EXECUTIVE SUMMARY

Research Synopsis: Over an 18 month period, beginning around January 2019, Subspace Labs conducted primary research and development towards the creation of a persistent, decentralized database anchored to a public, permissionless blockchain. While we did design and implement a novel architecture for a decentralized database service, known as Subspace Database (SSDB), we were not able to launch the service as we could not find a sufficient blockchain to anchor on to. Instead, we were forced to spend the majority of our research time (with a requested no-cost extension) to design and implement a suitable public, permissionless blockchain, based on proofs of space and time. Demonstrating the security of the blockchain architecture proved far more difficult than simply devising a working construction and it took multiple iterations of design and implementation work before we came upon a fully secure protocol. The end result was a secure, working implementation of a novel blockchain protocol in less than 10,000 lines of Rust and described separately in a technical white paper.

Main Results: The main contribution of this research is the design and development of a fundamentally new Layer I blockchain protocol, known as the Subspace Network Ledger (SNL). Compared to standard Nakamoto consensus, this protocol has much higher transaction throughput, much faster confirmation latency, and is several orders of magnitude more energy efficient (measured in watts per transaction). Most importantly, since it is based on proofs-of-storage it far more decentralized and ASIC resistant — anyone who has a hard drive may participate in consensus. Remarkably, it maintains the same security guarantees as Nakamoto consensus, namely that no attacker may successfully propose a double spend transaction who control less less than one-half of the storage resources pledged to the network. Compared to other proposed proof-of-stake and proof-of-space protocols it remains fundamentally open and permissionless, players may come and go at any time and do not rely on any special purpose nodes to maintain the security of the protocol.

Remaining Work: All that remains is to actually launch the blockchain as a public decentralized network, in order to demonstrate its security in the wild and show that sufficient demand exists for a secure storage protocol amongst space farmers. We plan to submit our final outcomes report within the next 90 days detailing the results of this final experiment. Following demonstrated security and traction we will be applying for a Phase II grant in Q4 of 2020, in order to complete and launch the Layer II network we originally set out to build.

ORIGINAL PROPOSAL REVIEW

Goals: We set out to create a decentralized database service that would allow developers to easily build web and mobile applications in which users would have full control and ownership over their personal data. The envisioned Subspace Network would allow anyone to monetize slack disk space on their personal computers or mobile devices, in the form of a native network utility token *subspace credits*. Developers and end users would then be able to reserve space on the network, in the form a simple, persistent key-value store database, by paying a monthly storage cost, also using Subspace Credits. From a developer point of view, the experience would be very similar to using Google Firebase or Amazon Dynamo DB. However, instead of the data being stored within a corporate data center, it would be stored across a decentralized device network. End users would also be able to pay their own cost of storage, similar to how they are currently able to pay for Google Drive or Apple iCloud space, allowing them to have full control over their data, through their associated Subspace private keys, in the same way that a Bitcoin user has full control over their money, through their Bitcoin wallet.

Architecture: At a high level, the original architecture called for a two-tiered network, divided between a core blockchain network and a distributed data network, similar to the distinction between the Bitcoin blockchain [1] and the Lightning network [2]. Layer I consisted of a Nakamoto style blockchain with consensus based on proof-of-space, known as the Subspace Ledger. The ledger would maintain the balance of all Subspace Credits, record smart storage contracts for the database protocol, and manage payments between storage hosts (device owners) and storage clients (developers and end users) in a fully autonomous fashion. Layer II, known as Subspace Database (SSDB), would then handle the actual data storage and retrieval, in a fully decentralized manner. SSDB called for a not-trivial technology stack including a WebRTC transport layer, a gossip network, a distributed hash table, a privacy-preserving cryptographic database schema, a proof-of-replication for storage, an ephemeral globally replicated hash table, and a byzaninte fault tolerant consensus algorithm to enforce membership uptime.

Expected Key Challenges:

1. Designing a novel WebRTC overlay network for Layer II that would allow for a low-latency reads and writes from SSDB across all expected host and client platforms.
2. Implementing or adapting a proof-of-space blockchain for Layer I, to be developed by another research team, as opposed to designing one ourselves from scratch.
3. Maintaining security against known attacks presented in the literature, primarily for Layer II, as we expected security of Layer I to be implied.
4. Implementing in a manner that is host and client agnostic, by using a Javascript as the primary language, relying on its rich ecosystem for cross-platform development.

LAYER II NETWORK DESIGN

Prior Work: Before submitting our Phase I proposal, we had created a proof-of-concept (PoC) implementation of the protocol in vanilla Javascript. This did not include a blockchain, nor was it designed to be secure against the full range of attacks. This PoC did demonstrate that data stored on a mobile device could be retrieved within a browser over a simple web app, using a WebRTC transport layer. The key lesson learned from this exercise was that Javascript was not the best language choice, as it is weakly typed and prone to quirky behaviors that are hard to debug.

Development Framework: We began the research by starting over with a new codebase written in Typescript, a strongly typed version of Javascript with the tagline: *Javascript that Scales*. We also adopted a modular approach, creating different repositories for different functionalities that could be swapped out between different implementation targets such as server, desktop, web, and mobile runtimes much more easily. This allowed us to focus more on running experiments and less on debugging errors.

