information

MDPI

Article

Monitoring Key Pair Usage through Distributed Ledgers and One-Time Signatures

Lucas Mayr *, Lucas Palma, Gustavo Zambonin, Wellington Silvano and Ricardo Custódio

Departamento de Informática e Estatística, Universidade Federal de Santa Catarina, Florianópolis 88040-900, Brazil; lucas.palma@posgrad.ufsc.br (L.P.); gustavo.zambonin@posgrad.ufsc.br (G.Z.); wellington.fernandes@posgrad.ufsc.br (W.S.); ricardo.custodio@ufsc.br (R.C.)

* Correspondence: lucas.mayr@posgrad.ufsc.br

Abstract: Private key management is a complex obstacle arising from the traditional public key infrastructure model. However, before any related security breach can be addressed, it must first be reliably detected. Certificate Transparency (CT) is an example of a certificate issuance monitoring strategy, developed to detect the possible malfeasance of certification authorities (CAs). To the best of our knowledge, CT and other detection mechanisms do not cover digitally signed documents made by an end user, which are also susceptible to CA misbehavior. We modify the CT framework to handle signed documents via logging certificates in the blockchain to enable the secure and user-friendly monitoring of one-time signatures, backdating protection, and effective CA misbehavior detection. Moreover, to demonstrate the feasibility of our proposal, we present distinct deployment scenarios and analyze the storage, performance, and monetary costs.

Keywords: Certificate Transparency; blockchain; digital signature; key management

Citation: Mayr, L.; Palma, L.; Zambonin, G.; Silvano, W.; Custódio, R. Monitoring Key Pair Usage through Distributed Ledgers and One-Time Signatures. *Information* **2023**, *14*, 523. https://doi.org/10.3390/info14100523

Academic Editor: Panayiotis Kotzanikolaou

Received: 23 August 2023
Revised: 14 September 2023
Accepted: 19 September 2023
Published: 26 September 2023

1. Introduction

Since its inception [1], the public key infrastructure (PKI) model has struggled with several challenges derived from the difficulties of managing private keys. Certificate revocation mechanisms are used to handle cases where a private key has been deemed unfit to sign any further messages. However, before a certificate can be revoked, it is imperative that the compromise is detected. Typically, due to the increased security of root and intermediate certification authorities (CAs), they are assumed to be secure enough such that this monitoring is unnecessary. Final CAs, those that issue end-user certificates, are known to be more easily compromised [2]. Hence, there exists a need to monitor certificate issuance by final CAs.

Systems that deal with the monitoring of issued certificates are usually named after the well-known Certificate Transparency (CT) protocol [3]. In this protocol, CAs submit "pre-certificates" to CT loggers, which respond with signed CT timestamps, acting as a promise that these certificates will be logged. Typically, logs are built as a Merkle tree and reconstructed with new certificates after some specified period; CAs typically embed these timestamps in the issued certificates as X.509 extensions. Systems that require CT timestamps only verify their respective signatures, while monitors and auditors check log integrity. Still, the addition of extra signatures can lead to certificate bloat, harming the efficiency of TLS protocols, depending on the signature algorithms chosen by loggers.

While CT manages to monitor the issuance of TLS certificates with some degree of success, adapting it to track end-user documents would require extensive changes in document signature generation processes, requiring end users to log every created signature by themselves. A widespread PKI environment is expected to have laypeople as the largest portion of its user base and cannot rely on the assumption that non-technical certificate holders would follow the required steps to register signed documents. Imposing extra steps in the creation of digitally signed documents ultimately hurts usability and restrains

adoption. Therefore, we argue that any proposal based on the CT framework must be transparent to the end user.

Furthermore, the addition of multiple extra signatures on the certificate must be addressed when future proofing any CT-like protocol. For instance, with the advent of quantum computing, classic signature algorithms such as RSA and ECDSA are fated to be replaced. Active research is being conducted on quantum-resistant cryptographic algorithms. However, the consequences of quantum resistance are an increase in the public key and signature sizes [4], which greatly affect CT artifacts. Hash-based signatures are an example of quantum-resistant schemes with small public keys and signature sizes [5], which are desirable properties when considering storage requirements.

Hash-based signatures rely exclusively on the pre-image resistance of its component cryptographic hash function and provide a framework to construct one-time signature schemes (OTSs) [6]. It is strongly recommended that key pairs from OTS schemes are used to generate a single signature since it is derived directly from a private key. In the context of a PKI framework, this implies that a private key need not be stored after being used once. Thus, we argue that OTS schemes are an important asset in solving private key management issues, relieving the end user from holding sensitive information and consequently improving usability. This security restriction is monitored via a CT-like framework to assert that documents are not signed with the same private key.

Contributions. We propose an augmentation to the traditional PKI model, which consists of logging end-user certificates and hashes associated with digitally signed documents in a blockchain. Any logging must be made by CAs to relieve laypeople of such responsibilities, increasing the usability of end-user certificate holders. As a consequence, our model requires a binding between certificate issuance and documents, i.e., a digital certificate unique to each digitally signed file. This is performed via the creation of new key pairs for each signed document or simply through the use of OTS schemes. Such bindings are then registered as transactions in the blockchain.

We achieve the effective monitoring of digital certificate issuance and creation of digitally signed documents, detecting possible malfeasance from CAs and/or the compromise of user credentials, and preventing the backdating of digital signatures. Additionally, our strategy is quantum-resistant, considering a suitable choice of signature schemes throughout the CAs, end users and blockchain networks. The distributed ledger technique prevents effective private key reuse, solving the largest practical restriction of an OTS scheme. We discuss the privacy and security implications of our proposal, present public and private blockchain models as alternative implementations, and analyze the storage and performance requirements considering a realistic PKI scenario.

Related work. Blockchain implementations of the CT protocol have been proposed with some improvements. Madala et al. [7] introduce an implementation of CT, where CAs must receive consent from the domain owner before a certificate can be issued. Their implementation is able to revoke certificates in the blockchain. In a similar manner, BB-PKI [8] requires that registration authorities (RAs) reach a consensus on the issuance of a certificate, which is then signed by multiple CAs through an out-of-band channel to lower the chances of successful impersonation attacks against RAs.

Blockchain-based PKIs have been proposed as an alternative to the traditional model. In these proposals, CT is assumed, as most PKI artifacts are committed to the blockchain as part of the process of issuing certificates. However, most proposals simply replicate the same procedures of a traditional PKI, opting to simply log its operations without much disruption to the usual procedures. However, some proposals create conditions for the issuance of digital certificates. Kubilay et al. [9] require that a consensus be reached through a dynamic threshold signature scheme before a certificate is issued. Another PKI implementation that aims to block certificate misissuance before it happens through the blockchain is the Pistis [10] framework. In this framework, certificate issuance is performed through the blockchain only when domain ownership is verified.

