StrongSalt

Decentralized Encrypted Search and Storage Protocol

for Building Low Latency, Privacy Preserving Applications

Ed Yu and Paul Sri

Version 1.2.1
5 March 2019

ABSTRACT

StrongSalt is a blockchain-based encrypted search and storage protocol for building low latency, decentralized applications. Application developers building on the StrongSalt platform will achieve data encryption by default, while their data remains fully searchable.

The StrongSalt protocol provides a five-layer abstraction model that characterizes secure and private communication functions at all layers served by an immutable and trustless distributed ledger. A hierarchical publish-subscribe pattern is leveraged in conjunction with roles-based nodes in order to facilitate node function expansion, service level agreements, and network scalability.

The StrongSalt network utilizes a blockchain to achieve its immutable and trustless nature, however, is blockchain agnostic and open to both permissioned and permissionless blockchain models. A protocol token is provided to enable economic exchange amongst application developers and service providers based on usage and consumption patterns.

Finally, the StrongSalt network operates on the Google-developed, UDP-based protocol, QUIC, instead of TCP allowing all communication to be encrypted and low latency at the network transport layer.

"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it." -- Mark Weiser

StrongSalt @ Telegram

Introduction

Much has changed in the past 10 years. As daily inhabitants sometimes we forget how quickly the world can change and how quickly technology can outpace everything around it. The technology landscape is replete with smartphones, smart watches, smart thermostats, voice assistants, smart speakers, babycams, mesh networks, self-driving vehicles, and fully-electric vehicles -- to name a few. [1]

The proliferation of artificial intelligence (AI) constructs and Internet of things (IoT) devices have not only helped to make our lives easier through learning and automation but have also dramatically increased Internet traffic and introduced a new set of privacy and security concerns. Privacy and security are becoming increasingly important as more and more of our lives are digitized and made available online -- most by our own doing. Instagram has 1 billion monthly active users (MAUs)[2]; in 2016 those users posted over 95 million times and liked posts 4.2 billion times a day[3].

The world has shifted and continues to shift from predominantly human-initiated interactions such as chat applications or web browsing to the broader set of IoT and AI-type interactions. These interactions can range from chatty and low-latency to high-bandwidth and persistent. Current Internet infrastructure and protocols are ill-equipped to handle the explosive growth of IoT and AI-based devices and software. The number of IoT devices is expected to reach 21.5 billion by 2025 with that number currently sitting at 7 billion in 2018. The total number of connected devices will be closer to 34.2 billion in 2025.[4]

Additionally, the tech industry has begun embracing decentralized applications, services, and protocols[5]. The traditional approach of building centralized applications definitely have their merits such as the ability to rollback transactions or having a customer service call center with which to speak. However, these merits come at a cost; namely security breaches, inability to scale, and fraudulent behavior.

These changes are big. Billions big.

When it comes to security, it should come as no surprise to learn that the global average cost of a security breach is $3.86 million USD[6], while the global annual cost is

---

[1] "Smart device - Wikipedia." https://en.wikipedia.org/wiki/Smart_device.

[2] "Instagram hits 1 billion monthly users, up from 800M in September ...." 20 Jun. 2018, https://techcrunch.com/2018/06/20/instagram-1-billion-users/.

[3] "24+ Instagram Statistics That Matter to Marketers in 2019." 5 Oct. 2018, https://blog.hootsuite.com/instagram-statistics/.

[4] "State of the IoT 2018: Number of IoT devices now at 7B – Market ...." 8 Aug. 2018, https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/.

[5] "Microsoft To Embrace Decentralized Identity Systems Built On ... - Forbes." 12 Feb. 2018, https://www.forbes.com/sites/ktorpey/2018/02/12/microsoft-to-embrace-decentralized-identity-systems-built-on-bitcoin-and-other-blockchains/.

[6] "Cost of Data Breach Study | IBM Security." https://www.ibm.com/security/data-breach.

forecast to be $2.1 trillion USD by 2020. In just the first half of 2018 alone over 4.5 billion were exposed as a result of security breaches. [7] Many companies such as Equifax, Ticketmaster, British Airways, and Yahoo! have each paid their own price for not adequately securing their centralized systems.

In addition to privacy and security, there are also bandwidth and latency challenges directly attributable to IoT. New Zealand's Internet traffic is expected to reach the equivalent of 72,000 DVDs every hour by 2020[8], while annual global Internet protocol (IP) traffic will reach 3.3 zettabytes (ZB) per year by 2021, or 278 exabytes (EB) per month. [9]

We have also seen a number of bad actors in recent years such as Wells Fargo[10], Facebook[11], and Aflac[12] each demonstrating how centralized authority can have rather large, negative ramifications if incentives are on the wrong side.

The winners of the data theft wars will inevitably be those who embrace security and privacy at their core. It will be those who choose a decentralized application model as opposed to those who largely dependent on centralized systems.[13]

Blockchain, however, is not without its drawbacks. When it comes to decentralization, there are some issues we as an industry have yet to deal with such as consensus algorithms that are both cost-effective and secure -- which is why we have proof of work (PoW) algorithms competing with delegated Byzantine fault tolerance (dBFT).

And as with most technology scaling problems, we have the blockchain trilemma forcing us to choose two from the three blockchain characteristics of cost / efficiency, correctness, and decentralization. [14]

One common pattern in the aforementioned issues of privacy, security, and fraud stands out; that of centralized

---

[7] "List of data breaches - Wikipedia." https://en.wikipedia.org/wiki/List_of_data_breaches.

[8] "IoT will help New Zealand double web traffic by 2020 - ReadWrite." 26 Jun. 2016, https://readwrite.com/2016/06/26/iot-will-help-new-zealand-double-web-traffic-2020-pl4/.

[9] "The Zettabyte Era: Trends and Analysis - Cisco." 7 Jun. 2017, https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html.

[10] "Wells Fargo timeline: Bank's 20-month nightmare - Business." 24 Apr. 2018, https://money.cnn.com/2018/04/24/news/companies/wells-fargo-timeline-shareholders/index.html.

[11] "Facebook and Cambridge Analytica: What You Need to Know as ...." 19 Mar. 2018, https://www.nytimes.com/2018/03/19/technology/facebook-cambridge-analytica-explained.html.

[12] "Former Aflac Employees Allege Fraud and Other Abuses - The Intercept." 11 Jan. 2018, https://theintercept.com/2018/01/11/aflac-fraud-lawsuit-sales-associates/.

[13] "Why decentralization could prove the most disruptive tech megatrend ...." 17 Oct. 2018, https://latest.13d.com/decentralization-megatrend-disruptive-tech-tim-berners-lee-369fafc82068.

[14] "DSHR's Blog: The Blockchain Trilemma." 9 Aug. 2018, https://blog.dshr.org/2018/08/the-blockchain-trilemma_9.html.

software systems and centralized authority figures.

We aim to paint the reader a picture of what the future can hold -- a less centrally designed world where applications are more robust to malicious attacks and incompetent management due to the very nature of how they were designed and built -- with decentralization, privacy, and security in mind.

What the StrongSalt team is building now will provide the building blocks necessary for today's software developers, companies, or anyone who cares deeply about preserving privacy to build the next generation of decentralized applications (dApps) that will be head and shoulders above what we have now in terms of privacy, security, and scalability.

StrongSalt has already begun the development of its protocol in earnest by building alongside the protocol a real-world decentralized file vault. Because this file vault is built on top of the StrongSalt protocol, it inherits core features such as encrypted data at rest, encrypted data in transit, and fully searchable encryption for any and all data stored in the file vault.

From a product and engineering perspective the StrongSalt team strongly believes the best way to go about building a platform is to consume the platform as you're building it -- colloquially known as eat your own dog food. [15]

This white paper will first provide the problem statement detailing the inherent flaws in today's industry. Then it will provide an overall summary of the proposed solution and a vision for how the StrongSalt team sees the future. Finally, this white paper will discuss adoption strategies for the proposed technology solution.

Problem Statement

Current Internet protocols, platforms for building applications, and applications themselves are designed: 1) assuming only human interaction, 2) lacking a focus on security and privacy, and 3) intentionally centralized.

The next generation of protocols, platforms, and applications must account for not only human interaction, but machine-to-machine interaction. Applications must treat security and privacy as first-class citizens[16], building data protections into the foundation. And lastly a decision must be made as to whether an application should be centralized or decentralized.

More often than not we hope that application and platform developers will build in a decentralized manner whatever can be to avoid the many pitfalls and eventualities described in the Introduction.

