future internet

MDPI

Article

Comparison of Distributed Tamper-Proof Storage Methods for Public Key Infrastructures

Fabian Honecker, Julian Dreyer and Ralf Tönjes *

Faculty for Engineering and Computer Sciences, University of Applied Sciences, 49076 Osnabrück, Germany
* Correspondence: r.toenjes@hs-osnabrueck.de

Abstract: Modern Public Key Infrastructures (PKIs) allow users to create and maintain centrally stored cryptographic certificates. These infrastructures use a so-called certificate chain. At the root of the chain, a root Certification Authority (CA) is responsible for issuing the base certificate. Every verification and certification step within the chain is based upon the security of said root CA. Thus, its operation security is of great concern. Since the root certificates are stored locally on the root CA, any Denial of Service (DoS) attack may render the whole certificate chain, which is based on of the attacked root CA, inoperable. Therefore, this article evaluates different approaches to a decentralized data storage system that is based on the Distributed Ledger Technology (DLT). To show the real-world potential of the proposed approaches, we also evaluate the different technologies using a novel PKI mechanism called Near Field Communication Key Exchange (NFC-KE). The results indicate that modern distributed data storage solutions such as Interplanetary Filesystem (IPFS) and SIA can have significant performance and decentralization benefits in comparison to purely Blockchain-based technologies like Hyperledger Fabric. However, they lack any Smart Contract functionality, which requires a software developer to implement verification mechanisms in centralized software solutions.

Keywords: Public Key Infrastructures; decentralized data storage; blockchain data storage; Certification Authority; decentralized trust; public key cryptography

Citation: Honecker, F.; Dreyer, J.; Tönjes, R. Comparison of Distributed Tamper-Proof Storage Methods for Public Key Infrastructures. *Future Internet* **2022**, *14*, 336. https://doi.org/10.3390/fi14110336

Academic Editors: Christoph Stach and Clémentine Gritti

Received: 28 October 2022
Accepted: 16 November 2022
Published: 18 November 2022

1. Introduction

Establishing a trust relationship between two parties that never met before is an inherently hard task. Nevertheless, this task is executed each time a user visits, e.g., a website that is secured using the Hypertext Transfer Protocol Secure (HTTPS). Here, the user needs to be confident that the response of the server is authentic and has not been tampered with. Otherwise, a malicious server impersonator could send unauthentic messages and thereby compromise the communication between the user and the server. To remedy this fundamental problem, modern cryptographical digital signature schemes are used in conjunction with certificate chains.

These signature schemes mostly rely on asymmetric public key algorithms to sign data packets using a private/secret key (SK_{serv}), e.g., of the server, and an accompanying public key (PK_{serv}) that is stored within a cryptographic certificate. Now, when using this mechanism on its own, again, an attacker could craft a malicious certificate and thereby impersonate the real server. Therefore, multiple intermediary parties are introduced within the system. These are called *Certification Authority* (*CAs*). Their purpose is to issue and maintain user certificates and thereby act as a trusted intermediary. To enhance the cryptographic trust between the Certification Authorities (CAs) and the certificate owners, multiple CAs are chained together, forming a certificate chain. This chain is based on a root CA that issues a root Certificate and stores its corresponding private key securely on the client. Additional CAs are added by issuing new certificates that are signed by the root CA and are only capable of issuing and verifying new certificates. Most commonly, these

certification chains have a length of two to four CAs and thus allow for a robust and secure digital trust relationship that forms the basis of modern web traffic.

This system has three significant problems that are under current research [1–3]: First, the chain of trust is based on the trust for the root CA. This trust is artificially generated, e.g., by the developers of modern Web browsers and operating systems that store the root certificates in the default keychain of the system. These certificates need to be trusted implicitly. Second, the maintenance effort for establishing and running a custom certification chain is beyond the scope of competencies of most Small and Medium Enterprises (SMEs). These enterprises often do not own a proper IT department to establish such a certification chain and are therefore depending on a proper solution provided by external contractors. The third problem is an inherent flaw in the design of the root CA and its centralized, on-device keystore. If a root CA has been compromised, the attacker has potential access to the file system and all of its files and certificates. This allows, in the worst case, for tampering of the certificate contents or maliciously uploaded certificates that are not properly issued. In cases when a formerly well-intended CA turns out to be fraudulent, the same above problems arise.

The Distributed Ledger Technology (DLT) offers a solution for problems one and three by using its decentralized execution and data storage concepts and, when combined with a lightweight Public Key Infrastructure (PKI) approach, also a potential real-world applicable solution for problem two. Therefore, we chose to adapt the Near Field Communication Key Exchange (NFC-KE) PKI scheme, which offers a lightweight public key exchange scheme. This scheme is particularly suited for SMEs since it does not require a certification chain for its operation.

Most research on the general topic of DLT-based CAs is focusing on novel protocol approaches for diversifying the realm of new signature verification techniques, aided by, e.g., Blockchain platforms [4–6]. A general pattern evolved among these works, showing that popular DLT frameworks like Hyperledger Fabric (HLF) [5] or Ethereum [6] are valid candidates for successfully implementing those novel PKI approaches. These approaches rely on the Smart Contract capabilities of said platforms, thereby neglecting distributed data storage systems such as the Interplanetary Filesystem (IPFS), which also use the DLT at its core.

Therefore, our works aims to reason the feasibility of implementing novel DLT-based PKI schemes using distributed file storage technologies. To show the advantages of a tamper-proof storage system that is based on the DLT for PKIs, this article evaluates a broad spectrum of modern DLT solutions in regards to their specific traits and benefits, as well as their performances under the given application scenario. The main contributions can be summarized as follows:

- Concept proposals for decentralized tamper-proof storage algorithms for PKIs, based on *private IPFS*, *public IPFS* and *Sia*.
- Real-world application and adaptation to the lightweight NFC-KE PKI scheme [7]
- A performance evaluation of the system using standardized Key Performance Indicators (KPIs), e.g., *execution times*

The remainder of this article is structured as follows: First, in Section 2, we provide a brief overview of other approaches to decentralize modern PKI schemes and use case scenarios for the decentralized IPFS. In the following Section, Section 3, we explain the new concepts for the decentralized and tamper-proof public key stores for each DLT framework by first introducing the selected DLTs and introducing the NFC-KE and its intricacies. After this, the concepts are implemented in Section 4 and evaluated for their performances afterward in Section 5. The performance evaluation uses the same KPIs as previously used in [7] to establish a common basis for comparison. The results of the performance analysis and its implications are further discussed in Section 6. Finally, Section 7 concludes this article. All abbreviations used in this article are listed and explained in the abbreviations table at the end of this article.

2. Related Work

The previously mentioned drawbacks of modern PKIs have been addressed previously. In their research paper [1], Kfoury et al. proposed a new methodology of creating a fully functional PKI that leverages the Ethereum Blockchain platform and its Smart Contract capabilities. Among other contributions, their proposed system offers a proper Preshared-Key (PSK) exchange and keystore concept that is decentralized and thus offers enhanced tamper-resistance. By using an Application Management Platform in conjunction with the public Ethereum Blockchain, the authors achieved an implementation that showed the real-world application potential of the proposed system. Their approach is, therefore, capable of replacing common CA implementations for their system.

Trust and usability issues of common PKIs have also been identified by Hepp et al. [2] in their extensive exploration of potentials and challenges of Blockchain-based PKIs. The authors research four distinct usage scenarios that may benefit from the introduction of the DLT or the Blockchain in particular. Concepts such as the PKI, Pretty Good Privacy (PGP), and X.509 are shown to be capable of benefiting from the introduction of a Blockchain-backed approach for fixing common security problems and trust issues. The authors conclude that, at the state of writing, the Blockchain-based PKIs offers a large number of potentials, although there are major challenges to solve, especially concerning speed and scalability.

