P2P Protocol Comparison

This comparative analysis of peer-to-peer (P2P) protocols is guided by OpenArchive's essential criteria for considering how to integrate a decentralized backend to their *Save* application. Criteria include usability, security, privacy, authentication, media authenticity, encryption and mobile compatibility. Deleting media and maintaining a persistent connection are considerable advantages. Usability is addressed through the sections on reliability and similar case studies, although these protocols generally emphasize technical affordances other than ease of use.

Protocol / Criteria	Arweave	BitTorrent	Filecoin	Freenet	Hyper/Pear	I2p	IPFS	Iroh	Peergos	SocketSupply	Source.Network	Veilid
Open Source	✅	✅	✅	✅	✅	✅	✅	✅	✅	✅	✅	✅
User Authentication	✅	❌	✅	✅	✅	✅	✅	✅	✅	✅	✅	✅
Media Verification Authenticity	✅	❌	✅	✅	✅	❌	✅	✅	✅	❌	✅	❌
Provenance / History	✅	❌	✅	❌	✅	❌	❌	❌	✅	❌	✅	❌
Encryption (at rest)	❌	❌	❌	✅	❌	❌	❌	❌	✅	❌	✅	✅
IP Address Privacy	❌(server)	❌	❌(server)	❌	❌	✅	❌	❌	❌(server)	❌	❌(server)	✅
Free to Use (Cost)	❌	✅	❌	✅	✅	✅	✅	✅	If self hosted	✅	❌	✅
Mobile Compatibility	❌	✅	❌	✅	✅	❌	✅	✅	✅	✅	✅	✅
Ability to Delete Media	❌	❌	❌	✅	❌	❌	❌	✅	✅	?	?	?
Persistent Connection	✅	✅	✅	✅	✅	✅	✅	✅	✅	?	✅	✅
Implementation Languages	Erlang, Javascript	C++	Rust, Go, C++	Rust	Javascript	C++, java	Go, Rust, Java, Javascript,	Rust, Python, Swift, Kotlin	Java	Python, Rust, Node.js	Go	Rust, Flutter
P2P	❌	✅	❌	✅	✅	✅	✅	✅	❌	✅	❌	✅

Veilid + Iroh - *Recommended*

Overview

Veilid is an excellent choice for peer-to-peer (P2P) networking and discovering connections. It borrows from both the Tor anonymising router and the InterPlanetary File System (IPFS), to offer encrypted and anonymous peer-to-peer connection using a 256-bit public key as the only visible ID. Even details such as IP addresses are hidden. Launched relatively recently (~January 2023), Veilid is actively being developed and improved, making it a fairly reliable choice for projects needing an open-source, peer-to-peer, mobile-first, networked application framework. However, it does not have good support for data storage and verification. Despite its strengths, users should be aware of its early stage of development which can lead to breaking changes.

Iroh, on the other hand, excels in handling large collections of data that change over time. While it does not provide IP privacy, it offers solid primitives for data storage, making it a valuable tool for managing dynamic datasets. Iroh is an actively maintained project with a focus on reliability and performance. Rust projects can import Iroh crates (i.e. libraries) directly, which shortens development time. Other languages can call Rust code via a C-API to keep development work streamlined. (Note: Rust is easily embeddable, and because the code needs to run across both iOS and Android for the *Save* app, Rust is especially relevant for the OpenArchive use-case.)

Combining Veilid and Iroh leverages the strengths of both protocols, effectively addressing each other's weaknesses. Integrating both protocols into a single process is feasible due to their Rust implementations, providing a seamless and efficient solution for decentralized applications. This proposed solution encapsulates the P2P code into its own process, which can be called from any other code and language using Inter Process Communication (IPC). This will simplify deployment to the app and facilitate integration with different languages (e.g. Swift/Kotlin).