StrongSalt aims to provide a protocol for building applications that addresses all of these concerns by providing network level, platform level, and application level support.

Vision

[15] "Eating your own dog food - Wikipedia." https://en.wikipedia.org/wiki/Eating_your_own_dog_food.

[16] "Using AWS with Security as a First Class Citizen | ThoughtWorks." 16 Nov. 2016, https://www.thoughtworks.com/insights/blog/using-aws-security-first-class-citizen.

The StrongSalt team believes that we are all born with certain unalienable rights, one of which is privacy.

The past few decades have given us great achievements in technology driven by technology giants like Apple, Amazon, and Google. However, at the same time we have also seen our privacy eroded, as it has become increasingly lower and lower friction to give it up. [17]

These technology giants were so concerned with harvesting our personal data, selling it, and making it available to all, that they didn't stop to ask if they should.

Technology has and will not only make our lives easier but can also provide a means to undo some of the privacy bartered with these giants. Technologies such as blockchain, encryption, and network protocols will be the building blocks of the StrongSalt network allowing us to construct trustless ecosystems, protect our data, and do it reliably and quickly.

We have the technology to build a colony on Mars[18] and print self-healing DNA[19]. We believe that the world is at a point where not only do we have the technologies available to us to achieve these goals, but the motivation.

Years of hacked personally identifiable information (PII), grossly mismanaged data, and an almost reckless approach to security have made us more than ready to embrace a more secure world.

The StrongSalt team is designing and building the best combination of technologies allowing for blockchain-based trustless systems, low-cost encryption that is searchable, and transport mechanisms that will carry us into the next generation of artificial intelligence (AI) and machine learning (ML) based communications.

Much as a strong and unique salt is required to adequately encrypt and protect data such as passwords, StrongSalt aims to be the keystone of a secure, trustless, and performant technological future.

Solution Overview

The StrongSalt team proposes a multilayer, hierarchical messaging system built on a blockchain economy with a UDP-based transport mechanism.

At the root of StrongSalt, a multilayer approach provides for separation of concerns (SoC)[20] and a modular approach to any potential functionality expansion or performance scaling. We will provide a brief overview here of each layer and go into more detail in the design and architecture sections.

---

[17] "As Facebook Raised a Privacy Wall, It Carved an Opening for Tech ...." 18 Dec. 2018, https://www.nytimes.com/2018/12/18/technology/facebook-privacy.html.

[18] "We have the technology to build a colony on the moon. Let's do it ...." 10 Dec. 2018, https://www.washingtonpost.com/opinions/we-have-the-technology-to-build-a-colony-on-the-moon-lets-do-it/2018/12/10/28cf79d0-f8a8-11e8-8d64-4e79db33382f_story.html.

[19] "1B start-up backed by Bill Gates prints new DNA - CNBC.com." 21 Dec. 2018, https://www.cnbc.com/2018/12/21/bill-gates-backed-start-up-ginkgo-bioworks-prints-synthetic-dna.html.

[20] "Separation of concerns - Wikipedia." https://en.wikipedia.org/wiki/Separation_of_concerns.

A few major components of the StrongSalt network include a hierarchical messaging system that builds on traditional single layer messaging systems, a blockchain database and corresponding economy to provide decentralized interactions and the appropriate incentives, and a network transport layer built on Quick UDP Internet Connections (QUIC).[21]

A hierarchical messaging system provides additional system flexibility and scalability to the messaging backbone as certain levels of communication can be designated for particular functions only.

A blockchain layer provides not only an economic system of incentives or rewards for network participants, but the requisite infrastructure to achieve trustless interaction and immutable, public transactions.

Finally, a UDP-based transport layer provides the low-latency, connectionless communication necessary for chatty AI and ML actors.

The messaging and API layers which sit above the blockchain layer provide critical abstraction necessary for developers to avoid worrying about secure message passing and blockchain economics.

An API layer sits on top of the messaging layer providing a publish-subscribe messaging pattern whereby message producers publish, and message consumers subscribe to interest-based channels.

The messaging layer works in conjunction with the blockchain layer to handle message passing and the corresponding tracking of those messages.

In addition, the StrongSalt network provides a system of hierarchical and role-based nodes that serve the ecosystem in a functionally scalable fashion as new roles and relationships between those roles can be developed to suit the future needs of the protocol.

The StrongSalt network provides for security in a multilayer approach with each layer being responsible for its own security as is most appropriate to the separation of concerns paradigm.

With the solutions described we will build a decentralized private information network (DPIN) with foundational concepts such as private information retrieval (PIR)[22] and oblivious transfer (OT)[23] providing the cryptographically secure underpinnings of searchable encryption.[24]

The StrongSalt network is a private information storage and retrieval platform because it focuses on security and privacy at its core, while being an information network because of its blockchain-based mesh network of hierarchical, role-based nodes.

Solution Overview: Decentralized Encrypted Search

The StrongSalt platform leverages research in the area of encrypted search by

[21] "QUIC - Wikipedia." https://en.wikipedia.org/wiki/QUIC.

[22] "Private information retrieval - Wikipedia." https://en.wikipedia.org/wiki/Private_information_retrieval.

[23] "Oblivious transfer - Wikipedia." https://en.wikipedia.org/wiki/Oblivious_transfer.

[24] "OverNest launches GitZero with encrypted code searching at ...." 9 May. 2016, https://techcrunch.com/2016/05/09/overnest-launches-gitzero-with-encrypted-code-searching-at-techcrunch-disrupt-battlefield/.

ensuring that all data is encrypted by default and all encrypted data is searchable.

As we continue to produce more and more data, more and more of our data contains private information. As our datasets get larger, search becomes critical to making our data accessible and useful. Finally, we often trade off between security of our data and efficiency in searching our data. Lean too much one way, and you've sacrificed privacy. Lean too much the other way, and you've created an impractical application.

StrongSalt is building decentralized encrypted search, which we define as a discovery mechanism for distributed, chunked index files. As the client-initiated search scatters the queries to the distributed indices, the results must be gathered and annotated with source attribution. Built into the query pipeline will be timeouts that short circuit slow sources and report on which sources were successfully queried.

The StrongSalt query mechanism shall employ some amount of natural language processing (NLP)[25] to provide convenience and disambiguation functions, such as when one searches for the term "lease", a rental agreement absent of the term "lease" should be returned.

In building the search ranking functionality for StrongSalt, we shall leverage evaluation measures such as discounted cumulative gain (DCG)[26] and its normalized equivalent nDCG, both below.

$$DCGp = \sum_{i=1}^{p} \frac{reli}{log2(i+1)}$$

$$nDCGp = \frac{DCGp}{IDCGp}$$

One area of encrypted search is called encrypted term query whereby the keys containing a word or series of words are returned as a result of building an encrypted, inverted index of all the terms. In the figures below one can see the data pre-storage as well as the queries on the encrypted index.

```
Landlord1 => {
  Name: John Smith,
  Age: 25,
  Properties: [
    {
      Address: 3571 Nevada Circle,
      City: San Jose,
      State: California,
      Zip: 92347,
      Years Owned: 5
    },
    {
      Address: 1245 Poplar Street,
      City: Las Vegas,
      State: Nevada,
      Zip: 92445,
      Years Owned: 1
    }
  ]
}
```

Figure 1:
Sample keys and objects for encrypting in StrongSalt

```
Plain Text Index

John  -> ["Landlord1"]
Smith -> ["Landlord1"]
Name  -> ["Landlord1", "Landlord2"]

Encrypted Index

hmac(John)  -> aes256(["Landlord1"])
hmac(Smith) -> aes256(["Landlord1"])
hmac(Name)  -> aes256(["Landlord1", "Landlord2"])
```

Figure 2:
Plain text term HMAC hashing and corresponding object AES-256 encryption

[25] "Natural language processing - Wikipedia." https://en.wikipedia.org/wiki/Natural_language_processing. Accessed 4 Mar. 2019.

[26] "Evaluation measures (information retrieval) - Wikipedia." https://en.wikipedia.org/wiki/Evaluation_measures_(information_retrieval). Accessed 4 Mar. 2019.

Approaches to encrypted search under consideration include fully-homomorphic encryption (FHE), oblivious RAMs (ORAM), property-preserving encryption, functional encryption, and structured encryption. [27] Most of these approaches are insufficiently secure on their own, while others are performance constrained by the number of documents, which is why we favor structured encryption as our design approach.

