A Scalable Bitcoin-based Public Key Certificate Management System

Chloe Tartan[1], Craig Wright[1], Michaella Pettit[2] and Wei Zhang[1]
[1]nChain Ltd., London, U.K.
[2]nChain AG, Zug, Switzerland

Keywords: Bitcoin, Blockchain, Public Key Infrastructure, Certificate Management, Certificate Transparency.

Abstract: The main challenges with traditional public key infrastructures arise from the detection of fraudulent public key certificates and the timely retrieval of an up-to-date record of revoked certificates. While Certificate Transparency logs help to detect falsified certificates in circulation, they do not address the prevailing issues with certificate revocation. Public blockchains such as Bitcoin can be used to create a transparent, tamper-proof log of events secured by the cryptographic work carried out by nodes in the network. In this paper, we present a Bitcoin-based certificate management system that exploits the scalability and low-cost features of its underlying blockchain infrastructure, while preserving user privacy. Based on a feasibility analysis, we estimate the capability to support 9000 certificate issuances, revocations, or updates per second at a cost of less than 0.005 USD per event. The immutability and auditability of records stored on the blockchain provides a universal view of public key certificates. A comparative analysis shows that our solution can significantly reduce the overhead endured by live certificate status retrievals and offers flexibility in certificate revocation. The revocation of a public key certificate is as simple as spending a Bitcoin transaction.

1 INTRODUCTION

Public key infrastructure (PKI) underpins the security of most public key cryptosystems. PKI schemes typically consist of a certificate authority (CA) that verifies the identity of a user and signs their public key certificate. The certificate binds the identity to the public key. It ensures that the public key used in a cryptosystem indeed belongs to the expected user.

Traditional PKI models are often centralised with a hierarchical structure. Certificate logging (Blagov & Helm, 2020) was introduced to bring more transparency to the CA process, and has become mandatory for web PKIs on account of CA misbehaviour and several security breaches (Van der Meulen, 2013) (Langley, 2015). Google's Certificate Transparency is the most widely deployed public log of SSL/TLS certificates (Scheitle, et al., 2018). Other alternatives to PKI include identity-based cryptography (Boneh & Franklin, 2001) (Boneh, Lynn, & Shacham, 2001) and certificateless public key cryptography (Al-Riyami & Paterson, 2003). Both alternatives require an implementation of a bilinear map, which is difficult due to its complexity (Galbraith, Paterson, & Smart, 2008).

One of the most challenging tasks for any PKI model is certificate revocation, which necessitates the timely retrieval of up-to-date records of revoked certificates compounded by the additional effort required to check any such records. One approach to handle revocation checks is to consult public certificate revocation lists (CRLs) (Cooper, et al., 2008), which are signed by CAs to ensure their integrity. This creates a significant overhead for certificate verification while also introducing potential implementation-related vulnerabilities (Hoogstraaten, 2012). Some web browsers default to not checking CRLs in order to gain performance efficiency (CERT Division, 2001).

Another approach is to use the Online Certificate Status Protocol (OCSP) (Santesson, et al., 2013) in which the status of a certificate is maintained on a dedicated server that facilitates status queries via a request-response mechanism. OCSP Stapling and Must-Staple (Wazan, et al., 2020) are extensions that address network latency and user privacy issues (Zhu, Amann, & Heidemann, 2016). However, several major browsers have opted for proprietary revocation mechanisms e.g., Google's CRLSets (Langley, 2012) and Firefox's CRLite (Larisch, et al., 2017) both of which crawl CRL servers and certificate logs periodically to update their list of valid certificates. While these browsers support traditional OCSP checks, failure to determine a certificate's status within an acceptable timeframe results in a soft-fail revocation check, where the certificate is accepted by default when its status is indeterminable. An adversary can therefore simply suppress the OCSP response in a denial-of-service (DoS) attack.

In recent years, blockchain technology has offered a promising future for PKI. The immutability and transparency of data stored on a blockchain (on-chain) offers a single source of truth for the PKI ecosystem, while the consensus mechanism secures and timestamps the on-chain data. The Certcoin protocol by Fromknecht, Velicanu, & Yakoubov (2014) defines a pure blockchain-based PKI that publicly links identities to public keys in the Namecoin network, while the work by Axon & Goldsmith (2017) adapts Certcoin to provide privacy to its users. However, neither of these methods propose an adequate solution for the reclamation of an identity in the case of key compromise (Kubilay, Kiraz, & Mantar, 2019).

Yakubov, Shbair, Wallbom, Sanda & State (2018) use Ethereum to design smart contracts that automate certificate management. While the solution benefits from the transparency of data stored on a public blockchain, it is not suitable for large-scale applications. This is due to the fundamental scaling limitations arising from Ethereum's account-based transactional model, where each user account is updated sequentially by a generalised state transition function (Wood, 2014). As a result, large transaction volumes cause congestion in the network that translates to higher transaction fees for its users (Etherscan, 2020). Ethereum-based certificates may therefore be difficult and expensive to manage. Applications that require fast processing can instead be realised with UTXO-based transactional models (Zhang, Xue, & Liu, 2019) such as Bitcoin, where parallel processing is supported.

1.1 Our Contributions

In this paper, we propose a Bitcoin-based certificate management system that can be readily implemented and addresses ongoing issues with certificate verification and revocation. We show how a public key certificate can be published in a Bitcoin transaction, which allows us to create a link between a certified public key and a Bitcoin transaction outpoint. To verify the certificate, a verifier checks whether the corresponding transaction outpoint is unspent. To revoke a certificate, one simply spends the transaction outpoint. The main contributions of the paper are:

- a low-cost blockchain-based system that can manage large volumes of certificates,
- a single source of truth for certificates and their status,
- a transparent and auditable log of timestamped certificates,
- an immutable record of certificates that is secure even when keys are compromised,
- a flexible revocation mechanism using Bitcoin's Script language that can accommodate different revocation requirements and key updates,
- an atomic verification mechanism that integrates large-value payments and identity verification into one Bitcoin transaction, and
- an obfuscating technique that preserves user privacy.