Additional Blockchain-based concepts for a decentralized PKI have been proposed by Papageorgiou et al. [3]. Their approach is focussing on being particularly applicable for the Internet of Things (IoT) and its special demands. The IoT is characterized, among other facets, by the dynamic joining and leaving of new or known network participants. Common PKI solutions, as the authors argue, are not always capable of coping with the corresponding scalability demands. Furthermore, the common monolithic PKI systems establish a single source of failure. In case of a PKI system outage, no trusted communication is possible anymore. To remedy these issues, the authors analyzed the abstract inner routines of common CA architectures and identified singular building blocks that can be decentralized. Using this method, they identified the verification component and swapped it out for a decentralized authorization component that offers a distributed information basis. To show the real-world applicability of the concept, an implementation using the HLF Blockchain Platform is provided. The results indicate that this solution is indeed applicable for IoT networks and is thus capable of replacing common monolithic PKI solutions.

In practice, the IPFS is often used in combination with a Blockchain-based Smart Contract platform for handling logic. Ramesh et al. [8] use the IPFS together with a private Ethereum-based application for secure data management that is especially suited for the IoT. Their approach is based upon a common IoT network structure, comprised of data consumers and producers. The exchanged data, which themselves shall be encrypted using specialized security measures, is stored decentralized on the IPFS. To allow for encryption and data verification, the Ethereum Smart Contracts allow for a custom programmability for the desired use case. In their specific setup, the authors used a Trusted Platform Module (TPM) for local key storage and verification operations. They were able to show that their system offers a throughput performance of eight transactions per second using a private Ethereum setup, thereby proving the real-world applicability. Though, the performance itself is mainly influenced by the data verification on behalf of the data consumers and producers.

Another application example of the IPFS being employed in combination with the Ethereum Blockchain is provided by Uddin et al. [9]. In their research paper, the authors propose a decentralized and secure data sharing system that is based upon the IPFS for handling the decentralized data storage and an Ethereum-based encryption and decryption mechanism. Furthermore, the Ethereum Smart Contracts are used for data identification and client authentication. Since their setup is not fundamentally new, the authors compare their proposed solution to previous works by Aslam et al. [10], Hasan et al. [11], Nizamuddin et al. [12], and Park et al. [13], which all have unique drawbacks in terms of security and

performance. Since their system is intended to be used on the public Ethereum blockchain, the authors also provide a cost estimation based on Ethereum transaction/gas prices. Their approach has shown to be real-world applicable and offers new security perspectives for similar use cases.

3. Concepts and Evaluation Methods

The following subsections will give a brief overview of the several technologies that are used for the concept of a tamper-proof data storage for PKIs. First, the DLTs used within this article are described. Furthermore, this section also includes a statement on how these concepts are evaluated later. For that, the main lightweight PKI scheme NFC-KE will also be explained and discussed.

IPFS Private

The IPFS [14] is a distributed Peer-To-Peer (P2P) network intended for storing and allowing access to data, websites, and applications. In contrast to centralized systems that manage the data access via location-based addressing, IPFS implements a content addressing scheme [15] for accessing data stored within it. Using this scheme, a distributed download of one single file via multiple peers in different locations is possible. Said peers are called *IPFS Peers* in the following.

The decentralized storage of files offers, beyond others, some noteworthy advantages:

1. **Redundancy:** Centrally stored files are lost, if the central file server is out of operation, e.g., after a fire hazard. Using IPFS, data redundancy can be achieved since data stored on one given peer is also replicated to multiple peers within the network. Therefore, the data availability is enhanced.
2. **Performance:** The data query can be accelerated because, using a custom routing scheme within the IPFS network, files can be queried from geographically close IPFS peers. This routing scheme is facilitated by distributed hash tables (DHTs) and is based on a P2P basis [16]. Thus, a file that has been uploaded in Brazil can be queried within a few milliseconds by a client located in Germany (Note: After complete file replication within the network and if there are IPFS peers closely located).
3. **Tamper-proof design:** For third parties, manipulation or deletion of a file stored within the IPFS is made almost infeasible since the data redundancy and content addressing schemes cryptographically and physically deny any attempt to alter data.

The content addressing works by cryptographically hashing the given files using the *SHA-256* hashing algorithm. Thus, two identical files will always produce the same data hash (called *Content identifier (CID)*). Furthermore, each file is split up into chunks of the same size to allow for an even more extended spreading of data. This is made possible by using Directed Acyclic Graphs (DAGs) and Merkle-DAGs in particular [17].

Fundamentally, the IPFS is a homogeneous network comprised of peers that are storing the desired files on their local hardware. Each peer is therefore responsible for storing and managing the data, as well as communicating with other network participants. In general, IPFS is a publicly available implementation that can be used either in a private deployment or using public Application Programming Interfaces (APIs) and frameworks, although the inner workings of the technology are common for every setup.

When using a public IPFS setup, each peer needs a financial incentive to store the data on its local hard drive. Within a private setup, this requirement is not given since every peer is set up on-demand and is well-intended by default.

Public IPFS-Filecoin

IPFS can be implemented in both a private and a public setup. Each paradigm involves different intricacies that need to be considered. On the one hand, public IPFS solutions offer a broad decentralization and may spread all around the globe while, on the other hand, private setups do not require a financial incentive to be run and also allow for generally better performance.

Filecoin [18] is a Blockchain-based P2P network that uses such a financial incentive to guarantee a persistent and reliable data storage. By using Filecoin, each client will pay a small fee for the data storage to the peers hosting the storage space. Those peers are essentially storage servers connected to the underlying IPFS network. After registering to the Filecoin network, they can offer their free available storage space to any client willing to pay them. Thus, a digital marketplace for decentralized data storage, which is backed by a Blockchain keeping track of the Filecoin transactions, is established.

Additionally, storage providers can promote special benefits like lower cost, higher redundancy, or enhanced access speed. After a client chose to store his data on a given provider's hardware, the provider needs to execute a *Proof-of-Replication*. This scheme, being beyond the scope of this article, is responsible for eliminating the *double-spending* problem and essentially ensures a proper storage operation that is provably correct.

Sia

Sia is a decentralized cloud storage platform that implements a different storage scheme than Filecoin or IPFS in general. Instead of lending storage space from a central provider, Sia allows clients to lend storage space to each other. The underlying Sia Blockchain only stores the storage contracts the clients agreed on. Each contract has a storage interval in which the data storage provider needs to guarantee the storage of the data. To prove this, the provider executes a *Proof-of-Storage* algorithm, thereby validating the contract.

Additionally, the storage providers get compensated for each successful proof using the native *SIA Token*. In return, the data client will have to spend some of his SIA Tokens for the data to be stored on the network. After the process is complete, chunks of the data will be replicated to 30 different peers within the Sia network and, using error correcting code (ECC), can be reconstructed using only ten unique peer responses [19].

Hyperledger Fabric

Hyperledger Fabric is a private/permissioned Blockchain platform that offers custom programmability using Smart Contracts and a fully custom network setup. In contrast to other, mostly public Blockchain platforms, the network structure of Fabric is heterogeneous. It is comprised of different nodes each with a different role, alhough each peer role may occur multiple times within the network.

1. Channels

On the top level, each custom Fabric network is based around a *Channel* [20]. It hosts a custom Blockchain together with a so-called *World State* database. The Blockchain is responsible for keeping the transactions logged and persisted within the network while the World State contains all the software-defined objects and their connected states. Fundamentally, it acts as a distributed database that is kept in sync using an ordering mechanism.

2. Organizations

Each Channel can be organized within several *Organizations* [21]. They allow for a more business-oriented network structure and encapsulate the clients within it from the rest of the network. Each Organization can host its own CA, which is responsible for issuing new client certificates. Additionally, Smart Contracts can be deployed on an organizational basis.

3. Peers

The Peers are physical entities within the network [22]. The Channel and Organization concepts in themselves are purely logical while the Peers are separate clients within the network. Each Peer hosts a full copy of the shared ledger, in this case, the Fabric Blockchain and the World State. To keep these in sync with other Peers and thus reach a consensual basis, each Peer will communicate with other Peers within the same Organization. Furthermore, the peers are responsible for executing and validating Smart Contracts being installed onto them. During network setup, an administrator may choose to configure some Peers as *Endorsement Peers*. Those Peers are on duty of executing, checking, and

endorsing Smart Contracts while non-Endorsement Peers only hold and maintain a copy of the Blockchain and the World State.