Bost, Minaud, and Ohrimenko in their paper titled "Forward and Backward Private Searchable Encryption from Constrained Cryptographic Primitives" state that "The security of an SSE scheme expresses the fact that the server should learn as little as possible about the content of the database and queries." Further, "An SSE scheme Σ is $L-adaptively-secure$, with respect to a leakage function L, if for any polynomial-time adversary A issuing a polynomial number of queries $q(\lambda)$, there exists a PPT simulator S such that: [28]

$$\left|\mathbb{P}\left[\text{SSEREAL}_{A}^{\Sigma}(\lambda,q)=1\right]-\mathbb{P}\left[\text{SSEIDEAL}_{A,S,\mathcal{L}}(\lambda,q)=1\right]\right|=negl(\lambda).$$

Solution Architecture

The StrongSalt architecture ensures that any data stored on its network is encrypted. It also ensures that same encrypted data is easily and efficiently searchable, eliminating any practical reasons for developers and businesses not to encrypt their users' and customers' data at rest.

In the world of encrypted search, when having to choose between security and efficiency or how much to lean in any one direction, StrongSalt favors achieving a balance between both. We are able to achieve this balance with structured encryption, a type of encryption that is specific to the data structure[29] being encrypted. Common examples include set encryption schemes and graph encryption schemes, each supporting its respective query types -- for example, set membership and neighbor queries, respectively.

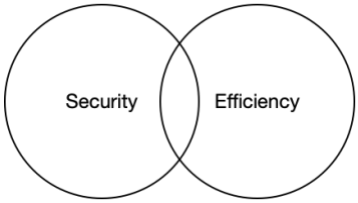


Figure 1:
StrongSalt's encrypted search solution balances between data security and query efficiency

Another concept important to dealing with large datasets is controlled disclosure whereby a data owner wishes to only grant access to a particular subset of data. Being able to do so without requiring the client to leak unnecessary information requires classifying computation or algorithmic effort to be performed locally (on a subset of data) instead of globally (across the entire dataset). [30]

Bost, Minaud, and Ohrimenko define a common leakage function in their paper titled "Forward and Backward Private Searchable Encryption from Constrained Cryptographic Primitives": the search pattern as follows:

In its internal state, the leakage function records the list Q of every search query, in the form (u, w), where u is the *timestamp*

[27] "Encrypted Search - Brown CS." https://cs.brown.edu/~seny/pubs/esearch.pdf.

[28] "Forward and Backward Private Searchable Encryption from …." https://eprint.iacr.org/2017/805.pdf.

[29] "Data structure - Wikipedia." https://en.wikipedia.org/wiki/Data_structure.

[30] "Structured Encryption and Controlled Disclosure - Cryptology ePrint …." https://eprint.iacr.org/2011/010.pdf.

(an index starting at 0 and increasing with every query) and w is the searched keyword. The search pattern is defined as a function $N \rightarrow P(N)$ with

$$sp(w) = \{u: (u, w) \in Q\}.$$

Thus, sp leaks which search queries relate to the same keyword. [31]

StrongSalt takes a multilayered approach in its overall design whereby each layer provides an abstraction from the layer below it allowing for more degrees of freedom and flexibility of implementation.

If one is familiar with the Open Systems Interconnection (OSI) model [32], StrongSalt takes a very similar approach dividing its functionality into layers much like the OSI model partitions a communication system into abstraction layers.

It should be noted that while the StrongSalt protocol goes lower in its protocol definition than most blockchain projects, it does so in order to solve real world latency problems. In particular the StrongSalt protocol is concerned with the transport layer, which is equivalent to both the ISO network and transport layers.

The StrongSalt protocol consists of the following layers (bottom up):

1. Transport Layer
2. Blockchain Layer
3. Messaging Layer
4. API Layer
5. Application Layer

[31] "Forward and Backward Private Searchable Encryption from ...." https://eprint.iacr.org/2017/805.pdf.

[32] "OSI model - Wikipedia." https://en.wikipedia.org/wiki/OSI_model.

These layers work together to create a cohesive environment in which to build secure, private, and decentralized applications with ease.

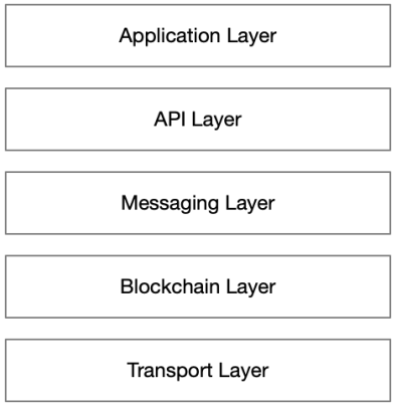


Figure 2:
The five (5) layers of the StrongSalt protocol

While each layer is independent and follows the separation of concerns (SoC) pattern, each layer also lives to serve the layer above it, while being served by the layer below. We will describe each layer's purpose and intended goal in relation to its surrounding layers in the next five sections.

Solution Architecture: Transport Layer

The Transport Layer much like the layer of the same name in both the ISO model and TCP/IP protocol stack provides host-to-host communication services for applications. In doing so it is responsible for services related to flow control, connections, and reliability. [33]

The StrongSalt team made a conscious decision not to go any lower in the network stack in order to maintain compatibility with the widest range of networking

[33] "Transport layer - Wikipedia." https://en.wikipedia.org/wiki/Transport_layer.

equipment as well as the research and development underway for Internet2 [34].

The StrongSalt protocol will be built on QUIC (Quick UDP Internet Connections), a transport layer alternative to the Internet protocol suite set of conceptual protocols, more commonly known as TCP/IP. “QUIC is an experimental network transport layer protocol initially designed, implemented, and deployed by Google in 2012.” [35]

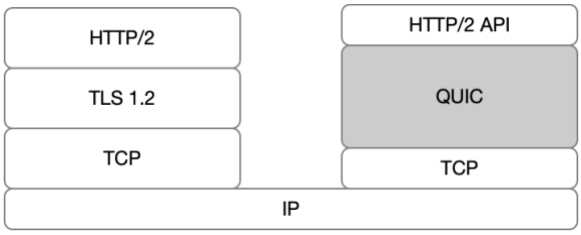


Figure 3:
Google’s description of where QUIC fits in the network protocol stack[36]

QUIC aims to solve a number of shortcomings with TCP namely:

1. perceived performance,
2. connection and transport latency, and
3. congestion.

Perceived performance: While HTTP/2 utilizes multiplexed connections, it cannot gracefully recover from an error on a single data stream, while QUIC allows for independent streams on each of the multiplexed connections, thus allowing for non-blocking behavior.

Connection and transport latency: QUIC is able to reduce connection and transport latency by reducing the number of roundtrip handshakes to just one, while utilizing User Datagram Protocol (UDP) instead of Transmission Control Protocol (TCP).

Congestion: Congestion is avoided by the use of bidirectional bandwidth estimation as part of the protocol itself.

In fact, both the IETF HTTP and QUIC working groups were moving to make QUIC a worldwide standard by requesting to rename the protocol to HTTP/3; this has now been finalized. [37]

All of this background is to say that QUIC is the ideal choice of technology when it comes to a transport layer for StrongSalt protocol behavior as well as decentralized networks. This is because the traffic patterns exhibited by our network’s nodes, such as short messages, replication requirements, and the Scatter/Gather pattern[38] tend to require a large number of connections and subsequent round-trip handshakes.

---

[34] "Internet2." https://www.internet2.edu/.

[35] "QUIC - Wikipedia." https://en.wikipedia.org/wiki/QUIC.

[36] "Redefining Internet Transport Presenter: Jana Iyengar." https://httpworkshop.github.io/workshop2015/presentations/iyengar-quic.pdf. Accessed 4 Mar. 2019.

[37] "HTTP-over-QUIC to be renamed HTTP/3 | ZDNet." 12 Nov. 2018, https://www.zdnet.com/article/http-over-quic-to-be-renamed-http3/.

[38] "How Does Scatter/Gather Work? – EEJournal." 9 Feb. 2017, https://www.eejournal.com/article/20170209-scatter-gather/.