Protocol / Criteria	Iroh	Veilid
Open Source	✅	✅

User Authentication	✅	✅
Media Verification Authenticity	✅	❌
Provenance / History	❌	❌
Encryption (at rest)	✅	✅
IP Address Privacy	❌	✅
Free to Use (Cost)	✅	✅
Mobile Compatibility	✅	✅
Ability to Delete Media	✅	✅
Persistent Connection	✅	✅
Implementation Languages	Rust, Python, Swift, Kotlin	Rust, Flutter
P2P	✅	✅

Additional Context

- **Reliability and age of projects:** Both Veilid and Iroh are relatively new and are actively being developed, ensuring ongoing improvements and support. However, this work may also introduce breaking changes. That being said, their active development communities contribute to their reliability and adaptability. Newer projects are also susceptible to pivots or significant changes.
- **Security considerations:** Veilid, being newer, may have more undiscovered security issues compared to Iroh, which has more established and scrutinized security measures (based on a review of the number of releases and patches to the code). In both cases, users should stay updated with the latest security patches and community discussions.
- **Similar projects and use cases**: Both these protocols are so new that there are limited implementations. VeilidChat, a chat application, is currently in active development. Iroh is being integrated in Jumpy by Fish Folk. Our research found no examples of projects similar to *Save* using these protocols.

I2P + BitTorrent - *Recommended*

Overview

I2P (Invisible Internet Project) is an alternative to Tor for keeping user IP addresses private. It provides a high level of anonymity and security through routing and encryption mechanisms. I2P's technology has been around for many years, making it a reliable choice for privacy-focused applications. The project is actively maintained, with regular updates and a strong community. Despite its strengths, I2P cannot find peers on the network based on a general "key," requiring the use of introduction points or trackers for peer discovery. It has an efficient C++ implementation, which we could run via FFI (Foreign Function Interface).

BitTorrent, by comparison, is renowned for its efficiency in data transfer. It has a proven track record for distributing large files quickly and reliably across the network. However, BitTorrent lacks support for IP privacy and transport-level encryption, which are critical for secure communications. Additionally, BitTorrent does not natively support mutable data, necessitating the integration of additional layers for handling data changes over time. Despite these limitations, BitTorrent remains a widely-used and well-supported protocol.

Combining I2P with BitTorrent leverages the strengths of both protocols, enhancing privacy and data transfer efficiency. Running I2P's C++ implementation via FFI ensures seamless integration with existing systems. This combination addresses the limitations of each protocol, creating a more secure and efficient solution for decentralized applications.

Protocol / Criteria	I2P	BitTorrent
Open Source	✅	✅
User Authentication	✅	❌
Media Verification Authenticity	❌	❌
Provenance / History	❌	❌
Encryption (at rest)	❌	❌
IP Address Privacy	❌	❌
Free to Use (Cost)	✅	✅
Mobile Compatibility	❌	❌

Ability to Delete Media	❌	❌
Persistent Connection	✅	✅
Implementation Languages	C++, java	C++, java, Rust
P2P	✅	✅

Additional Context

- **Reliability and age of projects**: I2P has been in development for over a decade, providing a mature and reliable platform for privacy-focused applications. BitTorrent, with its long history of successful deployments, offers a proven solution for efficient data transfer.
- **Security considerations**: I2P is designed with security and privacy as primary goals, offering robust protection against surveillance and tracking. BitTorrent, while efficient in data distribution, requires additional measures to ensure secure communications. Users should implement transport encryption and privacy-enhancing techniques when using BitTorrent.
- **Similar projects and use cases**: I2P and BitTorrent are utilized in various applications, including anonymous file sharing, secure communications, and distributed content distribution networks. As an example, I2P is used as a part of a recent free banking initiative.

By integrating I2P's privacy capabilities with BitTorrent's data transfer efficiency, developers can create secure and efficient decentralized applications. This combination addresses the weaknesses of each protocol, providing a comprehensive solution for privacy-conscious users and reliable data distribution.

Pear + Relays for IP privacy - *Recommended*

Overview

Pear is a versatile protocol that excels in managing data and provenance, making it easy to connect peers directly. It includes tools that simplify embed into mobile applications, enhancing its usability and accessibility. However, Pear does not provide IP privacy by default, requiring the addition of Relays. (Note, although IP addresses

aren't hidden, only individuals who possess the unique key for specific data would be able to discover the corresponding addresses.)

Relays add a layer of privacy to Pear by acting as intermediaries, obscuring users' IP addresses from others on the network. This setup requires users to configure relays as introduction points if they need enhanced IP privacy. Despite the need for additional configuration, the content remains hidden even if IP addresses are exposed.

