as the maintainer and lead contributor to several open-source software projects in the decentralized space including Simple Peer, the leading P2P framework for WebRTC [37]; Entangled State (ES) DHT, a privacy focused DHT implemented in JavaScript over WebRTC [38]; and the Detox network, a secure decentralized application platform built on top of ES-DHT [39]. Nazar has the technical acumen and experience needed to design a custom DHT that meets the needs of the subspace protocol.

John Heeter: Chief Creative Officer (CCO) and lead implementation engineer for the Subspace platform. John studied Art and Product Design at Savannah College of Art & Design (SCAD). He is also a full-stack JavaScript developer who has extensive experience building user interfaces on top of existing decentralized protocols that abstract away their complexity. He has co-founded two companies in the decentralized space including Suchflex and Boid (www.boid.com), both of which simplify the mining experience for blockchains in a way that allow the average user to participate in the network. John will focus on developing the client and host implementations of the subspace protocol.

All team members will work greater than full time at subspace over the course of the Phase I award. Each has gained deep experience developing Internet enabled apps in JavaScript, has worked at and cofounded their own startups, and have independently developed their own decentralized applications. With their complementary skills and experience, this team can solve the technical challenges required to bring Subspace to market.

Vision: Subspace aims to become the dominant player in the decentralized space and a household name for developers and users. To achieve this, Subspace must win the battle for developer mindshare by making the protocol as approachable and useful as possible. The protocol must be implemented across all hosting platforms including Android and iOS for mobile; Mac, Windows, and Linux on Desktop; and in Node JS on the server. The client API must be extended to work across languages and frameworks. Initially JavaScript will provide compatibility in the browser, mobile (React Native), desktop (Electron), and server (Node JS). The protocol must then be extended to Java (Android) and Swift (iOS) for mobile, followed by Python (Django) and Ruby (on Rails) for web, and Go and C++ for desktop and server apps.

***Subspace Web Services*:** The core Subspace protocol will eventually be expanded beyond a simple key-value store. SSDB could be abstracted into a file system (SSFS) by chunking large files across multiple records. SSDB could also store application code, which could then be remotely executed on a host device as a serverless function. Once the subspace includes the key primitives of a database, file system, and compute platform more interesting services may be created such as: a global authentication system, using a public-private key pair (SS-Auth); a content delivery network for static hosting (SS-CDN); a software package manager (SS-NPM); a domain name service (SS-DNS); and an immutable storage platform for distributed ledgers (SS-DLT). The goal is to become a decentralized alternative to a platform like Amazon Web Services that could manage the entire application development workflow.

***SpaceSuite*:** Subspace will also develop its own core suite of dApps using the protocol, targeted at the everyday user. These apps would begin as an extension of the host app, which would also serve as a social wallet. From here many core dApps would be built out including: SpaceBook, a decentralized social network; SpaceMail, a decentralized email client; SpaceGram, a decentralized messenger; SpaceDocs, a decentralized desktop publishing suite; and SpaceDrive, a decentralized file hosting service.

SpaceMart: In order to properly align incentives between users, developers, and third parties, Subspace will eventually launch a decentralized data marketplace. This would allow users to share their data with third parties of their choosing under a revenue sharing model along with the app used to generate the data. For example, a user on SpaceBook could choose to share their data with advertisers, taking 90% of the payment and allocating 10% back to the SpaceBook app wallet. Subspace would take a small transaction fee on these payments.

***BitBot*:** Subspace will also develop custom hardware optimized for the protocol, targeted at the average household. BitBot would be a secure personal storage device that connects to a home router via ethernet and is always online. It would consist of a low-cost computer connected to an array of inexpensive disks, effectively serving as a Network Attached Storage (NAS) device that acts like a Redundant Array of Inexpensive Disks (RAID) [40] device at the level of the Subspace network. The BitBot would have four key features: secure personal storage of user owned data; a hardware wallet for the user's Subspace private key; hosting data for the network with free space; and 'farming' the Subspace blockchain.