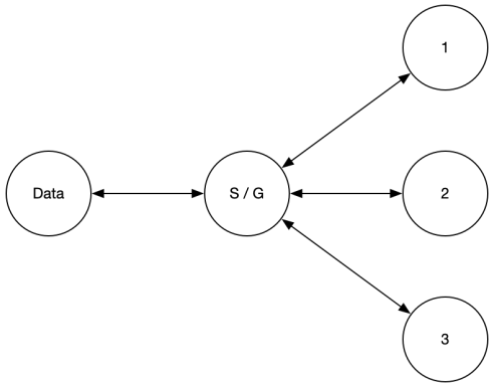


Figure 4:
A demonstration of the scatter / gather pattern

Solution Architecture: Blockchain Layer

Although the StrongSalt protocol utilizes a blockchain data structure and associated network constructs to provide the immutable and trustless nature of its network, StrongSalt is chain agnostic, and as such, does not require any particular blockchain. What this means and why is due to a number of reasons having more to do with the primary goal of the StrongSalt project rather than the specific underlying technology.

The StrongSalt team first and foremost aims to provide a platform on which to build the best secure, private, and decentralized applications. Whether that platform utilizes a private or public blockchain does not matter. Whether that platform utilizes on-chain, off-chain, or a mix of both layer 1 and layer 2 blockchain optimization [39] techniques does not matter.

Instead we choose to focus on delivering a usable and scalable blockchain-based protocol ensuring data privacy with as low a cost to developers and users as possible.

That being said, in the short term, we are interested in projects such as Stellar[40], Ontology[41], and Ethereum[42] for their performance characteristics and / or token creation ability.

In the long term we will likely develop our own blockchain; one that is built from the ground up to support the chatty or short and Scatter/Gather pattern of communication, rather than financial transactions.

What this might look like is a middle ground between signature-based blockchains like Bitcoin and smart contract-based blockchains like Ethereum. The reason for this is because the StrongSalt protocol calls for an if-this-then-that pattern, well-suited for routing and passing messages to and from connected smart devices.

$$if\ (x)\ \{\ y();\ \}$$

The if-this-then-that pattern is more complex than signature-based blockchains can handle, but utilizing a Turing complete blockchain solution would be overkill. We believe this level of complexity will provide the generalization necessary to build a scalable and performant, decentralized private information network.

---

[39] "Making Sense of Ethereum's Layer 2 Scaling Solutions: State ...." 12 Feb. 2018, https://medium.com/l4-media/making-sense-of-ethereums-layer-2-scaling-solutions-state-channels-plasma-and-truebit-22cb40dcc2f4.

[40] "Stellar - Develop the world's new financial system." https://www.stellar.org/.

[41] "Ontology." https://ont.io/.

[42] "Ethereum Project." https://www.ethereum.org/.

Blockchain Layer: Nodes

The blockchain layer provides the concept of nodes and node roles. Given that the blockchain layer functions as an abstraction layer, it is free to introduce any number of node roles so long as it maintains its interface with the layer above it as well as properly utilize the layer below it.

There are currently three types of nodes in the StrongSalt network:

1. Routing nodes,
2. Messaging nodes, and
3. Client nodes

Routing nodes play the role of broker for messaging nodes. While routing traffic between other routing nodes based on channel subscription, they also route lower level traffic such as message traffic between messaging nodes.

In addition to routing, routing nodes also play the role of accountant. They keep track of the work performed by messaging nodes such as message storage and message retrieval. The accounting is then written to the blockchain as transactions, whereby they become available as immutable transactions viewable publicly and verifiable by all on the public ledger that is the blockchain.

When it comes to compensation, the routing nodes' accounting results in payment to all nodes who have performed work. In this case both the routing nodes and messaging nodes will each earn a proportion of the StrongSalt tokens allocated for messaging activity.

Messaging nodes play the role of broker for client nodes. When a message is published, the messaging nodes forward this client traffic to routing nodes, ensuring that messages from one client will ultimately be received by all subscribed clients.

Messaging nodes are also responsible for providing notification services such as when a client receives a message on a channel to which they are subscribed. In this way messaging nodes implement the hierarchical publish-subscribe model of StrongSalt protocol's messaging backbone.

Finally, messaging nodes also provide message storage and message retrieval services as part of the message duplication and scattering process mentioned in the Transport Layer section.

Finally, if one day a better decentralization technology than blockchain comes to fruition, we will not hesitate to explore and adopt as we see fit.

Blockchain Layer: Consensus

When it comes to consensus algorithms, the StrongSalt team prefers a combination of Federated Byzantine Agreement (FBA)[43] and Proof of Stake (PoS)[44], owing to the centralized tendency of mining when Proof of Work (PoW) is concerned. [45]

---

[43] "The Stellar Consensus Protocol: A Federated Model for Internet-level …." https://www.stellar.org/papers/stellar-consensus-protocol.pdf.

[44] "What is Proof of Stake? – Hacker Noon." 6 Oct. 2017, https://hackernoon.com/what-is-proof-of-stake-8e0433018256.

[45] "Mining Centralization Scenarios – Jimmy Song – Medium." 10 Apr. 2018, https://medium.com/@jimmysong/mining-centralization-scenarios-b74102adbd36.

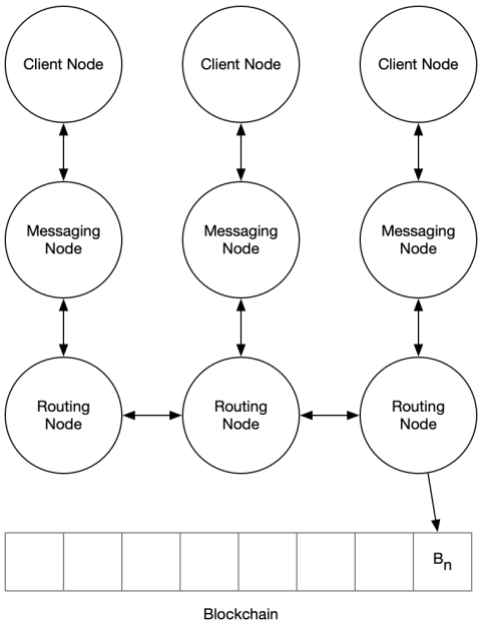


Figure 5:
Client, messaging, and routing node interaction; routing node writing to blockchain

Blockchain Layer: Token Economics

Token economics is a concept at the core of permissionless blockchain networks where anyone is free to join, participate, and either act good or bad. A properly designed token economic system favors good behavior. [46]

As such a utility token requires an ecosystem of motivated actors and corresponding incentives in order to provide value. The Blockchain Layer requires a protocol token in order to operate -- the StrongSalt Token. Depending on one's role within the StrongSalt network, one may be spending or earning tokens. And in one particular case, both spending and earning tokens.

[46] "Nuances Between Permissionless and Permissioned Blockchains." 17 Feb. 2018, https://medium.com/@akadiyala/nuances-between-permissionless-and-permissioned-blockchains-f5b566f5d483.

The StrongSalt network has the following actors:

1. Application developer;
2. Service Provider;
3. Hybrid application / service;
4. User.

Each of the actors will have a different use for tokens.

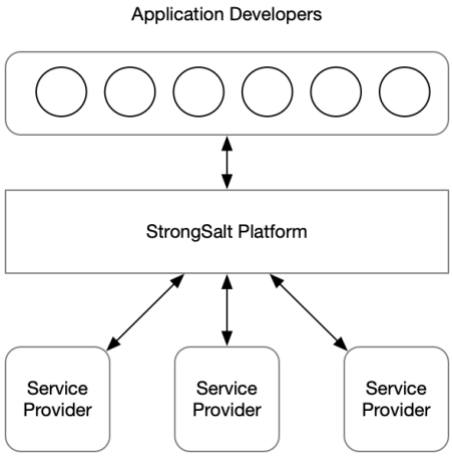


Figure 6:
Application developers accessing service providers via the StrongSalt platform

Application Developer: Companies such as LinkedIn who provide a professional social network platform may be interested in not only encrypting their users' data at rest, but also searchable encryption functionality. This would make LinkedIn and similar platform companies ideal candidates for leveraging the StrongSalt protocol.

In this case LinkedIn is an application developer on the StrongSalt platform, utilizing the StrongSalt protocol and APIs to provide a layer of privacy and security by default.

As a consumer of both storage and compute resources on the StrongSalt platform, LinkedIn would be charged API

access fees. For ease of use and simplicity both storage and compute usage can be abstracted at the API request / response level.

Service Provider: Companies such as Amazon, Google, and Microsoft will function as service providers with their existing cloud services platforms. Amazon has AWS[47], Google has GCP[48], and Microsoft has Azure[49].