4. Smart Contracts

In the Fabric domain, Smart Contracts are called *Chaincodes* since they are code fragments, written in high-level languages like Java or GoLang [23] and run on-chain using the underlying Fabric Blockchain. They are the fundamental way to alter the World State by adding, deleting, updating, and reading data from it. Thus, the Smart Contracts allow for decision-based data storage for data that can be stored within the World State database. Methods like cryptographical signature verification, decryption, and permission handling are all possible using the Smart Contract functionality of Fabric.

5. Ordering Service

After a Peer issues a new transaction, it needs to be stored on the Blockchain. Since there might be multiple transactions being issued at the same time within the network, a protocol for reaching a consensus must be employed. For this, the role of an *Orderer* has been created [24]. Using a Byzantine-Fault-Tolerant algorithm, the (multiple) Ordering nodes, which themselves are distinct clients within the network, are responsible for receiving the new transactions from the peers, sorting them by their timestamps and other metrics, and creating a new block for the Blockchain. The new block will hold the newly collected transactions and is sent to every peer within the network. Eventually, every peer will receive the new Block and append it to its local copy of the Blockchain and update the World State accordingly.

NFC-Key Exchange and Evaluation Methods

Modern PKI hierarchies and certification chains are often too cumbersome to set up and maintain, especially for SMEs. Some scenarios within the Industry 4.0 domain require the use of trusted communication between the involved parties. Mostly, these scenarios are based on sensor networks and low-cost devices. Thus, large server environments and multiple CA instances are too expensive in terms of resources and maintenance effort for these specific scenarios.

Nevertheless, modern cryptographic signature schemes are more often used in these communication scenarios. This process is also fostered by more modern and potent hardware on the sensor side that allows for faster signing of data packets. The authenticity of data is crucial in most industrial scenarios since no malicious sensor nodes shall be able to intervene within the network, e.g., by using unauthenticated data requests or by acting as a Man-in-the-Middle (MitM) and manipulating well-intended data packets.

Therefore, Dreyer et al. proposed a novel public key exchange scheme based on the Near Field Communication (NFC) as a transport medium [7]. The main idea of the concept assumes that each sensor is capable of signing a given data packet using its own secret key SK_S. The resulting signature can be verified using the corresponding public key PK_S. In practice, this basic concept works until a malicious actor joins the network and begins signing his own data packets. Therefore, the proposed NFC-KE concept involves a central PKI-like entity, called *Signing Hub*, that is responsible for verifying access to the network and thus allowing for communication. Since only previously authenticated publickeys will be added to the system, this process is called an *authentic public key exchange*.

The Signing Hub is comprised of a TPM connected, in this case, to a Raspberry Pi microcontroller. Additionally, it has an NFC-Reader attached to its serial port allowing for NFC tags to be read and written. The main challenge–response protocol of the NFC-KE is depicted in Figure 1.

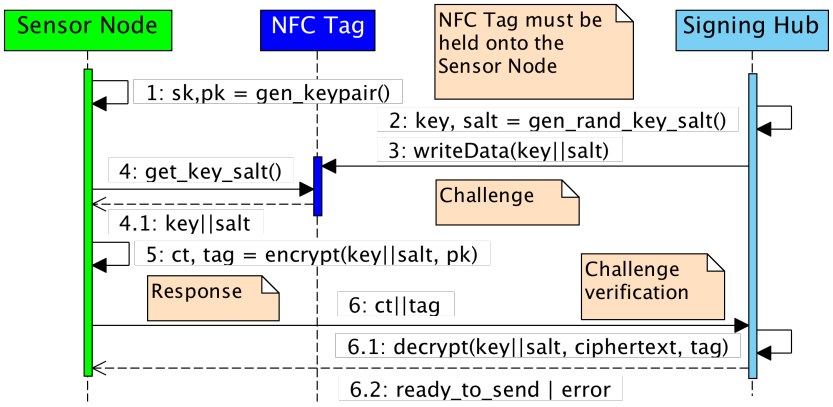

Figure 1. NFC-KE Phase 1: Fundamental public key exchange scheme [7].

The fundamental working principle of the NFC-KE requires a physical Proof-of-Possession scheme to be performed between the Sensor Node and the Signing Hub. Using the NFC, which requires a close physical distance for communication, a cryptographic token can be exchanged between the two parties, thereby implicitly proving authorization due to the low likelihood of an unauthorized NFC-communication. For further details on the NFC-KE, the reader is referred to the original paper [7].

Due to its lightweight setup and its ability to be implemented using low-cost hardware, the NFC-KE scheme is a valid candidate for low-resource PKI replacement in Industry 4.0 and Industrial Internet of Things (IIoT) environments in general.

The NFC-KE in its basic variant is susceptible to arbitrary file-tampering if an attacker is able to exploit a potentially unknown software vulnerability, e.g., by infiltrating the Raspberry Pi microcontroller. The properly exchanged authentic public keys of the Sensor Nodes are stored as certificate files on the Signing Hub. If an attacker had access to the filesystem, arbitrary upload and manipulation of said files could be possible. In the worst case, an attacker could be able to inject a nonauthorized public key and thus circumvent the whole authorization mechanism. This, therefore, creates a general single point of failure, as well as a setup vulnerable to public key tampering.

To remedy this problem, Dreyer et al. also proposed an extension to the former NFC-KE protocol that is using the HLF Blockchain for a more decentralized data storage of the public keys [25,26]. The new approach essentially replaces the standard filesystem-based storage approach of the NFC-KE scheme in favor of a Blockchain-based approach. Instead of storing and implicitly validating the newly exchanged Sensor Node's public key on the Signing Hub device itself, this functionality is outsourced to an external Blockchain Platform, in this case, HLF. The authors showed that using their approach, the NFC-KE experiences, on the one hand, a decrease in performance while, on the other hand being more tolerant against system outages, tampering attempts and unauthorized access due to the higher redundancy and tamper resistance of the HLF platform.

For their new approach, the authors introduced two new network participants, one being the Application Server and the other being the HLF instance (s. Figures 2 and 3). Generally, the Application Server is responsible for validating the requests originating from the Signing Hub on a network level and passing them on to the HLF instance. The latter is responsible for performing upper-level validations such as permission control and signature verification. Since HLF offers a fully programmable Smart Contract runtime, every signature verification can be performed on-chain, thus enabling an enhanced distributed trust. Additionally, each Sensor Node's public key, after being successfully exchanged, is stored within the World State of the HLF platform. This eliminates the need for a distinct file for each new key and thereby reduces the chances of arbitrary file tampering. For further details on the inner workings of the general extension scheme, the reader is referred to [25].