The paper is organised as follows: Section 2 provides an overview of Bitcoin along with relevant PKI standards and practices. In Section 3, we outline certificate issuance, revocation, verification, and key updates, together with the assignment of revocation rights. In addition, we describe the immutability of historical records, user privacy and atomic certificate verification. Section 4 contains empirical data to show the scalability and low-cost of our solution. Section 5 provides comparisons with other PKI solutions, and Section 6 concludes the paper.

2 PRELIMINARIES

This section describes preliminaries on Bitcoin, along with relevant PKI standards and practices.

2.1 Bitcoin

The Bitcoin blockchain can be viewed as a distributed platform service that offers scalability, transparency, immutability, and the availability of data. Data can be published on the blockchain and retrieved in the form of Bitcoin transactions. There are different blockchain implementations with a shared history tracing back to the original Bitcoin genesis block, such as Bitcoin Core (BTC), Bitcoin Cash (BCH) and Bitcoin SV (BSV). When implemented on Bitcoin SV, the solution in this paper benefits from scalability, the lowest cost, and data integrity. The Bitcoin SV protocol offers high transaction throughput, with over 2000 transactions per second (tps) on the main net (Blockchair, 2020) and 9000 tps on the scaling test net (Southurst, 2021). In what follows we will present the solution in accordance with the Bitcoin SV protocol for transactions, but it is understood that it can be easily adapted to any UTXO-based blockchain. Table 1 shows a simplified format of a Bitcoin transaction.

Table 1: Bitcoin Transaction Format.

Transaction ID			
Version		Locktime	
Inputs			
Outpoint	Unlocking Script		Sequence Number
Outputs			
Value	Locking Script		

We would like to highlight a few fields in the transaction format.

- **Outpoint:** a concatenation of a transaction identifier (TxID) and an index that identifies the output from the previous transaction.
- **Unlocking Script:** a script that contains data only (no operational codes). It is combined with the corresponding locking script for script execution, which is part of the transaction validation process. It usually contains a digital signature and a public key.
- **Locking Script:** a script that contains the conditions to spend the output. It usually contains a hash value check on a public key and a verification of a signature, which is known as Pay-To-Public-Key-Hash (P2PKH) script. The script can also be more complex to accommodate for more complicated spending conditions. In addition to its primary functionality, a locking script can be used as a data carrier, where a data payload can be appended to the opcode `OP_RETURN`.

A transaction outpoint can be either spent or unspent. The set of unspent transaction outpoints is called UTXO set. A static UTXO set can be derived from the history of the blockchain for a given block height. A live UTXO set also accounts for the transactions in the mempool (a set of validated transactions that are to be published on-chain). As a result, a live UTXO set can vary from one Bitcoin node to another.

A block consists of a set of Bitcoin transactions and a block header, which is of a fixed size of 80 bytes regardless of the number of transactions in that block. A Merkle tree (Merkle, 1979) is derived from the set of transactions in a block where the leaves consist of the transaction identifiers. The Merkle root is then included in the block header. It provides an efficient mechanism to prove transaction inclusion. Given a transaction and its Merkle proof, one can compute the Merkle root and compare it with the Merkle root in a block header. If they match, one can be convinced that the transaction has been validated and the transaction data has not been tampered with. This integrity and inclusion proof is referred to as simplified payment verification (SPV), a terminology coined in the original Bitcoin white paper. This lightweight mechanism makes Bitcoin applications low cost and scalable, since they can ignore other transactions and only maintain block headers, application-specific transactions and their Merkle proofs. As a result, applications built on the Bitcoin system can operate without hindrance from the ever-growing size of the blockchain.

The consensus mechanism in Bitcoin is based on proof of work. Each block header contains a 4-byte number that allows iterative hashing of the block header. The goal is to find a hash value that is less than a set value derived from the difficulty level specified in the block header. This allows the proof of work on a block header to be verified independently. Each block header also contains the previous block header hash and forms a chain of block headers. To change a historical block, one must redo the proof of work on all the following blocks. The earlier the block, the more proof of work is required to change it. This results in the immutable nature of historical data on-chain.

The Bitcoin network consists of Bitcoin nodes, also loosely known as miners. Their services include, but are not limited to, transaction validation and propagation, block construction and publication, and responding to queries for Merkle proofs and transaction status. The majority of Bitcoin nodes are incorporated businesses and can be identified via coinbase transactions. An initiative to establish miner IDs led by the Bitcoin SV community allows miners to build their reputation over time based on proof of work (nChain, 2020). This brings more transparency to the network and therefore leads to more overall trustworthiness.

2.2 Relevant Standards and Practices

In this section, we describe two standards that share similarities with our solution, namely OCSP Stapling (Eastlake, 2011) (Pettersen, 2013) and CT logs

(Laurie, Langley, & Kasper, 2013). The high-level architecture of these two approaches is the same, as shown in Figure 1.

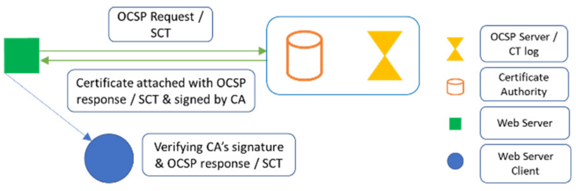
OCSP Request / SCT
Certificate attached with OCSP response / SCT & signed by CA
Verifying CA's signature & OCSP response / SCT
OCSP Server / CT log
Certificate Authority
Web Server
Web Server Client

Figure 1: High-Level Architecture of OCSP Stapling / CT Logs.

OCSP Stapling describes a way to append or 'staple' the short-term status of a certificate to the certificate itself. The CA signs the certificate status in the same way that it signs the certificate, both of which are cached on the web servers frequently. During certificate checks, OCSP stapling shifts the onus from the client (browser) to the web server, mitigating potential breaches of user privacy and reducing network latency as a result. OCSP Must-Staple (Hallam-Baker, 2015) is an advancement that addresses DoS attacks by ensuring that an OCSP response must be included with a certificate request for the public key to be deemed valid. However, its adoption among browsers remains limited.