The most basic use cases for the service providers involve making cloud storage and cloud compute resources available on the StrongSalt platform in the form of pluggable services in a services marketplace. To continue the example above, these service providers will be able to charge their customers, such as LinkedIn, for consumption of any and all of their services from the basic storage and compute to more advanced services such as dynamic RDBMs and columnar storage engines.

Hybrid: Application / Service: There exists a 3rd non-user actor in the StrongSalt network; companies such as MongoDB who provide a document store database[50] as a service act like both application developers and service providers. We call these hybrid applications / services.

In the first part of the hybrid scenario, MongoDB is an application developer on the StrongSalt platform, utilizing the StrongSalt APIs to enhance their product offering with encryption at rest and searchable encryption. In the second part of the hybrid scenario, MongoDB is a service provider, now offering its enhanced document store as a service to potential customers.

Because MongoDB both consumes resources and provides resources, it plays on both sides of the token economy.

Users: The applications built on the StrongSalt platform can have users that are businesses or users that are end-users depending on if they are of the LinkedIn type or of the MongoDB type, as in our examples above.

An end-user will typically not have to buy or spend tokens to utilize an application built on the StrongSalt protocol unless the application requires it. However, a user that is a business, specifically consuming resources on the StrongSalt platform in a programmatic fashion will typically incur resource consumption costs.

Solution Architecture: Messaging Layer

The Messaging Layer has been implemented as a publish-subscribe messaging pattern whereby the senders (publishers) and receivers (subscribers) of messages are unaware of each other and merely categorize messages and express interest in topics, respectively. The roles of publishers and subscribers in a publish-subscribe messaging pattern are clearly defined and delineated in order to provide the network scalability desired.

---

[47] "Amazon Web Services (AWS) - Cloud ...." https://aws.amazon.com/.

[48] "Google Cloud." https://cloud.google.com/.

[49] "Microsoft Azure." https://azure.microsoft.com/en-us/.

[50] "Document-oriented database - Wikipedia." https://en.wikipedia.org/wiki/Document-oriented_database.

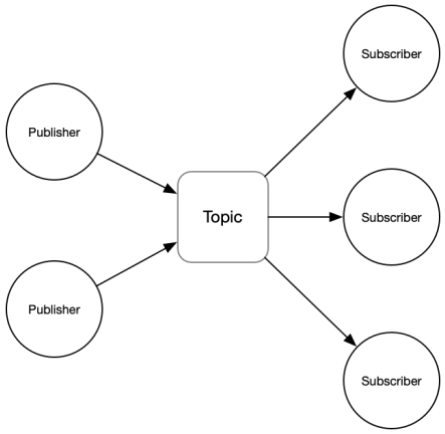
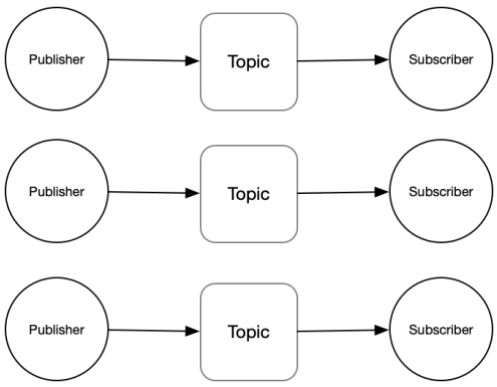


Figure 7:
Publisher / Subscriber pattern

The StrongSalt protocol, however, calls for a hierarchical publish-subscribe pattern[51], providing multiple layers of publishers and subscribers each creating their own messaging system.

Higher level brokers such as routing nodes route traffic at that level, while lower level brokers such as messaging nodes route traffic on their own level.

The hierarchical design affords a higher level of flexibility when it comes to scaling the layers below, the Blockchain and Transport Layers.



[51] "Distributed publish/subscribe networks - IBM." https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_8.0.0/com.ibm.mq.pro.doc/q005120_.htm.

Figure 8:
Hierarchical pub / sub pattern present at each node layer

The messaging layer also provides notification services as implemented by the messaging nodes in the Blockchain Layer. When a client connects to the network with its list of subscribed channels, the client will receive a set of notifications corresponding to any new messages in its queue. The client's initial connection and subsequent receipt of notifications is tantamount to a pull and push model, respectively.

Messaging Layer: Reactive System Manifesto

The Messaging Layer plays a critical role in the security and scalability of the overall StrongSalt network. As such we have adopted the Reactive System Manifesto[52] as a core tenet of StrongSalt. Because of how applicable this set of beliefs is to our cause, we have reproduced it in part below:

Responsive: The system responds in a timely manner if at all possible. Responsiveness is the cornerstone of usability and utility, but more than that, responsiveness means that problems may be detected quickly and dealt with effectively. Responsive systems focus on providing rapid and consistent response times, establishing reliable upper bounds so they deliver a consistent quality of service. This consistent behavior in turn simplifies error handling, builds end user confidence, and encourages further interaction.

Resilient: The system stays responsive in the face of failure. This applies not only to highly-available, mission-critical systems —

[52] "The Reactive Manifesto." 16 Sep. 2014, https://www.reactivemanifesto.org/.

any system that is not resilient will be unresponsive after a failure. Resilience is achieved by replication, containment, isolation and delegation. Failures are contained within each component, isolating components from each other and thereby ensuring that parts of the system can fail and recover without compromising the system as a whole. Recovery of each component is delegated to another (external) component and high-availability is ensured by replication where necessary. The client of a component is not burdened with handling its failures.

Elastic: The system stays responsive under varying workload. Reactive Systems can react to changes in the input rate by increasing or decreasing the resources allocated to service these inputs. This implies designs that have no contention points or central bottlenecks, resulting in the ability to shard or replicate components and distribute inputs among them. Reactive Systems support predictive, as well as Reactive, scaling algorithms by providing relevant live performance measures. They achieve elasticity in a cost-effective way on commodity hardware and software platforms.

Message Driven: Reactive Systems rely on asynchronous message-passing to establish a boundary between components that ensures loose coupling, isolation and location transparency. This boundary also provides the means to delegate failures as messages. Employing explicit message-passing enables load management, elasticity, and flow control by shaping and monitoring the message queues in the system and applying back-pressure when necessary. Location transparent messaging as a means of communication makes it possible for the management of failure to work with the same constructs and semantics across a cluster or within a single host. Non-blocking communication allows recipients to only consume resources while active, leading to less system overhead.

Messaging Layer: Generality of Nodes

The StrongSalt protocol is intentional in its design of a hierarchical node model. Keeping the core routing functionality separate and distinct from the messaging functionality allows us to easily provide additional node types at the secondary level of nodes.

For example, in addition to messaging nodes, one can imagine developing a storage node type that provides semi-permanent storage services. One can also imagine backup nodes for long term backup services and compute nodes for computing, big data, or machine learning services.

Messaging Layer: SLAs

With an architecture that supports any type of service-oriented node, StrongSalt is able to isolate users from existing service providers. As such, StrongSalt's network provides a service level agreement (SLA) [53] between its users and its service providers.

Imagine enterprise cloud services providers such as Amazon or Google offering their cloud services on the StrongSalt network, competing with each other and other providers fairly in a decentralized market driven by incentive-based token mechanics and financial models.

---

[53] "Service-level agreement - Wikipedia." https://en.wikipedia.org/wiki/Service-level_agreement.

In doing so, we believe that StrongSalt can help the net neutrality[54] cause by equalizing the service providers and at the same time avoiding vendor lock in.

Messaging Layer: Actions

The Messaging Layer works like any other queue-based system. When a client is connected to the network, it has these actions available to it:

1. Connect
2. Disconnect
3. Publish
4. Subscribe

Messaging Layer: Quality of Service

Each message, when published, has a Quality of Service (QoS) [55] measurement ascribed to it that determines its overall delivery performance. These levels are currently available:

1. At-most-once
2. At-least-once
3. Exactly-once

At-most-once delivery means that an attempt will be made to deliver the message, but there is no guarantee that the message will be delivered.

At-least-once means that multiple attempts will be made to deliver the message such that at least one is delivered, sometimes resulting in multiple duplicate messages being delivered.