TECHNICAL DISCUSSION

Subspace Database (SSDB) is a decentralized key-value store that is massively sharded amongst a network of Internet enabled host devices that may include mobile phones, tablets, desktop computers, and traditional servers (Figure 5). A JavaScript client library with a simple put(), get() Application Programming Interface (API) is provided to developers, allowing them to store and retrieve data inside any web, mobile, desktop, or server-side app. Anyone who uses these apps has full ownership and control of the data they generate, with their private key. Each record stored in SSDB is guaranteed to persist with a replication factor (R) in a manner that is self-healing as hosts leave the network. Space utilization is load balanced across all hosts proportional to the amount of space pledged. Hosts pledge free space to the network in return for regular distributions of Subspace credits, a native protocol token tracked on a blockchain using proof of space consensus. Storage payments are mediated through a series of smart contracts that are also stored on the blockchain. Farmers secure the chain by competing to add the next block and earn the block reward, based on who has the most valid proof of space. The probability of any given farmer adding a new block to the chain is proportional to the amount of storage space pledged to the network.

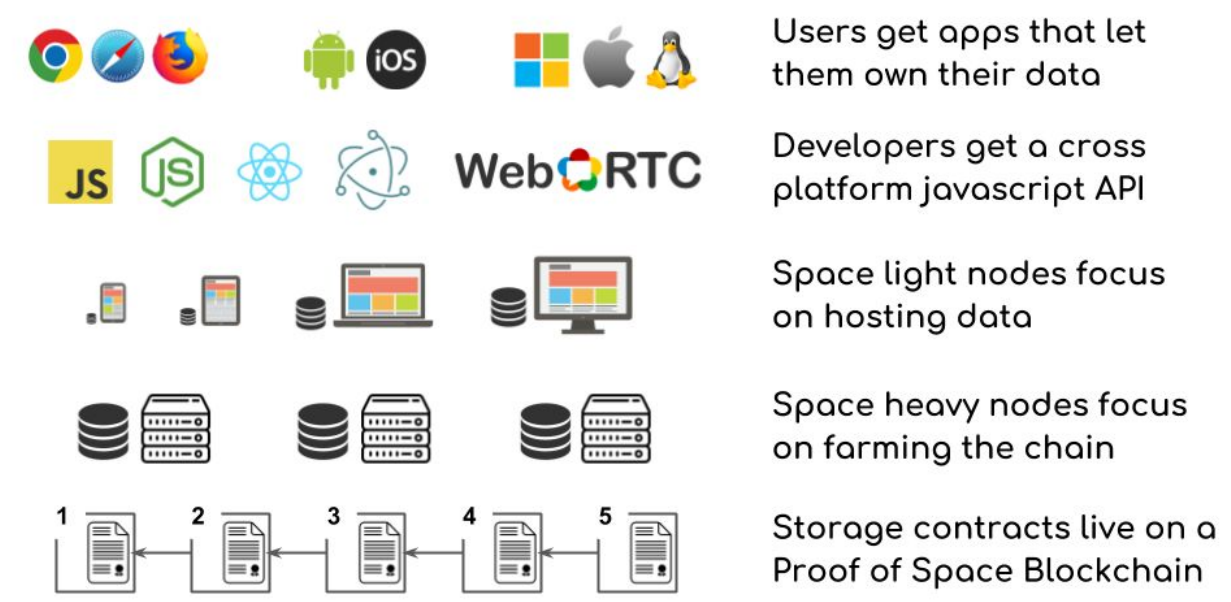

Figure 5: Protocol Stack

***Node Identities*:** Each node maintains a persistent cryptographic identity through an ED25519 PGP key pair. Private keys are stored on the device and public keys are shared with all other nodes on the network. A node's payment and routing address are the SHA256 hash of its public key. The PGP protocol was selected for its ability to handle multi-key encryption, allowing for the creation of fine grained permission schemes for SSDB records.