Some proposals include misissuance prevention as a feature of signature verification applications. In other words, verifiers can set conditions for when a signature can be deemed valid. For instance, Han and Hwang [11] introduced a trust model dubbed "conditional trust". Certificates with many trusted signatures are seen as more trustworthy than those with fewer signatures. This proposal utilizes the blockchain to communicate among many CAs and RAs. However, an important observation of models that embed many signatures in a certificate is that it may grow indefinitely with time.

The intersection of blockchain and OTS limits itself to using such schemes to improve aspects of blockchain as a replacement for classical digital signature algorithms. Most works set hash-based algorithms as the de facto OTS scheme. Vives [12] improves efficiency by replacing ECDSA signatures with a single hash-based signature to enable post-quantum authentication in the blockchain. Hyla and Pejaś [13] show how blockchains can create a beneficial environment to the archiving of digital documents by acting as a timestamp. They propose the usage of XMSS to increase blockchain longevity in the post-quantum scenario.

The Key Compromise Resilient Signature [14] (KCRS) system includes the end-user signature in the list of PKI artifacts that it logs in the blockchain. In this scheme, users have multiple key pairs, a master key pair certified by a digital certificate, and multiple "signing key pairs". The user derives the signing key pairs from the master key to sign symmetric key-encrypted messages sent through an out-of-band secure channel by the verifier and log the resulting signature in the blockchain. Thus, only the verifier and the signer know that the end user has generated a signature. This scheme loses the monitoring qualities of CT by weakening the link between the user and the asymmetric key pair.

To the best of our knowledge, there are no published works that deal with the private key reuse of OTS schemes by limiting the number of signed documents that can be validated. We note that most proposals draw heavily from the CT protocol and consequently focus on the perspective of TLS certificates. Research on the impact of post-quantum algorithms, certificate logging and monitoring, and usability for the signing of digital documents by laypeople is severely lacking.

2. Background

2.1. Cryptographic Hash Functions

Such functions are used to detect changes to given inputs, for instance, due to a noisy channel or malicious alterations in transit. We note that the hash of a message may also be called a *digest* or *fingerprint*.

Definition 1 (Ref. [15], Sec. 9)**.** *A function* $\mathcal{H} : \{0,1\}^* \to \{0,1\}^n$ *is a* cryptographic hash function *if it respects the following set of properties:*

(i) Performance*: it is easy to compute* $\mathcal{H}(x)$*.*

(ii) One-wayness*: it is computationally hard to find any* x *such that* $\mathcal{H}(x) = y$*, for some* y *picked at random from the range of* $\mathcal{H}$*.*

(iii) Pre-image resistance*: it should be computationally hard to find a original message* m *given its digest* y*, i.e., to find any* m *such that* $H(m) = y$*, where* y *is known.*

(iv) Second pre-image resistance*: it should be computationally hard to, given a message* m*, find another distinct message* m' *that generates the same digest* y*, i.e., to find* m' *such that* $H(m) = H(m')$*, where* m *is known.*

(v) Collision resistance*: it should be computationally hard to find any two distinct messages that output the same digest, i.e., to find any* m *and* m' *such that* $H(m) = H(m')$ *and* $m \neq m'$*.*

2.2. Digital Signature Schemes

Digital signatures are classic examples of how public key cryptography can be employed. Such algorithms provide three main properties: authentication, which is the validity of the identity information related to an entity; integrity, which is the guarantee

that the signed information has not been modified in transit; and non-repudiation, which is the assurance that the signer cannot deny having signed a given message. Oftentimes, the actual message being signed is the digest of the original message. A formal definition is given below.

Definition 2 ([16]). *Consider a security parameter* $\lambda \in \mathbb{N}$ *and any message* $m \in \{0,1\}^*$. *A* digital signature scheme *is a triple of probabilistic polynomial-time algorithms: (i) the* key generation $\textsc{Gen}(1^\lambda) \to (\mathsf{sk}, \mathsf{pk}) \in (\{0,1\}^* \times \{0,1\}^*)$, *which outputs private and public keys, bound by a predefined property; (ii) the* signature generation $\textsc{Sig}(\mathsf{sk}, m) \to \sigma \in \{0,1\}^*$; *and (iii) the* signature verification $\textsc{Ver}(\mathsf{pk}, m, \sigma) \to \{0,1\}$, *all of which must satisfy* $\Pr[\textsc{Ver}(\mathsf{pk}, m, \textsc{Sig}(\mathsf{sk}, m)) = 1] = 1$ *for any* $(\mathsf{sk}, \mathsf{pk})$.

Signature Policy

A signature policy is a set of rules that must be followed by both signers and verifiers when generating digitally signed documents, indicating which certificate extensions and artifacts must be included in the final signature for it to be correctly verified. Thus, before generating a signature, the user implicitly or explicitly selects the appropriate policy in relation to its intended use. Electronic signatures that follow signature policies are called advanced electronic signatures (AdESs) [17]. Hereafter, we consider advanced signatures as defined in ETSI EN 319 102-1 [18].

2.3. Winternitz Signature Scheme

Winternitz is an OTS scheme whose signatures are obtained from multiple applications of a cryptographic hash function $\mathcal{H}$ over the private key. We give our definition of the scheme based on [19] for readability, but we observe that there exist variants with fewer security requirements [20].

Consider $\lambda \in \mathbb{N}$ as the security parameter, $n \in \mathbb{N}$ to be the message length, a cryptographic hash function $\mathcal{H} : \{0,1\}^* \to \{0,1\}^n$, any message $m \in \{0,1\}^n$, and $w \geq 1$ to be the Winternitz parameter, usually a power of two, which represents the number of bits to be signed in parallel. Let $t_1 = \lceil \frac{n}{\log_2 w} \rceil$, $t_2 = \lfloor \frac{\log_2 t_1 (w-1)}{\log_2 w} \rfloor + 1$ and $t = t_1 + t_2$. We define $f^i(x)$ to be $f(f(\ldots f(x)\ldots))$ applied i times, with $f^0(x) = x$. We set $\lambda = n$ for practical purposes. We present the algorithms that form the signature scheme below.