Sharding Schema: The next major change was to revise the sharding schema, or the strategy by which client data would be distributed across the host network in a way that was fault-tolerant, load-balanced, and efficiently retrievable. Our original sharding implementation used a simple random peer sampling (RPS) algorithm that would lead to imbalances over time, required long query times, and was not sybil resistant. After a substantial review of the literature on sharding (not to confused with *blockchain sharding)*, we decided to implement a novel, hybrid approach that combined rendezvous hashing [3] and jump-consistent hashing [4].

Basic Design: Let us begin by noting that client data is never stored on a single host. Instead the contract is divided into shards and each shard is replicated across many different hosts on the network. The default implementation calls for a 1 GB storage contract consisting of 10x100 MB shards, with each shard replicated on four unique hosts. The end result is then 40 shards, each stored on a different host with 4 GB of total network space utilization. We now have two challenges for clients. First, given a read or write request, how do they know which shard the record is assigned to? Second, after the shard has been identified, how do they know which host on the network is currently storing the shard?

Jump Hash: The first problem, assigning records to shards, can be efficiently solved using a jump-consistent hash. Given a fixed set of shards (defined in the storage contract) and a unique value (the unique record id), a jump consistent hash allows us to efficiently and deterministically compute the correct shard for the record. Moreover, as new records are added they are assigned to different shards uniformly, achieving near-perfect load balancing.

Rendezvous Hash: The second problem, assigning shards to hosts, may be solved with a rendezvous hash. Once a host pledges space to the network, via a smart storage contract posted to the underlying blockchain, they must join the tracker, an eventually consistent, in-memory data-structure maintained locally by each peer on the network. The tracker maintains the contact info for all hosts on the network and serves as the routing table for clients making read and write requests to the network. When a host joins the tracker it is assigned a shard set through a rendezvous hash algorithm. Rendezvous hash efficiently solves the distributed k-agreement problem — given the same version of the tracker, any peer on the network will arrive at the same set of hosts for a given shard. Rendezvous hash also allows for weighting different hosts by their space pledged so that shards are evenly balanced between hosts on the network. Critically, as the tracker changes, reassignment of shards between hosts is kept to the theoretical minimum. While all peers on the network (including clients) are required to maintain an up-to-date copy of the tracker, computing the rendezvous hash is extremely efficient. The tracker itself is tiny, roughly 100 bytes per host (1 MB for 10k hosts) and may be further divided into multiple subnets for further scaling.

Distributed Hash Table (DHT): With respect to the networking stack, we were able to simplify the design while making it more robust. We originally proposed combining a Kademlia DHT [5] with a gossip network [6] over a WebRTC transport. Further analysis (and devising a more efficient sharding schema) showed that a K-DHT was no longer necessary, as its role was already filled by the tracker. The critical insight was that the tracker could be used not only as a means to track the uptime of hosts for fair storage payments but also as a routing table, effectively serving as a highly replicated local hash table, similar to a single-hop DHT [7]. A K-DHT would only serve to increase the latency of reads and writes, as a query would need to hop across many hosts on the network before the destination was found. While this would allow for more scalability, the tracker alone could already scale to a large number of hosts while it could later be extended to be more distributed if scalability proved to be an issue.

Transport Protocols: We did implement the WebRTC protocol with a gateway server that implemented the ICE protocol for NAT traversal for bootstrap nodes. However we learned that working with WebRTC is difficult and that many open connections are resource intensive on the client. Since WebRTC was only really needed for browser support, we extended the network module to also support TCP and UDP sockets, and then made the transport conditional on the relationship between two peers. For example, two desktop or mobile peers could connect much more efficiently over TCP, while a browser could only connect over WebRTC. While we initially attempted to reuse existing open source libraries for peer-to-peer networking, we found them to be either too general purposes, with a high degree of overhead, or custom-build for a different use-case. Instead we had to construct or own networking library, effectively from scratch. While this took significantly more time it did result in a very fast and efficient implementation.

Initial Results: By this point in the research we had effectively solved the first research question, by devising and implementing the correct distributed systems architecture and transport protocols needed to make SSDB (Layer II) work as envisioned. This milestone was completed within the first three months of our grant period. We then turned to the much more challenging cryptographic problems of selecting and implementing a distributed ledger, a proof-of-space, and a proof-of-replication.

INITIAL LAYER I LEDGER DESIGN AND CRYPTO PRIMITIVES

Mockchain: In the first iteration we began by implementing a very simple mock blockchain or *mockchain*, to fully map the workflow and test the end-to-end system. At this point we were expecting to implement a chain based on the design for the Chia network [8], which alternates proofs-of-space and proof-of-time to achieve a mining dynamic like Bitcoin at a fraction of the electricity cost. A proof-of-space consensus chain was desired (as opposed to the more common proof-of-work and more popular proof-of-stake mechanisms) as it dovetailed nicely with the proof-of-space that the host was already required to produce in order to demonstrate that it had the storage needed to actually be a host on the network, which allowed it join the tracker. This would allow the host to retain the proof-of-space on its free storage capacity and use this to “farm” (participate in consensus) on the blockchain while simultaneously storing useful data for the network. Note that as storage on the Layer II network is load balanced, any host can only utilize as much space as the network itself utilizes — e.g. if the network is at 30% capacity than any individual host will only have 30% of its locally pledged space occupied with client data. The remainder of the space would then be filled with the proof-of-space which would secure the ledger and allow for further incentives and rewards through farming.