***Transport Layer*:** Nodes form a relay network of stateful WebRTC data channels inspired by Kademlia, allowing them to communicate with each other using only the node ID [19]. The problem of network address translation (NAT) is handled by implementing the ICE protocol in WebRTC over gateway nodes. These publicly addressable nodes may relay signaling information between devices behind NATs, allowing for direct P2P connections via hole punching techniques [41]. WebRTC also allows for cross-platform connectivity and is the only means of connecting browsers directly to host devices.

***Gossip Protocol*:** Nodes disseminate signed gossip between their neighbors in the relay network including join, leave, failure, fault, and ping messages. These messages are used to disseminate updates to the membership pool, an eventually consistent local hash table maintained by all nodes. The membership pool tracks the current status, uptime, and public key of each node. This allows nodes to evenly distribute put(), balance(), and replicate() requests across the network through a Random Peer Sampling (RPS) algorithm, quickly verify digital signatures, and come to consensus on the uptime for any given node.

***Distributed Hash Table*:** A modified sloppy Kademlia DHT is layered on top of the WebRTC relay network and serves as a distributed index for SSDB [21, 42]. Given a record key the DHT will return an array of node IDs who currently hold that record. Kademlia is extended to only allow nodes to announce records given a valid put() or replicate() request. Records do not expire until any node can provide a valid leave, failure, or fault message. Hosts coordinating a put() request may also multi-announce ownership of records. Clients holding records in their LRU cache may still announce records with a traditional expiry period.

***Record Schema*:** SSDB records are stored locally on each node using a cryptographic encoding schema inspired by the BEP-44 specification (Figure 6), Each record is JSON encoded and includes a data field up to 1 MB in size. Regardless of the encoding format all records are digitally signed by the record owner and must reference a valid data contract ID.

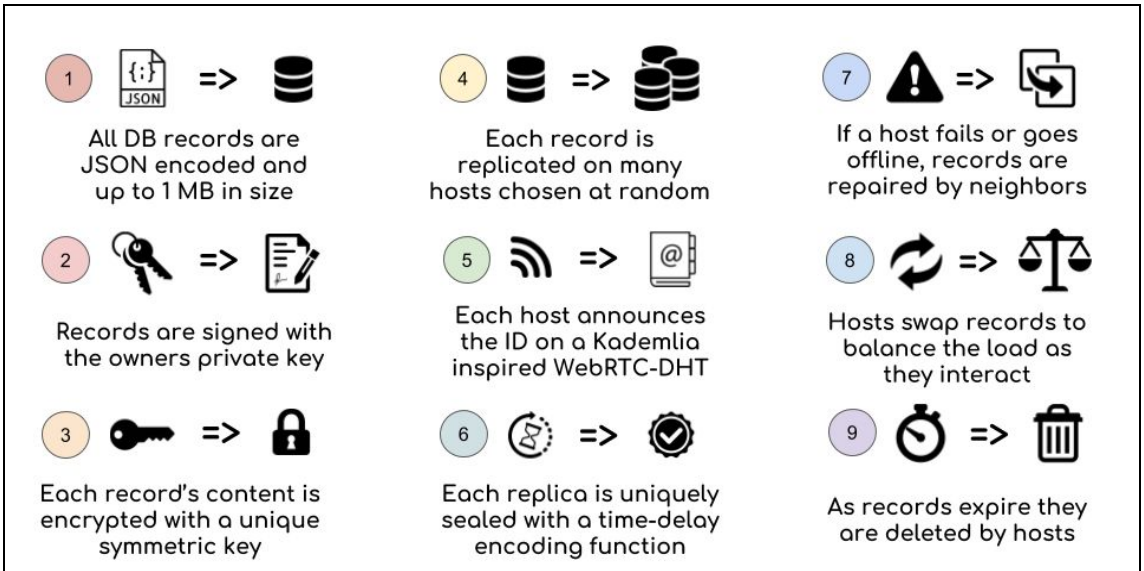

Figure 6: Subspace Database Schema

***Record Encryption*:** Content is encrypted by default with an AES 256 symmetric key that is unique to this record and included in the record schema. The symmetric key is asymmetrically encrypted using the private key of the record owner (for immutable records) or the private key of the record (for mutable records), and the public keys of any other identities that the owner would like to share the data with. Identities may be the key of other nodes or any arbitrary group schema that can be embedded into an SSDB record. It is also possible to specify fine-grained access control with read, write, and admin privileges for each identity.