$\textsc{Gen}(1^\lambda)$. Generate $\lambda \cdot t$ random bits and set the private key to be t random words of equal size, i.e., $\mathsf{sk} = (x_{t-1}, \ldots, x_0)$. The public key is $\mathsf{pk} = (\mathcal{H}^{w-1}(x_{t-1}), \ldots, \mathcal{H}^{w-1}(x_0))$, or the result of repeated applications of the cryptographic hash function to each element of the private key.

$\textsc{Sig}(\mathsf{sk}, m)$. Take m and slice it into a tuple with t_1 elements of w bits $B_1 = (b_{t-1}, \ldots, b_{t-t_1})$. (Without loss of generality, we assume $w \mid n$ such that no padding is required.) Calculate $c = \sum_{i=1}^{t_1} (w - 1 - b_{t-i})$ and slice it into an analogous tuple with t_2 elements of w bits $B_2 = (b_{t_2-1}, \ldots, b_0)$. Finally, let $\sigma_i = \mathcal{H}^{b_i}(x_i)$ for $0 \leq i \leq t-1$, and the signature is $\sigma = (\sigma_{t-1}, \ldots, \sigma_0)$.

$\textsc{Ver}(\mathsf{pk}, m, \sigma)$. Obtain B_1 and B_2 from m as per the procedure above. Then, the signature is valid if $\mathsf{pk} = (\mathcal{H}^{w-1-b_{t-1}}(\sigma_{t-1}), \ldots, \mathcal{H}^{w-1-b_0}(\sigma_0))$, and invalid otherwise.

The sizes of the public keys and signatures are $n \cdot t$. For public keys, the size can be reduced to n by hashing the concatenation of all public key elements in a predefined order.

2.4. Blockchain

Blockchain is a distributed database maintained by a decentralized network of nodes. Commonly, in blockchain implementations, every node in the network keeps a copy of the blockchain. To keep the copies cohesive, nodes must agree on the state of the blockchain by executing a consensus protocol. From a data structure perspective, we assume a blockchain is a linked list of blocks, with every block comprising a header and a list of transactions. Headers maintain metadata about the block and a hash pointer to the previous block, while the list of transactions keeps the data shared by nodes in the network.

Bitcoin [21] and Ethereum [22] are public blockchain networks, meaning that virtually anyone can integrate the network. Also, both are permissionless networks since no authorization mechanism regulates who can write and read data in the blockchain. In other words, any node in these networks can propose new transactions and read any data stored in the chronological list of blocks. However, we note that public and permissionless networks may only work for some blockchain use cases. For instance, a use case may exist where only authorized nodes should be part of the network, and just a subset of them should have writing permissions.

For such use cases, one can count on private and permissioned blockchains. Hyperledger Fabric is an outstanding example of such networks. It is part of the Hyperledger initiative [23] maintained by the Linux Foundation. In contrast to Bitcoin and Ethereum, Hyperledger Fabric does not have a cryptocurrency or a public network. Instead, one can deploy a private network where digital certificates identify nodes, and the policies regulate which nodes have the rights to write and read data.

3. The OTSCHAIN Framework

To solve the issue of effectively monitoring digitally signed electronic documents and, consequently, key pair usage, we propose a blockchain-based strategy where final CAs share a distributed ledger of issued digital certificates and message hashes. In our strategy, multiple compliant PKI infrastructures contribute to the same blockchain network since users may have one or more digital certificates issued by unrelated CAs. To improve the authenticity and non-repudiation of transactions, we propose to identify CAs in the network with the same key pair they use to issue digital certificates. As a result, CAs can read and write data on the blockchain. When reading data, CAs and other interested parties, such as end users and signature validation applications (SVAs), act as monitors. To properly validate a signed document, our strategy requires a connection to the blockchain in addition to the usual access to certificate chains and revocation data by the SVA.

We require the use of one-time signature schemes in our framework to improve usability for end users such that private keys need not be stored after the first document is signed. Additionally, we are able to prevent effective repeated OTS private key usage via blockchain semantic validation, addressing the main constraint of such schemes. A direct consequence of our restriction is the higher load for PKIs, particularly in the certificate issuance and the key generation step. We argue that hash-based OTS schemes have efficient key generation when considering sufficient and available entropy, compared to classic methods and even other quantum-resistant alternatives; only random bits are needed, with no additional constraints imposed [24].

Certificates used in the context of the CT protocol may include a certificate extension α of the signed timestamp response from the CT logs. Analogously, the usage of OTSCHAIN may also be identified by the presence of an extension β embedded into the certificate. While α generates unwanted certificate bloat, as it contains one or more digital signatures, β may hold a short list of ledger URIs, or it can be empty when such information is already known via out-of-band mechanisms. Therefore, we are able to minimize the overhead caused by CT-like proposals while maintaining the desired properties.

Any legislation that recognizes advanced signatures similarly to the European standard ETSI EN 319 102-1 [18] is able to use this proposal without any changes to their regulations. That is, advanced and qualified signatures and derived requirements are agreed upon by all parties involved in processing digitally signed documents. Notably, the Brazilian civil registry PKI allows the use of any certificate extension, and is currently using a proof-of-concept version of OTSCHAIN within their existing legal frameworks.

We define the entities and data that belong to the protocol for any $i, k \in \mathbb{N}$. The actors are a certification authority CA_i, an end-user U_i, and a CT-like monitor M_i. Data inserted in the blockchain are denoted as follows: a message hash H_i to be signed by U_i, a digital certificate C_i carrying a set of attributes $\alpha_i = (\alpha_i^1, \alpha_i^2, \dots)$, owned by U_i and issued by CA_i, and a tuple $\mathsf{L}_i = (\mathsf{C}_i, \mathsf{H}_i)$ linking certificate and message hash. Finally, we

define the blockchain elements: an asset A_i carrying a tuple L_i, a transaction payload P_i encoding A_i, the signed payload σ_{P_i}, a transaction request $R_i = (P_i, \sigma_{P_i})$, a transaction output $O_i = (A_1, A_2, \dots)$ (or $O_i = \emptyset$), a transaction T_i, and a block B_k containing T_i. The flowchart in Figure 1 depicts the interactions among these entities.