Existing PoS Chains: At the time there was no code available for the Chia network nor was there a white paper that formally described the system design and security. Instead, Chia had published some rather abstract papers on their proof-of-space and proof-of-time, while their founder Bram Cohen had given several public talks that outlined the protocol at a high level. We set about to create a minimal implementation of their design using a naive a proof-of-space (insecure) with mock proofs-of-time based on simple random timeouts. The goal was to expend as little effort while retaining the essence of the system and fully expecting code to released in the near future. Note that two other competing protocols, Filecoin [9] and Spacemesh [10], were basically at the same stage of development at this time, while the only live proof-of-space blockchain, Burstcoin, had already been shown to have several fundamental security flaws [11].

Implementation: Even though the mockchain was an insecure, minimal implementation it took much more time than expected as we had to go back and study several of the fundamental properties of blockchains as we learned that much of our understanding proved to be more folklore than fact. While time consuming, this effort was useful in that it allowed us to reconsider the abstractions between the Layer I and Layer II network, specifically with respect to the distinction between immutable (permanent) and mutable (ephemeral) storage. The key insight was that we could implement the chain in such a way that the archival state of the chain could be stored immutably on SSDB in a distributed fashion, rather than requiring all full nodes (farmers or hosts) to store a full copy of the ledger. This allowed for massive horizontal scaling of ledger state in a way that was not possible with a Layer I network alone. Furthermore, we were able to combine the mockchain with the Layer II network and demonstrate that the system basically worked (though insecure).

CONCURRENT NON-RESEARCH ACTIVITIES

User Research: In parallel with these research efforts we also sought to get early feedback from potential users on various aspects of the Subspace Protocol. We conducted over thirty customer interviews as part of the Beat the Odds Bootcamp program, mostly with protocol engineers and application developers. The key takeaway was that we had to focus on developer experience (DX), as the current landscape of crypto protocols was so hard to understand, poorly documented, and difficult to use by mainstream and decentralized developers alike. We then focused on refining our developer focused API and documentation over several follow up interviews and eventually decided on a simple API that was included in the first iteration of the protocol.

Fundraising: Prior to applying for our Phase I grant we had raised $35k in funding from angel investors. While talking with investors we received many questions on our token economic policy and what kind of market dynamics we expected to see within the storage space. This led us to write a Token White Paper [12] which described a token distribution and fundraising plan, analyzed the underlying value of our network, and presented an appropriate decentralized governance model. Over the course of the grant we raised an additional $240k in equity investment from venture investors.

BitBot Hardware: In our Phase I proposal we described a potential business model based on selling plug and play home electronic devices that could easily run the protocol. We began experimenting with ARM based single board computers, off the shelf hard disks and simple 3D printed enclosures. After devising a prototype that we could produce in small batches, we developed a simple software stack and were able to run the initial version of the protocol on this device. Based on demos and feedback to early users we decided to pivot slightly and modify the device to run existing crypto protocols, specifically Bitcoin and Ethereum. This was not as a miner with financial incentives, but a full node and wallet that simply provided trustless way to access the protocol. We setup a point of sale website and conducted a short 30 day marketing experiment to see if we could sell 50 devices to justify a small production run. We ran into many issues with payment providers delisting us as a scam and also being banned from advertising networks due to the crypto association. While we sold a few devices we came nowhere near our campaign goal. The key lesson learned was that very few people were willing to invest money in a crypto hardware product that had no direct financial incentives. This reinforced our view that the BitBot only made sense as a storage farmer for our network and we decided to postpone further development on monetization with hardware until we had the protocol fully operational.

PRIMARY RESEARCH PROBLEM

Within the first six months of the proposed nine month research timeline we had

1) Implemented and validated a basic version of the Layer II network (SSDB)
2) Implemented and validated a mock version of the Layer I blockchain
3) Gained valuable early feedback, conducted marketing experiments, and raised funding

We now had to make a choice. We could optimistically move forward with securing the Layer II network and fully implement across all host and client environments, hoping that a secure Layer I blockchain would be developed in the next three to six months that we could then integrate. Alternatively, we could pessimistically predict that such a blockchain would not be ready for some time, or may not actually be suitable for our Layer II network, and instead do the hard work and attempt to design our own. At the outset of the research we would have expected the second approach to be far beyond our technical abilities, but through the process of building the mockchain and implementing several new cryptographic primitives we now felt this was within our reach. We chose the latter.

We began as we had with all other problems, with an in depth literature review of proofs-of-space, proofs-of-storage and storage based consensus mechanisms for blockchains. The goal was to mimic the Bitcoin model of Nakamoto consensus [1] as much as possible, mainly by keeping the protocol simple while maintaining the security threshold at one-half of network resources, with work being replaced by space. There were a few early proposal (circa 2014) that we found noteworthy including the original Filecoin white paper [13] , Permacoin [14], and a proof-of-unique-blockchain-storage [15]. These proposals were all based on a proof-of-storage (useful data) vs a proof-of-space (random data). We followed this thread and stumbled across the idea of hourglass schemes [16] which was later re-stated as a proof-of-replication [17] by the Filecoin project. From these proposals and primitives we formulated a basic construction that eventually led to a simple, secure storage-based blockchain.

In a nutshell, consensus would be based on proof-of-storage of the blockchain itself. Each new block would specify an audit for some piece of a past block. As all farmers would store the block, pieces would be qualitatively differentiated by having each farmer store a unique encoding of the block based on their identity. In order to prevent farmers from computing this on-demand (instead of ahead of time) the encoding would be inherently sequential and slow in the forward direction. This would result in each farmer storing as many unique replicas of the ledger as its disk space allowed and having consensus power proportional to its disk space. While the high level idea sounds simple enough, the implementation was extremely tricky. After much trial and error, we did work out the details and come across simple, secure, and scalable protocol. Fortunately, we recognized this would take more time than originally planned and filed a no cost extension adjusting our initial nine-month research timeline to an eighteen-month plan.