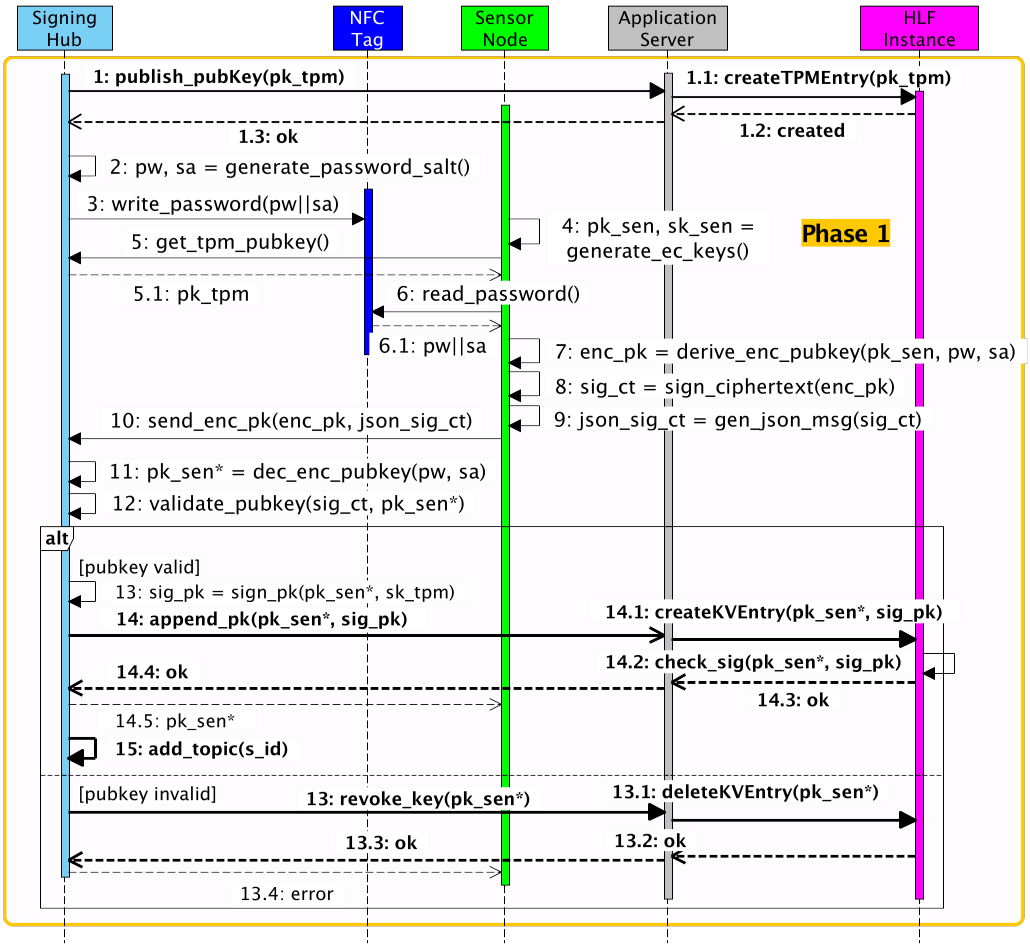

Figure 2. NFC-KE Phase 1 HLF extension [25].

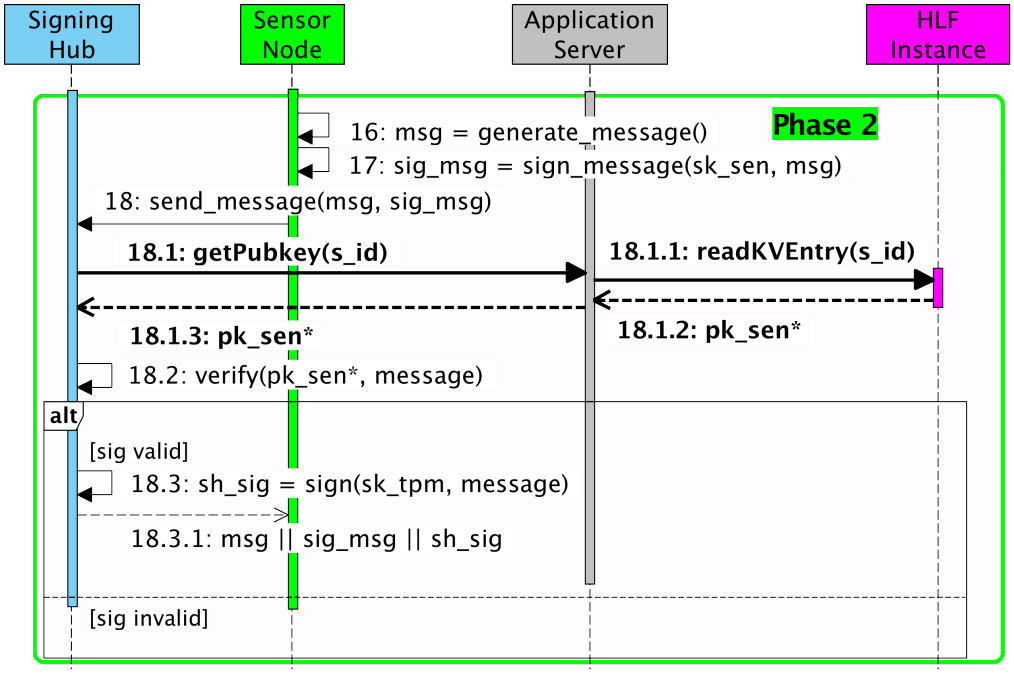

Figure 3. NFC-KE Phase 2 HLF extension [25].

The basic idea of outsourcing the filesystem keystore to an external DLT-based solution is also pursued by Papageorgiou et al. [3]. Their approach also includes a preceding analysis

phase of the system in regards to potential parts that may benefit from being outsourced to a DLT. Thus, for this article, we identified the filesystem in particular and analyzed how different DLT solutions can handle the same functionality as the HLF-based approach of [25].

4. Alternative Implementations and Concept Analysis

One goal of this article is to present an alternative approach for a DLT-based data/file storage that is suited for the given use case as a replacement for the filesystem keystore in the NFC-KE scheme. Apart from the file storage per se, the implemented NFC-KE extension using HLF is also capable of executing Smart Contracts, e.g., for verifying cryptographical signatures. The other DLTs mentioned in Section 3 lack such a feature entirely and focus mainly on the plain storage aspect of the data.

Therefore, a custom verification mechanism is required to map the Smart Contract functionality to a centralized entity. For all concepts explained in the following sections, the following assumptions are set:

- Each Sensor Node has a unique identificator ID_S
- All Sensor Node public keys will be called PK_S
- The corresponding file for a given PK_S will be named $< ID_S > .json$

4.1. Application Server

Since a stateless Application Server component is already present in the NFC-KE extension (s. Figure 2), it can be extended to support the verification of incoming signatures for new public keys. We chose to implement the Application Server using Node.js and JavaScript to achieve a universal basis for each DLT evaluation. Each of the proposed DLTs provide a custom NodeJS library and/or API endpoints for interacting with them [27–29]. Additionally, to reach enhanced modularity and exchangeability of the different technologies, object-oriented programming paradigms were leveraged, enabling a loosely coupled system.

When a new public key needs to be added to the system, the Application Server will perform a validation of the corresponding signature, following the NFC-KE scheme. For this, it has to query the publicly available public key of the TPM PK_{TPM}, which has to be stored immutably in the given DLT. The following subsections will describe this particular process in more detail. After querying PK_{TPM}, the Application Server can use the key for signature verification of the newly sent Sensor Node's public key PK_S. If the verification is successful, the key can be passed onto the given DLT and stored permanently.

Another problem arises when querying the newly added Sensor Node's public key since the presented DLT platforms do not offer a *Key-Value* mapping like HLF does. In the presented NFC-KE extension, the unique Sensor Node identifier ID_S is used as a key in the World State database and PK_S as the corresponding value. To map this scheme to the other DLT platforms, new concepts are needed and further described in the following subsections. Naive implementations could make use of the filename, e.g., Sensor_X.pem, and use it as an identifier. Since the presented DLTs are using content-based addressing, the unique content hash (called CID) of the file is used for addressing. Therefore, a mapping between the CID and the sensor id is required, resulting in the same fundamental problem.

Private IPFS

The fundamental IPFS system allows for file deletion using a garbage collection mechanism. Each file has a property called *Pinned*. If a file is pinned, the internal IPFS garbage collector will not delete the file. Else, if the file is unpinned, eventually the garbage collection will delete and remove the file from the network. Depending on the network size and speed, this process can take some time.

The pinning request has some intricacies that allow a direct *SensorID-Publickey* mapping, just like HLF. Each *Pin* has some metadata attached to it that can be arbitrarily altered by the user. Therefore, when uploading a file to the IPFS the Application Server will pin

the file accordingly and add the given Sensor ID ID_S to its metadata. Using integrated network functionalities, the IPFS can be queried for all (or specific) pins. The resulting list can then act as a direct ID_S-PK_S mapping.

The functionality of adding the TPM key to the IPFS instance also works similarly. The required pin for the TPM public key PK_{TPM} has a special identifier, e.g., TPM_{PK} and is programmed into the application server. This pin cannot be altered after being inserted.