***Distributed Ledger*:** While SSDB records are stored off-chain in a series of local shards on each node and the record-node mapping is stored on the DHT, a secure method of accounting for Subspace credits between peers is still needed. This is tracked within the Subspace Credit Ledger (SSCL), a blockchain maintained by proof of space consensus, but otherwise similar to Bitcoin. SSCL is also stored on SSDB as a series of linked immutable records.

***Smart Contracts*:** SSCL employs the bitcoin smart contract language, *Script*, allowing for the creation of storage contracts to mediate payments between hosts and clients (Figure 7). When a new host comes online, it will generate a storage contract specifying the amount of disk space and the reward interval. This space is seeded using the host's private key, yielding a dual purpose proof of space. This proof of space is embedded within a storage contract that is self-hosted on SSDB and gossiped to farmers for validation and inclusion in the ledger. In order for host to accept a put() request on behalf of the network, the request must reference a valid data contract. Any node may create a data contract by submitting a valid data transaction to the SSCL which encumbers Subspace credits into an agreement between the contract holder and the Subspace network. The network agrees to store an amount of data based on the Cost of Storage (CoS) published in the last block in return for the ability to pay these credits out to hosts. Client data contract funding transactions will always specify the *to*, or payment address, as the Subspace nexus contract. The nexus contract allows for simple aggregation and redistribution of credits by farmers to hosts. Credits will then be paid out to hosts based on the interval specified in their storage contract.