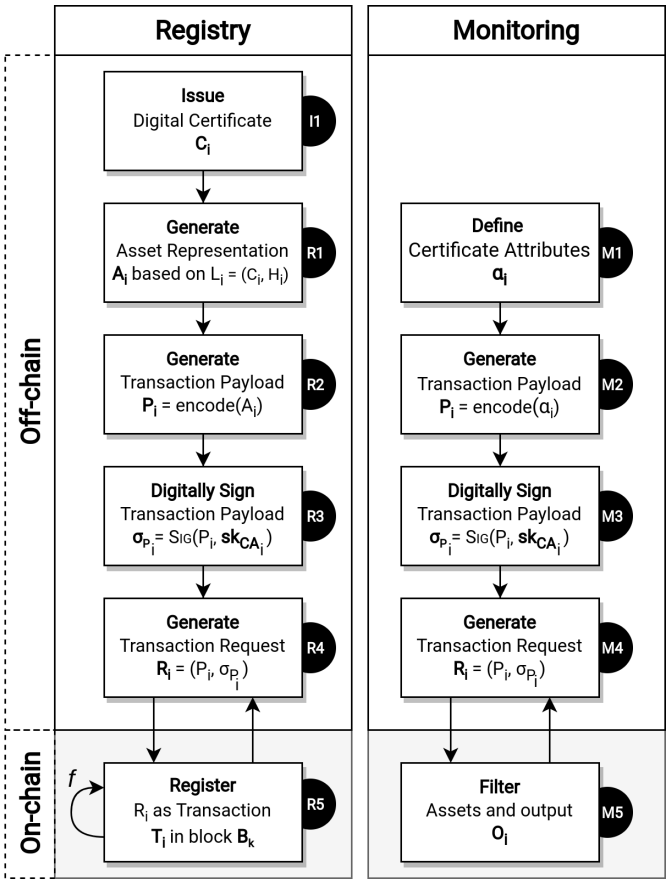

Figure 1. Flowchart of the registering and monitoring steps of OTSCHAIN.

In the context of the blockchain network roles, we propose a network, where every PKI can maintain a copy of the blockchain. Furthermore, root and intermediate CAs are not required to be part of the network or publish data since our proposal focuses on end-user certificates. However, every final CA is encouraged to maintain a copy of the blockchain and engage in the consensus protocol. SVAs should be nodes in the network as well, as they act as monitors, querying the existence of assets A_i to validate the document corresponding to H_i. Additionally, having a local copy of the blockchain speeds up document validation, which brings further usability to end users.

The left side of Figure 1 illustrates the general steps toward registering digital certificates in the blockchain. We recall that CA_i is the final CA that issues C_i. Step I1 in the flowchart represents the certificate issuance, which does not depend on the active blockchain network. The message hash H_i is obtained via an out-of-band mechanism, for instance, via the signature creation application. We remark that there are no interventions in the traditional certificate issuance procedures; it is merely depicted for completeness.

The first step R1 of the registry process consists of the generation of an asset representation A_i of the end-user digital certificate and message hash by the CA. Then, CA_i is ready to generate a transaction payload P_i in step R2, encoding the asset, using any format required by the blockchain network. Subsequently, in step R3 of the protocol, CA_i signs the transaction payload using its private key, which enables the generation of the transaction request R_i in step R4. Previous steps execute outside of the blockchain network

and culminate in step R5, which consists of storing the data as a transaction to be consumed in the monitoring process at a later time.

We note that between steps R_4 and R_5, the transaction proposal must be broadcast by CA_i, so it reaches the other nodes in the network, making the copies cohesive. The consensus protocol by which nodes will agree on the new state of the blockchain depends on the arrangements between nodes. For instance, they may agree on a majority policy, meaning that the data are stored in the blockchain if more than 50% of the nodes agree on the transaction request. We present a detailed discussion about deployment recommendations and measurements in Section 4.

Let ℓ be the set of all possible L_i. Step R5 contains the application of a function $f : \ell \to \{0, 1\}$. Consequently, our framework may be customized to check certain semantic restrictions. Examples of such verifications are (i) how recently the certificate was issued to prevent backdating; (ii) the certificate issuance rules according to the content of α_i, for instance, allowing only certain organizational units to participate in the network; and (iii) restricting node operations, for example, to impose upper bounds on attempts to register certificates by a faulty CA. The transaction fails if f returns a false response, which prevents the insertion of T_i in any block. We opt to register the offending request to improve the detection of malpractice and possible attacks.

In our proposal, f is set to control additional registration of certificates containing a previously logged public key. As previously discussed, we move log information out of the certificate itself to the blockchain. We require compliant SVAs to check the blockchain for the existence of the correct tuple L_i. In this manner, even if an OTS private key is used multiple times, only the registered hash associated with the submitted signed document will be valid under this policy.

After the registering process, A_i is available to be read by any node with permission, which enables the start of the monitoring process. The right side of Figure 1 details such procedures. In step M1, a monitor M_i defines a set of attributes α_i to search. Such attributes are encoded in a transaction payload in step M2 and digitally signed in step M3 to create a transaction request in step M4, analogous to the register operations. Finally, in step M5, the network consults the set ℓ for a matching L_i, and returns the suitable asset list to M_i.

We emphasize that monitors are independent; further external validation by these entities, such as the processing of specific documents, is, by design, out of the scope of our framework. Such behavior is particular to how a monitor operates and which services they offer. Moreover, as we require the registration of every certificate in the blockchain to properly validate a document, monitors are able to detect and warn users of every certificate that has been issued in their name. In other words, whenever an attacker uses a compromised credential, it must inevitably reveal itself by having the resulting certificate registered in the blockchain. Thus, monitors and end users may take appropriate steps to deal with this security breach as soon as possible.

Our strategy is suitable for a wide range of use cases, especially when long-term registry of signed documents is a hard requirement. For instance, consider the issuance of degree certificates by educational institutions, which may be subject to fraud [25]. In this case, as long as the blockchain network is online, students and other interested parties can validate a degree certificate by checking the registry of the signature on the OTSCHAIN. Moreover, since our framework ties any digital certificate to a single document, compromised private keys would not allow the issuance of valid degree certificates. Furthermore, through the use of quantum-proof OTS schemes, degrees can be protected from quantum attackers long before they become a real threat.

3.1. Privacy Impact

As our proposal requires the inclusion of entire end-user certificates in a blockchain, we must consider the ramifications of facilitating access to attributes contained in such certificates, considering data protection laws such as the EU GDPR [26] and the Brazilian LGPD [27]. We discuss the differences between registering TLS and end-user certificates,

the intrinsic trade-offs between privacy and monitoring services, the differences between a public and private blockchain privacy-wise, and how to use OTSCHAIN when pseudo-anonymity is desired.

Bernabe et al. [28] proposes eleven privacy-preserving challenges and four general solutions. We highlight "non-erasable data and on-chain data privacy" as the most relevant challenge to our proposal. Since blockchains tend to be immutable databases, the authors claim that even encrypted or hashed data are unsafe in the face of broken cryptographic primitives. That is even more complicated when referring to public blockchains, where data are publicly accessible by virtually anyone.