Protocol / Criteria	Hyper/Pear	Relays
Open Source	✅	
User Authentication	✅	
Media Verification Authenticity	✅	
Provenance / History	✅	
Encryption (at rest)	❌	
IP Address Privacy	❌	✅
Free to Use (Cost)	✅	
Mobile Compatibility	✅	
Ability to Delete Media	❌	
Persistent Connection	✅	
Implementation Languages	Javascript	
P2P	✅	

Additional Context

- **Reliability and age of projects:** Pear is the latest instantiation of the project formerly known as Dat, and has been around for several years. Its reliability and speed is better or comparable to many other protocols. Its active development and growing community support make it a good choice for modern applications. The use of relays to enhance privacy is a well-established practice in the P2P ecosystem, adding robustness to the solution.
- **Security considerations:** Pear provides robust mechanisms for data verification and provenance but lacks built-in IP privacy. Using relays mitigates this issue by hiding IP addresses, though users need to ensure proper configuration of these relays. The content remains encrypted and secure, even if the IP addresses are exposed.

- **Similar projects and use cases:** Pear can be compared to other protocols focusing on data integrity and peer connections, such as IPFS and Hypercore. Relays are commonly used in privacy-focused options like VPN or HTTP Proxies, providing a proven method for enhancing IP privacy. For instance, the Hyperswarm DHT Relay can be used with WebSockets, and the Blind Relay is designed for forwarding UDP packets. These examples demonstrate how relays can facilitate secure and private peer-to-peer connections by masking IP addresses and routing traffic through intermediary nodes.

By integrating Pear with relays, developers can create applications that balance efficient data management
with enhanced privacy. This combination leverages the strengths of Pear in data handling and the proven privacy benefits of relays, providing a comprehensive solution for decentralized applications requiring both data integrity and IP privacy.

Protocol review

Arweave

Overview

Arweave is a decentralized data storage protocol that aims to provide permanent and immutable storage. It refers to this storage as the 'permaweb'. Arweave is supported by a decentralized network and incentivized by a token. Users pay an upfront fee for storage, which is placed in an endowment to support the perpetual storage of data and assets. Arweave uses a proof of access mechanism.

To interact with applications, a user must be authenticated and the application in turn must have permissions from the user to interact with the network on the user's behalf. Additionally, every interaction (transaction) on the network has a unique signature. The signature is a hash of the various input parameters as well as the user's key that helps authenticate the transaction and point it to a particular user.

Developers need to create user friendly tools and interfaces that verify user wallet connection to applications, use wallet addresses and request permissions from users to perform actions on their behalf.

Pros

- **User authentication:** Arweave uses 4096 bit RSA-PSS key-pairs stored using the JWK (JSON Web Keys) format.
- **Media authenticity with immutable data:** Immutability ensures data has not been tampered with. This is important for applications that require data integrity and trust.
- **Provenance provided by blockchain transaction metadata:** Arweave's blockchain architecture provides data provenance, allowing users to track the history and ownership of stored data. This is beneficial for applications requiring detailed records of data provenance and history.

Cons

- **No encryption:** Arweave does not support encryption for data at rest, meaning that stored data is not inherently protected from unauthorized access. Users need to implement their own encryption methods if needed.
- **No privacy:** The protocol does not prioritize user privacy, which means that stored data is accessible to anyone if no additional encryption was implemented. This can be a significant drawback for users who need confidentiality.
- **Cost:** Arweave is not free to use. Users must pay an upfront fee for storage, which might be a barrier for accessibility.
- **Immutable**: Arweave does not support the deletion of media. Once data is stored, it remains permanently on the network. This immutability can be a disadvantage for users needing the flexibility to remove data.

Overall Grade

Arweave is recommended for users interested in permanent and immutable data storage. However, users seeking encryption and privacy features may need to consider other options or implement their own encryption mechanism. The use of public/private keypairs ensures that the necessary primitives for encryption are already in place. It's also important to note that Arweave is not free to use and does not support the deletion of media.

BitTorrent

Overview

BitTorrent is a well-known protocol that excels in data distribution. It is widely used for distributing open-source software, media content, and large datasets. Despite its popularity and effectiveness in distribution, BitTorrent lacks several critical security features.