PRIMARY RESEARCH RESULTS

The search for the correct construction played out over the next twelve months. Key questions included: defining the correct audit strategy, choosing the appropriate proof-of-replication, having the right notion of time, managing high-throughput chain growth with the right ledger data structure, and demonstrating the security of the protocol against all known attacks in the literature. The end result was a technical white paper [18] (revised many times) and two different implementations in code that eventually led to a working construction demonstrated over a test network.

Slicing the Ledger: The first challenge involved defining the sampling strategy by which farmers could prove that they were storing one or more unique replicas of the archival history of the ledger. To simplify and normalize this process, the ledger is divided into an ever-growing set of constant sized (4096 byte) pieces, as each block may be a different size (based on the number of transactions) and we require uniform sampling.

Single Piece Audits: The simplest and perhaps most straightforward choice is to audit exactly one piece, chosen at random, for each challenge. In effect, we are then sampling the replication factor of the ledger (under an assumption of load balanced replication). For example, if there are 256 replicas of the ledger, then for each audit of a single piece, we should see 256 encodings sampled by the entire network. We may then measure the quality of the encoding by computing a tag, as a Hash-Based Message Authentication Code (HMAC) over the encoding and the the challenge. We can then measure the quality of the tag by comparing it to to a target that is adjusted in relation to the expected number of encodings and some desired mean quality. This audit strategy requires that for each challenge, every encoding of the audited piece must be read from disk and hashed. In other words, the computational work done for consensus is linearly proportional to the replication factor of the ledger. Likewise, the security is also linearly proportional to the replication factor, as the most obvious attack involves attempting to create more encodings on-demand than exist across the honest network. The efficacy of on-demand encoding depends heavily on the chosen proof-of-replication and the target delay parameter. We may ratchet up the security threshold by increasing the scope of the audit, or specifying more than one piece which may be used as a valid encoding for a challenge, at the cost of requiring more honest work for each challenge as well.

Best Encoding Audits: Another option is to base the audits not on all encodings for a single piece, but for every encoding across the entire network. This may actually be done very efficiently if we hash each encoding and place the hash in a binary search tree, which is small enough to fit in main memory. We then find the closest encoded hash for a given challenge, measured by XOR distance or Hamming weight, with computational complexity logarithmic in the size of the plot, resulting in a single disk read for the highest quality encoding. Now the work done by the honest network is logarithmic in the space pledged, allowing it to scale gracefully as the network reaches many petabytes or exabytes of space, while the complexity of a brute-force on-demand encoding attack is proportional to the total space pledged by the honest network.

Compression Attacks: While the second approach is more elegant, it suffers from a subtle attack that proved difficult to resolve. The attack involves honestly pre-computing encodings but only retaining the encoded hashes stored withing the binary search tree. Note that while small enough to fit in RAM, these trees may also be stored on a Solid State Drive (SSD) which has low enough read latencies to not impose a meaningful delay on evaluation. Since the binary search tree is approximately 100x smaller than the plot itself, the attacker can leverage its storage by the same degree. During the audit process the attacker only has to identify the single best encoding across all of its trees and then re-derive that single piece on-demand. A motivated attacker, given sufficient pre-computation time, could employ this strategy to mount a 51% attack with as little as 1% of the storage on the network. Moreover, any farmer could "mine" its plot and gain a share of the farming rewards out of proportion to the disk space pledged.

Salted Hashes: The solution we eventually devised was actually quite simple. Note that we audit based on some encoded hashes, which are easy to re-derive for the honest node, who retains them on disk, while expensive to re-derive for the attacker, since they have been deleted. If we then periodically salt these encoded hashes with some random value that may not be predicted ahead of time, such as the hash of the last state block, we make it such that the cost of re-deriving hashes is negligible for the honest node, while very expensive for the attacker. The trick lies in finding the right value for this re-plotting window, which we plan to set as a daily interval until the network grows beyond a few petabytes, after which we will gradually ratchet up to a weekly interval.

Proof-of-Replication: Another key challenge lay in selecting the correct Proof-of-Replication (PoR) for encoding itself. The ideal PoR would be based on a well-studied cipher that was inherently sequential in the forward direction, inherently parallelizable in the reverse direction, with an efficient and widely available hardware implementation. This led us to focus on the Advanced Encryption Standard (AES) [19] and the AES-NI hardware instruction set [20]. We wanted to ensure that an ASIC manufacturer could not devise a better instruction set that would give an attacker a meaningful advantage over the honest network, so we completed a detailed study of AES [21] on various hardware platforms including the CPU, GPU, FPGA, and ASIC. The basic takeaway was that it would be possible to build an ASIC that could be between three and six times faster than AES-NI, an acceptable security margin for our protocol. We also looked heavily at time-asymmetric permutations that require far less work to decode than encode, as this would improve the energy-efficiency of the network and the speed at which solutions could propagate across the network (as each solution has to be verified before forwarding). This led us to develop a custom permutation, heavily inspired by Sloth algorithm [22] (slow time-hashed puzzle) and essentially an application of an earlier modular square client puzzle [23]. In the end, we decided to use the Sloth-based permutation due to its superior decoding efficiency. The analysis of the ASIC resistance of Sloth boils down to the choice of the prime size and the degree to which parallel computation may speed up the evaluation time. For a 256 bit prime, that parallelism is already accounted for on moder x86 architectures, while still offering a 100x speedup. We also thwart known ASIC techniques by using a different 256 bit base prime for each index of the block cipher.