In this context, some blockchain-based implementations have proposed to store only public data in the blockchain and maintain private counterparts in an external database, such as the InterPlanetary File System (IPFS) [29]. Moreover, some private blockchain networks present an integrated solution to the problem. For instance, Hyperledger Fabric offers "private collections", where nodes maintain the public part of the data in the blockchain and the private portion on a local database, where they can control which other nodes in the network may have access.

We remark that there exists a natural trade-off between privacy and transparency in our proposal due to the generation of evidence that shows the intent behind the signing of a document. Thus, if a signed document is contested in court, a robust monitoring system may help to prove that the signature was indeed generated by the original certificate holder. This logging framework can be augmented to include further user data to increase the amount of evidence for the receiver, such as IP addresses and identity provider tokens. However, we note that privacy is decreased with the amount of evidence provided.

Conversely, techniques that are able to obfuscate the data contained in the certificate, such as zero-knowledge proofs [30], or logging the hash of the certificate instead of itself, are able to improve privacy at the cost of monitoring capabilities. We note, however, that such techniques might damage monitoring to a degree where misissuance can no longer be detected and should be carefully considered before adoption in the context of our proposal.

Ultimately, digital certificates associate a key pair to an entity and thus are seen as public documents in the traditional PKI model. Logging certificates through our proposal facilitates document spread, a desirable property in many PKI implementations. We observe that the link between the entity and signature is made through the digital certificate itself; logging mechanisms only assist in spreading the public certificates. Thus, we argue that a PKI that requires pseudo-anonymity and wishes to incorporate logging and monitoring services should ensure anonymity directly in the certificates. While pseudo-anonymity may harm monitoring effectiveness, multiple uses of the same OTS private key are still prevented. Furthermore, if the pseudonym used in the certificate can be traced back to the user by a trusted party, monitoring is provided on a smaller scale.

PKI architects are free to choose the best way to implement the OTSCHAIN framework according to the particular purposes of the models. Infrastructures that wish to keep end-user information private can opt for a mix of permissioned blockchain and digital certificates with pseudonyms. Alternatively, those that do not wish to or cannot create their private blockchain can utilize pre-established blockchains, such as Ethereum, to increase the spread and monitoring capabilities of their logs. Finally, as data protection regulations vary wildly among countries, there must be considerable care in logging implementation pursuant to local laws.

3.2. Security Benefits

In this section, we discuss how OTSCHAIN facilitates the administration of several scenarios where participants misbehave, helping to detect or outright prevent situations in which the security of a PKI may be compromised. We remark that there are two main incentives for participants to behave in the traditional PKI model. Trustworthy participants have increased financial benefits, as opposed to those who have misbehaved, as the PKI is ultimately built on trust. Additionally, the threat of law action may be imposed by

regulators if misbehavior is frequent or severe enough. We list below several situations addressed more effectively by our model compared to a traditional PKI.

CA misbehavior. OTSCHAIN is built on top of the CT framework; consequently, it has the same intrinsic set of benefits as conventional implementations of CT, i.e., the detection of certificate misissuance by CAs. Those that are found to be misbehaving can be, in addition to any legal action, revoked in the traditional sense and, in the case of blockchains, have their write permissions removed from the blockchain network. A CA is deemed malicious if it does not follow the guidelines established by the protocol: (i) it does not register any binding L_i when a signature is generated; or (ii) registers L_i with incorrect information.

We recall that, for a document to be validated using our proposal, a SVA must obtain the L_i tuple containing the link between the message hash and the matching digital certificate. Thus, refusing to register a certificate, logging the wrong certificate, or failing to produce the correct hash all lead to an invalid signed document. We argue that adapting existing SVAs to be pursuant to OTSCHAIN can be done efficiently, as it amounts to querying a blockchain network and performing a small number of additional checks.

Our proposal may also be extended to broader threat models. We consider signature backdating attacks, in which rogue CAs publish signed claims allegedly made in the past but never before disclosed. To address this situation, only a small modification to the function f is necessary: certificates are required to have their issuance time attribute be "close enough" to the logging time. Indeed, this is crucial when protecting classical CAs from quantum attacks in the event that signed artifacts are held until a conventional signature algorithm becomes insecure.

Monitor misbehavior. A third-party rogue monitor may opt to respond to queries about certificates and documents wrongfully, to individually prevent the validation of a document, or to collude with a malicious CA to validate a document that has not been correctly logged in the blockchain. In OTSCHAIN, SVAs are free to either (i) choose a trustworthy monitor, (ii) query multiple monitors to form a consensus, or (iii) become a monitor themselves to ensure the correct response to a query.

End-user private key compromise. A major benefit of our proposal, when compared against traditional CT implementations, is that it includes the tracking of end-user documents in addition to CA actions. Particularly, we are able to detect when end-user credentials are stolen without any input from the certificate holder whatsoever. In this manner, we are able to increase protection to the most exposed and broadest class of entities in a PKI while still preserving usability.

4. Deployment and Measurements

This section presents two deployment scenarios for our proposal and briefly discusses the storage, performance, and monetary costs of each scenario. We first consider a private and permissioned blockchain, where only authorized nodes can read and write data. In the second scenario, we consider a public and permissionless alternative, where virtually anyone can participate. In both scenarios, we propose the deployment of a smart contract called OTSchain-SC, where nodes can read and write data in the blockchain, respectively triggering the registering and the monitoring procedures detailed in Section 3.

4.1. Private Blockchain

The Hyperledger Fabric platform is chosen as the blockchain of the first scenario. In this environment, participants cooperate as a consortium of CAs and other interested parties to deploy a private blockchain network. Thus, participants agree on the definition of OTSchain-SC, which specifies the function f and how assets are read and written. Consider id_i as a unique identifier of the binding L_i. We set id_i to the digest of the public key pk_i in the certificate C_i, as we are interested in checking the existence of any L_i that already contains pk_i. Then, the tuple $(\mathsf{id}_i, \mathsf{L}_i)$ is implemented as an asset of OTSchain-SC. We store certificate data in the well-known DER encoding to make full use of smart contract programming

languages available in Hyperledger Fabric (e.g., Go and Java), which already have robust libraries for handling X.509 certificates.

Moreover, owing to the permissioned aspect of Fabric, we restrict which nodes are able to start the registration procedure, as to only allow CAs to log data in OTSCHAIN. This registration process will necessarily trigger the most important procedure of the smart contract, which is the execution of function f. To execute the monitoring phase of the protocol, participants query the blockchain by calling the OTSchain-SC monitoring procedure, which, in our implementation, accepts a set of attributes submitted as JSON-encoded key–value statements. For instance, to obtain the list of certificates issued by Brazilian CAs, a monitor must send the statement `{"issuer":{"C":"BR"}}` to the monitoring function, which returns a list of filter-matching assets.

