Point Network - A Holistic Implementation of Web 3.0

Serge Var
<serge@pointlabs.org>

Dr. Amaury Hernandez
<amaury@pointlabs.org>

Darren Jensen
<darren@pointlabs.org>

Denis Voropaev
<denis@pointlabs.org>

Abstract. Web 3.0, also known as decentralized internet, is a term designating an envisioned peer-to-peer network, run on a consensus-defined set of protocols, open for everyone, decentralized, transparent, neutral, permissionless, censorship-resistant, and privacy-preserving, which could potentially replace the current generation of the internet protocols due to its improved security and privacy. This journey toward Web 3.0 started with applications such as Tor, Bittorrent, Bitcoin, Ethereum, and IPFS, and now blossomed into a myriad of projects under the label of "Web 3.0". However, few achieve these desired properties because, typically, only a few components are sufficiently decentralized. This white paper describes Point Network, a set of software components and protocols as proposed decentralized replacements to all standard parts of the modern internet. It includes a blockchain-powered consensus layer, a decentralized storage layer, decentralized domains and identities, and a web3-enabled browser that allows users to access the network directly. To exemplify how Point Network enables Web 3.0, we provide a set of common Web 2.0 applications reimagined using our architecture, such as Point Mail, end-to-end encrypted email, Point Social, a social network redesigned in the Web 3.0 way, and others. In this way, Point Network presents significant improvements on the core protocols of the legacy internet that can be achieved regarding privacy, security, censorship-resistance, and transparency. It thus fulfills the defined criteria for Web 3.0 in full.

Draft v0.5.1, last update: May 23, 2022

Table of Contents

Introduction

The internet's privacy and security, as they currently stand, present multiple flaws. Information security specialists, cryptographers, and free speech advocates are fighting an uphill battle.

Insecure protocols. The internet was not initially designed with security in mind [DeNardis, 2007]. Decades later, we still have to suffer from the security-as-an-afterthought approach. Hardware and software vendors are barely catching up with zero-day vulnerabilities, with lines of code and the number of dependencies growing geometrically, and attacks becoming increasingly more sophisticated [Dessouky et al., 2020];

Mass-surveillance. Government intelligence agencies, to conduct mass-surveillance in the name of national security, intentionally lower everyone's protection mechanisms by introducing backdoors into hardware and software [Lear, 2018], keeping zero-day vulnerabilities they find to themselves instead of helping patch them [Moshirnia, 2018] [Slupska et al., 2021], and hoarding everyone's private files, emails, photos, documents, pictures, calls, geolocation and browsing history in massive government data centers [Coddington, 2017]. Not only it has been revealed that this information gets abused by government employees for personal, career and political reasons [Slobogin, 2007] [Anderson, 2019] [Snowden2019], but it has become obvious that even highly classified information in intelligence agencies is not secure from theft[1] [Shane et al., 2017] [Brevini, 2017], and if unfettered access to the full archives of billions of people's digital lives falls into the wrong hands, the consequences are incalculable.

Censorship. The coalition of the largest social media corporations colloquially known as "big tech" enjoys unprecedented monopolistic control over the flow of information [van der Schyff et al., 2020] [Nadler and Collins, 2017] [Hallam and Zanella, 2017] [Harris, 2020] [Hubbard, 2020], runs opaque courts with unnamed unelected judges without effective appeal mechanisms, occasionally participates in synchronous events of removing social media accounts of the targets, also known as "deplatforming" [Rogers, 2020], and actively meddles in social discourse by censoring certain subjects, links and hashtags and using artificial intelligence to boost certain voices and suppress others [Gunitsky, 2015]. Alternative social media platforms could not solve the problem, because their Achilles heel is getting refused service by the hosting

[1] CIA lost hundreds of classified documents and cyberattack tools in the Vault 7 Leak https://nakedsecurity.sophos.com/2017/03/07/wikileaks-drops-huge-cache-of-confidential-cia-documents/, and the fact that a 29-year old NSA employee Edward Snowden was able to exfiltrate hundreds of thousands of top secret documents undetected until published [link]

providers[2]. Additionally, their domain names can be taken away at any moment [Kopel, 2013] [Molloy, 2021].

While the legal routes taken by public organizations and movements did not yield much change in these areas, technologists have been focused on trying to solve the problems with cybersecurity and cryptography protocols, algorithms and tools. What unites projects such as Bittorrent, Tor Network, Bitcoin, Ethereum, Signal, IPFS and the likes is the spirit of giving back the power to the internet users through decentralized peer-to-peer networks. A shared vision for what the internet should be like permeates their communities, and developers are working on bringing parts of it to reality with these projects. This vision of the global network that is open for participation, free from censorship, permissionless and privacy-preserving has been nicknamed "web 3.0", or "decentralized internet" [Alabdulwahhab, 2018].

Engineers behind Point Network have been inspired by the same ideals, and work towards the same goals. We believe that most of the building blocks necessary to build web 3.0 have already been invented, it is only a matter of developing a few additional components and linking them together.

There are several components that are crucial in the architecture of Point Network: 1) ***decentralized identities*** (which includes decentralized domains), 2) a ***decentralized storage*** layer (which uses Arweave), and 3) a web3-enabled browser (***Point Browser***) combined with a locally running proxy software (***PointProxy***). To achieve our goals, we combine several additional technologies and implementations.

In this paper we provide our vision of the future of web 3.0. We do not wish to cultivate misleading expectations that all of the features, or all of the decentralized applications described in here will be present in our first test implementations. Nonetheless, it marks our direction for where we believe web 3.0 technology is headed, and we expect engineering teams and dApp teams to get inspiration from this work, while not taking this as a definitive and final gospel dictating how it must look in its final form.

[2]A clear example is the shutdown of Parler [Otala et al., 2021] [Kierstead, 2021].

Point Network Architecture

Architecture Overview

Below is an overview of Point Network architecture (Figure 1).

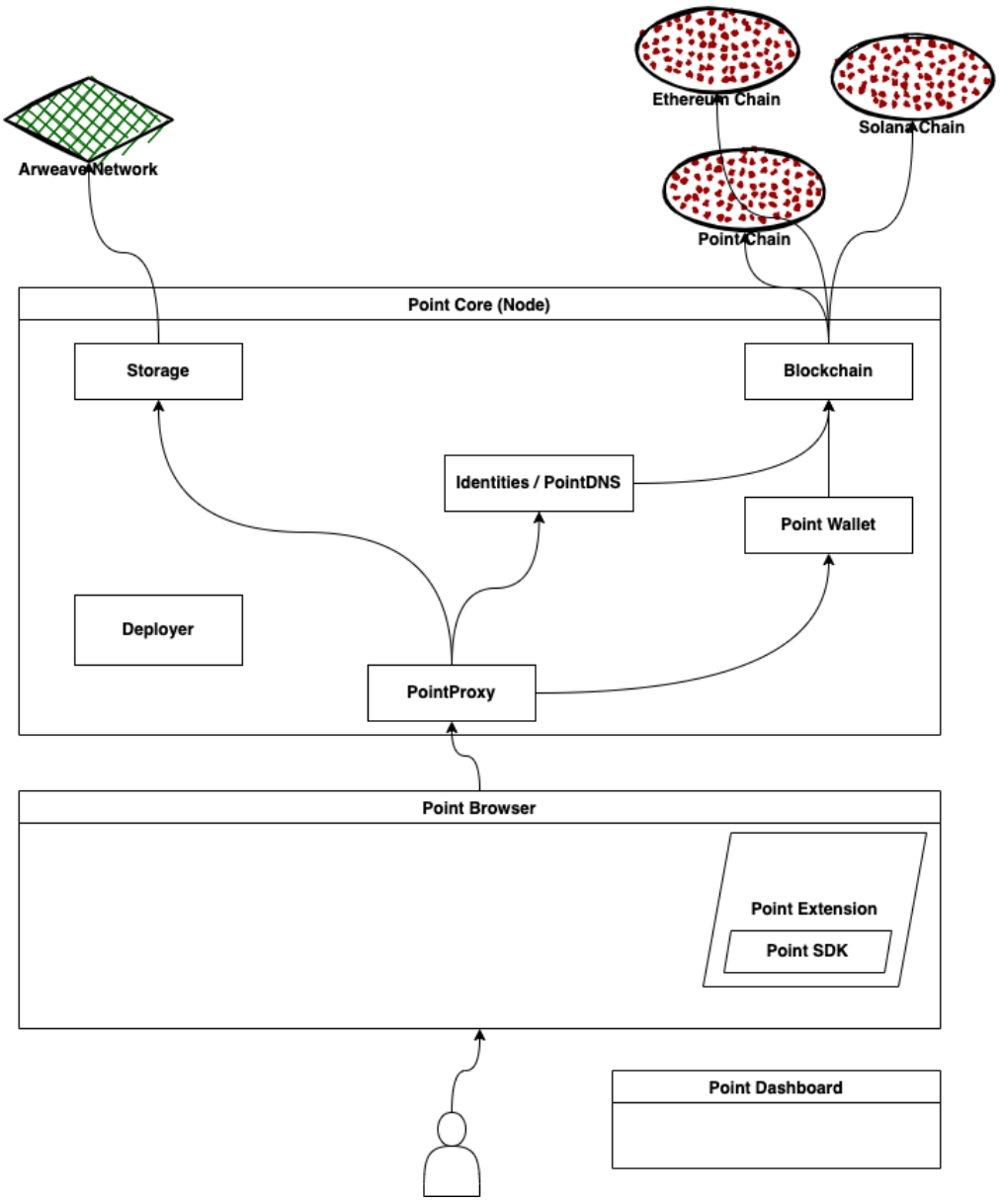

Figure 1. Simplified Point Network architecture

Point Dashboard installs, updates, and manages the rest of the software; it also shows the status of the various components to a user.