Public IPFS

The previous concepts apply specifically for the private IPFS setup and the provided IPFS Software Development Kit (SDK) [27]. For the public IPFS variant, *Filebase* [30] offers a free to use API that exposes all IPFS functionalities. Though, the ID_S-PK_S mapping can be handled easier by leveraging the Filebase API. To query a given file, a custom GET request needs to be crafted that contains the following file name as a parameter. On request, the API will return the complete object stored under that file name. Alternatively, if the CID of the file is known, the file can also be queried using public IPFS gateways [31].

An alternative way of accessing the public IPFS is the *Web3.Storage* API. In the background, Web3.Storage uses the IPFS with a financial incentive concept called *Filecoin*. Using this incentive system, data can be guaranteed to be persisted for the long term, since each data hoster will be rewarded with Filecoin tokens. Whenever a user uploads a new file to the network, he will receive a unique CID for said file. This can then be used to query the data from the network. For users of the Web3.Storage API, this feature is free to use [32].

Adding a new Sensor's public-key to the IPFS using the Web3.Storage API differs, again, in its way of mapping the ID_S to its corresponding PK_S. Since the user receives a unique CID of the file and can use this CID to query it, a direct search for the key is not trivially possible. For this purpose, the API offers a method to query all files uploaded via a given Web3.Storage account. This feature can be exploited to receive a full list of all files. Since by definition the uploaded files will follow the naming scheme $< ID_S > .json$, the returned list can be trivially searched for a given ID_S. The list, therefore, also contains a direct ID_S-CID mapping, allowing the public key storing concept to work.

To add the TPM's public key to the IPFS using the Web3.Storage API, the same methodology is used as in the previous approach using the private IPFS implementation. The TPM's public key is uploaded under a specific identifier which is programmed into the application server's logic.

Sia Skynet

In its abstract form, Sia offers the same functionalities as plain IPFS storage systems when using its *Skynet* API. For public key storage, Skynet offers a method to upload arbitrary data using a file and a corresponding filename. Again, following set conventions, the filename will be using the naming scheme $< ID_S.json >$. After uploading a file to the Skynet network, the user receives a so-called *Skylink*, which can be compared to the concept of a CID that is used in IPFS. Essentially, a Skylink allows direct access to the given file.

To support the query of an uploaded public key, an alternative approach needs to be proposed, since the previously described concepts that apply for private and public IPFS solutions do not apply to Sia Skynet. Other than Web3.Storage, for example, Skynet does not offer a direct API endpoint to query all uploaded files. Therefore, we propose an alternative approach using the available *SkyDB* [33] that is offered by the Sia Blockchain. Other than the Skynet system, the SkyDB is directly stored on the Sia Blockchain and thus only allows for small datasets to be stored. By adding a mapping file, called *devices.json* for this case, a mapping can be created for each new ID_S-PK_S pair using the Filename and the returned Skylink ID. On creation, the *devices.json* file will be stored on-chain and can be queried, edited, and updated at any time. After a successful query of the mapping file, the desired ID_S-PK_S mapping for the given Sensor Node can be extracted.

To support the special storing of the TPM's public key, this approach also follows the same principle as described in the previous section for IPFS.

4.2. Further Considerations

Every presented approach has some particularly minor accompanying problems, that require consideration.

Since the public APIs all rely on a properly working internet connection, increased network latency can be expected in comparison to private deployments. Public DLT setups offer enhanced global decentralization and therefore a more resilient data availability. As stated previously, data can be queried on a P2P basis and is thus more directly available. On the one hand, using the financial incentive system, e.g., Filecoin, data is guaranteed to be pinned/stored persistently on the public IPFS. There is still a remaining yet small chance of the uploading peer to disconnect from the network during initial upload. Then, the given file will not be pinned or uploaded. This financial incentive, on the other hand, renders the corresponding solutions more expensive to use since the public APIs all offer free plans that allow for a limited amount of data to be stored, e.g., 5GiB per day, or a decreased performance (e.g., latency, bandwidth, etc.) in comparison to a paid plan.

All presented alternative approaches to replace the central NFC-KE on-device public key storage lack the Smart-Contract-based verification functionality that enables a decentralized, fully autonomous signature verification. Since this feature is missing from all of the presented technologies, it is not part of the following performance comparison. It shall be noted that this is a major drawback in comparison to the previously proposed NFC-KE extension.

Nevertheless, current PKI setups also do not rely on a decentralized verification system, therefore legitimating this comparison of storage-only technologies.

5. Performance Comparison

To show the real-world benefits and problems of the previously proposed concepts, this section will contribute the following points:

- Uniform comparison basis for decentralized storage systems
- Real-world performance analyses of *IPFS <Private | Public>*, *Sia Skynet* and *HLF*.
- General and technology-specific implementation advice, based on performance results

When testing the private deployments, in this case, IPFS private and HLF, the setup can be individually created based on requirements or limitations, e.g., by the physical hardware. For this evaluation, the IPFS private setup is tested using two, four, and eight peers, respectively, and HLF in its setup with four orderers, two logical organizations, and two peers per organization, thus four peers in total. The latter configuration is chosen to establish a basis for comparison between the results of this analysis and the performance evaluation in [25].

To establish an experimental baseline, the same Signing Hub setup was chosen with the exact configuration as previously setup in [7,25], comprised of a Raspberry Pi 3B+ with an Infineon SLB9670 TPM 2.0 attached to it. Since the network in which the different Sensor Nodes and the Signing Hub operate is also a point of concern, this performance evaluation will consider a basic WiFi-based network using 802.11ac and an Commercial off-the-shelf (COTS) industrial access point. No special Quality of Service (QoS) presets or other performance-enhancing preferences were set or changed. All created peer instances were set up in a virtualized container environment using Docker. Finally, the Application Server was built and run using the following hardware specifications:

- **OS:** Arch Linux 64-Bit, Kernel Version 5.19.3-arch1-1
- **CPU:** AMD Ryzen 9 3900X, 12 Cores, 24 Threads, 3.80–4.60 GHz
- **RAM:** 31.3 GiB DDR4-3200, CL14-14-14-34
- **Network**: 1 GiB Ethernet Link

Since the NFC-KE can be separated into two distinct phases that each have different requirements in terms of filesystem access and availability, the performance analyses will also differentiate between *Phase 1* and *Phase 2*, where Phase 1 mainly consists of the

authentic public key exchange (cf. Figure 2) and Phase 2 covers the following authentic message exchange between the Signing Hub and the Sensor Node (cf. Figure 3).

5.1. Authentic Key Exchange-Phase 1

First, every previously described DLT and, if applicable, their different setups are compared in terms of execution times. Using the Boxplot representation shown in Figure 4 of the collected data, the mean values and standard deviations of the datasets can be extracted directly.

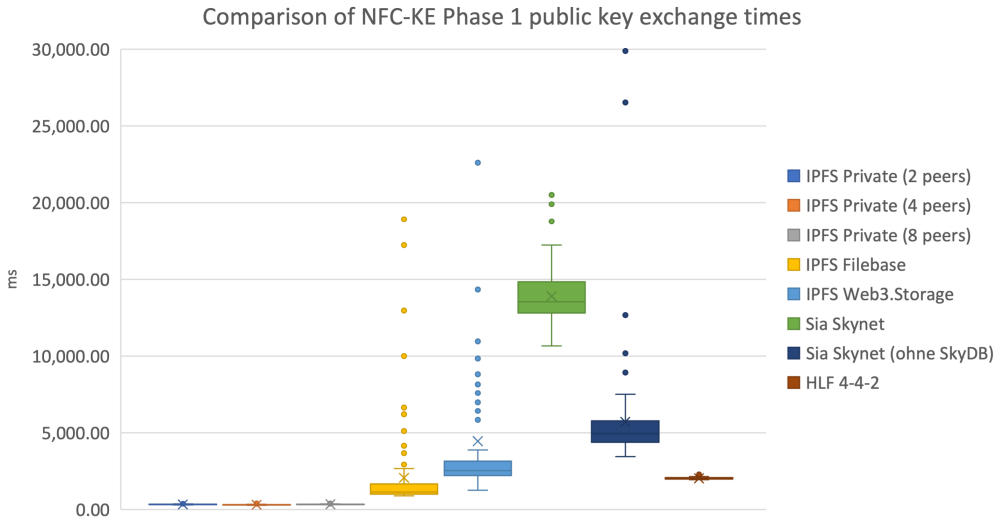

Figure 4. NFC-KE Phase 1 execution time comparison.