Certificate transparency (CT) logs are append-only Merkle trees that provide a public record of certificates issued by different CA organisations to support the detection of falsified certificates. While Google's CT is the most widely deployed log in the web PKI ecosystem, any CA can choose to maintain their own log. It is recommended that CAs also publish their certificates to third-party logs (Laurie, Langley, & Kasper, 2013). A central authority monitors and audits each CA's log to ensure that the information is consistent.

A signed certificate timestamp (SCT) is generated by a CT log when the certificate is submitted for publication at the point of issuance. The SCT acts as proof of a certificate's existence in a log. It is possible to fetch an SCT using an OCSP request (DigiCert, 2021). While there is no minimum number of proofs required, Google currently recommends including at least three such proofs with a certificate to account for any misinformation arising from compromised or misbehaving logs (Google, 2013).

The general operation of a CT log is complex and requires a high degree of availability to fulfil client requirements. As logs grow, they become more expensive and difficult to operate (Matsumoto, Szalachowski, & Perrig, 2015). Temporal sharding can be used to address these scaling issues by limiting the date range of CT logs. This results in multiple logs per CA, each of which is typically limited to one-year segments (Lynch, 2018). However, increasing the number of CT logs increases the cost of monitoring for domains and auditors (Google, 2013).

3 CERTIFICATE MANAGEMENT ON-CHAIN

The proposed solution aims to address the issues of certificate revocation in PKI and offers an all-in-one set of features including transparency, privacy, auditability, and immutability of historical records. The high-level architecture of the solution is shown in Figure 2.

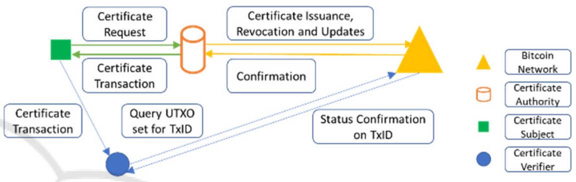

Figure 2: High-Level Architecture of a Bitcoin-Based Certificate Management System.

Our solution does not require a new blockchain. It adapts the existing PKI practices and utilises the Bitcoin network infrastructure. A certificate authority, say Charlie, connects to the Bitcoin network by connecting to a Bitcoin node or a group of nodes, say Natalie. A user, say Alice, submits a certificate request to Charlie and is issued a certificate in the form of a Bitcoin transaction. A certificate verifier, say Bob, queries the Bitcoin network for the status of the outpoint in the certificate transaction. If it is unspent, then the certificate is valid. Otherwise, it is invalid. All certificates and their change of status are recorded in Bitcoin transactions and published on the blockchain for transparency and auditability. The verification of certificates requires interactions with the Bitcoin network.

The detail of the process is described in the next section, followed by other features and extensions such as flexibility in revocation, immutability of historical records, privacy enhancements, and atomic certificate verification.

3.1 Certificate Life Cycle

In this section, we describe how to issue, revoke, update and verify certificates using Bitcoin transactions. The mechanisms can be applied to any certificate type that links an identity to a public key.

3.1.1 Issuance

Alice submits a certificate request to the CA, Charlie. After verifying Alice's identity, Charlie creates a Bitcoin transaction that can be viewed as a certificate for Alice's public key.

Table 2 shows an example of a certificate issuance transaction. Charlie assigns x satoshis to himself (PK_C) from the outpoint $TxID_0||0$ and embeds an issuance message for PK_A as an `OP_RETURN` data payload. This `OP_RETURN` does not invalidate the script in accordance with the Bitcoin SV implementation (Bitcoin SV, 2021).

Table 3 shows an example of the issuance message format. Note that the data payload can be an X.509 V3 certificate in its entirety, which would provide great interoperability. It also can be of any other format, e.g., OpenPGP (Callas, Donnerhacke, Finney, Shaw, & Thayer, 2007). In the extension field, Charlie adds some identification evidence, which can be the hash digest of documents that Alice has provided to verify her identity. This would be used to prove that Charlie has conducted the identification due diligently.

Table 2: Certificate Issuance Transaction.

$TxID_1$			
Version	1	Locktime	0
Inputs			
Outpoint	Unlocking Script		Sequence Number
$TxID_0\|\|0$	<SIG_C> <PK_C>		0xFFFFFFFF
Outputs			
Value	Locking Script		
x	`OP_DUP OP_HASH160` <$H(PK_C)$> `OP_EQUALVERIFY OP_CHECKSIG OP_RETURN` <Issuance Message PK_A>		

Table 3: Issuance Message.

Message Type	Issuance
Issuer Name	Charlie
Subject Name	Alice
Subject Public Key	PK_A
Subject Public Key Algorithm	ECDSA
Extensions (optional)	Identification Evidence

The digital signature (SIG_C, PK_C) from Charlie in the unlocking script not only signs the transaction but also attests to the issuance message for PK_A as it is part of the signed message. This implies that there is no need to include an explicit signature in the issuance message.

Charlie then sends the transaction to the Bitcoin node Natalie for publication to the blockchain. Charlie can wait for the next block to be published before passing on the transaction to Alice as her public key certificate. This often takes around ten minutes, occasionally more. However, Bitcoin nodes can implement an API (nChain, 2021) that allows users to request for instant confirmation of the acceptance of their transactions. Nodes with this capability are the ideal candidates to connect to.

The transaction $TxID_1$ effectively binds the validity of the certificate of public key PK_A to the Bitcoin outpoint $TxID_1||0$. If $TxID_1||0$ is unspent, then PK_A is a valid public key that belongs to Alice.

As Bitcoin transactions allow a large number of outputs, Charlie can combine multiple issuances into one issuance transaction. Each certificate can be independently identified by the transaction outpoint. The "Locktime" and "Sequence Number" can be used to schedule the issuance of certificates (Bitcoin SV, 2021). Both batching and scheduling mechanisms are also applicable to both revocation and key updates.

3.1.2 Revocation

In the case that the certificate corresponding to Alice's public key PK_A needs to be revoked, Charlie can spend $TxID_1||0$ to invalidate the certificate. Charlie creates a Bitcoin transaction as shown in Table 4. Charlie assigns x' satoshis to himself (PK_C) from the outpoint $TxID_1||0$ whose spending status is tied to the validity of the certificate on PK_A. Charlie also embeds a revocation message in the output as the `OP_RETURN` data payload. The revocation message may take the format in the example shown in Table 5.