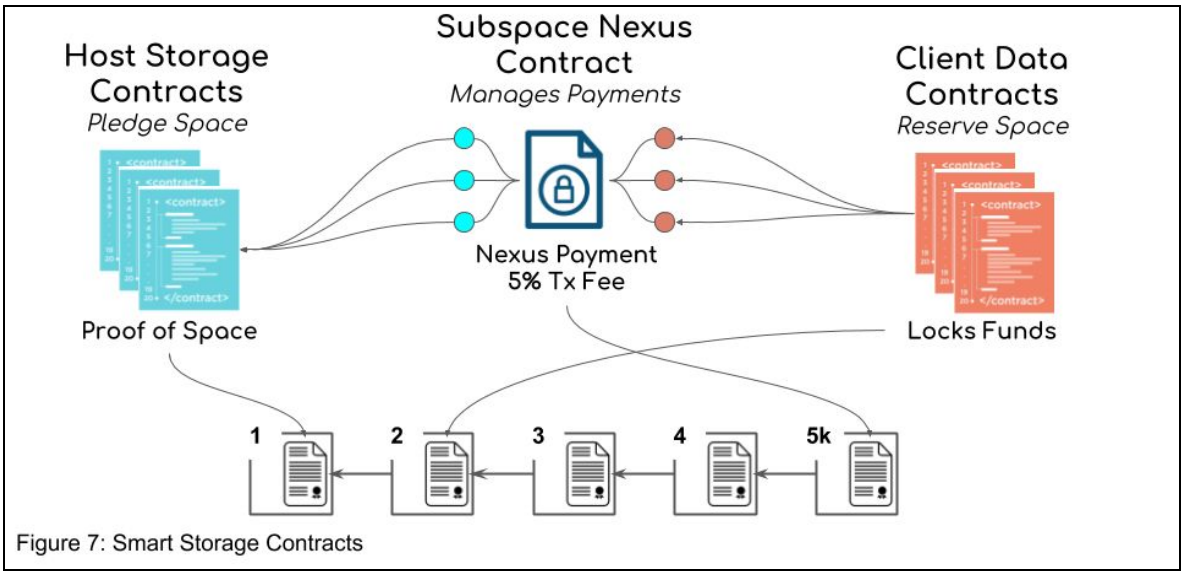

Figure 7: Smart Storage Contracts

Key Objectives & Research Questions

***Design the WebRTC Overlay Network*:** Create the underlying transport and communications layer as a novel implementation of Kademlia over Web-RTC and extend with an anti-entropic gossip protocol and a simple key-value store.

1. Can a DHT and gossip protocol be efficiently combined into a single overlay network using Web-RTC, considering that the POC did this at the cost of O(log(log N) latency?
2. Can the network be designed such that it may reach Internet scale (in the manner of BitTorrent) while maintaining fast lookups (< 300 ms) for a good user experience?
3. Can record state be arbitrated in such a way that it is abstracted away from the application developer? For example, by using vector clocks or conflict free replicated data types (CRDTs)?
4. Will developers pay $10 per GB - Month for a decentralized key-value store?

Extend the Network with a Proof of Space Ledger: Extend the core protocol to store the Subspace credit ledger as a series of immutable records. Implement a secure dual-purpose proof of space that serves the needs of host attestation of space and ledger consensus.

1. Can SSDB be used to store permanent immutable records or does the ledger need to be an independent data structure?
2. Can a secure proof of space ledger be implemented?
3. Can the same proof of space serve the needs of both hosts and farmers?
4. Will hosts and farmers be sufficiently financially incentivized by the market value of Subspace credits to participate in the network?

Secure the Network from Attack: Harden the protocol against malicious hosts employing generation, outsourcing, and data hostage attacks without sacrificing performance.

1. Can a Verifiable Delay Function be tuned for a Proof of Replication such that it works across a range of host device compute performance profiles?
2. Does BEP-44 present the simplest solution to the generation attack?
3. Can a low overhead solution to the data hostage attack be implemented?

Implement and Test the Protocol: Embed the protocol in host apps on each device category and expose an API for each target runtime. Test a network of host and client devices at various scales for performance and security.

1. Can the Subspace protocol run on host devices without incurring significant background overhead or degrading the general user experience of the device?
2. Will incumbent app platforms (Google Play and Apple App Store) allow the protocol to pass their screening process?
3. Can the network perform at scale in a secure manner?
4. Will end users show a preference towards decentralized apps for certain use case?

Research & Development Plan

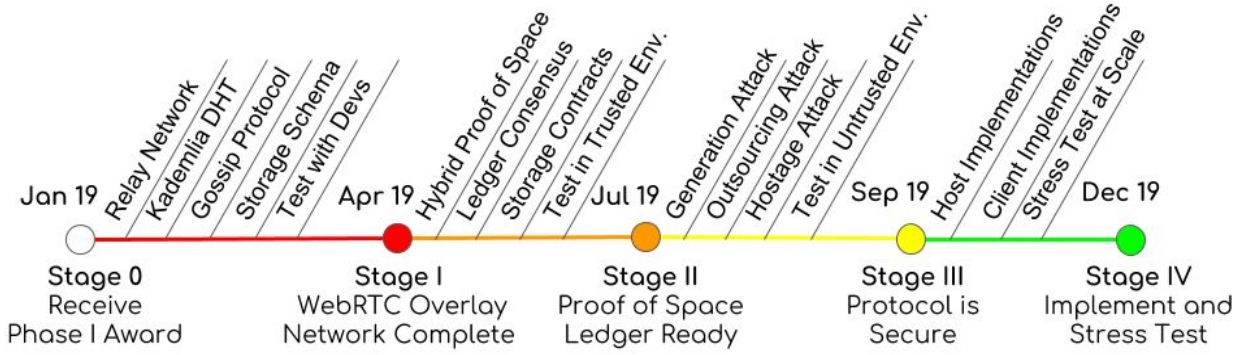

Figure 8: R&D Timeline

Stage I: Develop a WebRTC Overlay Network (16 weeks)