4.2. Public Blockchain

For the second deployment scenario, we propose to use the public and permissionless Ethereum blockchain. This network allows the deployment of decentralized applications, "dapps", using the Solidity programming language. In this case, the network does not identify participants nor enables default restrictions of which nodes can read or write data in the blockchain. Thus, any participant in the Ethereum network would have access to data stored in OTSCHAIN by default; indeed, assets in this scenario are entirely public. Nevertheless, it is possible to restrict the logging of L_i to CAs by defining a set of addresses allowed to execute the registering procedure in OTSchain-SC smart contract.

In other words, even in the public implementation of our proposal, it is possible to restrict the register operation only to trusted CAs. The only requirement is that every CA would need to previously inform the smart contract of its address in the Ethereum network. The register procedure in the public version creates the same kind of asset defined in the previous scenario. However, the monitor function works differently from the Hyperledger Fabric version since Solidity does not support standard libraries to process X.509 certificates. Consequently, all such management routines must be implemented as native functions in OTSCHAIN.

4.3. General Implications

A critical aspect to consider when choosing a blockchain network is the positive and negative impacts of the underlying consensus protocol. In the literature, authors have already investigated the high energy consumption of consensus protocols [31], such as proof of work, which is adopted by a wide range of blockchain networks like Bitcoin [21]. However, this is not necessarily the case for the current default implementation of, for example, Ethereum and Hyperledger Fabric. The first recently migrated to a proof-of-stake consensus protocol, and the second uses a crash fault-tolerant protocol called Raft [32].

Furthermore, blockchain technology is constantly evolving, which means that any OTSCHAIN deployment, public or private, may require an update in the future, either to avoid known breaches of current blockchain network implementations or to take advantage of new features. In this context, our strategy could be easily adapted to any other blockchain network with support for smart contracts since it does not require any particular feature that only exists in the recommended deployment scenarios, i.e., Ethereum and Hyperledger Fabric. In the migration process, in order to share data between networks, data-sharing strategies and sharding [33] may be employed.

4.4. Discussion

We now turn to the practical consequences of OTSCHAIN. Storage costs are heavily dependent on the number and size of issued certificates. In our model, the number of certificates issued is exactly the number of digital signatures generated. Therefore, we estimate storage costs according to such usage statistics. To better reflect the feasibility of our framework, we consider several quantum-resistant signature schemes for CAs as chosen by NIST in its standardization process [34] and two real-world PKIs as the basis of

our analysis: Brazil, with approximately $\sigma_{\mathsf{low}} = 2^{24}$ signatures per year [35]; and Estonia, with approximately $\sigma_{\mathsf{high}} = 2^{27}$ signatures per year [36].

The size of a single tuple $\mathsf{L}_i = (\mathsf{C}_i, \mathsf{H}_i)$ depends on the algorithm choice. We hereafter consider Winternitz to be the choice for end users due to the previously mentioned benefits. Winternitz has a public key size of $\frac{n}{8}$ bytes; let ψ be the size of the signature made by the CA. We let κ be the certificate "header" size in bytes, i.e., any attributes and necessary encoding, apart from the public key and signature. Then, the size of L_i is precisely $(\kappa + \psi + \frac{n}{8}) + \frac{n}{8}$ bytes. By separating the public key, signature, and other data in C_i, we are able to estimate the size of L_i considering several scenarios for κ and ψ. We define $\kappa_{\mathsf{min}} = 128$ for a minimal certificate with barely any semantic information, $\kappa_{\mathsf{avg}} = 512$ for a certificate with few attributes, and $\kappa_{\mathsf{max}} = 1536$ for a certificate with liberal use of extensions, attributes, and names. Our findings are summarized in Table 1.

Table 1. Uncompressed storage size estimation of a blockchain containing bindings between certificates and hashes, in GB/year, considering various security levels n, several quantum-resistant signature algorithms and parameters, certificate sizes κ and two contrasting signature usage scenarios σ_{low} and σ_{high}.

n	Algorithm	σ_{low}			σ_{high}		
		κ_{min}	κ_{avg}	κ_{max}	κ_{min}	κ_{avg}	κ_{max}
256	Falcon-512	13.41	19.41	35.41	107.25	155.25	283.25
	Dilithium-2	40.81	46.81	62.81	326.50	374.50	502.50
	SPHINCS$^+$-128 s	125.75	131.75	147.75	1006.00	1054.00	1182.00
384	Dilithium-3	54.95	60.95	76.95	439.62	487.62	615.62
	SPHINCS$^+$-192 s	257.00	263.00	279.00	2056.00	2104.00	2232.00
512	Falcon-1024	24.00	30.00	46.00	192.00	240.00	368.00
	Dilithium-5	75.80	81.80	97.80	606.38	654.38	782.38
	SPHINCS$^+$-256 s	469.50	475.50	491.50	3756.00	3804.00	3932.00

We observe that the storage cost of L_i is dominated by the signature size ψ. Moreover, storage may be a problem in the long term, as the blockchain grows endlessly if the network does not decide on archival procedures. In the private scenario, creating a new instance of OTSCHAIN from time to time is allowed since every new instance of the smart contract can have its own blockchain in the Hyperledger Fabric network. Older chains can be kept at least until every certificate registered has expired. Hence, the network may agree on a size threshold to determine when to migrate to a new smart contract instance. This solution cannot be applied to the public version because Ethereum has only one chain of blocks for every smart contract created.

Furthermore, in the Ethereum network, storage is directly connected to monetary costs due to *gas* transaction costs, which in turn, reflect the amount of Ether spent to process an operation. For example, consider the implementation of step R5 in Figure 1, where the smart contract simply stores an array of assets. The *gas* cost is calculated using the metrics specified by Kostamis et al. [37]. Considering Falcon-512 as the algorithm with the smallest ψ, OTSCHAIN requires 617,700 *gas* for κ_{min} and 1,590,100 *gas* for κ_{high}. We remark that there exists an upper bound of *gas* spent per block; Kostamis et al. set this bound to 8,000,000, which may be increased by the network in the future. Either way, algorithms with large ψ such as SPHINCS$^+$-192 s, with a cost of 11,380,400 *gas* for κ_{min}, are generally unsuited to be used in Ethereum.