Table 4: Certificate Revocation Transaction.

$TxID_2$			
Version	1	Locktime	0
Inputs			
Outpoint	Unlocking Script		Sequence Number
$TxID_1\|\|0$	<SIG_C'> <PK_C>		0xFFFFFFFF
Outputs			
Value	Locking Script		
x'	`OP_DUP OP_HASH160` <$H(PK_C)$> `OP_EQUALVERIFY OP_CHECKSIG OP_RETURN` <Revocation Message>		

Table 5: Revocation Message Example.

Revocation Message	
Issuer Name	Charlie
Reason for Revocation	Key Compromised
Extension (optional)	

Note that it is not critical to include PK_A in $TxID_2$, since the revocation transaction is not for users to check. The purpose of this transaction is to ensure that any check on the certificate transaction fails, as the validity of the certificate is bonded to the spending status of the certificate transaction outpoint. However, the data payload may be an X.509 V2 revocation message to provide interoperability. While it is more informative and auditable to include a revocation message, the absence of one also works.

Charlie signs the revocation transaction and sends it to the Bitcoin node Natalie. When Charlie obtains a confirmation from Natalie, it can be viewed as a confirmation that the revocation is successful. As for certificate issuance, this can take around 10 minutes and occasionally more, while it would take a matter of seconds or less if Natalie has implemented an API that offers instant confirmations.

3.1.3 Update

It is good practice to update cryptographic keys regularly and is a necessity if they are compromised or lost. An ideal key update mechanism would be low cost and take effect in a short time. In our solution, key updates are done by combining revocation and issuance into a single Bitcoin transaction. Suppose Alice needs to update her public key PK_A to PK_A', Charlie spends the certificate transaction outpoint for PK_A to create a certificate transaction for PK_A' as shown in Table 6.

Table 6: Certificate Update Transaction.

$TxID_3$			
Version	1	Locktime	0
Inputs			
Outpoint	Unlocking Script		Sequence Number
$TxID_1\|\|0$	<SIG_C'> <PK_C>		0xFFFFFFFF
Outputs			
Value	Locking Script		
x'	OP_DUP OP_HASH160 <$H(PK_C)$> OP_EQUALVERIFY OP_CHECKSIG OP_RETURN <Update Message PK_A'>		

The update message in the output can take the format shown in Table 7, and for interoperability, an X.509 V3 certificate can be used as an alternative.

Table 7: Updated Public Key Data Payload.

Certificate for PK_A	
Issuer Name	Charlie
Subject Name	Alice
Subject Public Key	PK_A'
Subject Public Key Algorithm	ECDSA
Extension (optional)	PK_A updated

Note that we do not need to have an explicit output to represent revocation. It is the action of spending $TxID_1||0$ that invalidates the certificate. On the other hand, spending $TxID_1||0$ requires Charlie's signature, which implies that it can be used as an input to issue a new certificate as described in Section 3.1.1. In this case, the output can be utilised to represent the issuance of the certificate for PK_A'.

Charlie signs the transaction and sends it to the Bitcoin node Natalie. After receiving confirmation from Natalie, Charlie passes on $TxID_3$ to Alice as the new certificate reference for her public key PK_A'. If Natalie offers instant confirmation service, then the key update can take effect immediately.

3.1.4 Verification

To verify the certificate of Alice's public key, Bob needs to conduct three checks:

1. the integrity of the transaction data,
2. the relevant information embedded in the transaction data, and
3. the status of the transaction outpoint.

Bob can either obtain the transaction data and its Merkle proof from Alice or retrieve both from the Bitcoin network by referencing the transaction ID $TxID_1$. Bob must have an up-to-date copy of the block headers to verify the Merkle proof and ensure the integrity of the transaction data. Any attempt to modify the data will invalidate the Merkle proof.

After being convinced that the transaction data has not been tampered with, Bob parses the data to read the locking script and the unlocking script. The locking script should contain a data payload indicating that the public key PK_A belongs to Alice and is certified by Charlie.

The unlocking script should contain Charlie's public key PK_C. We assume that PK_C is certified in the same way as PK_A. If Charlie is a root CA or a subordinate CA, Bob may conduct regular checks on the certificate of PK_C instead of every time it appears in an unlocking script. In this case, we assume that Bob has a list of trusted certified public keys to look up PK_C. Otherwise, Bob must conduct the same verification on the certificate of PK_C and follow the chain of certificates until he reaches a certified public key that is on his list. One criterion for a certificate to be maintained in the trusted list can be its expected lifetime, for example, at least 10 years (IdenTrust Services, 2021).

The last step is for Bob to query the Bitcoin network on whether the certificate transaction outpoint is unspent. In this step, we assume that Bitcoin nodes are trusted. More precisely, we assume that their response to the query is genuine and authenticated. The assumption is particularly important when the response is unspent. Verifying whether a transaction outpoint is spent involves pinpointing the spending transaction and verifying its validity, which is relatively easy. Verifying that a transaction is unspent would require verifying all transactions that come after it, which is far more computationally costly for users. Nevertheless, knowing that there is a way to check their response, it is reasonable to assume that Bitcoin nodes are honest and trusted.

Bob can either make a static query or a live query. A static query is to ask for the status of the outpoint at a time in the past, either using block height or date and time. As the response is based on the published blocks, a static query can be made to any Bitcoin node. A live query reflects the status in a timely manner. In addition to published transactions, it takes the transactions that are validated and have not yet published into account. It is available if Bob connects to one of the Bitcoin nodes to which Charlie connects.

For Bob, a live query would generally be preferred as it provides the real-time status of the certificate. There are scenarios, such as auditing historical usage of certificates, where static queries may be more appropriate.

For Natalie, responding to a live query is more costly since it requires a snapshot of the dynamic UTXO set that is continuously updated by incoming transactions, while responding to a static query only requires a look up on a static data set.