1. Design the peer relay network over Web-RTC using k-buckets **(3)**
2. Extend the relay network with a custom designed DHT **(5)**
3. Extend the relay network with a custom gossip protocol to track member state **(4)**
4. Construct the SSDB API while arbitrating state in a simple and clear manner **(2)**
5. Test the API with early developers for usability and performance feedback **(2)**

Stage II: Extend the Network with a Proof-of-Space Ledger (12 weeks)

1. Create a hybrid Proof of Space that allow hosts to interactively pledge space and farmers to non-interactively seed a solution set to the block challenge **(3)**
2. Create the ledger data structure and consensus methods by extending SSDB **(5)**
3. Implement client and nexus smart storage contracts in *Script* **(2)**
4. Test the farming and hosting protocol with a core set of trusted early adopters **(2)**

Stage III: Secure the Network and Ledger from Attack (8 weeks)

1. Implement BEP-44 using PGP & ECDSA to prevent the generation attacks **(2)**
2. Construct a Proof of Replication using VDFs to prevent the outsourcing attack **(2)**
3. Develop a Byzantine consensus method for preventing the hostage attack **(2)**
4. Test hosting and farming with an open network of untrusted early adopters **(2)**

Stage IV: Create Host & Client Apps to Stress Test the Network (12 weeks)

1. Implement the full protocol within a mobile, desktop, and server host **(3)**
2. Extend the client API to run in the browser, mobile, and desktop **(3)**
3. Test the network for performance at different scales and under different loads **(6)**

REFERENCES

[1] Gartner, Inc. “Gartner Forecasts Worldwide Public Cloud Revenue to Grow 21.4% in 2018.” April 12 2018. URL: https://www.gartner.com/newsroom/id/3871416

[2] Lucas Mearian. “Blockchain Growth Makes Developers a Hot Commodity.” May 1, 2018. Computer World URL: https://www.computerworld.com/article/3235972/it-careers/blockchain-moves-into-top-spot-for-hottest-job-skills.html

[3] G. Decandia, D. Hastorun, M. Jampani, G Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Voshall, and W. Vogels. Amazon, Seattle. “Dynamo: Amazon’s highly available key-value store.” In *Proceedings of twenty-first ACM SIGOPS* symposium on Operating systems principles (2007), ACM Press New York, NY, USA, pp. 205–220. URL: https://www.allthingsdistributed.com/files/amazon-dynamo-sosp2007.pdf

[4] Satoshi Nakamoto. “Bitcoin: A Peer-to-Peer Electronic Cash System.” 2009 URL: https://bitcoin.org/bitcoin.pdf

[5] Vitalik Buterin. “Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform.” URL: https://github.com/ethereum/wiki/wiki/White-Paper

[6] BigchainDB GmbH, Berlin, Germany. “BigchainDB 2.0: The Blockchain Database.” May 2018 Paper version 1.0 URL: https://www.bigchaindb.com/whitepaper/bigchaindb-whitepaper.pdf

[7] Brian M. Platz, Andrew “Flip” Filipowski, Kevin Doubleday. “FlureeDB, A Practical Decentralized Database.” November 7th, 2017. URL: https://flur.ee/assets/pdf/flureedb_whitepaper_v1.pdf

[8] Jeremiah Wagstaff. “Subspace: A Decentralized Database of Edge Devices.” June 12th 2018. URL: https://subspace.github.io/paper/

[9] Wolk, Inc. “Wolk SwarmDB: Decentralized Database Services for Web 3.” October 15, 2017. URL: https://wolk.com/whitepaper/WolkTokenGenerationEvent.pdf