Notion of Time: In a proof-of-work blockchain such as Bitcoin, the time between blocks is a by-product of the proof-of-work. In a resource efficient blockchain such as proof-of-space or stake time has to be simulated. Burst, the first proof-of-space blockchain, handled this with a notion of deadlines, or time before which a block would not be recognized by the honest network. Spacemint (and most proof-of-stake protocols) instead use a notion of timeslots, which ends up creating more work for the honest network. Chia proposes using an actual proof-of-time, or a inherently sequential proof-of-work, in order to deal with the subtle yet devastating history rewriting attack. As we show later, Subspace resists the history rewriting attack, and this is not needed. Instead we chose to go with the simplest approach and simulate proofs-of-time, as Burst did. The difference is that we would rather have a short block time, as close to the network delay as possible, so as to have a higher transaction throughput and lower confirmation latency. This also significantly reduces the time that an attacker has to carry out a time-space trade-off attack by encoding on-demand. The result is we produce roughly three blocks per second, with up to 128 transactions per block, for a throughput of up to 500 transactions per second on a single chain, with confirmation latency at K=18 of six seconds.

Managing Chain Growth: The final problem was figuring out the right data structure to handle the small network delay and high block throughput such that all nodes would quickly converge to the same shared ledger history. We also had to account for the added problem of simulation, or that each node can try many different proofs to create a large tree and then choose the fastest (or highest quality) path through the tree. We do this my making a manageable degree of simulation (roughly 10) the default strategy and carefully prune branches of the block tree as new blocks are confirmed, resulting in a single longest chain [24]. We also experimented heavily with the Prism family [25, 26, 27] of protocols and explored the GHOST [28] heaviest sub-tree rule as well. In the end we decided that the simplest approach to stick with a single chain, since so much else of what we were doing was already radically different.

Code Implementation: Our first attempt at the Ledger was implementing the mockchain within the SSDB project in Typescript. While a valuable learning exercise, it was clear that the complexity of a blockchain required a project entirely of its own. In the second attempt we implemented the Ledger as a stand-alone Typescript project and made steady progress for a while. However, as we moved further along with the implementation the it became more and more difficult to debug issues with the protocol. Even though Typescript provides strongly typed interfaces, Javascript itself is still a garbage collected language with automated memory management. We found that the overhead of the Javascript runtime paired with the opaqueness of the memory management paradigm led to many strange behaviors and memory leaks where the garbage collector was not freeing variables as expected. The end result was that the Typescript project would simply not run for a large plot or blockchain. Finally it became clear that we would need to write the Ledger in a systems language such as C, C++ or Rust. Much of the blockchain community has migrated to the Rust ecosystem, and for good reason. Rust provides automated memory management in a way that provides the efficiency of a language like C without the overhead of a garbage collector. The end result, after eighteen months of trial and error, was a working implementation in less than 10,000 lines of Rust.

LAYER I SECURITY SUMMARY

Following Nakamoto consensus, no attacker who controls less than one-half of the physical storage resources dedicated to the network may produce a chain faster than the honest network. Likewise, block confirmation is probabilistic, and we may apply a k-deep confirmation rule based on an assumption of relative attacker power to determine a reasonable wait period before considering a block “confirmed”. This must be adjusted slightly to handle prediction attacks and weak clock synchronization.

Time-Space Trade-Off Attacks: Since every encoding on the network is audited during each round of consensus, then even for a small network, an attacker may only gain a negligible advantage by plotting-on-demand. In other words, even a massively parallel attacker could not encode many pieces on demand within the block time, if relying on computation alone.

Compression Attacks: An attacker may attempt to honestly pre-compute encodings and then retain only the encoded hashes for many different node ids, effectively compressing their plots by several orders of magnitude, in order to gain block rewards out of proportion to their storage power or mount a double-spend attack with less than one-half of the storage resources. We defend against these attacks by salting the encoded hashes with a random value, based on the archival state of the ledger, which updates at a regular interval. Now the attacker must continuously regenerate encodings in order to maintain their advantage, driving up the electricity costs drastically, while double spend attacks quickly become infeasible due to the massive computational resources needed to out-plot the network in such a short time interval.

Long-Range Attacks: Proofs-of-time were proposed by Chia network to prevent the history rewriting attack on proof-of-space blockchains. Any such long-range attack on Subspace is constrained by the fact that all space plotted by the network is a commitment to the canonical archival state, in contrast with all proof-of-space blockchains were all plots are valid for any history. An attacker would have to quickly re-plot while generating the alternate history and simultaneously convince all honest nodes to abandon their plot and switch to the new plot. This allows us to use resource-efficient simulated proofs-of-time, removing the reliance on time lords and allowing the protocol to be truly decentralized. Note that this only requires weakly synchronized clocks and agreement on genesis time to prevent producing blocks into the future.