Exactly-once delivery is difficult if not impossible[56] by some opinions, though more importantly is the most expensive and correspondingly has the worst performance. [57]

QoS is an important feature of most networking applications, as not all traffic is created equal. For example, if you are browsing the web, and a packet fails to arrive at its intended destination, that packet can be retransmitted, and the web page will still load, render, and be viewable. However, if you are on a Skype video call, and a few packets get lost, the impact is much higher; the audio from the video call could be garbled, and retransmitting the lost packets would not help because that moment in the conversation will have passed. [58]

Messaging Layer: Key Functions

There are several key functions that are needed to be performed on messages in the StrongSalt Messaging Layer:

1. Message Scattering
2. Message Gathering
3. Message Duplication
4. Message Expiration

---

[54] "What Is Net Neutrality? The Complete WIRED Guide | WIRED." https://www.wired.com/story/guide-net-neutrality/.

[55] "Quality of service - Wikipedia." https://en.wikipedia.org/wiki/Quality_of_service.

[56] "You Cannot Have Exactly-Once Delivery – Brave New Geek." 25 Mar. 2015, https://bravenewgeek.com/you-cannot-have-exactly-once-delivery/.

[57] "Message Delivery Reliability • Akka Documentation." https://doc.akka.io/docs/akka/2.5/general/message-delivery-reliability.html.

[58] "The Basics Of QoS | Network Computing." 15 Aug. 2016, https://www.networkcomputing.com/networking/basics-qos/402199215.

Each of these is required in a distributed and decentralized system.

Message scattering is necessary for long messages if we are to ensure network performance and storage requirements. Long messages must be chopped into smaller message fragments and scattered among the network nodes.

Message gathering is required as a result of the network supporting message scattering. Once a message has been chopped into smaller fragments and scattered, a process to gather and merge is responsible for reconstructing the original message before it can be delivered to its final destination.

Message duplication is useful for a network that desires resilience through redundancy, and particularly important for a decentralized environment where no guarantees exist for node operator minimums, whether it be architectural or political decentralization. [59]

Message expiration along with the combination of encrypting and scattering is a key feature of the StrongSalt protocol. This is to deter behavior such as the collecting and deciphering of expired messages.

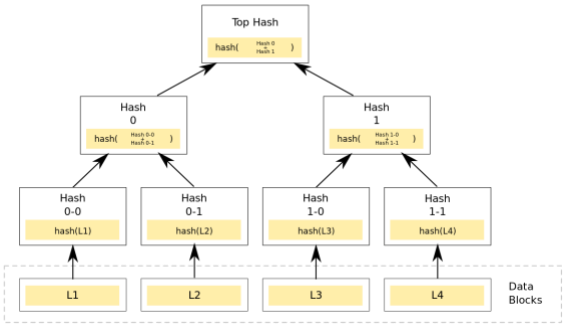
It should be noted that although all messages have an expiration, due to the fact that StrongSalt is not a permanent storage network, it is possible to build a highly scalable storage network on stop of it.

Janitorial services for expired messages include garbage collecting and discarding, however, determining whether or not the garbage collected messages were indeed removed can be a costly activity. [60]

Messaging Layer: Merkle Trees

In computer science Merkle trees or hash trees play an important role in the recordkeeping of chunked data. Their structure provides efficient and secure verification of the contents of data structures, especially large ones, such as a blockchain database. [61]

The StrongSalt network utilizes Merkle trees for the scatter / gather activities mentioned in the Transport Layer section. Much like a BitTorrent tracker tracks which peers have which file copies, a Merkle tree tracks all pieces of a message, ensuring none are subject to tampering or destruction. [62]



[59] "The Meaning of Decentralization – Vitalik Buterin – Medium." 6 Feb. 2017, https://medium.com/@VitalikButerin/the-meaning-of-decentralization-a0c92b76a274.

[60] "Costs of a Real World Ethereum Contract – Hacker Noon." 10 Aug. 2017, https://hackernoon.com/costs-of-a-real-world-ethereum-contract-2033511b3214.

[61] "Merkle Trees – Hacker Noon." 15 Dec. 2017, https://hackernoon.com/merkle-trees-181cb4bc30b4.

[62] "BitTorrent tracker - Wikipedia." https://en.wikipedia.org/wiki/BitTorrent_tracker.

Figure 9:
Merkle tree diagram[63]

Merkle trees are used in other areas of the StrongSalt network as well:

1. Blockchain Layer Accounting
2. Missing Message Fragment Identification

The Blockchain Layer is responsible for tracking which nodes are doing what work. It does so by storing a map of the message hashes to messaging nodes, and in doing so, performs the accounting of the temporary storage function the messaging nodes play.

Multiple messages can be hashed together in one transaction, providing significant storage savings on the blockchain. In addition to reducing transaction size and helping the network to scale, message batching provides a certain level of privacy at the transaction level, while preserving auditability. This is conceptually similar to cryptocurrency tumbling[64] whereby identifiable cryptocurrency funds are mixed with others to obscure the provenance of the funds.

Because we are using a Merkle tree to store the scattered message fragments, it becomes trivial to identify which message fragments are missing during the gathering phase of scatter / gather. This is due to the highly performant method of querying for the presence of a hash in a Merkle tree, namely $O(log\ n)$ time complexity, as compared to typical time complexity, which is proportional to the number of leaf nodes in the binary tree. [65]

Solution Architecture: API Layer

StrongSalt's API Layer is the entryway into the StrongSalt platform. Whether you are a service provider providing cloud storage functionality by way of the StrongSalt API or are an application developer consuming the StrongSalt APIs, these APIs are how all the non-user ecosystem actors will interface.

```
/v1/encrypt/

{
  first_name: "alice",
  ssn: 123456789,
  score: 88
}
```

Figure 10:
Sample API call to store data encrypted by default

StrongSalt's API Layer is a thin layer on top of the Messaging Layer. However, due to StrongSalt's layered architecture, this thin layer provides access to a feature-rich platform -- encrypting data by default and ensuring that data doesn't get lost forever by making it fully searchable.

The API Layer is also modeled after the publish-subscribe pattern. For much of the same reasons above, we believe the pub-sub pattern to be the best way to provide the decoupling desired for the message senders and receivers.

While sitting just below the Application Layer, the API Layer provides an abstraction such that the Application Layer need not be aware of how the publish-subscribe functionality is implemented by

[63] "Merkle tree diagram - Wikipedia." https://upload.wikimedia.org/wikipedia/commons/thumb/9/95/Hash_Tree.svg/1200px-Hash_Tree.svg.png.

[64] "Cryptocurrency tumbler - Wikipedia." https://en.wikipedia.org/wiki/Cryptocurrency_tumbler.

[65] "Merkle tree - Wikipedia." https://en.wikipedia.org/wiki/Merkle_tree.

the Messaging Layer, or that there even exists a Messaging Layer.

Meanwhile the API Layer is not concerned with how the publish-subscribe API is being used -- a user is no more than an opaque string its client is manipulating, as in the opaque data type concept. [66]

```
/v1/encrypt/

/v1/decrypt/

/v1/search/

/v1/share/
```

Figure 11:
Fundamental API endpoints to read, write, search, and share data

Solution Architecture: Application Layer

The StrongSalt Application Layer is where developers will build their applications, decentralized or not, that will take advantage of StrongSalt's full protocol stack. This means that applications built on the StrongSalt platform will have security and privacy built in, protecting their users' data by default. These same applications will also have data encryption and searchable encryption by default, all while leveraging a low latency transport layer.

As we will discuss in the future use cases section, the Application Layer will be home to both platform providers and service providers looking to build encryption and searchable encryption natively into their offerings.

Any application developer is free to implement their own utility token with its own token mechanics and economic model on top of the StrongSalt protocol token.

Solution Architecture: Security

When it comes to security, StrongSalt takes a unique approach to ensuring that data stored on its network is always secure. Security is implemented via a layered approach, where each layer implements its own security features, while at the same time avoiding duplication of security efforts among the other layers.

Though each layer addresses its own particular security concerns, the security of the overall system should be more than the sum of that of each layer. [67]

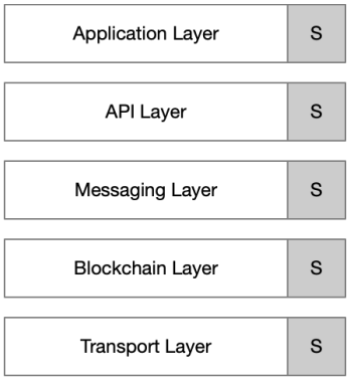


Figure 12:
Each layer is responsible for its own security

The Transport Layer leverages the encryption-by-default provided by the QUIC protocol, namely a more efficiently brokered TLS handshake.[68] This ensures that communication between nodes of the Blockchain Layer is secure, and only what is required to be exposed is exposed.