Point Browser is a fork of the Mozilla Firefox browser, which is configured to be a web 3.0 client communicating with Point Node. In particular, it has a hard-coded setting that makes all requests go through a port taken by PointProxy, a local proxy software that captures every request. Point Browser is also packed with **Point Extension**, which enables **Point SDK** to be used in decentralized applications, sends notifications, manages the confirmation window, etc.

Point Core (Node) is the backend part running on a user's computer. It contains the following modules:

- **PointProxy**, which is a local proxy software listening on a specific localhost port, capturing all requests from Point Browser, communicating with other components of Point Node, and returning back the response;
- **Storage** module, which uploads and downloads files from and to Arweave decentralized storage network using content-addressable hashes;
- **Blockchain** module, responsible for communicating with various blockchains;
- **Wallet**, responsible for managing user's private keys, sending and receiving tokens and cryptocurrencies on multiple chains, and fetching transaction history;
- **Identities** module, communicating with **PointDNS** smart contract to associate identities with public keys, and retrieve information about decentralized domains;
- **Deployer**, a software tool for deploying decentralized applications and websites onto Point Network;
- and various other extra modules not shown in the simplified Figure 1 above for simplicity.

Request/Response Lifecycle in Point Network

A typical lifecycle of a request-response combo on Point Network looks like this. We will use `https://example.point/user/mike` as our example URL for a website on the decentralized internet, to where a user navigates in ***Point Browser***.

DNS request. Normally, a browser issues a DNS request to a centralized DNS server, turning an alphanumeric domain name into an IP address to reach out to next. However, there is no `example.point` domain on the legacy internet[3]. Instead, a locally running proxy software

[3] Conversely, all connection to the legacy internet is severed from inside Point Browser. Content-addressable Web3 is fully isolated from the "old internet" similarly to how blockchain scripts (smart contracts) are run in environments isolated from the real world, so much so that in order to retrieve any data from the "external world" special "oracles" and "oracle networks" are needed.

called ***PointProxy***, which is part of Point Network software and whose local port is hardcoded in Point Browser's settings, intercepts all requests, including DNS requests, and returns `localhost` for all DNS requests.

HTTP request. Next, a browser typically contacts the server using the provided IP address with an HTTP request (optionally wrapped in SSL/TLS layer for traffic encryption). The HTTP request includes: a *hostname,* a *path* to the requested file, *method* type (GET, POST and others), *variables/body*, and other *metadata/headers.* In the case of Point Network, the HTTP request gets intercepted by PointProxy.

Preparing a *root directory hash* and a *routes file hash*. From the *Hostname* part of the request (`example.point` in our case). PointProxy looks at the key-value storage attached to that domain in the Identities contract to establish which *root directory hash* and *routes file hash* are currently attached to this domain by the identity owner.

> Let's assume that the *root directory hash* in our example is `cf5a...04ef`. When downloaded from Arweave, this returns a directory with folders `css/`, `js/` and files `layout.zhtml`, `index.zhtml` and `user.zhtml` (content-addressed by hash as well).
>
> Let's assume that the *routes file hash* in our example is `48ad...21de`. When downloaded from Arweave, this returns a JSON file: `{"/": "index.zhtml", "/user/:name": "user.zhtml"}`

Routing the request. In our example, the requested URL matches the route `/user/:name` pointing to `user.zhtml` inside the root directory (and `"mike"` will become the value of a template variable `name`). If the URL hadn't matched any route, PointProxy would try to establish whether it matches a path inside the root directory, in which case, the contents of this file would be returned.

Rendering a template. If the file ends with ".zhtml" (as it does in our example case), it first passes through a ZHTML renderer. ZHTML is a backend language available to Point Network dApps, inspired by templating languages such as Jinja. The output of a ZHTML renderer is HTML, which can be parsed by the browser.

Special routes. There is a list of special routes that are considered reserved for extra functionality. For instance, there is a route group `/_storage/{id}` which, when invoked on any domain, returns a file from decentralized storage based on the supplied `id`.

This is done in order to discourage developers of web3 to continue using unsafe centralized facilities to store, and instead forces

Point Storage

The most valuable part of the current internet, without which most of the algorithms would have nothing to operate on and would have been without use, is the content. A new kind of internet would need to store the content somewhere, while adhering to the strict standards of censorship-resistance, surveillance-resistance, and openness we set in defining this new network. Point Network engineers have considered several options that are available, and whether they are suitable for the set purpose.

Cloud storage. This is the default option for the current generation of the internet. In the majority of the cases, a file's location is defined by *http* or *https* protocol designation, an IP address or a domain name which gets resolved to an IP address, and a path to that file. Here are the downsides of cloud storage that we believe makes it unsuitable for web3.0:

> ***No data integrity.*** The server on the other end is free to serve any content, and users, lacking data integrity verification tools, would generally miss it if it were modified. For example, when one stores files on *Google Drive* or *Amazon Web Services*, they can never be sure that the files they download back have not been modified. It is even more dangerous in storing assets for web applications; for instance, *ProtonMail* is known for their ability to decrypt email in the browser and never have the decryption keys sent to any server, and people can freely inspect JavaScript files from *ProtonMail* to make sure this is how it works; however, absent data integrity checks, a target would never notice if *ProtonMail* or a party having access to their servers decided to serve a modified JavaScript file to a user, which would in fact capture the password and the decryption keys and exfiltrate them onto an attacker's server.
>
> ***Man-In-The-Middle attacks.*** Because of the lack of data integrity checks, using cloud storage is also vulnerable to the *man-in-the-middle attacks*, where a party located in between a server and a client has the ability to intercept and modify the data being exchanged. On the way to the server, there are routers and sometimes open Wi-Fi networks, internet provider companies, Internet Exchange Points (IXPs), and government agencies in different countries, all of which can be a point of attack. While the use of HTTPS/SSL makes it more difficult to modify the server-client communications, and a majority of modern websites is using HTTPS, there are known situations where due to the centralized nature of SSL authorities, the signing authorities were compromised and valid malicious certificates were generated, which even led to an (albeit delayed) emergency exclusion of those SSL authorities from browsers, but the damage might have already been done at that point [Taylor, 2019] [Alwazzeh et al., 2020].

Potential for censorship. There are plenty of widely known cases when cloud storage providers removed content without the consent of the uploader, citing various reasons, or even whole accounts and the content [Augier, 2016] [Martiny, 2018].

Clouds disappear. Cloud storage is not always perfectly reliable, and there are known incidents in which not only the information was inaccessible for a prolonged period of downtime, but sometimes even irrecoverably lost [Gagnaire, 2012]. For example, on March 10, 2021, a major fire broke out at an OVHcloud datacenter in Strasbourg, France, damaging the equipment and the building, with some websites and clients having their content irretrievably destroyed[4].

Breach of privacy. Cloud providers have direct access to the hardware, therefore, without extra mechanisms, they have access to the full contents of people's private data. [Shakor, 2019] There have been cases where journalists' private Google Drive documents, which they did not share with anyone, were mistakenly censored citing presence of prohibited materials in violation of the Terms of Service, and the users were locked out of their Google accounts[5].

Blockchain as a storage. Next, a blockchain itself was considered as the storage for web 3.0. Some decentralized social media, such as Steem (now HIVE), attempted to use it. However, it presented additional problems.

Problematic scalability. In a blockchain-powered network, every full node has to store everything that is on the blockchain, otherwise participants endanger themselves to a risk of nobody storing a particular transaction/account data, which means it would be lost forever. Which means that if a web3.0 project uses blockchain as the storage layer, and somebody uploads a random image, every full node in the world has to now store it. At least for the most known cryptocurrencies in the world, there is a decrease in the number of full nodes, due to two factors: 1) the requirements for hosting a full node grow superlinearly, including the growing storage requirements, which increases the costs[6], and 2) while the costs are rising, the incentive provided for full node operators to continue running the nodes stays at 0.

Prohibitively high price of storage for the user. Because of the aforementioned issue (all full node operators have to store all blockchain data), blockchains employ economic algorithms, counteracting abusing the blockchain as a storage layer indiscriminately.

---

[4] https://us.ovhcloud.com/press/press-releases/2021/fire-our-strasbourg-site

[5] "Google is locking people out of documents, and you should be worried": https://mashable.com/article/google-docs-locking-people-out

[6] According to https://etherscan.io/chartsync/chaindefault, the storage required for a synchronized Ethereum full node has exceeded 500 GB

There are costs associated with storing data on the blockchain. These costs, compared to the projected storage needs of the new generation of the internet, make it practically impossible to consider this as an option. For example, in 2017 it was estimated[7] that to store 1 GB of data on Ethereum it would cost around $76,000, and it should follow that with a price increase and a demand increase this cost would continue to grow.

Outsourcing to other blockchains does not solve the problem. In 2019, Vitalik Buterin, creator of Ethereum, proposed[8] using another blockchain, Bitcoin Cash, as a data storage layer for Ethereum 2.0, since the storage price on Bitcoin Cash chain has been much lower than on Ethereum. The limitation of this approach is that with the shift from using high-demand Ethereum chain as a data layer to using Bitcoin Cash, the demand for Bitcoin Cash transactions would start to increase, which, in turn, would increase the data storage cost on Bitcoin Cash, creating the same problem.