Another relevant aspect that must be touched on is performance. In the Hyperledger Fabric version, writing data in the blockchain strongly depends on the network configuration. More precisely, the response of endorsing nodes and the execution of the Raft consensus protocol define the time required to process a transaction. On the other hand, in the Ethereum implementation, writing data in the blockchain depends not only on the

network configuration but also on the fees that miners require to execute transactions. In general, higher fees increase the chances of a transaction being accepted in a new block.

The performance cost of reading data depends on the complexity of the query function since monitors may be nodes on the network independent of the deployment scenario. Therefore, we consider that reading data in the blockchain would be similar to reading data from a standard database, i.e., with minimal overhead to SVAs. Finally, we highlight that the proposal is feasible in Hyperledger Fabric and in the Ethereum network. However, the second option suffers primarily from the monetary aspect. Modifying OTSCHAIN to store $\mathsf{L}_i = (\mathcal{H}(\mathsf{C}_i), \mathsf{H}_i)$ would reduce these requirements for storage-constrained systems but at the expense of deteriorating the monitoring capabilities of the framework.

5. Final Remarks

The existence of rogue CAs, end-user credential theft, and human error are some of the problems that PKI implementation faces. We address the primary challenge associated with these issues, i.e., their swift detection. We define a framework that expands on the benefits of the CT protocol via the inclusion of a blockchain network. Our rationale is to include the monitoring of end-user signatures in an accessible and transparent manner by shifting the logging process to CAs. Furthermore, we enable the secure use of OTS schemes through semantic checks processed by smart contracts. We discuss the privacy and security implications of registering end-user certificates bound to message hashes in the blockchain. Storage cost and performance estimations show that our model is feasible and compatible with current PKI implementations, maintaining security while increasing usability to end users and acting as a first responder to misbehaving entities.

Future work. There exist several blockchain implementations with specific characteristics such that one cannot easily predict how they will behave when paired with OTSCHAIN. Therefore, we suggest searching for an optimal selection of blockchain for each use case, such as specific consensus algorithms, throughput, and privacy requirements. Similarly, investigating advanced mechanisms, such as blockchain sharding, may improve the performance and scalability aspects of the deployment. Moreover, the function f may be augmented to include other semantic checks, for instance, adapting the framework to accept few-time or conventional signature schemes by allowing a specific number of uses of the private key. We note, however, that modifying f requires careful consideration, as it might cause privacy and security implications. Furthermore, while the changes proposed to the validation process are straightforward, a formal analysis may be conducted through security protocol verification tools, such as Tamarin [38] or AVISPA [39].

Author Contributions: Conceptualization, L.M., L.P., G.Z., W.S. and R.C.; methodology, L.M., L.P., G.Z., W.S. and R.C.; software, L.M., L.P., G.Z., W.S., and R.C.; validation, L.M., L.P., G.Z., W.S., and R.C.; formal analysis, L.M., L.P., G.Z., W.S., and R.C.; writing—original draft preparation, L.M., L.P., G.Z., W.S., and R.C.; writing—review and editing, L.M., L.P., G.Z., W.S., and R.C. All authors have read and agreed to the published version of the manuscript.