[10] Pavel Bains and Neeraj Murarka. “Bluzelle: A Decentralized Database for the Future.” Whitepaper v1.4. Dec 11, 2017. URL: https://whitepaperdatabase.com/bluzelle-blz-whitepaper/

[11] Viktor Trón, Aron Fischer, Dániel A. Nagy, Zsolt Felföldi, Nick Johnson. “Swap, Swear and Swindle: Incentive system for swarm.” May 2016 Ethersphere Orange Papers 1. URL: http://swarm-gateways.net/bzz:/theswarm.eth/ethersphere/orange-papers/1/sw%5E3.pdf

[12] Viktor Trón, Aron Fischer, Nick Johnson. “Smash-proof: auditable storage for swarm secured by masked audit secret hash.” May 2016. Ethersphere Orange Papers 2. URL: http://swarm-gateways.net/bzz:/theswarm.eth/ethersphere/orange-papers/1/sw%5E3.pdf

[13] Xuma Network. “Xuma: Decentralized Database for Developers.” Whitepaper v1.01. March 31st, 2018. URL: http://www.xuma.network/assets/xuma_white_paper_v1.pdf

[14] David Vorick, Luke Champine. “Sia: Simple Decentralized Storage.” November 29th, 2014 URL: https://sia.tech/sia.pdf

[15] Juan Benet. “IPFS - Content Addressed, Versioned, P2P File System.” (DRAFT 3) 2014. URL: https://ipfs.io/ipfs/QmR7GSQM93Cx5eAg6a6yRzNde1FQv7uL6X1o4k7zrJa3LX/ipfs.draft3.pdf

[16] Shawn Wilkinson, Tome Boshevski, Josh Brandoff, James Prestwich, Gordon Hall, Patrick Gerbes, Philip Hutchins, Chris Pollard, Vitalik Buterin “Storj: A Peer-to-Peer Cloud Storage Network.” December 15, 2016 v2.0 URL: https://storj.io/storj.pdf

[17] Protocol Labs :Filecoin: A Decentralized Storage Network.” January 2nd, 2018 URL: https://filecoin.io/filecoin.pdf

[18] Jeremiah Wagstaff: “Hex: A Distributed Application Protocol for the Revision Control of Engineering Documents.” February 24th 2018. URL: https://rg3l3dr.github.io/hex-paper/

[19] Austin Middleton. *Peer-Relay Library* URL: https://github.com/xuset/peer-relay

[20] Feross Aboukhadijeh and many others. *BitTorrent-DHT Library* URL: https://github.com/webtorrent/bittorrent-dht

[21] Petar Maymounkov, David Mazières. “Kademlia: A Peer to Peer Information System Based on the XOR Metric.” In: Druschel P., Kaashoek F., Rowstron A. (eds) *Peer-to-Peer Systems.* IPTPS 2002. Lecture Notes in Computer Science, vol 2429. Springer, Berlin, Heidelberg. 2002. URL: https://pdos.csail.mit.edu/~petar/papers/maymounkov-kademlia-lncs.pdf

[22] Arvid Norberg and Andrew Loewenstern. BitTorrent.org “BEP-5: DHT Protocol.” 2008 URL: http://www.bittorrent.org/beps/bep_0005.html

[23] Robbert van Renesse, Dan Dumitriu, Valient Gough, Chris Thomas Amazon.com, Seattle “Efficient Reconciliation and Flow Control for Anti-Entropy Protocols.” LADIS ‘08 Proceedings of the 2nd Workshop on Large-Scale Distributed Systems and Middleware Article No. 6 2008 URL: https://www.cs.cornell.edu/home/rvr/papers/flowgossip.pdf

[24] Secure Scuttlebutt. URL: https://ssbc.github.io/scuttlebutt-protocol-guide/

[25] Bram Cohen. “Incentives build robustness in BitTorrent.” In *Proc. of IPTPS*, 2003. URL: http://www.bittorrent.org/bittorrentecon.pdf

[26] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. “Proofs of Space.” URL: https://eprint.iacr.org/2013/796.pdf