IPFS. In May 2014, Protocol Labs introduced IPFS, which is a protocol and peer-to-peer network for storing and sharing data in a distributed file system. It is now commonly used by a large number of dApps as a storage layer. In a hybrid Ethereum+IPFS setup, the content is stored and served on the IPFS network, but only its hash, which is taking only 32 bytes, is stored on the blockchain. This property is known as ***"content-addressing"*** (see below), where a file is identified not by a malleable URL, but by the hash of its contents. The downside of using IPFS for Web 3.0 is in the lack of incentives for the network to store and serve the data. For example, if a user wants to publish photos of their pets on decentralized Facebook, they would have to stay online in order to serve this content, as, presumably, not a lot of participants would be interested in storing it. As soon as the original user goes offline (and they are the only party that has been serving the content), the content goes offline as well, and becomes unreachable. It is possible for other users of the network to "pin" the content they want to help store when the original user is offline, but, as mentioned, IPFS lacks incentives to ensure that Web 3.0 content is always online and accessible, which is important for the decentralized internet to be functioning reliably.

Filecoin. Because of lack of incentives in IPFS, in 2017 Protocol Labs announced that they began working on another project called Filecoin, which would allow users to pay to decentralized storage providers to host and serve their content while they're offline. This architecture is closer than all previous approaches to the ideal requirements for a Web 3.0 storage layer, however, it has several downsides too. One of them is that the zero-knowledge proofs required to establish that storage providers ("storage miners") continue to store the files, require expensive hardware, not affordable for a majority of users that would like to become

[7] https://ethereum.stackexchange.com/a/896
[8] https://ethresear.ch/t/bitcoin-cash-a-short-term-data-availability-layer-for-ethereum/5735

storage providers, as a result decreasing the level of censorship-resistance. Another issue is that users have to strike deals with specific storage providers, specifying a time period for which their files should be stored. After the period expires, it is up to the users to strike more deals with other storage providers to renew the storage. If the users do not do it, the content becomes lost, as it is not in the storage providers' interests to continue to use the space for the expired content as opposed to new deals. This might lead to a situation where a user that has been paying for the files a lot of people were relying upon, suddenly stops paying, and the content disappears.

Arweave. Despite the fact that there are a number of other projects implementing decentralized storage, reviewing them falls out of the scope of this paper. The gradual review of potential candidates, from cloud storage, to blockchains, to IPFS, to Filecoin, serves to help understand the specific requirements that are needed for the Web 3.0 storage layer. This makes it apparent why Arweave was eventually chosen for the Point Network storage layer.

Arweave is a peer-to-peer network for decentralized storage, defined in Arweave Litepaper [Williams, 2018]. It is described as "a low cost, high throughput, permanent storage", but there are several specific properties that "checked all boxes" and made it desirable to be adopted as a storage layer for Web 3.0.

- **Incentives for storage delegation are built-in.** Unlike IPFS, when you upload something to Arweave, you pay to the Arweave network and it takes care of storing the files. A web3 user doesn't have to be online or pay to centralized pinning services for others to be able to access that user's posts, pictures or other files. The Arweave network takes the responsibility for storing and serving the files continuously.
- **Decentralization of storage.** Unlike IPFS pinning services, which can go down, get censored, or disappear, users don't have to rely on a centralized entity to keep their files online. Just like other peer-to-peer networks, Arweave network is composed of decentralized storage nodes run by different participants. Which means that there is no one centralized entity to attack.
- **Redundancy built-in and homogeneous.** On Filecoin, users can always choose several storage providers to store and serve their files, but this is not required, and some might prefer not to do it, in order to save on fees, which would decrease redundancy to $n=1$ and could increase risks for this file to become unavailable should that one storage provider go offline or stop performing its duties. On Arweave, the redundancy factor is homogenous: when the system "notices" certain files are becoming rare, the system automatically increases the payment for them, so that other network participants download those rare files and they stop being rare.
- **Permaweb, stored "forever".** Instead of constantly having to pay for the file storage lease, like on Filecoin, on Arweave a user has to make a one-time payment to the

blockchain, which then gradually pays storage miners from that amount to store the files. The initial cost a user pays to upload data to the Arweave network covers the first 200 years of storage. If data storage declines are anything greater than 0.5% per year, this simply adds to the number of years that the data will be stored. The result is comparatively cheap data storage costs over time.

This shows why Arweave was eventually chosen as a storage layer for Web 3.0 in Point Network. Nevertheless, in order for Arweave to function as such, it needed to be extended with extra layers of functionality and mechanisms, which are described in the following sections.

Storing and Retrieving Files

Since Point Network is using the Arweave network for storage, the details on how the files are stored and the information on data availability guarantees are described in Arweave's litepaper [Williams, 2018]. However, there are extra additions from the Point Network side on top of the standard Arweave storage mechanism, necessary to make it a storage layer for Web 3.0.

Content addressing. Some decentralized file storage networks, such as IPFS, have built-in content addressability, which means that the way by which a file is referred to, its primary ID, is the hash of its contents, which is useful for ensuring data integrity. This is not the case with Arweave, at least from the external API side. When a file is stored, its hash (which is referred to as **`data_root`**), is part of the transaction. But in the event when one needs to search the network for a file, one normally uses not **`data_root`**, but **`txid`** – a transaction ID in which this file was submitted to the network. An Arweave node or an Arweave gateway then return the data from that transaction if they have it.

Point Network is using a content addressing schema, where each file's ID is a 256-bit `keccak256` variant of SHA-3 hashing function. We then go around the issue of searching by **`txid`** by adding a special tag to the transaction, with a key `__pn_chunk_X_Y` (where X and Y are integers representing the current namespacing version of our experimental tests), and a value being the hash of the file. Then, Point Network stores the file hash as a **`bytes32`** field in Solidity. When the file needs to be retrieved by its hash, first we find the transaction ID by the hash ID, querying GraphQL API of Arweave storage counterparty using the special tag, getting a list of candidate transaction IDs in which the file appeared, and then we ask for the file based on **`txid`**.

Trust-less environment. One of the common use cases of Arweave is downloading or viewing a file based on its transaction ID navigating to a URL on one of Arweave gateways in their browser, such as `arweave.net`. However, that places trust in the Arweave gateway. We

assume that no Arweave node nor Arweave gateway can potentially be trusted with the valid data being returned based on txid or hash, or that a transaction marked with a tag for the hash requested really contains a file hashing to this ID. Therefore, for the data integrity to be preserved, the storage layer of Point Network never blindly accepts the file returned. It verifies that the file hashes to the hash requested. If it does not, the storage layer goes down the list of other potential transactions. If none of them hash to the desired content, it treats the file as unavailable.

Chunking. We define a chunk size limit as 262144 bytes = 256 KiB. If a file exceeds the limit, we do not attempt to upload the whole file to Arweave, and instead split it into chunks based on the chunk size limit, and uploads them in parallel. Each chunk has its ID based on the same hashing schema as a file with the chunk's contents would. Then, Point Network creates and uploads a special `chunkinfo` chunk, the contents of which is serialized[9] metadata about the file (all chunk IDs assembled in a Merkle tree), including the full description of the Merkle tree. Then, that chunkinfo chunk's ID becomes the file ID, by which it is referred. To differentiate a chunkinfo chunk from a file or a chunk, we write several bytes of chunkinfo prologue, which is practically unlikely to happen at the beginning of a normal file.

On retrieval, the same happens in reverse: when we download the file by its ID, we discover, using the prologue, that what we downloaded instead is a chunkinfo chunk. Then we parse it, extract the chunk IDs, download the chunks, reassemble them into the complete file, and return it as the result.

Next Version: In the current implementation, the `chunkinfo` chunk contains a JSON dictionary with the following fields:

- `type`: a string indicating the type of the chunk. In this case, it is equal to `'file'`
- `chunks`: an array of chunk IDs
- `hash`: designates a used hash type (defaults to 'keccak256')
  - The name of the field might be confusing. We are considering renaming it to `hashType`
- `filesize`: an unsigned integer field representing the size of the file in bytes
- `merkle`: a field containing the Merkle tree for the chunks, squashed into a 1D array.

The `chunkinfo` chunk allows for easier discovery of the chunks based on file ID. Without it, a client would have to obtain information about chunk IDs from somewhere, plus obtain Merkle proofs that the chunks are parts of the requested file. Persisting this information on the same paid storage level increases immediate accessibility of this information. However, it might be beneficial to consider whether it is possible to skip the `chunkinfo` chunk in the

[9] We initially propose using JSON for all serializations described herein.

future and use the chunk's Merkle tree root hash itself as the file hash, and not the hash of the `chunkinfo` chunk.

Additionally, consider replacing the term `chunkinfo` with `fileinfo` for more clarity.

Payment. Storing files on Arweave blockchain requires payment in Arweave's native token AR. To work around that, an ecosystem of *Arweave Bundlers* has been proposed and created, which are network participants running Arweave nodes with expanded functionality. First, they guarantee that if the payment has arrived, the data will be included in the weave and seeded. Second, the payment can be done in a currency/token different from AR. In our case, payments could be done with Point Network's native token, POINT, which is then automatically exchanged by bundler operators for AR to pay for the Arweave storage fees.

Directories

Directories in Point Network are represented by files with a map of filenames and subdirectories as keys and hashes pointing to the files as values.