Pros

- **High reliability and speed:** Renowned for its ability to efficiently distribute large files by splitting them into smaller pieces and downloading them from multiple sources simultaneously. This method ensures high reliability and speed, even with large datasets. For example, many Linux distributions use BitTorrent to distribute their installation files due to its efficiency and resilience.
- **Free to use:** Bittorrent is an open protocol and free to use, making it accessible for a wide range of applications without the need for licensing fees. This has contributed to its widespread adoption in both personal and commercial use cases. Additionally, there are numerous implementations of the BitTorrent protocol, offering a variety of features and optimizations. Its general recognition and established user base further enhance its utility and reliability.

Cons

- **No user authentication**: BitTorrent does not include mechanisms for authenticating users, which means there is no built-in way to verify the identity of peers in the network. Users need to be cautious about the sources of the torrents they download, as there is a risk of encountering torrents published with harmful or malicious content.
- **Media authenticity:** BitTorrent uses content addressing to ensure the integrity of files, which helps in detecting tampered or corrupted files. However, the protocol does not provide features for verifying the provenance or authenticity of media files, making it possible for users to unknowingly download content from untrusted sources. This tradeoff is similar to other P2P protocols like Veilid, Hypercore, and Iroh, which also focus on content integrity but may not offer robust provenance verification.
- **No encryption:** BitTorrent does not offer encryption for data at rest, meaning files stored on users' devices are not protected against unauthorized access. Users must rely on external solutions to secure their data.

Overall Grade

BitTorrent is not recommended for scenarios requiring high security and privacy due to its lack of user authentication, media authenticity verification, and encryption for stored data. However, it remains a highly effective and widely-used protocol for efficient data distribution, especially for non-sensitive content.

Filecoin

Overview

Filecoin is designed for decentralized storage, focusing on security and scalability. It is built on top of the same software powering the IPFS protocol. Its use cases include NFT.Storage, which utilizes Filecoin to provide a simple decentralized storage solution for NFT content and metadata, and the Shoah Foundation and Internet Archive, which leverage Filecoin to back up their data

Pros

- **Chain of custody:** Filecoin maintains a detailed chain of custody for stored data, allowing users to track the history and ownership of files. This feature is crucial for applications that require verifiable records of data provenance.
- **Encryption is possible**: While encryption is not enabled by default, Filecoin allows users to implement encryption for their data. This flexibility enables users to secure their data according to their specific requirements.
- **Open source**: Filecoin is an open-source protocol, promoting transparency and community-driven development. This openness fosters innovation and allows developers to contribute to and build upon the existing technology.

Cons

- **Does not allow media deletion:** This immutability can be a drawback for users who need the flexibility to remove or update data.
- **Usage requires payment:** Since Filecoin operates on a token-based economy, users are required to pay for storage services. This cost can significantly limit accessibility.

Overall Grade

Recommended for users who prioritize secure storage and open-source technology. However, the need for payment and the lack of media deletion capabilities may be limiting factors. Filecoin is a robust option for applications requiring scalable and verifiable storage solutions supported by a large and active development community.

Freenet

Overview

Freenet is a distributed, decentralized alternative to the centralized World Wide Web, designed to protect freedom of speech and privacy. It operates as a peer-to-peer network where users' devices collectively host and share services, ensuring anonymity and security without central control. Freenet is structured to prevent censorship and surveillance, making it a robust option for privacy-conscious users.

Pros

- **Some IP address privacy:** Freenet offers some level of IP address privacy by routing communications through multiple nodes, which helps obscure the origin of data requests and provides a degree of anonymity for users. However, it does not employ a MixNet, and for stronger IP privacy, additional measures may need to be implemented.
- **Open source:** Freenet is an open-source project, allowing anyone to inspect, modify, and contribute to the codebase. This transparency helps build trust and encourages community participation.
- **Free to use:** Freenet is completely free to use, removing financial barriers for users who need access to a decentralized and private network.
- **Compatible with a wide range of devices:** The lightweight Freenet Core can be installed on various devices, including computers and smartphones, making it accessible to a broad audience.

Cons

- **No provenance/history tracking**: Freenet does not provide features for tracking the history or provenance of data, limiting its use in applications requiring detailed records of data ownership and changes.

Overall Grade

Freenet is recommended for users who prioritize privacy and decentralized network structures. However, it may not be suitable for applications requiring user authentication, media verification, or data mutability.