---

[66] "Opaque Data Types - IBM." https://www.ibm.com/support/knowledgecenter/en/SSGU8G_12.1.0/com.ibm.sqlr.doc/ids_sqr_165.htm.

[67] "Defense in Depth | US-CERT." 13 Sep. 2005, https://www.us-cert.gov/bsi/articles/knowledge/principles/defense-in-depth.

[68] "draft-ietf-quic-tls-18 - Using TLS to Secure QUIC - IETF Tools." 22 Jan. 2019, https://tools.ietf.org/html/draft-ietf-quic-tls-18.

The Blockchain Layer implements its security via a trustless model utilizing a blockchain to decentralize authority. A token economic model with mechanics and incentives for service providers and consumers ensures the appropriate behaviors are ones that are financially beneficial. [69]

The Messaging Layer is responsible for ensuring messages are securely sent and received as implemented by the hierarchical publish-subscribe pattern. This is accomplished by ensuring all connections to and from the Blockchain Layer nodes are encrypted and appropriate use of security certificates is employed.

The API Layer is capable of providing additional services such as an encryption API that application developers can call to encrypt and decrypt their data. This obviates the need for application developers to make decisions in areas for which they may not be well-versed such as cryptography and encryption scheme selection -- AES 128 vs. 256 or ECB vs. CBC vs. CTR vs. GCM.[70]

The Application Layer is responsible for security of users and users' content, such as data retention or message expiration.

Security: Blockchain

The StrongSalt team has made a critical decision when it comes to blockchain security -- StrongSalt code will not be stored on the blockchain as is the case with typical smart contract, blockchain-based virtual machines (VMs).[71]

StrongSalt code will live on computing nodes, segregated from the blockchain. There are a number of benefits to designing the network in this way, but the primary reasons have to do with scalability and security. By not incorporating the code into the blockchain itself, the blockchain is free to do what it is best at doing -- providing token economics and sharing a distributed accounting ledger. This also has the side effect of reducing the chances of the blockchain becoming an attack vector for the code.

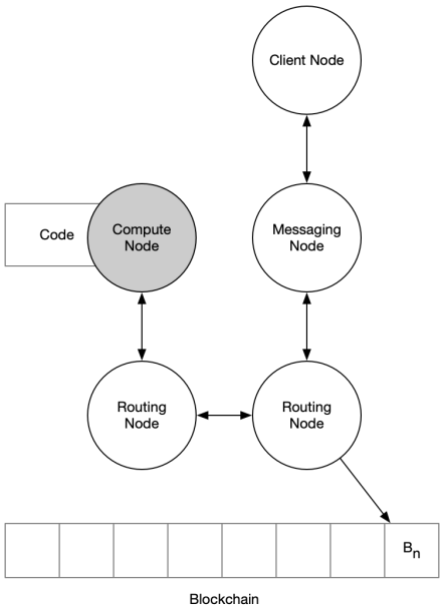


Figure 13:
Code will live on compute nodes not the blockchain

Security: Certificates

The StrongSalt network requires the use of public key certificates for all actors. A

[69] "Basecoin (aka the Basis Protocol): the worst idea in ... - Preston Byrne." 13 Oct. 2017, https://prestonbyrne.com/2017/10/13/basecoin-bitshares-2-electric-boogaloo/.

[70] "Block cipher mode of operation - Wikipedia." https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation.

[71] "Getting Deep Into Ethereum: How Data Is Stored In Ethereum?." 3 Aug. 2018, https://hackernoon.com/getting-deep-into-ethereum-how-data-is-stored-in-ethereum-e3f669d96033.

public key certificate provides information about the public key being represented, information about the identity of its owner, and the digital signature of the entity that has verified the certificate's contents. [72]

Public key certificates along with public key infrastructure (PKI)[73] are often employed in situations where a higher level of security is required during identity confirmation and data validation. [74]

A device attempting to connect to the StrongSalt network must present a public key certificate signed by a certificate authority (CA)[75] trusted by the StrongSalt network. The device's certificate which can be acquired by node assignment or burned in at the factory during manufacturing is then verified by a node on the StrongSalt network.

Nodes on the StrongSalt network in turn must also procure a public key certificate in order to be node operators which also verify device certificates. In order to bootstrap the StrongSalt network, select nodes will function as certificate managers.

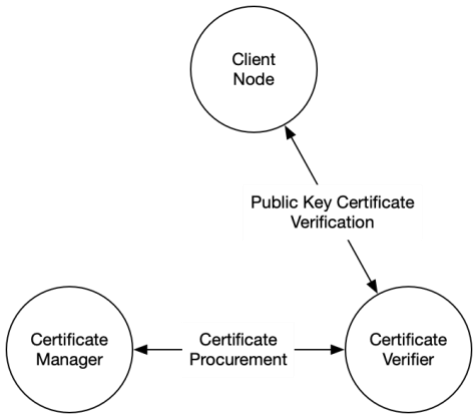


Figure 14:
Public key certificate procurement and verification among nodes

Security: Security Primitives

Because complexity is the enemy of execution, we have purposefully selected for use protocols and cryptographic ciphers that are well-researched and well-understood. This ensures the core of StrongSalt's protocol can be easily audited by academic researchers and industry technologists.

We have selected the latest, widespread protocols and cryptographic ciphers, but will be evaluating and updating as necessary in an ongoing fashion as mathematics, quantum computers, and hardware advances.

All numbers are to be encoded using network big-endian order.[76]

Authentication

AES-256 in CTR mode + HMAC[77]

[72] "Public key certificate - Wikipedia." https://en.wikipedia.org/wiki/Public_key_certificate.

[73] "Certificates and Public Keys - Windows applications | Microsoft Docs." 30 May. 2018, https://docs.microsoft.com/en-us/windows/desktop/seccrypto/certificates-and-public-keys.

[74] "Public key infrastructure - Wikipedia." https://en.wikipedia.org/wiki/Public_key_infrastructure.

[75] "Certificate authority - Wikipedia." https://en.wikipedia.org/wiki/Certificate_authority.

[76] "Network byte order and host byte order - IBM." https://www.ibm.com/support/knowledgecenter/en/SSB27U_6.4.0/com.ibm.zvm.v640.kiml0/asonetw.htm.

[77] "HMAC - Wikipedia." https://en.wikipedia.org/wiki/HMAC.

Hashing Algorithm

SHA256[78]

Diffie-Hellman Key Exchange[79]

Curve25519[80]

Signatures

Ed25519 / EdDSA[81]

Public Key Length

256-Bits (32 bytes)

Signature Length

512-Bits (64 bytes)

Security: Security Attacks

In 2015 there was an article titled "How Secure and Quick is QUIC? Provable Security and Performance Analysis" presented at the IEEE Symposium on Security and Privacy. This article presented a provable, more suitable security model for QUIC in that QUIC must consider reordering and packet injection issues normally handled by TCP, lest it sacrifice security for latency. [82]

This new model, Quick Authenticated and Confidential Channel Establishment (QACCE)[83], considers attackers who can initiate and observe communications in addition to intercept, drop, reorder, or modify any exchanged messages. The study found that "QUIC can successfully protect against such strong attackers and provide QACCE Security." [84]

Adoption

Adoption: IoT Messaging Protocol

The typical evaluation criteria for assessing protocols includes, but are not limited to the following[85]:

- Confidentiality
- Low protocol overhead
- Unstable network tolerance
- Low power usage
- Millions of connected clients
- Push communication

Because the StrongSalt protocol can demonstrate all of these traits, we believe it is highly suitable for secure IoT communication use cases.

Adoption: Existing Solutions

[78] "SHA-2 - Wikipedia." https://en.wikipedia.org/wiki/SHA-2.

[79] "Diffie–Hellman key exchange - Wikipedia." https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange.

[80] "Curve25519 - Wikipedia." https://en.wikipedia.org/wiki/Curve25519.

[81] "EdDSA - Wikipedia." https://en.wikipedia.org/wiki/EdDSA.

[82] "How Secure and Quick is QUIC? Provable Security and Performance ...." https://ieeexplore.ieee.org/abstract/document/7163028/.

[83] "Authenticated Confidential Channel Establishment and the Security of ...." 1 Oct. 2017, https://dl.acm.org/citation.cfm?id=3146433.

[84] "QUIC: Performance and Security at the Transport Layer – IETF Journal." 12 Nov. 2016, https://www.ietfjournal.org/quic-performance-and-security-at-the-transport-layer/.