For Charlie, connecting to multiple Bitcoin nodes in different geolocations and sending the transactions to each of them simultaneously would reduce the latency of transaction propagation in the Bitcoin network. This would allow Bob to have a wider choice of Bitcoin nodes to connect to, which would be especially useful if Bob has limited access to Bitcoin nodes. Charlie may choose to shift the extra cost to Alice as an optional feature. However, any business model for Alice, Charlie or Natalie is beyond the scope of this paper.

3.2 Rights to Revoke

As described in Section 3.1.2, revoking a certificate for a public key is achieved by spending a transaction outpoint. The right to spend the transaction outpoint gives the right to revoke the certificate. The locking script in a Bitcoin transaction can accommodate different spending conditions and therefore different revoking requirements.

As of now, we assume that the CA Charlie can revoke certificates. This reflects the current setup in a general PKI model. However, the locking script can be adapted to allow any designated entity to revoke the certificate. For example, by replacing the hash value $H(PK_C)$ with $H(PK_A)$ in the output of the certificate issuance transaction shown in Table 2, Charlie effectively assigns Alice the right to revoke her own certificate. This is advantageous when Alice needs to revoke the certificate of her public key because her private key has been compromised. She does not have to communicate with Charlie and wait for his response to her revocation request. She can spend the certificate transaction outpoint immediately to revoke her public key certificate.

On the other hand, Charlie may want to revoke Alice's public key on account of misbehaviour. To accommodate both requirements, a 1-out-of-2 multi-signature can be used, in which case either Alice's signature or Charlie's signature will be able to spend the outpoint. An example locking script can be constructed as:

```
OP_DUP OP_HASH160 OP_DUP
<H(PK_A)> OP_EQUAL
OP_IF
 OP_DROP
OP_ELSE
 <H(PK_C)> OP_EQUALVERIFY
OP_ENDIF
OP_CHECKSIG
OP_RETURN
<Issuance Certificate PK_A>.
```

This can be generalised further to m-out-of-n scenarios for any integer $m \geq 1$ and $n \geq 2$. If $m \geq 2$, ECDSA threshold signature schemes (Gennaro & Goldfeder, 2018) can be used, in which case the locking script would have the same format as in Table 2 with a single public key hash. Multi-party revocation is useful in increasing the robustness of the revocation mechanism. For example, a regulatory body can be introduced as part of a 2-out-of-3 revocation scheme. In this case, Charlie cannot unilaterally decide on Alice's misuse of her public key. He requires approval from the regulatory body in order to revoke Alice's public key.

3.3 Mitigating Compromised CAs

When a CA is compromised, it may require revoking all the certificates issued by that entity. This can be done by spending all the certificate transactions originating from the CA. In Section 4, we show how the Bitcoin network has the capability to handle this scale of large volume transactions in a relatively short timeframe. On the other hand, if the verification of the certificate of the compromised CA can be triggered remotely, then spending the corresponding certificate transaction outpoint would serve the purpose of revoking all certificates issued by the CA.

Another feature offered by our solution is the integrity of the historical records even after the private key of a CA is compromised. In traditional PKI, when the private key of a CA is compromised, actions can be taken to prevent any new certificate from being issued. However, it is possible for the attacker to use the private key to create a certificate that would be deemed valid in the past. Because of this possibility, the entire history of certificates from that CA becomes non-trustworthy. The damage done to the historical record becomes irreversible. This problem does not exist with our solution. As all the certificates and their status are recorded on the blockchain, any changes made to the history would require redoing all the proof of work, which is economically inviable and computationally infeasible.

3.4 User Privacy

The privacy issue with OCSP stems from the fact that a user's browsing history is exposed by their certificate status queries, since all the queries are directed at the OCSP server instead of each individual web server. OCSP Staple solves the problem by shifting the burden of status queries to the CA and back to the web servers. Our solution can adopt the same approach to protect user privacy. In our case, the Bitcoin network replaces the OCSP server. Only the CA interacts with the Bitcoin network, and the transaction statuses are signed and passed on to the web servers with a limited lifetime. While this approach offers great interoperability, it does not utilise the benefits of our revocation mechanism.

An alternative approach is to hide the link between a certificate and the corresponding transaction outpoint by using a cryptographic hash function. Charlie can replace the issuance message in the certificate transaction with a hash value derived from the message. Both Alice and Charlie can keep a copy of the issuance message. When Bob makes a query to the Bitcoin node Natalie, Natalie would not be able to identify which certificate the query is about.

By obtaining the issuance message from Alice, Bob can verify its integrity by checking whether the message leads to the same hash value in the certificate transaction. To enhance user privacy, the certificate can be updated frequently, preventing any traffic analysis that attempts to establish the hidden link. When updating the certificate, the CA could combine the revocation of a certificate of one entity with the issuance of a certificate of another as well as combining several updates into one transaction. This would further obfuscate the link between a certificate and its transaction outpoint. Note that recording hash values on-chain represents a series of certificate events that cannot be altered, and the transparency of these commitment-like records ensures that our solution facilitates trustworthy auditing.

3.5 Atomic Certificate Verification

In Section 3.1.4, we assume that Bitcoin nodes are trusted in the sense that their response to a query on whether a transaction is unspent is genuine and authenticated. This assumption can in fact be omitted by designing each certificate to be single use whereby we attempt to "spend the certificate" instead of querying the Bitcoin nodes for a spending status.

A transaction outpoint can be spent only if it is unspent. If the accepted validity of a certificate leads to an action, for example, accepting a payment, then that action will be executed at the same time as the verification of the certificate, thereby making the exchange atomic. This verification mechanism implies that the certificate can only be verified once, but multiple certificates for the same identity can be issued and combined into the same transaction.

This solution integrates with the Bitcoin system at its protocol level by utilising the double spending prevention mechanism for certificate verification. The trust on the Bitcoin system can be inherited by the certificates. As a result, an atomic solution may not be practical for certificates with high verification demands, i.e., domain certificates, but could be of use for secure user identification in online payments.