When compared with the other technologies, the *IPFS private* setup is executing the NFC-KE process the fastest. On average, the public key exchange happens within 286 to 334 ms, thereby not being influenced by the number of peers within the network. Additionally, the distribution of data points and the number of significant outliers is also lower when compared to the other DLT setups. IPFS Filebase is also executing the NFC-KE Phase 1 faster than the previously proposed HLF extension, although this setup experiences more outliers due to increased network jitter and API restrictions. On average, the authentic public key exchange has been completed within 992 and 1654 ms. The public IPFS setup based on Web3.Storage is significantly slower in comparison to the Filebase setup. With a mean completion time within 1921 and 2353 ms, the Web3.Storage setup is the slowest IPFS-based storage solution and also experiences the most outliers in direct comparison to the Filebase and private IPFS setups. The Sia Skynet setups share the last places among all tested setups in terms of execution times. Since the proposed setup uses the integrated SkyDB for public key query, two different API endpoints need to be contacted. To get a deeper understanding of the different execution times, we also included a setup without the SkyDB and thus without the public key query functionality. Using this setup, only one public key can be stored and queried which is only useful for testing. The results show that even without the SkyDB functionality, the Skynet-based approach is significantly more unstable and slower in comparison to all other DLT setups. With a mean execution interval of 4377 to 5746 ms, the Skynet solution does not offer any in-time storage solution. By adding the SkyDB backup functionality, and thus enabling the main NFC-KE Phase 1, the execution times range between 12.8 to 14.8 s. When compared to other setups, the outliers are less scattered in the latter setup case.

The stark performance degradation between the public and private setups is to be expected since the external APIs are highly reliant on the network performance. Furthermore, another main result of this particular comparison is the comparably fast execution times of the private IPFS solution in comparison to the HLF execution times. Even the

public Filebase approach is, on average, faster than the proposed HLF NFC-KE extension, although being more prone to outliers in the execution times and thus less reliable.

5.2. Authentic Message Sending-Phase 2

The second phase of the NFC-KE is, in terms of real-world impact, the most important since the corresponding execution times will directly influence the responsiveness of the system. Generally, Phase 2 is querying the file storage system for the given Sensor Node's public key to verify the message signature, originating from said Sensor Node. Therefore, lower execution times will allow for faster message processing. A significant differentiation needs to be made between the two phases because Phase 1 must be executed only once for each Sensor Node, whereas Phase 2 is executed repeatedly afterward. Thus, the impact of particularly bad Phase 1 execution times is less significant than comparatively high execution times in Phase 2.

For the performance comparison in Phase 2, the number of simultaneously sending Sensor Nodes influences the general performance of the Signing Hub. Therefore, the following evaluation will, additionally, include performance numbers for one to six simultaneously sending Sensor Nodes that are virtualized and sending data within an interval of one to two seconds. The following Figure 5 shows the mean execution times of each DLT scenario and also differentiates between each Sensor Node configuration.

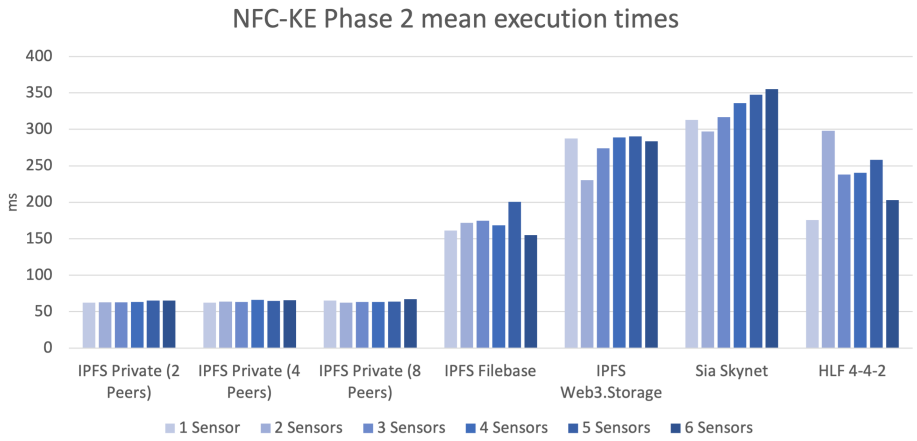

Figure 5. NFC-KE Phase 2 mean execution time comparison.

As can be seen, by the resulting mean execution times, the number of Sensor Nodes does, in this case, not influence the performance numbers in a significant way. In the case of the IPFS-based solutions, these are almost neglectable. Again, the private IPFS deployments offer the lowest execution times at almost 60 ms constantly. The Filebase setup is considerably faster than the HLF approach with a 150 ms mean execution time. The general pattern that already set itself apart in Section 5.1 does repeat itself in this case, since the Sia Skynet solutions also take the most time for Phase 2 execution with over 310 ms mean execution time. Note that for this comparison leaving out the SkyDB functionality like previously described in Phase 1 (cf. Section 5.1) does not apply here, since the message-sending part is inherently reliant on the public key query functionality. When comparing the IPFS-based solutions, only the Web3.Storage approach falls behind the Phase 2 execution times of the HLF-based approach.

To get a deeper understanding of the Sensor Node number's influences, Figure 6 shows the particular execution times of the IPFS private scenario using eight peers within the network.

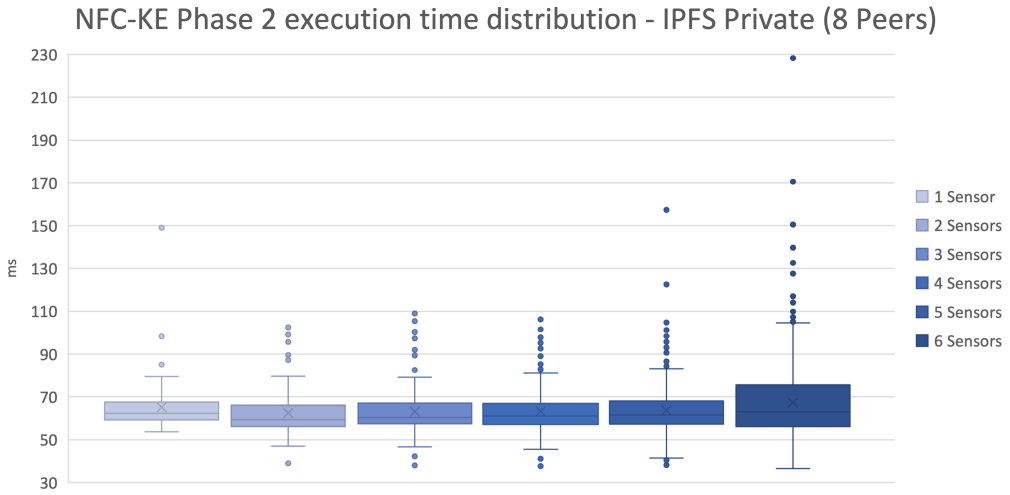

Figure 6. NFC-KE Phase 2 execution time distribution-IPFS private-8 Peers.

Here, the mean and median execution times remain almost constant among all tested Sensor Node numbers. The standard deviation and the number of outliers increase proportionally with increasing numbers of Sensor Nodes. Thus, the reliability of the system is slightly decreasing. This effect is only noticeable when visualizing the performance numbers for the private IPFS setup. Other DLT setups do not show the given behavior, due to other effects being more influential to the generally longer execution times such as latency, network bandwidth, and congestion.

6. Discussion

The results presented in Section 5 indicate that modern distributed file-storage systems, based on the DLT, are viable candidates for replacing centralized filesystem public key stores. Some technologies, especially those which can be set up in a private configuration, are capable of executing transactions significantly faster than, e.g., their public counterparts. Therefore, some specific technologies may not be suitable for every desired use case due to a lack of required performance.