[85] "MQTT and IBM MessageSight: Secure, reliable communications for ...." 28 Jan. 2015, https://www.ibm.com/developerworks/websphere/techjournal/1501_maynard/1501_maynard.html.

Though encryption protocols like the one from Signal[86] already exist, most provide end-to-end encryption primarily suited for real time messaging between two parties.[87]

The StrongSalt protocol aims to provide one-to-many and many-to-many[88]communication, and thus requires a protocol designed from the ground up that not merely accommodates, but natively supports multiple senders and multiple receivers.

Telegram[89] is another popular messaging application, however, it is just that -- an application. It also does not encrypt multi-party communication[90], and is at the same time centralized. The founders of Telegram have plans to provide a decentralized platform, but nothing has been released yet. [91]

The rest of the messaging apps and their associated protocols follow a centralized model and do not provide the type of encryption a secure, private information network requires.

The StrongSalt protocol will provide the foundation upon which any messaging app can build upon without each having to worry about security, encryption, and latency on their own.

[86] "Signal >> Home." https://signal.org/.

[87] "Signal >> Specifications >> The Double Ratchet Algorithm." 20 Nov. 2016, https://signal.org/docs/specifications/doubleratchet/.

[88] "Many-to-many - Wikipedia." https://en.wikipedia.org/wiki/Many-to-many.

[89] "Telegram Messenger." https://telegram.org/.

[90] "In contrast, Telegram does no encryption at all for ... - Hacker News." https://news.ycombinator.com/item?id=16117487.

[91] "Telegram to Debut 'Test Version' of Blockchain Platform TON 'This ...." 16 Oct. 2018, https://cointelegraph.com/news/telegram-to-debut-test-version-of-blockchain-platform-ton-this-autumn-say-investors.

Adoption: Tor

The Tor[92] project was started in 2002, and while successful at what it was designed to do, does not satisfy the requirements for a generic communication platform where communication can be peer-to-peer and bidirectionally initiated.

Tor's primary use case is anonymous website access[93], which can be seen as one-to-one communication where a user or client accesses information on a server. A generic communication platform cannot be limited to one-to-one or even n-to-one communication, but rather must support m-to-n communication.

In addition, a generic communication platform should support offline communication, which does not make sense in the Tor anonymous website access pattern.

From a security perspective, Tor uses SHA1 instead of the recommended SHA256 version of the cryptographic hash function due to its need to maintain backward compatibility with hybrid encryption.

Finally, the type of data or traffic Tor was designed to anonymize is high-latency TCP-based activity. Given this, Tor more closely resembles an application level protocol, rather than a low-latency network level protocol built on the QUIC protocol.

Adoption: Future Use Cases

There are several obvious use cases for building on top of the StrongSalt protocol given its decentralized privacy features:

- Genetic Testing

[92] "Tor." https://www.torproject.org/.

[93] "Who uses Tor?." https://www.torproject.org/about/torusers.html.en.

- Decentralized video conferencing
- Enterprise group collaboration
- Data escrow

Use Case: Genetic Testing

Imagine the genetic testing company 23andMe[94] that popularized home based testing for health reports, genetic traits, and family history. Over the years they have amassed a trove of genetic data that they can now sell to pharmaceutical companies like GlaxoSmithKline (GSK).

Rather than sell their users' genetic data in a one-time transaction, 23andMe can share or rent the data to GSK via the StrongSalt platform. 23andMe's data never leaves the StrongSalt platform, and as such, 23andMe retains full control over data access and usage. In fact, 23andMe can grant certain permissions to the original genetic data's owner in a multi-level data ownership model.

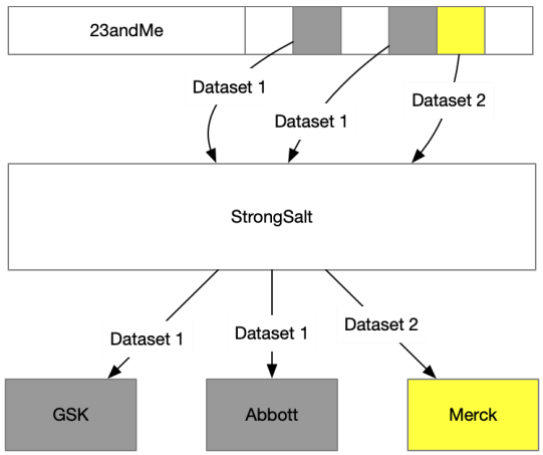


Figure 15:
23andMe sharing data selectively among pharmaceutical companies

From an economics perspective, GSK will pay less for access to data vs. a one-time data dump, but ultimately the 23andMe data is worth more under 23andMe control and subsequently can be shared and rented to other pharmaceutical companies.

Use Case: Decentralized Video Conferencing

For those who care about preserving privacy when it comes to video conferencing whether it be for personal use or business use, the only way to ensure complete privacy is to ensure that no centralized authority has access to the internal operations of the platform or network.

In this case we would build a decentralized video conferencing application where the content and data are routed and coordinated by a decentralized network of operators.

Use Case: Enterprise Group Collaboration

Currently, many cloud-based enterprise group messaging products and platforms store most if not all of their data in the cloud. As a result, the companies that operate these platforms retain complete control over your conversations, emails, and business data. More often than not, this means these companies can also view your private data whenever they want.

Though these same companies may use marketing speak such as "encryption" and "firewalls", as long as they continue to perform the encryption and hold the encryption keys, data privacy does not exist.

Utilizing the StrongSalt protocol we can build a decentralized enterprise group collaboration product or platform that is truly private.

Use Case: Data Escrow

Imagine a data escrow service. Payments are guarded by smart contracts. Though generic blockchain smart contracts can

94 "23andMe." https://www.23andme.com/.

provide generalized escrow services, StrongSalt will focus on data escrow services only.

StrongSalt would provide a list of pre-coded smart contracts, each with its own custom logic specific to the escrow type. The complexity would be hidden behind an intuitive user interface such that the application would have mass appeal.

The escrow service would only release the product, say a digital piece of art, and charge the buyer the corresponding tokens after the conditions of the smart contract were satisfied.

In addition, we could instruct for a preview to only be downloadable once the tokens are in escrow, and the tokens not released to the seller until the full artwork was downloaded.

Adoption: Node Types

Throughout the course of this paper we have mostly discussed storage services, however, one can imagine any number of useful node types providing services for the decentralized applications to consume. Similar to how the Internet evolved, first you need the network, then you can provide the services.

Examples of additional node types and services include long term data storage and computation services. These services will provide work in exchange for tokens as part of the token economic model brokered by the Blockchain Layer discussed above.

Adoption: Decentralized Cloud

In many ways, the StrongSalt network functions as a decentralized cloud such as Amazon Web Services (AWS) or Google Cloud Platform (GCP). This is in large part due to the service provider abstraction provided by the StrongSalt protocol. In fact, both Amazon and Google can participate in the StrongSalt economy by offering their services in an a la carte fashion on the StrongSalt network.

Summary

Today we face ever-increasing amounts of data creation. By some estimates we are creating 2.5 quintillion bytes of data every day. [95] Inevitably that data contains more and more personally identifiable information (PII)[96] Based on the scale and severity of data attacks in 2018 alone, it is clear that businesses are not sufficiently protecting its users data. [97]

The StrongSalt team is building a set of protocols and a platform to ensure that data is encrypted by default. We are making that encrypted data fully searchable, easy to store and retrieve, and in a cost-effective ecosystem. There should be no excuses to properly protect the world's data -- especially when the ramifications of not doing so are so dire. [98], [99]

---

[95] "How Much Data Do We Create Every Day? The Mind-Blowing Stats ...." 21 May. 2018, https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/.

[96] "Personally identifiable information - Wikipedia." https://en.wikipedia.org/wiki/Personally_identifiable_information.

[97] "The 21 scariest data breaches of 2018 - Business Insider." 30 Dec. 2018, https://www.businessinsider.com/data-hacks-breaches-biggest-of-2018-2018-12.

[98] "The Equifax Data Breach: What to Do | Consumer Information." 8 Sep. 2017, https://www.consumer.ftc.gov/blog/2017/09/equifax-data-breach-what-do.

[99] "Equifax Says Cyberattack May Have Affected 143 Million in the U.S. ...." 7 Sep. 2017,

Contact

Come chat with the StrongSalt team on Telegram! We'd love to meet you and learn what brought you here.

https://www.nytimes.com/2017/09/07/business/equifax-cyberattack.html.