I2P

Overview

I2P (Invisible Internet Project) is a decentralized, peer-to-peer network designed to provide strong privacy protections for its users. It allows for anonymous communication and data sharing, ensuring that users' IP addresses remain hidden. The I2P network uses layered encryption and a distributed architecture to protect users' identities and data. An observer cannot see a message's contents, source, or destination.

I2P uses cryptography to achieve a variety of properties for the tunnels it builds and the communications it transports. I2P tunnels (paths through a series of routers) use NTCP2 and SSU2 transports to conceal the traffic being transported over the protocol. Connections are encrypted from router-to-router, and from client-to-client (end-to-end). Forward-secrecy is provided for all connections. Because I2P is cryptographically addressed, I2P network addresses are self-authenticating and only belong to the user who generated them.

Pros

- **Strong IP address privacy:** I2P offers strong IP address privacy by routing communications through multiple nodes, ensuring that users' IP addresses remain hidden and their activities anonymous.
- **Open source:** : I2P is an open-source project, allowing anyone to inspect, modify, and contribute to the codebase. This transparency helps build trust and encourages community participation.

- **Free to use:** I2P is completely free to use, removing financial barriers for users who need access to a decentralized and private network.

Cons

- **No media verification authenticity:** The protocol lacks mechanisms for verifying the authenticity of media, making it possible for tampered or fake content to be distributed.
- **No provenance/history tracking:** I2P does not provide features for tracking the history or provenance of data, limiting its use in applications requiring detailed records of changes.
- **No built in file exchange:** I2P is not a file transfer protocol; it focuses on connecting computers together but does not specify what is being exchanged.
- **Limited mobile compatibility:** While I2P can be used on mobile devices, its compatibility and performance on these platforms are limited compared to other protocols.

Overall Grade

I2P is recommended for users who prioritize strong privacy protections and decentralized communication. However, it may not be suitable for applications requiring media verification, provenance tracking, or encryption at rest. Its strong focus on anonymity and censorship resistance makes it ideal for privacy-focused applications, but users should be aware of its limitations in data management and security features.

IPFS

Overview

InterPlanetary File System (IPFS) is a protocol and network designed to create a peer-to-peer method of storing and sharing hypermedia in a distributed file system. It facilitates decentralized file storage, allowing users to store and access files across a distributed network of nodes. IPFS aims to make the web faster, safer, and more open by replacing traditional location-based HTTP with content-addressed IPFS.

Pros

- **High reliability and mobile compatibility:** IPFS is reliable and there are implementations for most platforms, including mobile devices. Its decentralized nature ensures that files are always accessible, even if some nodes go offline.
- **Free to use and open-source, fostering a broad adoption:** IPFS is an open-source project, allowing anyone to use, inspect, and contribute to the codebase. This openness fosters broad adoption and continuous improvement by the community. Additionally, it is free to use, making it accessible to a wide range of users and applications.

Cons

- **Lacks user authentication and robust privacy:** IPFS does not include built-in user authentication mechanisms or robust privacy features, meaning users must implement their own solutions to control access and ensure privacy.
- **No ability to guarantee deletion of media:** Once data is added to IPFS, it is distributed across multiple nodes, making it difficult to guarantee complete deletion of the data. This can be a disadvantage for users who need the flexibility to remove data from the network.

Overall Grade

IPFS is moderately recommended for users needing efficient and scalable file distribution without stringent security requirements. It excels in providing a decentralized and robust file storage system but may not be suitable for applications requiring strong user authentication, privacy, or data deletion capabilities.

Iroh

Overview

Iroh is a protocol for syncing bytes. At its core, it's a peer-to-peer network built on a *magic socket* that establishes QUIC connections between peers. Peers request and provide *blobs* of opaque bytes that are incrementally verified by their BLAKE3 hash during transfer.

Iroh is organized into three layers. Each higher layer depends on functionality in the layer below it. Documents rely on blobs, and blobs rely on connections. Its data exchange system provides efficient tunnel establishment and peer discovery.

The protocol supports a growing set of languages, embedding nodes directly in your project without any need to call out to an external API. It ships with SDKs for numerous languages and platforms, including Rust, Python, Swift and Kotlin.