In the current implementation, directories are JSON files, having the following structure:

- *type*: the string 'dir'
- *files*: an array of elements, pointing to files and/or subdirectories:
    - if the element is a pointer to a file, it is an object with the following structure:
        - *type*: the string 'fileptr'
        - *name*: the file name inside the directory
        - *size*: the size of the file in bytes
        - *id*: file ID
    - If the element is a pointer to a subdirectory, it is an object with the following structure:
        - *type*: the string 'dirptr'
        - *name*: the subdirectory name inside the directory
        - *size*: the recursive size of all the files in the directory in bytes
        - *id:* file ID of the file representing the subdirectory

Point Storage implementation must implement method(s) allowing traversing the directory structure, by recursively downloading subdirectories and files.

Private Files

Until now, we were discussing files which are intended to be exposed publicly on the network. If the client needs to keep some of the files, it should apply a symmetric encryption algorithm on top of the data for the chunk, such as AES. To not burden the client with storing these symmetric keys, the entropy for this symmetric key could be deterministically derived from the identity private key (yet it should not be possible to retrieve that private key from the symmetric encryption key; this is achieved by using hash functions and HD wallets).

As a result, by merely having the identity private key it would be possible to regenerate the AES keys on demand to decrypt the files. AES keys can be different and random for each chunk/file, yet still deterministically regenerative, if we add to the entropy the hash of the chunk/the file, before we hash it to produce the final AES key. This will allow sharing the keys with other users when needed, without concern of exposing keys to other files.

An alternative scheme might be to use randomly generated AES keys, encrypt them with the identity public key, and add at the beginning of each chunk/file. That way, the client is still not burdened with storing the keys, and can always extract the AES keys by decrypting them using public-key cryptography, although it increases the storage needed for each chunk.

Settings Files

The client, using the algorithm described, turns data into storage pledges, and RSA key pairs (redkeys). However, it still begs the question of where this metadata should be stored.

We propose to store it on storage providers as well, encrypted with AES (see Private Files section). That way, clients only have to keep track of that file's ID (they can store it on the blockchain, encrypted with AES or RSA), and if they have to log into Point Network on a new device, it would download this data, decrypt it and continue from there. And since the data is replicated across several storage providers, plus the fact that the file is encrypted and storage providers cannot distinguish it from normal data, the concern that providers might censor this data is alleviated.

Similarly, this can be extended to all settings and profile data for Point Network users. These settings can be stored in a private directory on Point Storage, and in case they need to log into Point Network on a new device, the software would download the directory and apply the settings. Additionally, by implementing sequence ordering for settings files, updates to them could be shared between devices, synchronized in real-time.

Small Files

Next Version: If a user has to store many small files, it is unreasonable and expensive to create an Arweave transaction for each of them. In the future versions, we can consider augmenting the directory data structure in such a way, that some of the files might be pointers with byte offsets into the middle of a chunk, allowing to store multiple files in one chunk. It would also be an appropriate time and place to consider adding compression, if needed.

This needs to be discussed further, because if pointers into the middle of chunks are only defined inside the directory schema, we cannot leverage this for pointers to files in Solidity, as they are *bytes32* pointing to chunkinfo chunks or files, and not directories; perhaps this mid-chunk file pointer object could also be added to the chunkinfo chunks, otherwise the format for referencing files on the blockchain has to be changed and augmented with byte offsets.

Transit Files

When you have files that are not fully formed at the time they are requested (such as audio and video streams, for example), it might be reasonable for the source of the file to incrementally sign over the Merkle tree of all the current chunks and distribute these signatures with each next produced chunk, plus the signature should be containing the hash of the first chunk and the sequence nonce, proving that the source of the file stream intended to augment the stream starting with the first chunk, with the current chunk.

That way, clients can refer to this stream by its ID (consisting of the ID of the first chunk, and the identity of the uploader, to avoid two uploaders colliding with each other hash-wise) and specifying the nonce of the last chunk they have. The network will help resolve these requests into new chunks that are being uploaded and attached to this stream ID as continuation of the stream.

Identity Management

Identity is an important part of web3.0. By identity here we do not mean KYC identities, or personal identities corresponding to a person, but rather alphanumeric handles registered on the blockchain, and everything that can be attached to it. A person/organization can own several identities.

Before considering identities, we need to discuss key pairs, because identities are attached to public keys.

Key Pairs

It is a standard in blockchain applications to generate a key pair of a public and a private key, and identify a user by the public key. Commonly, cryptocurrency addresses are derivations of public keys. Resources (such as balances, smart contracts, etc.) are then attached to these public keys, and in most cases, every action that modifies these resources has to be signed by the corresponding private key to be considered valid by the network, proving the ownership.

One of the mechanisms to increase user privacy has been to use HD (Hierarchical Deterministic) wallets [Khovratovich, 2017] [Banupriya, 2021], allowing to derive many additional key pairs from any key pair in a deterministic way (meaning that if we discard all the information but have the original private key and the path to the subkey, we can always restore the subkey), but have no apparent public relationship between any of them. We propose the following multi-level hierarchy of HD keys:

- There is a master (“seed”) key pair derived from a mnemonic [BIP39]. This mnemonic is the only thing a user is supposed to store in a secure and safe place, and enter on every device they want to log into Point Network, since it provides access to all the subkeys.
    - The key pairs on the first sublevel are representing keys that attach to different identities. A user can switch between these identities in Point Browser (See the relevant sections below). From the public point of view, these key pairs are not correlated and allow them to act on web 3.0 independently, preserving privacy. These subkeys, in turn, help derive additional sub-sub-keys, which can be used for following purposes:
        - The main key pair which is attached to an identity
        - Keys for holding cryptocurrency, with different purposes: normal wallets, “credit card”-like accounts that are used to subscribe to services requiring periodic payments (see Subscriptions section below), vault (see Vaults section below), keys that deterministically provide entropy to derive keys to use on other blockchains such as Bitcoin keys, Ethereum keys etc.
        - Keys used to communicate on the DHT network, and to pay for storage/retrieval

Identities

An identity is an alphanumeric handle with ownership, set on the blockchain and attached to a public key, with consensus that this public key is authorized to act on behalf of this identity.

In Point Network, an identity is that person’s:

- **public payment handle**, which people can use to transfer funds to them/send them invoices, instead of traditional human-unreadable cryptocurrency addresses (read more below on how to preserve privacy while sending cryptocurrency to a public handle);
- **email address** (see Decentralized Email section);
- **website domain**, i.e. a person with a handle @mike automatically gets access to the domain space mike.point and all its subdomains (see Decentralized Websites section);
- **handle for Instant Messaging** (see Decentralized Instant Messaging);
- **handle in decentralized social networks**;
- and other purposes.

Practically, an identity is used to authenticate a user in all web3 dApps. On web3, users will not have to make up a password to register on each website, remember the passwords, and enter them on every login, because they will be logged in automatically with their identity.

Key-Value Store, Domains and Websites

Each identity has a publicly-viewable key-value store attached to it, with only the owner being allowed to write to it.

Identity Transfer

Identities can be transferred to someone else, sold and auctioned off. Point Network should have this functionality built-in, to not force it to be reinvented several times and scattered across many smart contracts.

For privacy reasons, the reasonable approach would be to discard the key attached to the identity after that identity is transferred (not remove it completely, but mark it as abandoned), and not recycle it further attaching another identity.

Applications should be aware that the owner of the identity might change at any time (the public key would change in such a case), and therefore have functionality that takes care of the identity transfer and creates a new user when this event is emitted.

Delegation of Control

It should be possible to delegate control over an identity to another key temporarily, or to another smart contract, or to a multi-signature smart contract (which is especially useful for enterprise use cases, for example, having a marketing department and legal department sign off on any transaction which is an update to the company's decentralized Twitter feed, or having an engineering team sign an update to the source code on decentralized Github before it gets

into the block). This also allows the management key to reorganize and to take away control from specific public keys e.g. in case an employee contract is terminated.

Subidentities

Owning an identity allows the owner to register subidentities, and to delegate access to them if needed. A subidentity adds a prefix to the original identity separated by a dot, e.g. an identity `store.samsung.point` is a subidentity of `samsung.point`.

In decentralized websites, the subidentity becomes a subdomain, and in decentralized email, it becomes an email address inside that identity.

Decentralized applications should treat subidentities the same way as standalone entities. The difference between standalone identities is that the identity one level higher would be an owner of all its subidentities and retains control over them, even if subidentities are delegated. This might especially be useful for enterprise applications.

> **Next Version:** The spirit we desire to bring into the web3 is not one of hostility and fierce competition (also colloquially referred to as "maximalism" towards any favorite project), but collaboration and interoperability between different projects. Therefore, a consideration should be given on how to make Point Network identities interoperable with other decentralized naming system projects. For example, native Point Network identities could be given a suffix **.point**, while it would equally support projects such as Ethereum Name Service (**.eth**), Unstoppable Domains (**.crypto**), Namecoin (**.bit**), and others; to enable such use cases, Point Network would query the respective blockchains to establish the ownership. This should be weighed against the possibility of causing confusion.

Multi-Signature Actions on Identities

For additional security, identities (and all resources associated with it, such as balances or decentralized websites) could be owned by an *M-of-N multi-signature (multisig)*. For instance, in a 3-of-5 multisig, it requires any 3 or more persons from a designated list of 5 controlling identities to make an action associated with the controlled identity valid on the network.

Traditionally, multi-signatures have been used to secure funds in a wallet, allowing to only spend funds when the transaction is signed by several parties.