Costless Simulation Attacks: Since proofs-of-replication are computationally cheap to evaluate for nodes who honestly pre-plotted, the rational strategy is to solve on every fork, or branch, of the pending block tree, to hedge ones bets. It has been shown that the maximum advantage for “simulating” different trees approaches *e* (Euler’s Number) and that 99% of this advantage is subsumed within the first ten simulations [8, 24]. Following these protocols, we make simulation the default behavior. We note that the balance attack [26] against this strategy is not feasible in our case as the block time is already at the network propagation delay. This means that any balance attacker would see a slower growth rate for blocks that it selectively reveals, making the balance attack ineffective.

KEY DELIVERABLES PRODUCED

The following products can be seen as our *proof-of-work* for this research ;-)

Subspace Database Network (Code): *Q1 2019* www.github.com/subspace/subspace

The Layer II Database network implemented in Typescript, including the mock blockchain. This project could not be launched without a secure Layer I blockchain.

Subspace Credit Primer (Paper): Q2 2019: G-Drive Link

A token white paper describing the economics and market dynamics for the Subspace Network

BitBot Marketing Experiments (Website): Q2 2019 www.subspace.store

A plug-and-play personal storage device and full blockchain node for the Subspace Network, market tested as a full blockchain node for existing protocols. Experiment showed the market was too small to justify production due to a lack of financial rewards for users.

Subspace Core (Code): Q3 2019: www.github.com/subpsace/subspace-core

Our first attempt at implementing a proof-of-unique-blockchain-storage based ledger in Typescript. This project had unresolved security issues and scaling challenges with Typescript.

Beyond Bitcoin: The Subspace Network Ledger (Paper): Q4 2019: G-Drive Link

Summary of the key ideas for the Subspace Network Ledger, modeled after the Bitcoin white paper. This paper went through (and continues to go through) many revisions as we have improved on the architecture.

AES Security Analysis (Paper): Q1 2020: G-Drive Link

An in depth study of the potential hardware speedups for the Advanced Encryption Standard (AES) as this serves as the basis for our proof-of-replication and proofs-of-time.

Subspace Core Rust (Code): Q2/3 2020 www.github.com/subspace/subspace-core-rust

Our second and final attempt to create a novel blockchain, this time done in pure rust and based on lessons learned from the two previous attempts. We plan to go live with this implementation in the Q3 of this year.

REMAINING PHASE I ACTIVITIES

During the roughly three months remaining until our Phase I outcomes report is due we plan to go live with the Subspace Ledger and establish a base of long-term users. We plan to focus on scaling the available storage on the network, improving the user experience and ensuring there is sufficient demand for space amongst developers (not just supply). Once we have a stable, secure implementation with a strong and growing user base we will be able to continue work on the originally planned Subspace Database (SSDB) as a Layer II network on top of the blockchain.

Launching Mainnet: While we have proven out the core consensus mechanism and security properties of the Subspace Ledger we still need to add a few more basic features to make it a minimum viable product. Luckily we have already implemented all of these features in the earlier Typescript iteration so it is mostly a matter of porting over the remaining code and ensuring is correctly integrated into the new architecture. We expect to have this complete by Sep 1st 2020.

Farmer Acquisition (Supply Side): The initial marketing challenge will be acquiring new users on the supply side (space farmers). We plan to focus on PC Gamers as our first target market as they have powerful machines (fast plotting), lots of free disk space, their machines are almost always online (desktop configuration) and they are often open to experimenting with cutting edge technology. To facilitate this we plan to reboot our web and social media presence around the narrative of the Subspace Ledger. We will also be setting up a Discord community chat server and Discourse forum for discussion of the protocol. We plan to onboard new farmers slowly and carefully, with a focus on continuous improvement based on user feedback. We will also be focused on establishing credibility within the open-source blockchain developer community and attempting to leverage their skills and expertise to continue to improve the protocol.

Improving User Experience: Once the protocol goes live we expect that changes to the back-end consensus mechanisms will be minimal and plan to devote most of our efforts towards building out the front-end user experience. This includes moving from a Command Line to a Desktop App user interface, building out a web-based blockchain explorer and wallet, and getting listed on exchanges to provide liquidity for the token.

Developer Acquisition (Demand Side): One way we plan to grow demand for the protocol token (beyond simple speculation) is building out an immutable storage API and a developer console that let developers use the ledger as a back-end for their applications, much like dApp state may be stored on the Ethereum blockchain, though at a much lower cost, since the storage is sharded. More specifically developers will be able to submit put() transactions to the network that store encrypted data directly on the ledger. To ensure proper scalability, once the ledger exceeds the storage capacity of the average farmer, we must also implement a K-DHT for record retrieval as described in the technical white paper.

PLANNED PHASE II ACTIVITIES

While we had originally planned to launch both a Layer I Blockchain and Layer II Storage network during the Phase I grant period, we were ultimately unable to do so. We were unable to rely on another project for this work and had to do it ourselves, taking up considerable time and doubling the proposed research timeline. Now that we have a working, secure Layer I chain we plan to re-scope our Phase II proposal to focus on the re-design, integration, and launch of the Layer II storage network (SSDB).

Revised Layer II Design: The current architecture for Layer II is over twelve months old. In that time, the field of distributed storage has advanced significantly as well as our understanding of the problem. We plan to start by reconsidering and revising key aspects of this architecture. First, we would like to make the design more generic and extensible. Instead of focusing on a decentralized database, we would like to focus on decentralized cloud services more generally. We can then design a framework for a database, a file-system, and a serverless function runtime that can co-exist on the same network, possibly with the same token. Second, the proofs-of-space for host pledges are susceptible to time-space trade-offs. They were designed for an earlier iteration where the blockchain was pure proof-of-space. We need to rethink this entire approach and select a more recent construction of a proof-of-space or proof-of-storage. Third, the sharding approach we have chosen sufferers from high communications complexity and heavy bandwidth usage. While we expect that it will work in the wild, we would prefer if it did so in a more efficient manner. From a scalability perspective we would also prefer to have a more formal design for integrating the DHT into the current tracker, or implementing it as a layered hierarchy of sub-networks.