[27] Burstcoin. URL:http://burstcoin.info

[28] Sunoo Park, Albert Kwon, Georg Fuchsbauer, Peter Gazi, Joel Alwen and Krzysztof Pietrzak. “SpaceMint: A Cryptocurrency Based on Proofs of Space.” Financial Cryptography and Data Security 2018. URL: https://eprint.iacr.org/2015/528.pdf

[29] Hamza Abusalah, Joel Alwen, Bram Cohen, Danylo Khilko, Krzysztof Pietrzak1, and Leonid Reyzin. “Beyond Hellman’s Time-Memory Trade-Offs with Applications to Proofs of Space” In: Takagi T., Peyrin T. (eds) *Advances in Cryptology – ASIACRYPT 2017*. ASIACRYPT 2017. Lecture Notes in Computer Science, vol 10625. Springer, Cham URL: https://eprint.iacr.org/2017/893.pdf

[30] Bram Cohen and Krzysztof Pietrzak. “Simple Proofs of Sequential Work.” In: *Advances in Cryptology* – EUROCRYPT 2018, pp.451-467 URL: https://eprint.iacr.org/2018/183.pdf

[31] Arvid Norberg and Steven Siloti. BitTorrent.org *BEP-44: Storing arbitrary data in the DHT* 2014 URL: http://www.bittorrent.org/beps/bep_0044.html

[32] Juan Benet, David Dalrymple and Nicola Greco. Protocol Labs. “Proof of Replication Technical Report.” July 27th, 2017 URL: https://filecoin.io/proof-of-replication.pdf

[33] Arjen K. Lenstra and Benjamin Wesolowski. “A random zoo: sloth, unicorn, and trx” Cryptology ePrint Archive, Report 2015/366. URL: https://eprint.iacr.org/2015/366.pdf

[34] Boneh, Dan, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. “Verifiable Delay Functions.” In *Advances in Cryptology—CRYPTO*, vol. 18. 2018. URL: https://eprint.iacr.org/2018/601.pdf

[35] Wesolowski, Benjamin. “Efficient verifiable delay functions.” June 2018. Cryptology ePrint Archive, URL: https://eprint.iacr.org/2018/623.pdf

[36] Leslie Lamport, Robert Shostak, and Marshall Pease. SRI International. “The Byzantine Generals Problem*”* In *ACM Transactions on Programming Languages and Systems,* Vol. 4, No. 3rd, July 1982, Pages 382-401. URL: https://people.eecs.berkeley.edu/~luca/cs174/byzantine.pdf

[37] Feross Aboukhadijeh, Nazar Mokrynksyi and many others. *Simple Peer* URL: https://github.com/feross/simple-peer

[38] Nazar Mokrynskyi. *Entangled Stated DHT Framework*. URL: https://github.com/nazar-pc/es-dht

[39] Nazar Mokrynskyi. *Detox Core.* https://github.com/Detox/core

[40] Bryan Ford, Pyda Srisuresh, Dan Kegel. “Peer-to-Peer Communication Across Network Address Translators.” In *ATEC ‘05 Proceedings of the annual conference on USENIX Annual Technical Conference* Pages 13-13. Anaheim, CA — April 10 - 15, 2005 URL: http://www.brynosaurus.com/pub/net/p2pnat/

[41] David A. Patterson, Garth Gibson, and Rany H. Katz. “A Case for Redundant Array of Inexpensive Disks (RAID).” Published in *Sigmod ‘88 Proceedings of the 1988 ACM SIGMOD international conference on Management of Data*. Pages 109-116. URL: https://www.cs.cmu.edu/~garth/RAIDpaper/Patterson88.pdf

[42] Baumgart, Ingmar, and Sebastian Mies. "S/kademlia: A practicable approach towards secure key-based routing." In *Parallel and Distributed Systems, 2007 International Conference on*, pp. 1-8. IEEE, 2007.