However, in Point Network, the effects of multisig mechanisms can be expanded to any action on web3. As an example, an identity handle, and a corresponding domain name could be owned by a 3-of-5 multisig. In that case, any proposed update to a website will not be accepted

immediately, but the parties involved in the multisig would receive notifications in Point Browser, asking them to accept, revise or reject the proposed update to the website code.

In another scenario, a multisig might be put on a repository in Decentralized Github, such as for a code commit to be merged to the master/main branch, it has to be approved by several people first, and not just anyone having write access to the branch (a feature current centralized Github lacks).

This presents a significant security improvement to the legacy internet mechanisms, where a successful attack on one party which has access to a resource is often enough to compromise the resource and inject malicious code into a website or a server. This improvement might especially be useful when adding audit companies into the mix as one of the multisig parties on a certain dApp.

Additionally, multi-signatures are not limited to simple M-of-N setups, and can be customized and extended with additional functionality, such as veto powers, emergency modes, and so on.

Social Recovery

Not requiring a user to create passwords for authentication with websites and using mnemonics and keys instead already alleviates many security concerns [Buterin, 2021], such as remembering to create strong passwords and remembering to have a different password for each website. Passwordless logins also protect from the databases of passwords becoming important targets for attackers. However, similarly to other cryptocurrencies, it shifts the burden of security to each individual user. When a user's private key/mnemonic gets compromised, there might still be some mechanisms to protect them from having their key stolen.

A user might want to specify a number of trusted individuals they know (e.g. their friends) as their recovery agents. Then, as soon as that user notices notifications about action they did not take, put their identity into Emergency Mode, which would allow agents to sign a multi-signature transaction to help recover identity ownership and the associated resources attached back to the user, to their new securely generated set of keys. Note that with this set up the user does not give these individuals power over the account while it's not been put into Emergency Mode. Moreover, the transaction designating these individuals as recovery agents can be stored in an encrypted form on the blockchain (in order to timestamp it but hide the contents), and only revealed when the account is put into Emergency Mode, so that nobody other than the owner could know which individuals are designated as recovery agents for any identity or whether this functionality is enabled at all.

Having this functionality implies adding delay periods for important actions, such as transferring an identity, or moving large amounts of money out of a wallet (see Vault section); otherwise, the user might notice the breach too late, after there would be nothing left to recover.

It would be practical to think of other recovery mechanisms to support, such as if the person simply loses their key. They can include social recovery, escrow recovery, backup keys etc. All the recovery mechanisms must not be compulsory and must be made optional.

Initial Identity Distribution

One of the problems naming systems encounter is the abuse by cybersquatters, registering a large amount of domains/identities in hopes to sell them for large sums of money to high profile people and companies. This could negatively impact the ecosystem, because when these people and companies would check out the project and find out that their identity is taken by cybersquatters, this might be an important factor discouraging them from using it.

We propose, to perform initial identity distribution, to have a temporary centralized oracle[10] in place, attached for the first 9 months after launch, monitoring Twitter feed of a certain activation hashtag, which would allow Twitter handles to be migrated to Point Network identities. This would not preclude people from registering identities that are not taken by Twitter handles, if they don't wish to register using their Twitter account or if they don't have one. After this period, the oracle would be detached, and anyone would be able to claim the identities that were not claimed, sold off in an auction.

[10] Alternatively, the temporary centralized oracle can be replaced by a temporary decentralized oracle *network*, using off-chain workers.

Point Browser

Introduction

Point Browser is a web 3.0 browser, configured to work with Point Node. It is a fork of Mozilla Firefox with extra functionality and configuration changes.

Point Browser is not merely a browser with the ability to access Ethereum or IPFS networks, as, for instance, Brave Browser does. Just as Tor Browser is used to access Deep Web (.onion domains), Point Browser is used to access Web 3.0 (.point domains). Therefore, it is not in the same category as and is not competing with Google Chrome, Mozilla Firefox, or Brave Browser, but rather using Firefox as a stable browser software to provide users of Point Network with a familiar browsing experience.

PointProxy Request Capture

The most important configuration option for Firefox in Point Browser setup is that it is set to proxy all requests through a port on which PointProxy is listening and capturing all requests, processing them using other Point Node modules, and returning responses back to the browser.

Therefore, even though `.point` domains do not exist on the legacy internet, PointProxy, by capturing all DNS and HTTP(S) requests, is able to imitate the existence of these domains in the legacy paradigm by returning valid DNS and HTTP responses to the browser.

Isolation from Web 2.0

Web 3.0, as defined by Point Network implementation, is a network completely isolated from the legacy internet, Web 2.0, just like blockchains are isolated from the external world, so much so that *oracles* are needed to send information from the external world to a blockchain. As such, in Point Browser, legacy internet websites such as facebook.com or google.com cannot be opened. If tried by a user, Point Browser will redirect the user to their normal default browser instead.

Similarly, decentralized applications running in Point Browser are unable to reach any IP addresses on the legacy internet. They can only access decentralized storage and decentralized domains.

This decision makes Point Network a completely separate network, which is designed to be growing separately from the legacy internet, and due to its clear boundaries can be called Web 3.0.

If it were possible to access Web 2.0 content from Point Browser, it is presumed that a large number of developers, working on decentralized applications, pressed by deadlines or motivated by other factors, could go back to their old ways, storing website assets (including JavaScript code) on centralized storage, using centralized domains, and pinging analytics servers by IP addresses. By severing the connection to Web 2.0, it forces all developers to develop decentralized applications in the way they're completely decentralized not just on paper, but in implementation as well.

Point Extension

Point Extension is a browser extension that is prepackaged with Point Browser which provides a helpful user interface and helps dApps communicate with Point Node.

The main extension window, which is opened when clicked on the extension icon, contains a compact summary of the wallet balances, and allows quick access to additional actions, such as accessing Wallet, Explorer, Contacts, different dApps and so on.

Point SDK

To help developers efficiently use Point Network capabilities, Point Extension makes available PointSDK, which becomes accessible on any page as `window.point`.

Point SDK contains functions which help perform various API calls to Point Node, related, for instance, to reading from and writing into decentralized storage, managing subscriptions, etc.

Legacy "Web3.js" SDK

A large number of developers in the blockchain space are familiar with a library called *web3.js* produced by the Ethereum community, and associated APIs. One popular blockchain access software called "MetaMask" popularized this approach, and it was solidified in EIP-1193.

Point Extension is responsible for mimicking this API in order to help with dApp migrations using as few modifications as possible. It makes available a "web3 provider" object at `window.ethereum` which can then be used with *web3.js* and similar libraries to interact with Point Chain.

Other chains

dApps on Point Network are not limited to only using Point Chain. Although the UI and the domains must reside on Point Chain in any case, Point Network is designed to allow users to

switch between several chains. For instance, users can choose to interact with Uniswap (a popular dApp) on Point Chain, then switch to Uniswap on Ethereum Chain, and then interact with Uniswap deployed on Polygon Chain.

Because the Point Network project is not just a blockchain, but a whole software suite, it can connect to several chains simultaneously, depending on *chainId*.

Similarly, it is not limited to Ethereum-compatible chains. Point Network can be extended to work with different APIs, such as APIs to access Solana Chain, Polkadot Chain, or Bitcoin Chain, in order to allow users to interact with Solana dApps, or pay invoices in Bitcoin, directly from Point Browser, by similarly exposing API endpoints customized to a specific chain (e.g. mimicking the behavior of Phantom Extension for Solana or polkadot{.js} extension for Polkadot).

Confirmation window

Because every request, including blockchain interaction requests from dApps, is captured by PointProxy, it allows Point Network to perform access control before deciding whether to fulfill a request.

By default, and with some exceptions, *read* blockchain operations are allowed. dApps are allowed to query blocks and transactions from every available chain.

However, *write* operations, such as sending assets, calling smart contract methods requiring a transaction, etc., have to undergo access control. For that purpose, before fulfilling a request, Point Extension opens a ***confirmation window***, displaying the request details to the user and asking whether they intend to perform this action.

Web3 Login

One of the exceptions where a confirmation window is needed for *read* operations, is “Log In As”. As mentioned in the “Identity Management” section, a user can have several identities. In that case, without additional mechanisms, it would have been confusing for Point Network to determine which identity to expose to the dApp so that it can load related information (such as the user name and the user’s notifications in a decentralized social media dApp).

For that purpose, a confirmation window for “Log In As” is triggered by request from a dApp, allowing the user to choose which identity to load.

Notification manager

Point Extension is also responsible for generating user notifications, when they are needed. At the start of Point Node, and during its operation, it is scanning the blockchain for events related to the user. It might be new deposits to or withdrawals from the user's accounts, an arrival of a new email message on Point Mail, or a custom dApp-generated event.

In that case, Point Extension uses browser API to show a notification, and open the associated page when the notification is clicked.

Internal Apps

Point Network is prepackaged with internal applications, making it easy for the user to access core Point Node functionality. Most of the apps are accessible using the URL `https://point/`.

Point Home

This is the first page that is opened when the user connects to Point Network. It can contain a welcome message, basic information about the user's account, and various links suggesting where to start exploring Web3.

Point Wallet

An important part of the Point Network's user interface is the Wallet application.

It allows common operations done from cryptocurrency wallets, such as sending cryptoassets (cryptocurrencies, tokens, NFTs etc.), receiving them, browsing transaction history etc. We will describe several modifications proposed to enhance the functionality and usability.