Funding: This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior—Brasil (CAPES)—Finance Code 001, and by the Operador Nacional do Registro Civil de Pessoas Naturais (ON-RCPN) do Brasil.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Kohnfelder, L.M. Towards a Practical Public-key Cryptosystem. Bachelor's Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1978.
2. Van der Meulen, N. DigiNotar: Dissecting the First Dutch Digital Disaster. *J. Strateg. Secur.* **2013**, *6*, 46–58. [CrossRef]
3. Laurie, B.; Langley, A.; Kasper, E.; Messeri, E.; Stradling, R. Certificate Transparency Version 2.0. RFC 9162, Internet Engineering Task Force. 2021. Available online: https://www.rfc-editor.org/info/rfc9162 (accessed on 20 August 2023). [CrossRef]
4. Kampanakis, P.; Panburana, P.; Daw, E.; Geest, D.V. *The Viability of Post-quantum X.509 Certificates*; Cryptology ePrint Archive, Paper 2018/063; ISARA; 2018. Available online: https://www.isara.com/resource-center/the-viability-of-post-quantum-x.509-certificates.html (accessed on 20 August 2023).
5. Hülsing, A. W-OTS$^+$—Shorter Signatures for Hash-Based Signature Schemes. In *Progress in Cryptology— AFRICACRYPT 2013 6th International Conference on Cryptology in Africa, Cairo, Egypt, 22–24 June 2013*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2013; Volume 7918, pp. 173–188. [CrossRef]
6. Buchmann, J.; Dahmen, E.; Ereth, S.; Hülsing, A.; Rückert, M. On the Security of the Winternitz One-Time Signature Scheme. In *Progress in Cryptology—AFRICACRYPT 2011 4th International Conference on Cryptology in Africa, Dakar, Senegal, 5–7 July 2011*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6737, pp. 363–378. [CrossRef]
7. Madala, D.S.V.; Jhanwar, M.P.; Chattopadhyay, A. Certificate Transparency Using Blockchain. In Proceedings of the 2018 IEEE International Conference on Data Mining Workshops (ICDMW), Singapore, 17–20 November 2018; pp. 71–80. [CrossRef]
8. Garba, A.; Hu, Q.; Chen, Z.; Al, M.R.A. BB-PKI: Blockchain-Based Public Key Infrastructure Certificate Management. In Proceedings of the 2020 IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Yanuca Island, Cuvu, Fiji, 14–16 December 2020; pp. 824–829. [CrossRef]
9. Kubilay, M.Y.; Kiraz, M.S.; Mantar, H.A. KORGAN: An Efficient PKI Architecture Based on PBFT through Dynamic Threshold Signatures. *Comput. J.* **2021**, *64*, 564–574. [CrossRef]
10. Li, Z.; Wu, H.; Lao, L.H.; Guo, S.; Yang, Y.; Xiao, B. Pistis: Issuing Trusted and Authorized Certificates With Distributed Ledger and TEE. *IEEE Trans. Parallel Distrib. Syst.* **2022**, *33*, 1636–1649. [CrossRef]
11. Han, K.; Hwang, S.O. A PKI without TTP based on conditional trust in blockchain. *Neural Comput. Appl.* **2020**, *32*, 13097–13106. [CrossRef]
12. Vives, S.J. Synced Hash-Based Signatures: Post-Quantum Authentication in a Blockchain. In Proceedings of the Actas del IX Congreso Iberoamericano de Seguridad Informática, Buenos Aires, Argentina, 1–3 November 2017.
13. Hyla, T.; Pejaś, J. Long-term verification of signatures based on a blockchain. *Comput. Electr. Eng.* **2020**, *81*, 106523. [CrossRef]
14. Xu, L.; Chen, L.; Gao, Z.; Fan, X.; Doan, K.; Xu, S.; Shi, W. KCRS: A Blockchain-Based Key Compromise Resilient Signature System. In *BlockSys 2019: Blockchain and Trustworthy Systems, Guangzhou, China, 7–8 December 2019*; Communications in Computer and Information Science; Springer: Singapore, 2019; Volume 1156, pp. 226–239. [CrossRef]
15. Menezes, A.J.; van Oorschot, P.C.; Vanstone, S.A. *Handbook of Applied Cryptography*, 1st ed.; CRC Press: Boca Raton, FL, USA, 1996.
16. Goldreich, O. *Foundations of Cryptography: Volume 2, Basic Applications*, 1st ed.; Cambridge University Press: Cambridge, UK, 2004.
17. Kutyłowski, M.; Błaśkiewicz, P. Advanced Electronic Signatures and eIDAS—Analysis of the Concept. *Comput. Stand. Interfaces* **2023**, *83*, 103644. [CrossRef]
18. *ETSI EN 319 102-1 V1.3.1*; Electronic Signatures and Infrastructures (ESI); Procedures for Creation and Validation of AdES Digital Signatures; Part 1: Creation and Validation; European Telecommunications Standards Institute: Sophia Antipolis, France, 2021.
19. Dods, C.; Smart, N.P.; Stam, M. Hash Based Digital Signature Schemes. In *Cryptography and Coding 2005: Cryptography and Coding: 10th IMA International Conference, Cirencester, UK, 19–21 December 2005*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3796, pp. 96–115. [CrossRef]
20. Hülsing, A.; Kudinov, M. Recovering the Tight Security Proof of SPHINCS$^+$. In *ASIACRYPT 2022: Advances in Cryptology—ASIACRYPT 2022*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2022; Volume 13794, pp. 3–33. [CrossRef]
21. Nakamoto, S. *Bitcoin: A Peer-to-Peer Electronic Cash System*; 2008. Available online: https://ssrn.com/abstract=3440802 (accessed on 20 August 2023).
22. Wood, G. Ethereum: A Secure Decentralised Generalised Transaction Ledger; Ethereum Foundation: Zug, Switzerland, 2022.
23. Hyperledger Foundation. *An Introduction to Hyperledger*; Hyperledger Foundation: San Francisco, CA, USA, 2018.
24. Hülsing, A.; Butin, D.; Gazdag, S.L.; Rijneveld, J.; Mohaisen, A. XMSS: eXtended Merkle Signature Scheme. RFC 8391, Internet Engineering Task Force. 2018. Available online: https://www.rfc-editor.org/rfc/rfc8391.html (accessed on 20 August 2023).
25. Palma, L.M.; Vigil, M.A.G.; Pereira, F.L.; Martina, J.E. Blockchain and smart contracts for higher education registry in Brazil. *Int. J. Netw. Manag.* **2019**, *29*, e2061. [CrossRef]
26. European Parliament and the Council of the European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016. *Off. J. Eur. Union* **2016**, *L119*, 1–88.
27. Brasil. Lei nº 13.079, de 14 de agosto de 2018. Lei Geral de Proteção de Dados Pessoais (LGPD). *Diário Oficial da União* **2018**, *157*, 59–64.
28. Bernabe, J.B.; Canovas, J.L.; Hernandez-Ramos, J.L.; Moreno, R.T.; Skarmeta, A. Privacy-Preserving Solutions for Blockchain: Review and Challenges. *IEEE Access* **2019**, *7*, 164908–164940. [CrossRef]
29. Benet, J. IPFS—Content Addressed, Versioned, P2P File System. *arXiv* **2014**, arXiv:1407.3561v1. https://doi.org/10.48550/arXiv.1407.3561.

30. Fiege, U.; Fiat, A.; Shamir, A. Zero knowledge proofs of identity. In Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, New York, NY, USA, 1 January 1987; pp. 210–217.
31. Sedlmeir, J.; Buhl, H.U.; Fridgen, G.; Keller, R. The Energy Consumption of Blockchain Technology: Beyond Myth. *Bus. Inf. Syst. Eng.* **2020**, *62*, 599–608. [CrossRef]
32. Ongaro, D.; Ousterhout, J. In search of an understandable consensus algorithm. In Proceedings of the 2014 USENIX Annual Technical Conference (USENIX ATC 14), Philadelphia, PA, USA, 19–20 June 2014; pp. 305–319.
33. Luu, L.; Narayanan, V.; Zheng, C.; Baweja, K.; Gilbert, S.; Saxena, P. A secure sharding protocol for open blockchains. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna Austria, 24–28 October 2016; pp. 17–30.
34. Alagic, G.; Apon, D.; Cooper, D.; Dang, Q.; Dang, T.; Kelsey, J.; Lichtinger, J.; Liu, Y.K.; Miller, C.; Moody, D.; et al. Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process; Internal Report 8413-upd1; National Institute of Standards and Technology (NIST): Gaithersburg, MD, USA, 2022. [CrossRef]
35. Instituto Nacional de Tecnologia da Informação. *Relatório de Gestão 2022*; Instituto Nacional de Tecnologia da Informação: Brasilia, Brazil, 2022.
36. Kuik, S. *In 20 Years, More than 800 Million Digital Signatures Have Been Given in Estonia | RIA*; Information System Authority: Harjumaa, Estonia, 2022.
37. Kostamis, P.; Sendros, A.; Efraimidis, P. Exploring Ethereum's Data Stores: A Cost and Performance Comparison. In Proceedings of the 2021 3rd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS), Paris, France, 27–30 September 2021; pp. 53–60. [CrossRef]
38. Meier, S.; Schmidt, B.; Cremers, C.; Basin, D. The TAMARIN prover for the symbolic analysis of security protocols. In *Computer Aided Verification: 25th International Conference, CAV 2013, Saint Petersburg, Russia, 13–19 July 2013*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 696–701.
39. Armando, A.; Basin, D.; Boichut, Y.; Chevalier, Y.; Compagna, L.; Cuéllar, J.; Drielsma, P.H.; Héam, P.C.; Kouchnarenko, O.; Mantovani, J.; et al. The AVISPA tool for the automated validation of internet security protocols and applications. In *Computer Aided Verification: 17th International Conference, CAV 2005, Edinburgh, Scotland, UK, 6–10 July 2005*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 281–285.