In general, the performance numbers lead to the conclusion that the IPFS private setup is the most reliable and, in terms of performance, the most responsive solution for a high-performance system. Other implementation candidates on the IPFS basis have shown to be capable of being a proper alternative to the existing HLF NFC-KE extension because their performance numbers are either lower or equal to each other. Only the Sia Skynet solutions did not prove to be a valid replacement alternative for the HLF approach due to significantly higher execution times among all test scenarios. There are some additional points to the HLF performance results that need acknowledgement. First, the HLF implementaion requires a consensus among all participants to synchronize the underlying Blockchain state. Only after this process is complete, the stored data can be queried by external parties. Furthermore, HLF internally performs multiple cryptographical operations such as signing, verifying, and hashing transaction data, thereby increasing performance numbers. These points all add up to form the resulting performance of HLF.

When discussing the real-world application potential of the presented DLTs for the use within PKIs, the level of decentralization needs to be considered as well. The private variants of the presented DLTs are generally less decentralized and autonomous than the public variants. This is due to the restricted access to the given network since a private DLT instance is mostly created by one single administrator and then spread to known clients within the network, e.g., a local data center or multiple office computers. Therefore, decentralization is a point to consider since these network clients are, again, connected within the same network and may be located geographically close to each other. Nevertheless, a network administrator may be able to geographically decentralize private

IPFS peers, e.g., by using different data centers or renting a virtual server. This would, in return, require a higher maintenance effort and would not scale dynamically because of a single authority needing to manage all peers simultaneously. Due to their public access nature, the public DLTs offer global access to the underlying network and, thus, allow for increased redundancy and outage resilience.

There are also some points, that require attention when it comes to choosing a DLT for the PKI use case. DLT Platforms such as HLF or Ethereum [34] offer custom programmability and can be used as a decentralized, trusted execution environment that can enhance the trustworthiness of the given PKI. Each operation can be cryptographically signed and later verified by the DLT network. Enabling this distributed consensus renders the whole network more tamper-resistant and less prone to external intervention. In this particular case, the previously proposed NFC-KE extension using HLF used the integrated Smart Contract functionality to verify and store the exchanged public keys. This functionality had to be moved away from the DLT because the evaluated distributed file storages do not support a custom Smart Contract functionality, leaving the centralized Application Server the duty of verifying the received public keys. Therefore, the HLF approach offers increased security while not offering a true global decentralization or, in some cases, acceptable performance numbers.

Other security considerations arise when focusing on the internal structure of the private IPFS network in particular. Since the network peers are configured on a P2P basis and inform each other about changes in the internal network structure, any attacker who is able to infiltrate one IPFS peer is also capable of reconfiguring the entire private IPFS network. By using advanced concepts that are under current development, these issues can be remedied by employing different roles and responsibilities among the different IPFS peers. Therefore, further network and Operations Security (OpSec) measures need to be employed when setting up a private IPFS deployment.

In this article, only public APIs were considered for interacting with the IPFS or Sia network respectively. These technologies also offer the option to set up a custom peer instance which can connect to the given public network. For this, a distinct wallet containing cryptographic material needs to be stored and maintained, thereby adding new maintenance effort to the overall architecture. Though, by using such a configuration, latency and execution times are presumably lower since the connection to the custom peer, and thus to the whole P2P network, can be established without further routing. Additionally, by setting up one or more custom public IPFS peers, a pinning service would be required to take advantage of a public IPFS network. If such a pinning service is not used, no third-party IPFS peer will replicate the data due to lack of incentive.

Furthermore, the evaluation approach chosen in this article has been carried out in a controlled environment. In particular, the same hardware and physical network configurations were used for testing all aforementioned approaches. Additionally, the evaluation only focuses on examining the most significant KPIs that are relevant for a real-world application, in particular the execution times. It can be argued, that other indicators such as scalability, ease of use, availability or read/write latency times of the DLT are also of concern. Evaluating these KPIs was beyond the primary scope of this article.

7. Conclusions

This article compares four different distributed file storage technologies in different network configurations with each other and evaluates them in terms of performance and applicability for use within PKIs. The comparison is intended to assist developers and researchers alike to better decide on which technology is best suited for the intended use case. The most notable points to consider, when choosing a distributed file storage system are the content addressing paradigm as well as the lack of custom programmability.

The technical comparison of the storage system was conducted by comparing integrity, availability, persistency, and performance limitations of the several DLTs. The integrity of the data is warranted by either the CIDs or Skylinks. In this case, the private IPFS

setup has shown to be the most promising candidate for the use, e.g., within modern Industry 4.0 applications because of its custom setup, its significantly better performance in comparison to the public setup or other technology and, generally, its improved resilience. Further means to increase the overall network security must be employed by the network administrators.

The availability of the public DLT variants has shown to be considerably higher than the private counterparts, given the assumption that a private DLT instance is hosted in a geographically small space, e.g., a data center. Though, in terms of persistency, public DLT solutions require an incentive for each peer within the given network to persistently store the desired files. Since data storage is a paid resource on the networks, deletion of files is possible. The private DLT counterparts do not require such an incentive and, therefore, attain a higher level of data persistency.

The performance analyses of the different DLTs also indicate that the private IPFS setup is, at least for the intended use within PKIs, the most suited candidate if high performance is mandatory. Since it lacks any custom programmability, its feature set is limited to only the decentralized data store which may reduce the trustworthiness of the overall system. This feature is offered by the already existing NFC-KE HLF extension, although the performance of this approach is significantly lower than the private IPFS approach.

Future research may focus on combining the decentralized Smart Contract functionality of, e.g., HLF and decentralized file storage such as IPFS and evaluating the resulting performance. Following this approach, a PKI would benefit from the distributed consensus and the decentralized, redundant file storage. Additionally, focusing on purely DAG-based DLT technologies such as *IOTA*[35] can also benefit the overall scope of evaluating new DLT-based PKIs by changing the fundamental data structure of the filesystem. Since this article focused on the lightweight operation and low maintenance requirements of the overall system, future research may also evaluate the performance of our setup using the aforementioned custom public IPFS and Sia peers respectively.

Author Contributions: Conceptualization, F.H. and J.D.; data curation, F.H.; formal analysis, J.D.; funding acquisition, R.T.; investigation, F.H.; methodology, F.H.; project administration, R.T. and J.D.; resources, F.H. and J.D.; software, F.H. and J.D.; validation, R.T. and J.D.; visualization, F.H. and J.D.; writing—original draft, F.H.; writing—review and editing, J.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: All datasets can be found on Github via https://github.com/JulianD267/Tamper-Proof-Storage-Methods accessed on 27 October 2022.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

API	Application Programming Interface
CA	Certification Authority
CID	Content identifier
COTS	Commercial off-the-shelf
DAG	Directed Acyclic Graph
DHT	distributed hash table
DLT	Distributed Ledger Technology
ECC	error correcting code
HLF	Hyperledger Fabric
HTTPS	Hypertext Transfer Protocol Secure
IIoT	Industrial Internet of Things

IoT	Internet of Things
IPFS	Interplanetary Filesystem
KPI	Key Performance Indicator
MitM	Man-in-the-Middle
NFC	Near Field Communication
NFC-KE	Near Field Communication Key Exchange
OpSec	Operations Security
P2P	Peer-To-Peer
PGP	Pretty Good Privacy
PKI	Public Key Infrastructure
PSK	Preshared Key
QoS	Quality of Service
RTT	Round Trip Time
SDK	Software Development Kit
SME	Small and Medium Enterprise
TPM	Trusted Platform Module