Multi-Chain Support

Because Point Node has the ability to connect to several chains simultaneously, Point Wallet should not only provide access to cryptoassets on Point Chain, but also on other chains, such as Ethereum, Polygon, Solana, and others. This way, a user would potentially have the ability to eventually interact with any cryptocurrency blockchain without leaving Point Browser, provided that a Point Wallet plugin for that chain exists.

Send to Identity Handles

A user should be able to not just use the legacy way of specifying a recipient (by submitting a long string of hexadecimal or base-58 characters which is someone's cryptocurrency address), but by directly specifying the identity handle. Extra care should be taken to make sure the user hasn't mistyped the handle (e.g. by providing a drop-down list with extra information about the identity as they type, by providing quick access to identities from the contact list, and by including a button to paste the name from another place).

In transaction history, Point Wallet should attempt to resolve cryptocurrency addresses to their identity handles instead, where possible.

In the future, more blockchain naming systems should be supported (such as Ethereum Name System using .eth handles, Unstoppable Domains using .crypto handle, Namecoin with .bit addresses etc.)

Send to Identity Handles on Different Chains

It is possible to create a mechanism by which a sender can send assets to an identity handle not only on Point Chain, but on other chains as well, such as Ethereum, Bitcoin or Solana.

One way to achieve this is through users publishing their addresses on these chains in the identity key-value store (*ikv*). This way, Point Wallet would infer the type of asset and the chain from the user's "From" field, and attempt to detect the existence of a cryptocurrency address on that chain for the identity selected in the "To" field, and use this address to fulfill the transaction request.

The proposed above requires a user's interaction to generate addresses for these chains beforehand. One improvement might be to pre-generate these addresses (with private keys derived using HD Wallets derivations, and as such, always accessible to the recipient), and automatically publish them.

Despite the improvements, both methods would still lack privacy (the same address per chain would be used by all senders). In the "Privacy" section below we discuss potential approaches to create uncorrelated addresses for increasing transaction privacy. Readers should keep in mind that most of those approaches could potentially be applicable not only to Point Chain transactions, but also to addresses on different chains (i.e. the ideal scenario is for a sender to be able to create unlimited addresses for the recipient on any existing blockchains, retaining a mathematical guarantee that the recipient is able to detect and access the funds in these addresses, with or without a help of an additional encrypted message to point to the right

address, but at the same time, for any external observer, leaving these addresses uncorrelated both with each other and with the identity).

Vaults (Two-Factor Authentication in Web3)

One of the worst case scenarios that can happen to a cryptocurrency user is their private key/seed phrase being compromised and in the hands of a malicious party, which would then lead to a theft of the funds and any other cryptoassets.

In the "Identity Management" section, "Social Recovery" was discussed as one approach to strengthen security. "Vaults" is another way, which does not involve relying on actions by third parties.

The idea behind vaults is, first, to be able to create a delay on a certain wallet (presumably, containing enough valuable cryptoassets that would warrant the extra complications in lieu of extra security) before finalizing any action on the blockchain. Thus, a successful transaction is split into a broadcast intention, a time delay which starts from the broadcast intention, and a completion, if the intention wasn't contested.

In a hypothetical scenario, a user *U* creates a vault, configured with a 24 hour delay, and deposits 100,000 POINTs into its address. Whenever *U* needs to spend some funds from the vault, it broadcasts an *intention transaction* first, then waits for 24 hours, and the transaction is then automatically confirmed. If the vault's private key becomes compromised and an attacker *A* attempts a theft of funds, *A* cannot simply complete a withdrawal transaction. *A* would first have to create an intention transaction, and *U* would receive a warning notification about this transaction in Point Wallet. *U* can then recognize that the transaction is fraudulent, and has 24 hours to broadcast a *cancellation transaction*, negating the intention transaction and resetting the timer. This, without extra mechanisms, would still lead to loss of funds, but it would lead to a tug-of-war between *U* and *A*, in which case, both of them would not receive anything from this, and this would make targeting vaults less attractive to attackers.

However, this can be improved further to allow full recovery of the vault. A simple variation would be for *U*, when creating the vault, to generate an offline key pair, never exposing the seed phrase online, and then grant that generated wallet address emergency powers on the vault. In the event of the dispute between *U* and *A*, *U* would prove to the network that *U* is the original owner by using the seed phrase (on a hardware that is known not to be compromised) to sign a recovery transaction and recover the vault's contents into a wallet address known to be safe.

Note that on its own, the seed phrase stored offline, cannot be used to access the vault. The vault's private key is always required to perform any action on it. Any person accidentally discovering the offline seed phrase might not even be aware that it is connected to the vault in

any way (in this case, the vault smart contract would not use the address openly in plain-text, but hashed with some "salt" nonce). But in the event of a dispute, the offline seed phrase would serve to complete the two-factor authentication and, in conjunction with the vault's private key, be used to recover the vault.

Privacy

Privacy has been a pain point for many blockchain systems. For instance, in Ethereum it is common for a person to have one wallet address, and then conduct all business from that address. Then, it is trivial for anyone to browse all transactions and interactions with dApps for that person. The privacy level here is similar to a person walking into a supermarket with a paper attached to their back, listing all previous stores they walked into, the amounts they spent there and when, and, in some cases, what they bought.

It takes colossal effort in rigorous research, development and testing to develop privacy solutions correctly. Therefore, the first priority for Point Network is to create a base for the decentralized internet, focusing more on the censorship-resistance part in the beginning. However, the privacy preservation side of decentralized internet is of comparable importance, therefore, sufficient research and development efforts should also be allocated to it.

Privacy preservation is a spectrum, and it can be upgraded gradually. These are several mechanisms that could be used, separately or altogether, with varying levels of strength:

- **Switching Identities in the Browser.** Point Node should support switching between several identities, by using the HD Wallet approach described in the "Identity Management" section.
- **Switching Identities in the Dashboard.** Point Dashboard should support the ability to open a completely different environment when a user logs in with a different seed phrase (e.g. no cache leaking between identities).
- **Feejoin.** One of the common privacy solutions is known as *coinjoin*, a multi-party transaction where the parties mix their coins together and take back their amounts, but an external observer has no way of knowing which party owns which part of the output. While coinjoin is something Point Network could use to increase privacy, it is possible to also adapt this idea for web3 interaction privacy, what we call *feejoin*. Right now, if anyone wants to interact with any dApp smart contract (e.g. posting a tweet on decentralized social media) from their identity, they must have a sufficient balance of tokens in that address. This exposes the amount of funds they have, something which they might desire to not advertise. Even if a user chooses to top-up the identity account every once in a while from another one, links to that main account can be trivially established. *Feejoin* mechanism would allow *anyone* to pay for *any* transaction, and then,

when several people desire to interact with dApps, would automatically merge their fees, such as one person would pay for the transaction of another, and vice versa. This would make it difficult for external observers to establish which funded wallet belongs to which identity (and the identity's wallet can have 0 tokens), because it would be unclear whether the wallet that has paid for the web3 interaction belongs to the owner of the identity, or some other *feejoin* party.

- **Payment requests.** Traditionally, when a blockchain user desires to receive a payment from another party, they send their main wallet address. Point Network could provide a mechanism whereby each time a user clicks 'Receive', a different subaddress is generated. There is no need to remember the seed phrase for that subaddress, because it uses the HD Wallet mechanism from the "Identity Management" section. However, for external observers, the addresses would be completely uncorrelated. The difficult parts of implementing this would then be to efficiently query and consolidate those balances in the user interface, and merge them when a payment is needed. This would help shift Point Network from account-based (traditional in Ethereum-like systems) to UTXO-based privacy level (traditional in Bitcoin-like systems), while still preserving beneficial account-based capabilities for web3.
- **Offline payment requests.** Payment requests described above require a user to be online when the subaddress is generated, but this might not always be the case. For this, a mechanism might exist by which a sender could generate a unique shared wallet address, deposit the transaction amount there, and when the recipient comes online, Point Network would detect the payment and add as part of the balance, while for an external observer it would be uncorrelated with any other addresses or payments to the recipient. A naïve approach would be to simply create a new cryptocurrency address, send funds there, and share the private key to it in an encrypted message to that identity (which would designate a successful completion of the transfer by the recipient capturing the payment into another recipient-controlled wallet, and would also allow the sender to take back the funds in the unsuccessful case or if the recipient never comes back online). There might be more ways to do this more efficiently, such as blending keys of the parties in such a way that a sender could create unlimited unique addresses with guaranteed ability of the recipient (and only the recipient, as opposed to the previous scenario) to generate the private key required to capture the funds, while it would take external parties an impractical effort if they try to attempt to establish the ownership by a bruteforce attack. More research is needed to establish the possibility of offline payment requests and a path to their implementation.
- **Existing privacy projects.** Finally, lessons from privacy-preserving projects such as Monero could be taken, to eventually provide Monero-grade privacy to the network.

Point Explorer

Traditionally, software that allows users to browse the contents of a blockchain in a human-readable format has been called *block explorers*. Normally, it is run on a centralized server, and users are left with the only option to trust the data provider.

By making Point Explorer an internal application as part of the auditable software suite, running on a user's computer instead of on a centralized server, it would be possible to increase the level of security and trust to the data output, and move block explorers further into decentralization.

Point Node could still use other nodes on the network to query the data, but it would run ad-hoc checks (Merkle proof validation, state transition result validation etc.) in order to verify the validity of conclusions before displaying them in the browser.