Pros

- **Supports verification and privacy with Authors & Permissions:** Iroh provides robust cryptographic verification and privacy features through its Authors & Permissions system. Read access is granted by sharing the document's public key, and write access by sharing the private key. This ensures read capabilities at the network level, similar to Hypercore, but does not imply encryption at rest.
- **Allows for mutable document handling:** The protocol supports mutable documents, enabling dynamic data updates and modifications.
- **Documents as Pub/Sub swarm:** This means that documents are shared and updated in real-time using a system where changes to the document are published and subscribed to by others. Each document has a unique identifier, which is its public key, making it easy to find and update. This allows for efficient and timely distribution of data changes to everyone who needs them.

Cons

- **Nascent tech, early in development:** Iroh is still in its early stages of development, which may result in instability and ongoing changes as the protocol matures.
- **Media verification authenticity and chain of custody are not fully robust:** The mechanisms for media verification authenticity and chain of custody are not yet fully developed, which may be a limitation for applications requiring strong data integrity guarantees.

Overall Grade

Iroh is recommended for its excellent wire protocol data exchange between peers and user management features. Its support for encryption, privacy, and mutable documents makes it a strong candidate for projects requiring efficient and secure data synchronization. However, its early stage of development and incomplete media

verification features should be considered when evaluating its suitability for specific applications.

Pear (formerly Hyper)

Overview

Pear is designed for decentralized data structures, offering scalable solutions for collaborative environments. It leverages peer-to-peer networking to provide efficient data distribution and real-time updates across nodes, making it suitable for applications that require strong data integrity and reliability.

Pros

- **Strong support for media verification authenticity and chain of custody:** Pear ensures that media can be verified for authenticity and maintains a clear chain of custody, which is essential for applications that need to track the history and integrity of data.
- **Implements reliable data replication:** The protocol offers robust data replication, ensuring that data is consistently and reliably available across the network. This reduces the risk of data loss and improves overall reliability.
- **Free to use with a robust support for encryption:** Pear is free to use and provides strong support for encryption, enhancing the security of data transmissions across the network.

Cons

- **Does not have inherent IP privacy:** Despite its strong encryption, Pear does not offer inherent IP privacy features. Users must implement additional measures to ensure complete IP privacy.
- **Cannot delete media:** Once data is stored on the Pear network, it cannot be deleted. This immutability can be a drawback for users who need the flexibility to remove or update data.
- **Javascript reliance**: Pear uses Javascript, which can be heavy to implement, although the project has implemented some workaround functionality.

Overall Grade

Pear is recommended for collaborative and scalable environments requiring strong data integrity features. It is particularly well-suited for applications that need reliable data replication and robust verification mechanisms. However, Pear has limited IP privacy and media control options.

Peergos

Overview

Peergos offers a decentralized, privacy-focused file storage and sharing platform with a strong emphasis on security. It aims to provide a secure and user-friendly environment for storing and sharing files while maintaining user privacy. Peergos uses end-to-end encryption and a distributed architecture to ensure data security and privacy.

Pros

- **Support for provenance and media authenticity:** Peergos provides mechanisms to verify the authenticity of media and maintain a clear provenance, which is crucial for applications requiring data integrity and trust.
- **Ability to delete media:** Users can delete media when necessary, offering flexibility and control over stored data.
- **Maintain a persistent connection:** Peergos ensures persistent connections for reliable data access and sharing.
- **Strong privacy and encryption:** Peergos is one of the most privacy and encryption-focused platforms, outside of having MixNets for full IP privacy.

Cons

- **Key features still in development:** While Peergos supports many advanced features, some aspects of its reliability and the ability to handle chain of custody are still being developed. This can affect the overall robustness of the platform.
- **This is a paid service:** Peergos requires payment for its services, which may limit accessibility for some users looking for free storage solutions.
- **Limited client-only operation:** While Peergos can be self-hosted and networked between hosted instances, it cannot be run entirely on a client without the use of Peergos servers.

Overall Grade

Peergos is recommended for users who prioritize privacy and security in a user-friendly environment. Its robust support for provenance, media authenticity, and the ability to delete media make it suitable for secure file storage and sharing. However, users should consider the ongoing development of certain features and the cost associated with the service.

SocketSupply

Overview

SocketSupply is a newer entrant in the P2P space, focusing on secure communications and file sharing. It allows developers to build mobile and desktop applications for any operating system using familiar web technologies like HTML, CSS, and JavaScript. SocketSupply aims to provide a seamless development experience while ensuring data security and privacy.