References

1. Kfoury, E.; Khoury, D. Distributed Public Key Infrastructure and PSK Exchange Based on Blockchain Technology. In Proceedings of the 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Halifax, NS, Canada, 30 July–3 August 2018; pp. 1116–1120. [CrossRef]
2. Hepp, T.; Spaeh, F.; Schoenhals, A.; Ehret, P.; Gipp, B. Exploring Potentials and Challenges of Blockchain-based Public Key Infrastructures. In Proceedings of the IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Paris, France, 29 April–2 May 2019; pp. 847–852. [CrossRef]
3. Papageorgiou, A.; Mygiakis, A.; Loupos, K.; Krousarlis, T. DPKI: A Blockchain-Based Decentralized Public Key Infrastructure System. In Proceedings of the 2020 Global Internet of Things Summit (GIoTS), Dublin, Ireland, 3 June 2020; pp. 1–5. [CrossRef]
4. Schaerer, J.; Zumbrunn, S.; Braun, T. Veritaa-IoT: A Distributed Public Key Infrastructure for the Internet of Things. In Proceedings of the 2022 IFIP Networking Conference (IFIP Networking), Catania, Italy, 13–16 June 2022; pp. 1–9. [CrossRef]
5. Melo, W.; Machado, R.C.S.; Peters, D.; Moni, M. Public-Key Infrastructure for Smart Meters using Blockchains. In Proceedings of the 2020 IEEE International Workshop on Metrology for Industry 4.0 & IoT, Roma, Italy, 3–5 June 2020; pp. 429–434. [CrossRef]
6. Singla, A.; Bertino, E. Blockchain-Based PKI Solutions for IoT. In Proceedings of the 2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC), Philadelphia, PA, USA, 18–20 October 2018; pp. 9–15. [CrossRef]
7. Dreyer, J.; Fischer, M.; Tönjes, R. NFC Key Exchange—A light-weight approach to authentic Public Key Exchange for IoT devices. In Proceedings of the 2021 IEEE 7th World Forum on Internet of Things (WF-IoT), New Orleans, LA, USA, 14 June–31 July 2021; pp. 374–379. [CrossRef]
8. Ramesh, V.K.C.; Kim, Y.; Jo, J.Y. Secure IoT Data Management in a Private Ethereum Blockchain. In Proceedings of the 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC), Madrid, Spain, 13–17 July 2020; pp. 369–375. [CrossRef]
9. Uddin, M.N.; Hasnat, A.H.M.A.; Nasrin, S.; Alam, M.S.; Yousuf, M.A. Secure File Sharing System Using Blockchain, IPFS and PKI Technologies. In Proceedings of the 2021 5th International Conference on Electrical Information and Communication Technology (EICT), Khulna, Bangladesh, 17–19 December 2021; pp. 1–5. [CrossRef]
10. Aslam, F.; Javaid, N. Blockchain-based secure data sharing platform for research data rights management over the Ethereum network. In *This Work Is Submitted as Major Assignment for the Fulfilment of the Graduate Course, Research Methodology in Information Technology (RMIT)*; 2019. Available online: https://www.researchgate.net/publication/339989757_Blockchain-Based_Secure_Data_Storage_for_Distributed_Vehicular_Networks (accessed on 27 October 2022).
11. Hasan, H.R.; Salah, K. Proof of Delivery of Digital Assets Using Blockchain and Smart Contracts. *IEEE Access* **2018**, *6*, 65439–65448. [CrossRef]
12. Nizamuddin, N.; Hasan, H.; Salah, K.; Iqbal, R. Blockchain-based framework for protecting author royalty of digital assets. *Arab. J. Sci. Eng.* **2019**, *44*, 3849–3866. [CrossRef]
13. Park, J.S.; Youn, T.Y.; Kim, H.B.; Rhee, K.H.; Shin, S.U. Smart contract-based review system for an IoT data marketplace. *Sensors* **2018**, *18*, 3577. [CrossRef] [PubMed]
14. Protocol Labs, Inc. Interplanetary Filesystem. Technical Report. 2022. Available online: https://docs.ipfs.tech/ (accessed on 11 October 2022).
15. Protocol Labs, Inc. IOTA Wiki. Technical Report. 2022. Available online: https://wiki.iota.org/learn/about-iota/an-introduction-to-iota (accessed on 11 October 2022).
16. Protocol Labs, Inc. IPFS: Distributed Hash Tables (DHTs). Technical Report. 2022. Available online: https://docs.ipfs.tech/concepts/how-ipfs-works/#distributed-hash-tables-dhts (accessed on 11 October 2022).

17. Protocol Labs, Inc. IPFS: Directed Acyclic Graphs (DAGs). Technical Report. 2022. Available online: https://docs.ipfs.tech/concepts/how-ipfs-works/#directed-acyclic-graphs-dags (accessed on 11 October 2022).
18. Protocol Labs, Inc. Filecoin Docs. Technical Report. 2022. Available online: https://docs.filecoin.io/get-started/overview/ (accessed on 12 October 2022).
19. Filebase, Inc. What Is the Difference Between IPFS and Sia? Technical Report. 2022. Available online: https://docs.filebase.com/storage-networks/what-is-the-difference-between-ipfs-and-sia (accessed on 12 October 2022).
20. Linux Foundation. Channels. Technical Report. 2018. Available online: https://hyperledger-fabric.readthedocs.io/en/latest/channels.html (accessed on 12 October 2022).
21. Linux Foundation. Organization. Technical Report. 2021. Available online: https://hyperledger-fabric.readthedocs.io/en/latest/glossary.html#organization (accessed on 12 October 2022).
22. Linux Foundation. Peers. Technical Report. 2021. Available online: https://hyperledger-fabric.readthedocs.io/en/latest/peers/peers.html (accessed on 12 October 2022).
23. Linux Foundation. Smart Contracts and Chaincode. Technical Report. 2020. Available online: https://hyperledger-fabric.readthedocs.io/en/latest/smartcontract/smartcontract.html (accessed on 12 October 2022).
24. Linux Foundation. The Ordering Service. Technical Report. 2022. Available online: https://hyperledger-fabric.readthedocs.io/en/latest/orderer/ordering_service.html (accessed on 12 October 2022).
25. Dreyer, J.; Tonjes, R.; Aschenbruck, N. Decentralizing IoT Public- Key Storage using Distributed Ledger Technology. In Proceedings of the 2022 International Wireless Communications and Mobile Computing (IWCMC), Dubrovnik, Croatia, 30 May–3 June 2022; pp. 172–177. [CrossRef]
26. Dreyer, J.; Tönjes, R.; Aschenbruck, N. Towards securing Public-Key Storage using Hyperledger Fabric. In Proceedings of the 2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), Shanghai, China, 2–5 May 2022; pp. 1–3. [CrossRef]
27. Filebase Team. API Documentation. Technical Report. 2022. Available online: https://docs.filebase.com/api-documentation (accessed on 13 October 2022).
28. Protocol Labs, Inc. API Documentation. Technical Report. 2022. Available online: https://github.com/ipfs/js-ipfs (accessed on 13 October 2022).
29. Skynet Devs. Skynet Developer Guide. Technical Report. 2021. Available online: https://docs.skynetlabs.com/ (accessed on 13 October 2022).
30. Filebase Team. Introducing Support for IPFS, Backed by Decentralized Storage. Technical Report. 2022. Available online: https://filebase.com/blog/introducing-support-for-ipfs-backed-by-decentralized-storage/ (accessed on 13 October 2022).
31. Protocol Labs, Inc. IPFS Public Gateway Checker. Technical Report. 2022. Available online: https://ipfs.github.io/public-gateway-checker/ (accessed on 13 October 2022).
32. Protocol Labs, Inc. Web3.Storage Docs. Technical Report. 2022. Available online: https://web3.storage/docs/ (accessed on 18 October 2022).
33. Vorick, D. SkyDB: A Mutable Database for the Decentralized Web. Technical Report. 2020. Available online: https://blog.sia.tech/skydb-a-mutable-database-for-the-decentralized-web-7170beeaa985 (accessed on 18 October 2022).
34. Wackerow, P. Ethereum Development Documentation. Technical Report. 2022. Available online: https://ethereum.org/en/developers/docs/ (accessed on 18 October 2022).
35. IOTA Foundation. The Complete Reference for IOTA. Technical Report. 2022. Available online: https://wiki.iota.org/#core-libraries (accessed on 18 October 2022).