Additionally, there are some extra views that could be specific to Point Explorer and which should be supported, such as displaying and verifying NFTs, and exploring the list of identities and all properties, transactions and assets associated with them (for example, a detailed history of dApp deployments on a certain decentralized domain).

Potential Applications

This section describes potential applications that could become possible once Point Network is fully implemented. It does not mean that the authors seek to implement all of the described applications, but it provides general guidelines and suggestions for teams that would be attempting to work on them.

Censorship-resistant websites

One type of content that can exist on web3 is simply static websites. One of the differences between them and websites on web 2.0 is that websites on Point Network do not reside in a central location, and instead live on decentralized internet, which makes them resistant to censorship attempts.

Since every part of the tech stack is decentralized (domains, storage, identities), there is no central point of failure to attack.

Another beneficial property of websites residing on web3 is that every update to the website is a transaction on the blockchain. This allows all updates to be publicly auditable, and also allows to create *multisig setups*, whereby every update has to be approved by N out of M parties,

which improves security by shifting the requirements needed to make malicious modifications to a website from compromising any one party (any developer working on the website), which is the situation right now on web2.0, to compromising several members of the team (see “Multi-Signature Actions on Identities” subsection under “Identity Management”).

Interacting with dApps on several chains

- - - -

Note to the reader: this section is not finalized yet, please follow our roadmap:

https://roadmap.pointnetwork.io

and the URL this document is hosted on for new versions:

https://pointnetwork.io/link/whitepaper

- - - -

Decentralized End-to-End Encrypted Email

> See https://penumbra.zone/crypto/primitives/fmd.html and the linked https://eprint.iacr.org/2021/089

To hide metadata/relationship graph: Hide by fuzzy recipient and hiding sender inside

Decentralized Social Media

Decentralized Facebook, Reddit and Discord

Decentralized Youtube

Decentralized Telegram

Decentralized Zoom

Audio video calls

Supporting Communities and Content Creators

Storage Archives

Decentralized Github

Digital Collectibles (e.g. art NFTs and game assets)

Signatures and Badges

Manageable Personal Data/KYC Identities

Digital Canaries

E-commerce

Zero-knowledge targeted advertising

DAO, Crowdfunding and Governance

Point Network for Enterprise

Conclusion

References

[Al Hasib and Haque, 2008] Al Hasib, A. and Haque, A. A. M. M. (2008). A comparative study of the performance and security issues of AES and RSA cryptography. *Proceedings - 3rd International Conference on Convergence and Hybrid Information Technology, ICCIT 2008*, 2(May):505–510.

[Alabdulwahhab, 2018] Alabdulwahhab, F. A. (2018). Web 3.0: The Decentralized Web Blockchain networks and Protocol Innovation. *1st International Conference on Computer Applications and Information Security, ICCAIS 2018*, pages 1–4.

[Anderson, 2019] Anderson, P. D. (2019). Edward Snowden: Permanent record. *Ethics and Information Technology*, 22(2):129–132.

[Baumgart and Mies, 2007] Baumgart, I. and Mies, S. (2007). S/Kademlia: A practicable approach towards secure key-based routing. *Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS*, 2.

[Bellare et al., 2000] Bellare, M., Kilian, J., and Rogaway, P. (2000). The security of the cipher block chaining message authentication code. *Journal of Computer and System Sciences*, 61(3):362–399.

[Benet, 2014] Benet, J. (2014). IPFS - Content addressed, versioned, P2P file system.

[Benet and Greco, 2017] Benet, J. and Greco, N. (2017). Filecoin: A decentralized storage network. *Protocol Labs*.

[Brevini, 2017] Brevini, B. (2017). WikiLeaks: Between disclosure and whistle-blowing in digital times. *Sociology Compass*, 11(3):1–11.

[Buterin, 2014] Buterin, V. (2014). Secret sharing and erasure coding: A guide for the aspiring Dropbox decentralizer.
https://blog.ethereum.org/2014/08/16/secret-sharing-erasure-coding-guide-aspiring-dropbox-decentralizer/. Accessed: 2021-07-10.

[Coddington, 2017] Coddington, A. (2017). *Mass Government Surveillance: Spying on Citizens*. Cavendish Square Publishing LLC.

[Dale Liu et al., 2009] Dale Liu, Caceres, M., Robichaux, T., Forte, D. V., Seagren, E. S., Ganger, D. L., Smith, B., Jayawickrama, W., Stokes, C., and Jan Kanclirz, J. (2009). Next generation SSH2 implementation: Securing data in motion. *Syngress Publishing*, pages 41–64.

[Daniel and Tschorsch, 2021] Daniel, E. and Tschorsch, F. (2021). IPFS and friends: A qualitative comparison of next generation peer-to-peer data networks. arXiv e-prints.

[DeNardis, 2007] DeNardis, L. (2007). A history of internet security. *Information Society Project*, pages 681–704.

[Dessouky et al., 2020] Dessouky, G., Frassetto, T., Jauernig, P., Sadeghi, A. R., and Stapf, E. (2020). With great complexity comes great vulnerability: From standalone fixes to reconfigurable security. *IEEE Security and Privacy*, 18(5):57–66.

[Feigenbaum and Koenig, 2014] Feigenbaum, J. and Koenig, J. (2014). On the feasibility of a technological response to the surveillance morass. *Cambridge Security Protocols Workshop*.

[Gunitsky, 2015] Gunitsky, S. (2015). Corrupting the cyber-commons: Social media as a tool of autocratic stability. *Perspectives on Politics*, 13(1):42–54.

[Hallam and Zanella, 2017] Hallam, C. and Zanella, G. (2017). Online self-disclosure: The privacy paradox explained as a temporally discounted balance between concerns and rewards. *Computers in Human Behavior*, 68:217–227.

[Harris, 2020] Harris, T. (2020). Unregulated tech mediation, inevitable online deception, societal harm. *Center for Humane Technology*.

[Hubbard, 2020] Hubbard, S. (2020). Opinion: Forget bias, the real danger is big tech's overwhelming control over speech. https://edition.cnn.com/2020/10/28/perspectives/section-230-hearing-big-tech/index.html. Accessed: 2021-07-09.

[Kierstead, 2021] Kierstead, A. (2021). The parler shutdown. https://law.unh.edu/blog/2021/01/parler-shutdown. Accessed: 2021-07-09.

[Kopel, 2013] Kopel, K. (2013). Operation seizing our sites: How the federal government is taking domain names without prior notice. *University of California*, 28:859–900.

[Lear, 2018] Lear, S. (2018). The fight over encryption: Reasons why congress must block the government from compelling technology companies to create backdoors into their devices. *Cleveland State Law Review*, 66(2).

[Molloy, 2021] Molloy, D. (2021). Opinion: Forget bias, the real danger is big tech's overwhelming control over speech. https://www.bbc.com/news/world-middle-east-57570044. Accessed: 2021-07-09.

[Moshirnia, 2018] Moshirnia, A. (2018). No security through obscurity: Changing circumvention law to protect our democracy against cyberattacks. *Brooklyn Law Review*, 83(3):1279–1344.

[Nadler and Collins, 2017] Nadler, J. and Collins, D. (2017). RE: Competition in the Digital Marketplace. *Americans for Financial Reform Education Fund*.

[Otala et al., 2021] Otala, J. M., Kurtic, G., Grasso, I., Liu, Y., Matthews, J., and Madraki, G. (2021). Political polarization and platform migration: A study of parler and twitter usage by United States of America congress members. *Companion Proceedings of the Web Conference 2021*, pages 224– 231.

[Politou et al., 2020] Politou, E., Alepis, E., Patsakis, C., Casino, F., and Alazab, M. (2020). Delegated content erasure in IPFS. *Future Generation Computer Systems*, 112(September):956–964.

[Produit, 2018] Produit, B. (2018). Using blockchain technology in distributed storage systems. *University of Tartu*, pages 1–14.

[Rivest et al., 1978] Rivest, R., Shamir, A., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Massachusetts Institute of Technology*.

[Rogers, 2020] Rogers, R. (2020). Deplatforming: Following extreme Internet celebrities to Telegram and alternative social media. *European Journal of Communication*, 35(3):213–229.

[Shane et al., 2017] Shane, S., Rosenberg, M., and Lehren, A. W. (2017). Wikileaks releases trove of alleged c.i.a. hacking documents. https://www.nytimes.com/2017/03/07/world/europe/wikileaks-cia-hacking.html. Accessed: 2021-07-09.

[Slobogin, 2007] Slobogin, C. (2007). Privacy at risk: The new government surveillance and the Fourth Amendment. *University of Chicago Press*.

[Slupska et al., 2021] Slupska, J., Lowrie, J., Irani, L., and Stefan, D. (2021). How secrecy leads to bad public technology. *UC San Diego*. [Snowden, 2019] Snowden, E. (2019). Permanent record. *MacMillan*.

[Storj, 2018] Storj (2018). Storj: Decentralized cloud storage network framework. *Storj Labs*.

[Tiwari, 2019] Tiwari, A. (2019). Big tech monopoly: Effects, desirability and viable regulations.

[van der Schyff et al., 2020] van der Schyff, K., Flowerday, S., and Furnell, S. (2020). Duplicitous social media and data surveillance: An evaluation of privacy risk. *Computers and Security*, 94.

[Al Hasib and Haque, 2008] Al Hasib, A. and Haque, A. A. M. M. (2008). A comparative study of the performance and security issues of AES and RSA cryptography. *Proceedings - 3rd International Conference on Convergence and Hybrid Information Technology, ICCIT 2008*, 2(May):505–510.