Formal Security Analysis: The latest iteration for the Layer II network is only secure against the most commonly known attacks and relies on informal proofs. Based on our challenges with provable security for our Layer I network, we expect that formal security for Layer II will prove to be very challenging and should be dealt with in the earliest stages (before writing code). It is easy to build something that appears to work only to learn later that it is susceptible to a subtle attack that requires significant re-work. Given the two year timeline required for moving the Bitcoin Layer II Network, Lightning, from proposal [2] to working securely in production, we expect to encounter similar challenges and must factor that into our Phase II timeline.

Focus on Developer Experience: Whereas the go-to-market approach for the Layer I network is primarily a problem of supply, the challenge for Layer II will be demand. If we cannot ensure demand amongst developers for this service then it will be for nought. Our marketing experiments have shown that this demand exists, but the crux will be in presenting our protocol to developers in as simple a manner as possible. During customer interviews we found that there are many more traditional developers interested in experimenting with decentralized web technologies than we would have expected. The only thing holding them back is the mental load required to grasp the underlying protocol and the often poor user experience. We plan to focus on this problem throughout our Phase II research, not just at then end, as if an afterthought.

CONCLUSION

Goals Review: Returning back to our original research goals, we now see that two out of four were completed successfully. First, we did design and implement a novel architecture for a Layer II Network, Subspace Database (SSDB). Second, we did implement a blockchain architecture based on proofs-of-space and time. However, since we were unable to implement an existing chain we had to do the hard work of researching, constructing, and analyzing the security of a new protocol ourselves. *This proved to be the key technical challenge of our research*. It took significantly more time than we had originally allocated for the grant and has prevented us from yet completing the last two research goals. While we did complete an in-depth security analysis of the Layer I blockchain, we have not yet been able to do so for the Layer II network. Nor have we yet implemented this Layer II network across the range of host and client devices. We instead plan to focus on these last two tasks during our Phase II grant.

Contributions: The main contribution of this research is the design and development of a fundamentally new Layer I blockchain protocol, known as the Subspace Ledger. Compared to standard Nakamoto consensus, this protocol has much higher transaction throughput, much faster confirmation latency, and is several orders of magnitude more energy efficient (measured in watts per transaction). Most importantly, since it is based on proofs-of-storage it far more decentralized and ASIC resistant — anyone who has a hard drive may participate in consensus. Remarkably it maintains the same security guarantees as Nakamoto consensus, namely that no attacker may successfully propose a double spend transaction who control less less than one-half of the storage resources pledged to the network. Compared to other proposed proof-of-stake and proof-of-space protocols it remains fundamentally open and permissionless, players may come and go at any time and do not rely on any special purpose nodes to maintain the security of the protocol.

Feasibility and Broader Impacts: With respect to feasibility, we have learned much over the last eighteen months. From a technical perspective we believe the protocol to be fundamentally sound and secure. The challenges revolved more around composition and correct implementation of existing cryptographic primitives and techniques than in designing something fundamentally new. Regarding commercial feasibility, we now believe that a basic token appreciation strategy is the most viable path to commercial success for Subspace Labs. If the company owns a significant fraction of the tokens at launch and slowly divests itself of those tokens over time, this could prove to be a significant revenue stream if the network proves useful for users and developers. Finally, considering the massive energy expenditures required to power the Bitcoin network and cryptocurrencies generally, the broader impacts of this work may be an environmentally sustainable model for permissionless blockchains

Next Steps: Over the next six months we will be focused on launching and growing the Layer I network. Once we can demonstrate traction in the wild we will return to the original goal of extending with the Layer II Network, and launch Subspace Database.

References

[1] Nakamoto, Satoshi. Bitcoin: A peer-to-peer electronic cash system, 2008.
https://bitcoin.org/bitcoin.pdf

[2] Poon, Joseph, and Thaddeus Dryja. "The bitcoin lightning network: Scalable off-chain instant payments." (2016).
https://lightning.network/lightning-network-paper.pdf

[3] Lamping, John, and Eric Veach. "A fast, minimal memory, consistent hash algorithm." *arXiv preprint arXiv:1406.2294* (2014).
https://arxiv.org/pdf/1406.2294.pdf)

[4] Thaler, David, and Chinya V. Ravishankar. "A name-based mapping scheme for rendezvous." *Technical Report CSE-TR-316-96, University of Michigan*. 1996.
http://www.eecs.umich.edu/techreports/cse/96/CSE-TR-316-96.pdf

[5] Maymounkov, Petar, and David Mazieres. "Kademlia: A peer-to-peer information system based on the xor metric." *International Workshop on Peer-to-Peer Systems*. Springer, Berlin, 2002.
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.502.6326&rep=rep1&type=pdf

[6] Demers, Alan, et al. "Epidemic algorithms for replicated database maintenance." *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*. 1987.
http://bitsavers.trailing-edge.com/pdf/xerox/parc/techReports/CSL-89-1_Epidemic_Algorithms_for_Replicated_Database_Maintenance.pdf