Consider as an example that Bob would like to make an online payment to Alice for purchasing an item that is of large value. To comply with AML5 (Council of EU, 2018), Alice must verify Bob's identity. Alice can require that the payment transaction includes a transaction outpoint corresponding to Bob's public key certificate that represents his identity. If the transaction is accepted by the Bitcoin network, then Alice knows that Bob's certificate is valid at the time of payment.

Let us assume that Bob has a certificate outpoint $TxID_B||0$ that can be spent by either Bob or the issuer Charlie (Section 3.2). He also has an unspent outpoint containing the funds for payment in $TxID_{fund}||0$. Bob creates a transaction that spends one of the outpoints in his certificate transaction. He also includes an input that enables payment to be given to Alice. He sends the transaction shown in Table 8 to Alice together with $TxID_B$.

Table 8: Atomic Verification Transaction.

$TxID_{AB}$			
Version	1	Locktime	0
Inputs			
Outpoint	Unlocking Script		Sequence Number
$TxID_{fund}\|\|0$	$<SIG_B> <PK_B>$		0xFFFFFFFF
$TxID_B\|\|0$	$<SIG_B> <PK_B>$		0xFFFFFFFF
Outputs			
Value	Locking Script		
x	OP_DUP OP_HASH160 $<H(PK_A)>$ OP_EQUALVERIFY OP_CHECKSIG		

Upon receiving $TxID_{AB}$ and $TxID_B$ from Bob, Alice checks that $TxID_B$ indeed contains Charlie's signature and Bob's public key. She then sends $TxID_{AB}$ to the Bitcoin node Natalie. After receiving confirmation from Natalie, Alice is convinced that Bob's certificate is valid, and she has been paid. The compliance of the regulation and the acceptance of payment are integrated into one action, which is the validation of the transaction by the Bitcoin nodes. This offers Alice a significant computational saving.

4 FEASIBILITY ANALYSIS

In general, one of the main issues with blockchain-based solutions relates to high transaction fees. A justification for choosing Bitcoin SV arises from its extremely low transaction cost. As shown in Figure 3 (Blockchair, 2021), the average fee per Bitcoin SV transaction has been much lower than 0.005 USD since October 2019. According to the Bitcoin SV (2021) node implementation, nodes can configure the minimal acceptance and relay fee rates themselves, which is currently set to 0.5 satoshi per byte and 0.25 satoshi per byte, respectively. Assuming a certificate transaction 1000 bytes in size and a price of 200 USD for 1 BSV, the transaction fee to issue a certificate is roughly 0.001 USD. Note that 1 BSV is 10^8 satoshis. The cost goes up linearly with the price of BSV, but even at a price of 10,000 USD for 1 BSV, the cost of issuing a certificate is under 0.05 USD. Note that the price for an SSL/TLS certificate can range from around 10 USD to over 300 USD per year.

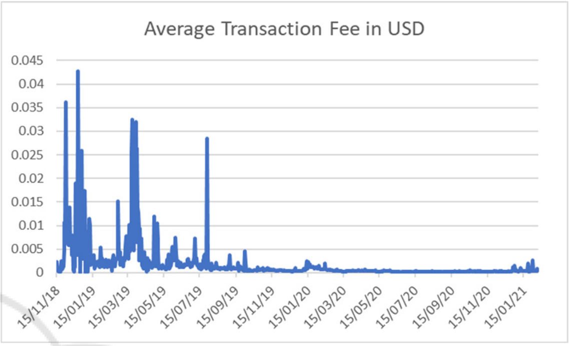

Figure 3: Average Transaction Fee in USD for Bitcoin SV.

In addition to low transaction fees, the blockchain also offers high transaction throughput via bigger block sizes. The estimated capability is 2GB per block (Southurst, 2021) at the time of writing, which equates to 9,000 transactions per second. It would take only one week to issue 4 billion certificates on the blockchain, which is the number of certificates currently included in Google (2021) CT logs.

Bitcoin as a blockchain is subject to reorganisation, which is triggered when more than one Bitcoin node finds a valid block around the same time. Our solution does not rely on any consensus to be reached on the most up-to-date chain tip or block headers, but on the perspective of the Bitcoin nodes that both the CA and certificate verifier are connected to. Provided those Bitcoin nodes as a group remain honest and share the same view of the status of certificate transactions among themselves, then reorganisation will have no impact on our solution.

Taking advantage of the scalability and low cost offered by Bitcoin SV means that our certificate management system can be readily implemented.

5 COMPARATIVE ANALYSIS

In this section, we analyse two different PKI solutions to frame the comparative advantages and disadvantages of our system. We confine each discussion to the features of our solution with reference to a given benchmark solution.

In our first analysis, we choose the scheme proposed by Yakubov, Shbair, Wallbom, Sanda, & State (2018) as our benchmark blockchain-based PKI since it explicitly supports certificate revocation. The authors propose a PKI management framework that uses Ethereum smart contracts for the registration, verification, and revocation of on-chain X.509 (v3) certificates. As with our solution, the framework takes advantage of an existing network infrastructure that makes it readily implementable and uses a permissionless ledger that is secured by large-scale networks of nodes. However, the Ethereum blockchain upon which Yakubov et al.'s framework is implemented cannot support the level of scalability needed for a large volume of certificate transactions as demonstrated by our chosen Bitcoin implementation. The transaction fees are also higher in the Ethereum network. The authors quote a 70 USD fee to publish a CA-issued certificate on-chain, which is significantly more expensive than our estimated cost of under 0.005 USD per certificate.

An in-depth analysis by Kubilay, Kiraz, & Mantar (2020) reveals several privacy and security issues in Yakubov et al.'s proposal, highlighting a critical security flaw with their revocation scheme in the case that a CA is compromised or corrupted. The flexibility of our solution means that revocation is not constrained by CAs, since our users can be bestowed with the rights to revoke their own certificates.

Our second analysis uses log-based PKIs (Matsumoto, Szalachowski, & Perrig, 2015) as a benchmark for certificate transparency. There are several notable similarities between our solution and CT logs. Both architectures are inherently distributed. Both solutions bring transparency to the CA process by creating a publicly verifiable store of issued certificates. Both solutions use cryptographically secure hash functions to create an append-only, tamper-proof record of certificates.