Pros

- **Free to use:** SocketSupply is completely free to use, making it accessible for developers and users without financial barriers.
- **Open source:** As an open-source project, SocketSupply allows anyone to inspect, modify, and contribute to the codebase. This transparency helps build trust and encourages community participation.
- **Cross-platform development:** Developers can build mobile and desktop applications for any operating system using HTML, CSS, and JavaScript, streamlining the development process and reducing the need for multiple codebases.
- **Strong focus on mobile:** SocketSupply has a strong focus on optimizing mobile battery usage and speed.

Cons

- **Uncertainties about the ability to delete media:** It's unclear whether SocketSupply has the ability to delete media once it is stored, which may be a limitation for users who need the flexibility to remove data.
- **APIs for encryption:** While SocketSupply provides APIs for encryption, developers need to handle the encryption details for each application.
- **Tightly coupled with their webview:** The code is tightly coupled with their webview, making it harder to integrate with existing mobile apps.

- **No specific file exchange protocol:** SocketSupply does not have a specific file exchange protocol; it simply relays packets to their destinations. This means applications need to bring their own logic for handling binary blobs.

Overall Grade

SocketSupply is recommended for those looking to offer cross-platform options. Its ease of use, open-source nature, and support for web technologies make it a strong choice for developers aiming to build secure and versatile applications. However, it is best suited for new apps and can be harder to integrate into existing ones. Potential limitations regarding media deletion should also be considered.

Source.Network

Overview

Source.Network aims to integrate blockchain technology for secure and verifiable transactions alongside data storage. By leveraging blockchain, Source.Network provides a decentralized and transparent platform where data integrity and authenticity are guaranteed. It supports a wide range of applications, particularly those that require robust security and verifiable records.

Pros

- **Strong support for encryption, privacy, and chain of custody:** Source.Network ensures that all data transactions are encrypted and private, with a clear chain of custody. This makes it suitable for applications that require high levels of security and traceability.
- **Open-source with good mobile compatibility:** As an open-source project, Source.Network encourages community involvement and transparency. It is also designed to work well on mobile devices, expanding its usability across different platforms.

Cons

- **DefraDB does not offer access control or data encryption:** DefraDB, one of the protocol's components, lacks built-in access control and data encryption. Additionally, the default configuration exposes the database to the network, which can be a security risk if not properly managed.

- **Usage requires payment:** Since Source.Network operates on blockchain technology, users are required to pay for transactions. This cost can significantly limit accessibility.

Overall Grade

Source.Network is recommended for users interested in blockchain integrations with strong security features. However, it requires further clarification on the encryption of DefraDB and its default configurations. It is a strong choice for applications needing secure, verifiable transactions and data storage but requires careful handling of its database configurations.

Veilid

Overview

Veilid is an open-source, peer-to-peer, mobile-first, networked application framework that offers features for secure and private data handling. This makes it one of the top picks for network functionality.

The framework is conceptually similar to I2P and Tor, but faster and designed from the ground up to provide all services over a privately routed network. This article from the Veilid launch provides a good high-level overview of the intended functionality.

The framework enables development of fully-distributed applications without a blockchain or a transactional layer at their base. It uses UDP, TCP, and Websockets to facilitate communication and data transfer.

Pros

- **Strong encryption and privacy features:** Veilid provides robust encryption and privacy, ensuring that data is secure and user anonymity is maintained.
- **Mobile-first:** The framework is designed with mobile compatibility in mind, making it ideal for developing applications that need to perform well on mobile devices.
- **Free to use:** Veilid is completely free to use, which removes financial barriers for developers and users looking to build or use secure, distributed applications.

Cons

- **Nascent tech, early in development:** Veilid is still in its early stages of development, which may result in instability and ongoing changes as the protocol matures.
- **Does not have good support for data storage:** The framework does not provide built-in solutions for data storage, meaning applications need to figure out how they want to store data themselves. However, blob storage is on their roadmap.
- **Limited data handling for large datasets:** While Veilid ensures that anything sent over the network is verifiable, it focuses on small bits of data being sent between peers and does not handle large datasets natively.

Overall Grade

Veilid is highly recommended for users needing secure and manageable P2P solutions with privacy as a priority. Its strong encryption, privacy features, and mobile-first design make it a compelling choice for developing secure, distributed applications. However, its early stage of development and limited support for data storage and handling of large datasets should be considered.