[Alabdulwahhab, 2018] Alabdulwahhab, F. A. (2018). Web 3.0: The Decen- tralized Web Blockchain networks and Protocol Innovation. *1st International Conference on Computer Applications and Information Security, ICCAIS 2018*, pages 1–4.

[Alwazzeh et al., 2020] Alwazzeh, M., Karaman, S., and Shamma, M. N. (2020). Man in the middle attacks against ssl/tls: Mitigation and defeat. *Journal of Cyber Security and Mobility*, pages 449–468.

[Anderson, 2019] Anderson, P. D. (2019). Edward Snowden: Permanent record.
Ethics and Information Technology, 22(2):129–132.

[Augier, 2016] Augier, M. (2016). Trustworthy cloud storage. Technical report, EPFL.

[Banupriya et al., 2021] Banupriya, S., Kottursamy, K., and Bashir, A. K. (2021). Privacy-preserving hierarchical deterministic key generation based on a lattice of rings in public blockchain. *Peer-to-Peer Networking and Ap- plications*, 14(5):2813–2825.

[Baumgart and Mies, 2007] Baumgart, I. and Mies, S. (2007). S/Kademlia: A practicable approach towards secure key-based routing. *Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS*, 2.

[Bellare et al., 2000] Bellare, M., Kilian, J., and Rogaway, P. (2000). The se- curity of the cipher block chaining message authentication code. *Journal of Computer and System Sciences*, 61(3):362–399.

[Benet, 2014] Benet, J. (2014). IPFS - Content addressed, versioned, P2P file system.

[Benet and Greco, 2017] Benet, J. and Greco, N. (2017). Filecoin: A decentral- ized storage network. *Protocol Labs*.

[Brevini, 2017] Brevini, B. (2017). WikiLeaks: Between disclosure and whistle- blowing in digital times. *Sociology Compass*, 11(3):1–11.

[Buterin, 2014] Buterin, V. (2014). Secret sharing and era- sure coding: A guide for the aspiring dropbox decentralizer. https://blog.ethereum.org/2014/08/16/secret-sharing-erasure-coding-guide-aspiring-dropbox-decentralizer/. Accessed: 2021-07-10.

[Buterin, 2021] Buterin, V. (2021). Why we need wide adoption of social recov- ery wallets.

[Coddington, 2017] Coddington, A. (2017). *Mass Government Surveillance: Spying on Citizens*. Cavendish Square Publishing LLC.

[Dale Liu et al., 2009] Dale Liu, Caceres, M., Robichaux, T., Forte, D. V., Sea- gren, E. S., Ganger, D. L., Smith, B., Jayawickrama, W., Stokes, C., and Jan Kanclirz, J. (2009). Next generation SSH2 implementation: Securing data in motion. *Syngress Publishing*, pages 41–64.

[Daniel and Tschorsch, 2021] Daniel, E. and Tschorsch, F. (2021). IPFS and friends : A qualitative comparison of next generation peer-to-peer data net- works. *arXiv e-prints*.

[DeNardis, 2007] DeNardis, L. (2007). A history of internet security. *Informa- tion Society Project*, pages 681–704.

[Dessouky et al., 2020] Dessouky, G., Frassetto, T., Jauernig, P., Sadeghi,

1. R., and Stapf, E. (2020). With great complexity comes great vulnera- bility: From stand-alone fixes to reconfigurable security. *IEEE Security and Privacy*, 18(5):57–66.

[Feigenbaum and Koenig, 2014] Feigenbaum, J. and Koenig, J. (2014). On the feasibility of a technological response to the surveillance morass. *Cambridge Security Protocols Workshop*.

[Gagnaire et al., 2012] Gagnaire, M., Diaz, F., Coti, C., Cerin, C., Shiozaki, K., Xu, Y., Delort, P., Smets, J.-P., Le Lous, J., Lubiarz, S., et al. (2012). Downtime statistics of current cloud solutions. *International Working Group on Cloud Computing Resiliency, Tech. Rep*.

[Gunitsky, 2015] Gunitsky, S. (2015). Corrupting the cyber-commons: Social media as a tool of autocratic stability. *Perspectives on Politics*, 13(1):42–54.

[Hallam and Zanella, 2017] Hallam, C. and Zanella, G. (2017). Online self- disclosure: The privacy paradox explained as a temporally discounted balance between concerns and rewards. *Computers in Human Behavior*, 68:217–227.

[Harris, 2020] Harris, T. (2020). Unregulated tech mediation, inevitable online deception, societal harm. *Center for Humane Technology*.

[Hubbard, 2020] Hubbard, S. (2020). Opinion: Forget bias, the real danger is big tech's overwhelming control over speech. https://edition.cnn.com/2020/10/28/perspectives/section-230-hearing-big-tech/index.html. Accessed: 2021-07-09.

[Khovratovich and Law, 2017] Khovratovich, D. and Law, J. (2017). Bip32- ed25519: hierarchical deterministic keys over a non-linear keyspace. In *2017 IEEE European Symposium on Security and Privacy Workshops (Eu- roS&PW)*, pages 27–31. IEEE.

[Kierstead, 2021] Kierstead, A. (2021). The parler shutdown. https://law.unh.edu/blog/2021/01/parler-shutdown. Accessed: 2021-07-09.

[Kopel, 2013] Kopel, K. (2013). Operation seizing our sites: How the federal government is taking domain names without prior notice. *University of Cal- ifornia*, 28:859–900.

[Lear, 2018] Lear, S. (2018). The fight over encryption: Reasons why congress must block the government from compelling technology companies to create backdoors into their devices. *Cleveland State Law Review*, 66(2).

[Martiny et al., 2018] Martiny, I., Miers, I., and Wustrow, E. (2018). Proof of censorship: Enabling centralized censorship-resistant content providers. In *International Conference on Financial Cryptography and Data Security*, pages 99–115. Springer.

[Molloy, 2021] Molloy, D. (2021). Opinion: Forget bias, the real danger is big tech's overwhelming control over speech. https://www.bbc.com/news/world- middle-east-57570044. Accessed: 2021-07-09.

[Moshirnia, 2018] Moshirnia, A. (2018). No security through obscurity: Chang- ing circumvention law to protect our democracy against cyberattacks. *Brook- lyn Law Review*, 83(3):1279–1344.

[Nadler and Collins, 2017] Nadler, J. and Collins, D. (2017). RE: Competition in the Digital Marketplace. *Americans for Financial Reform Education Fund*.

[Otala et al., 2021] Otala, J. M., Kurtic, G., Grasso, I., Liu, Y., Matthews, J., and Madraki, G. (2021). Political polarization and platform migration: A study of parler and twitter usage by Nnited States of America congress members. *Companion Proceedings of the Web Conference 2021*, pages 224– 231.

[Palatinus et al., 2013] Palatinus, M., Rusnak, P., Voisine, A., and Bowe, S. (2013). Bips/bip-0039.mediawiki at master · bitcoin/bips.

[Politou et al., 2020] Politou, E., Alepis, E., Patsakis, C., Casino, F., and Alazab, M. (2020). Delegated content erasure in IPFS. *Future Generation Computer Systems*, 112(September):956–964.

[Produit, 2018] Produit, B. (2018). Using blockchain technology in distributed storage systems. *University of Tartu*, pages 1–14.

[Rashid et al., 2019] Rashid, M., Singh, H., and Goyal, V. (2019). Cloud storage privacy in health care systems based on ip and geo-location validation using k-mean clustering technique. *International Journal of E-Health and Medical Communications (IJEHMC)*, 10(4):54–65.

[Rivest et al., 1978] Rivest, R., Shamir, A., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Massachusetts Institute of Technology*.

[Rogers, 2020] Rogers, R. (2020). Deplatforming: Following extreme Internet celebrities to Telegram and alternative social media. *European Journal of Communication*, 35(3):213–229.

[Shakor et al., 2019] Shakor, M. Y., Khaleel, M. I., and Abed, F. S. (2019). Enhancing cloud storage privacy (csp) based on hybrid cryptographic tech- niques. *Journal of the University of Garmian*, 6:1.

[Shane et al., 2017] Shane, S., Rosenberg, M., and Lehren, A. W. (2017). Wikileaks releases trove of alleged c.i.a. hacking docu- ments. https://www.nytimes.com/2017/03/07/world/europe/wikileaks-cia-hacking.html. Accessed: 2021-07-09.

[Slobogin, 2007] Slobogin, C. (2007). Privacy at risk: The new government surveillance and the Fourth Amendment. *University of Chicago Press*.

[Slupska et al., 2021] Slupska, J., Lowrie, J., Irani, L., and Stefan, D. (2021).
How secrecy leads to bad public technology. *UC San Diego*.

[Snowden, 2019] Snowden, E. (2019). Permanent record. *MacMillan*.

[Storj, 2018] Storj (2018). Storj: Decentralized cloud storage network frame- work. *Storj Labs*.

[Taylor, 2019] Taylor, A. (2019). Decrypting ssl traffic: best practices for secu- rity, compliance and productivity. *Network Security*, 2019(8):17–19.

[Tiwari, 2019] Tiwari, A. (2019). Big tech monopoly: Effects, desirability and viable regulations.

[van der Schyff et al., 2020] van der Schyff, K., Flowerday, S., and Furnell, S. (2020). Duplicitous social media and data surveillance: An evaluation of privacy risk. *Computers and Security*, 94.

[Williams and Jones, 2018] Williams, S. and Jones, W. (2018). Arweave light- paper.