The necessity to provide multiple proofs (i.e., SCTs) highlights that a CT log alone does not represent a single source of truth. Using our system, certificates are timestamped and stored on a unified distributed data set that acts as a single source of truth since it is securely backed by proof of work. Our solution addresses the longstanding issue of certificate revocation, while log-based PKIs do not handle revocation themselves (Google, 2013).

Although the transparency of both solutions disincentives misbehaving or compromised CAs, it does not prevent the production of fraudulent certificates. Thus, auditing and monitoring is necessary for both solutions. However, this cost could be minimised using our system by publishing certificates on a unified blockchain database instead of in multiple logs. CAs could reduce the burden of running their own CT logs by exploiting the existing and reliable network of densely connected Bitcoin nodes. Our low-cost solution also meets the scaling requirements of unbounded certificate logs without any added complexity and the creation of data silos i.e., via temporal sharding (Lynch, 2018).

The comparison table below summarises our findings for each analysis. For brevity, 'Y' refers to Yakubov et al.'s scheme; 'L' refers to log-based PKIs; and 'B' represents our Bitcoin-based certificate management solution.

Table 9: A Comparative Analysis of our System.

PKI Solution	Y	L	B
Public/Permissionless	✓	✓	✓
Economies of Scale	✗	✗	✓
User Privacy	✗	✓	✓
Certificate Transparency	✓	✓	✓
Single Source of Truth	✓	✗	✓
Immutability of History	✓	✗	✓
Revocation Mechanism	✓	✗	✓
Flexible Revocation Rights	✗	✗	✓
Key Update Mechanism	✗	✗	✓
Atomic Certificate Verification	✗	✗	✓

6 CONCLUSIONS

In this paper, we have proposed a Bitcoin-based certificate management system that addresses ongoing issues with certificate revocation in PKI and preserves the features of existing solutions that support transparency and user privacy. Our solution can be readily implemented on Bitcoin SV, a scalable and low-cost implementation of the Bitcoin protocol, while maintaining a high degree of compatibility with established PKI models. It offers a single source of truth with transparency of public key certificates, their status, and their event logs. The distributed nature of the blockchain offers availability of the data; the proof of work consensus mechanism secures the immutability of the data; and the data structure of the blockchain supports third-party audits of the certificate logs. Our solution can achieve instant revocation under the assumption that both the certifying entity and the certificate verifying entity connect to the same Bitcoin node or group of nodes. While the assumption that Bitcoin nodes are trusted offers a general certificate verification mechanism, an atomic verification mechanism integrated with the Bitcoin system at the protocol level inherits the trust from the system. The limitation that a certificate can only be verified once is mitigated by having multiple outpoints representing the same certificate.

The commonalities between a certificate management system and the Bitcoin system allow us to delegate a significant amount of work to the Bitcoin system and achieve great savings for the certificate management system. The extra gain is the security that is induced by the proof of work, which prevents the history from being malleated even when CA's private key is compromised.

ACKNOWLEDGEMENTS

The authors would like to thank K. Molloy, M. S. Kiraz and O. Vaughan for their invaluable contributions to the paper. In addition, the authors gratefully acknowledge the reviewers of the SECRYPT conference for their insightful feedback. C. Tartan would like to thank T. Tartan and E. Parry for the fruitful discussions. W. Zhang would like to thank J. Zhang and A. Zhu for their sweet support.

REFERENCES

Al-Riyami, S. S., & Paterson, K. G. (2003). Certificateless Public Key Cryptography. *International Conference on the Theory and Application of Cryptology and Information Security* (pp. 452-473). Springer.

Axon, L., & Goldsmith, M. (2017). PB-PKI: A Privacy-Aware Blockchain-Based PKI. *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications* (pp. 311-318). SECRYPT.

Bitcoin SV. (2021). *Bitcoin SV node software.* Retrieved from Github: https://github.com/bitcoin-sv/bitcoin-sv

Blagov, N., & Helm, M. (2020). State of the Certificate Transparency Ecosystem. *Network Architectures and Services*, 43-48.

Blockchair. (2020). *Bitcoin SV block with over 1.3 millions transactions*. Retrieved from Blockchair: https://blockchair.com/bitcoin-sv/block/635141

Blockchair. (2021). *Bitcoin SV Average Transaction Fee*. Retrieved from Blockchair: https://blockchair.com/bitcoin-sv/charts/average-transaction-fee-usd

Boneh, D., & Franklin, M. (2001). Identity-Based Encryption from the Weil Pairing. *Annual International Cryptology Conference* (pp. 213-229). Springer.

Boneh, D., Lynn, B., & Shacham, H. (2001). Short Signatures from the Weil Pairing. *International Conference on the Theory and Application of Cryptology and Information Security* (pp. 514-532). Springer.

Callas, J., Donnerhacke, L., Finney, H., Shaw, D., & Thayer, R. (2007, November). *RFC 4880: OpenPGP Message Format.* Retrieved from Request for Comments. IETF.: https://tools.ietf.org/html/rfc4880

CERT Division. (2001). *2001 CERT Advisories.* Software Engineering Institute, Carnegie Mellon University. Retrieved from https://resources.sei.cmu.edu/asset_files/WhitePaper/2001_019_001_496192.pdf

Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., & Polk, W. T. (2008, May). *RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.* Retrieved from Request for Comments. IETF.: https://tools.ietf.org/html/rfc5280

Council of EU. (2018). *Factsheet - 5th Anti Money Laundering Directive* . Retrieved from Official Website of European Union: https://ec.europa.eu/info/files/factsheet-main-changes-5th-anti-money-laundering-directive_en

DigiCert. (2021). *SCT Delivery*. Retrieved from Certificate Transparency: https://www.digicert.com/faq/certificate-transparency/enabling-ct.htm

Eastlake, D. 3. (2011, January). *RFC 6066: Transport Layer Security (TLS) Extensions: Extension Definitions.* Retrieved from Request for Comments. IETF.: https://tools.ietf.org/html/rfc6066

Etherscan. (2020). *Ethereum Average Gas Price Chart*. Retrieved from Etherscan: https://etherscan.io/chart/gasprice

Fromknecht, C., Velicanu, D., & Yakoubov, S. (2014, May). *CertCoin: A NameCoin Based Decentralized Authentication System*. Retrieved from Technical Report MIT: https://courses.csail.mit.edu/6.857/2014/files/19-fromknecht-velicann-yakoubov-certcoin.pdf

Galbraith, S., Paterson, K., & Smart, N. (2008). Pairings for Cryptographers. *Discrete Appl. Math. 156 (16)*, 3113-3121.