[7] Monnerat, Luiz, and Claudio L. Amorim. "An effective single-hop distributed hash table with high lookup performance and low traffic overhead." *Concurrency and Computation: Practice and Experience* 27.7 (2015): 1767-1788.
https://arxiv.org/pdf/1408.7070.pdf

[8] Cohen, Bram and Pietrzak, Krzyyztof. “The Chia Network Blockchain” Jul 9 2019.
https://www.chia.net/assets/ChiaGreenPaper.pdf

[9] Benet, J., and N. Greco. "Filecoin: A decentralized storage network." *Protoc. Labs* (2018): 1-36.
https://filecoin.io/filecoin.pdf

[10] Moran, Tal, and Ilan Orlov. "Simple Proofs of Space-Time and Rational Proofs of Storage." *Annual International Cryptology Conference*. Springer, Cham, 2019.
https://eprint.iacr.org/2016/035.pdf

[11] Park, Sunoo, et al. "Spacemint: A cryptocurrency based on proofs of space." *International Conference on Financial Cryptography and Data Security.* Springer, Berlin, 2018.
https://static1.squarespace.com/static/59aae5e9a803bb10bedeb03e/t/5a70b725085229767bf30d79/1517336365243/space-mint.pdf

[12] Wagstaff, Jeremiah. “Subspace Credit Primer”. *Subspace Labs* (2018).
https://drive.google.com/file/d/1TalXm8c1XxlRJYzuB6BJ42W04NlbJUkB/view?usp=sharing

[13] Benet, J. "Filecoin: A Cryptocurrency Operated File Storage Network. Retrieved January 19, 2016." (2014).
https://filecoin.io/filecoin-jul-2014.pdf

[14] Miller, Andrew, et al. "Permacoin: Repurposing bitcoin work for data preservation." *2014 IEEE Symposium on Security and Privacy*. IEEE, 2014.
http://soc1024.ece.illinois.edu/permacoin_full.pdf

[15] Lerner, Sergio. Proof of unique blockchain storage. *Bitslog*:
https://bitslog.com/2014/11/03/proof-of-local-blockchain-storage/. 2014.

[16] Van Dijk, Marten, et al. "Hourglass schemes: how to prove that cloud files are encrypted." *Proceedings of the 2012 ACM Conference on Computer and communications security*. ACM, 2012.
http://www.ccs.neu.edu/home/alina/papers/Hourglass.pdf

[17] Benet, Juan, David Dalrymple, and Nicola Greco. "Proof of replication Technical Report." *Protocol Labs*, July 27 (2017).
https://research.filecoin.io/assets/proof-of-replication.pdf

[18] Wagstaff, Jeremiah. “Beyond Bitcoin: The Subspace Network Ledger” *Subspace Labs* (2020)
https://drive.google.com/file/d/13ou7lrjrn06J4-6X9yncO0NcFgIDl6Gy/view?usp=sharing

[19] FIPS-197. “Specification for the Advanced Encryption Standard (AES)”. *National Institute for Standards and Technology*. Nov 26th 2001.
https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

[20] Gueron, Shay. “Intel Advanced Encryption Standard (AES) New Instructions Set” *Intel Corporation* May 2010.
https://www.intel.com/content/dam/doc/white-paper/advanced-encryption-standard-new-instructions-set-paper.pdf

[21] Dolecek, John and Jeremiah Wagstaff. “A Cross-platform Analysis of the Suitability of the Advanced Encryption Standard (AES) for Proofs-of-Time (PoT) and Proofs-of-Replication (PoR)” *Subspace Labs* (2020).
https://drive.google.com/file/d/1rEg7uXjRbKMt9eTEVwjJTNNGVpbvaQ3e/view?usp=sharing

[22] Lenstra, Arjen K., and Benjamin Wesolowski. "A random zoo: sloth, unicorn, and trx." *IACR Cryptol. ePrint Arch.* 2015 (2015): 366.
https://eprint.iacr.org/2015/366.pdf

[23] Jerschow, Yves Igor, and Martin Mauve. "Modular square root puzzles: Design of non-parallelizable and non-interactive client puzzles." *computers & security* 35 (2013): 25-36.
https://www.tdr.wiwi.uni-due.de/fileadmin/fileupload/I-TDR/Forschung/Modular_Square_Root_Puzzles/COSE-35-2013.pdf

[24] Fan, Lei, Jonathan Katz, and Hong-Sheng Zhou. "A Large-Scale Proof-of-Stake Blockchain in the Open Setting."
https://pdfs.semanticscholar.org/0623/6e710256e3120f342045dae0439a9f602258.pdf

[25] Bagaria, Vivek, et al. "Prism: Deconstructing the blockchain to approach physical limits." Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. 2019.
https://dl.acm.org/doi/pdf/10.1145/3319535.3363213

[26] Wang, Xuechao, et al. "Proof-of-Stake Longest Chain Protocols Revisited." *arXiv preprint arXiv:1910.02218* (2019).
https://128.84.21.199/pdf/1910.02218v2.pdf

[27] Trifecta: the Blockchain TriLemma Solved. Trifecta Blockchain Team. Oct 2019.
https://www.trifectachain.com

[28] Sompolinsky, Yonatan, and Aviv Zohar. "Secure high-rate transaction processing in bitcoin." *International Conference on Financial Cryptography and Data Security*. Springer, Berlin, Heidelberg, 2015.
https://fc15.ifca.ai/preproceedings/paper_30.pdf