Gennaro, R., & Goldfeder, S. (2018). Fast Multiparty Threshold ECDSA with Fast Trustless Setup. *Conference on Computer and Communications Security* (pp. 1179-1194). ACM SIGSAC.

Google. (2013). Retrieved from Certificate Transparency: https://sites.google.com/site/certificatetransparency/

Google. (2021). *Working together to detect maliciously or mistakenly issued certificates*. Retrieved from Certificate Transparency: https://certificate.transparency.dev/

Hallam-Baker, P. (2015, October). *RFC 7633: X.509v3 Transport Layer Security (TLS) Feature Extension.* Retrieved from Request for Comments. IETF.: https://tools.ietf.org/html/rfc7633

Hoogstraaten, H. (2012). *Black Tulip Report of the investigation into the DigiNotar Certificate* . Technical Report. Fox-IT BV.

IdenTrust Services. (2021, March). IdenTrust Global Common Certificate Policy. Page 91. Retrieved from https://www.identrust.com/sites/default/files/resources/IGC-CP-v1.5.3_03012021.pdf

Kubilay, M. Y., Kiraz, M. S., & Mantar, H. A. (2019).

CertLedger: A New PKI model with Certificate Transparency Based on Blockchain. *Computers and Security, 85*, 333–352.

Kubilay, M. Y., Kiraz, M. S., & Mantar, H. A. (2020). KORGAN: An Efficient PKI Architecture Based on PBFT Through Dynamic Threshold Signatures. *The Computer Journal*, 1-23.

Langley, A. (2012, January). *CRL Set Tools*. Retrieved from GitHub: https://github.com/agl/crlset-tools

Langley, A. (2015, March). *Maintaining digital certificate security*. Retrieved from Google Security Blog: https://security.googleblog.com/2015/03/maintaining-digital-certificate-security.html

Larisch, J., Choffnes, D., Levin, D., Maggs, B. M., Mislove, A., & Wilson, C. (2017). CRLite: A Scalable System for Pushing All TLS Revocations to All Browsers. *2017 IEEE Symposium on Security and Privacy (SP)* (pp. 539-556). IEEE.

Laurie, B., Langley, A., & Kasper, E. (2013, June). *RFC 6962: Certificate Transparency.* Retrieved from Request for Comments. IETF.: https://tools.ietf.org/html/rfc6962

Lynch, V. (2018, April). *Scaling CT Logs: Temporal Sharding*. Retrieved from DigiCert: https://www.digicert.com/dc/blog/scaling-certificate-transparency-logs-temporal-sharding/

Matsumoto, S., Szalachowski, P., & Perrig, A. (2015). Deployment Challenges in Log-Based PKI Enhancements. *Proceedings of the Eighth European Workshop on System Security* (pp. 1-7). ACM.

Merkle, R. C. (1979). *US Patent No. US4309569A.*

nChain. (2020). *BRFC-Miner ID.* Retrieved from Bitcoin SV Specs: https://github.com/bitcoin-sv-specs/brfc-minerid

nChain. (2021). *BRFC-mAPI*. Retrieved from Bitcoin SV Specs: https://github.com/bitcoin-sv-specs/brfc-merchantapi

Pettersen, Y. (2013, June). *RFC 6961: The Transport Layer Security (TLS) Multiple Certificate Status Request Extension.* Retrieved from Request for Comments. IETF.: https://tools.ietf.org/html/rfc6961

Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., & Adams, C. (2013, June). *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP.* Retrieved from Request for Comments. IETF.: https://tools.ietf.org/html/rfc6960

Scheitle, Q., Gasser, O., Nolte, T., Amann, J., Brent, L., Carle, G., Wählisch, M. (2018). The Rise of Certificate Transparency and Its Implications on the Internet Ecosystem. *Internet Measurement Conference* (pp. 343-349). Boston: ACM SIGCOMM.

Southurst, J. (2021, January). *Interview with Brad Kristensen.* Retrieved from CoinGeek: https://coingeek.com/its-over-9000-tps-bitcoin-sv-hits-new-transactions-per-second-record/

Van der Meulen, N. (2013). DigiNotar: Dissecting the First Dutch Digital Disaster. *Journal of Strategic Security, 6(2)*, 46-58.

Wazan, A. S., Laborde, R., Chadwick, D., Venant, R., Benzekri, A., Billoir, E., & Alfandi, O. (2020). On the Validation of Web X. 509 Certificates by TLS Interception Products. *IEEE Transactions on Dependable and Secure Computing*, 1-1.

Wilson, D., & Ateniese, G. (2015). From Pretty Good to Great: Enhancing PGP using Bitcoin and the Blockchain. *International Conference on Network and System Security* (pp. 368–375). Springer.

Wood, G. (2014). *Ethereum: A Secure Decentralised Generalised Transaction Ledger Petersburg*. Retrieved from https://ethereum.github.io/yellowpaper/paper.pdf

Yakubov, A., Shbair, W. M., Wallbom, A., Sanda, D., & State, R. (2018). A Blockchain-Based PKI Management Framework. *The First IEEE/IFIP International Workshop on Managing and Managed by Blockchain (Man2Block) colocated with IEEE/IFIP NOMS* (pp. 1-6). IEEE.

Zhang, R., Xue, R., & Liu, L. (2019). Security and Privacy on Blockchain. *ACM Computing Surveys, 52(3)*, 1-34.

Zhu, L., Amann, J., & Heidemann, J. (2016). Measuring the Latency and Pervasiveness of TLS Certificate Revocation. *International Conference on Passive and Active Network Measurement* (pp. 16-29). Springer.