A Distributed Public Key Infrastructure for the Web Backed by a Blockchain

Report for the final evaluation of the project work of course
CS391: Design Project

by

Group 9

Al Kahaf Ahmad - 202011007
Ashish Kumar Gupta - 202011013
Avichal Bansal - 202011015
Hrishubh Bhandari - 202011026
Sanskar Patel - 202011054

Under the guidance of

Dr. Gaurav Pareek Sir

IIITV-ICD

Department of Computer Science and Engineering

Indian Institute of Information Technology Vadodara
International Campus Diu

November 2022

Contents

1 Introduction

There are countless of electronic devices around us, ranging from desktop computers and servers to smartphones and e-Passports. Many of these devices need a digital identity which can be verified as legitimate. Without a trusted digital identity, there is no way of authenticating the remote party, and it becomes impossible to establish a secure communication channel. Digital identity is most commonly managed using something called certificates, a digital document containing an identifier, a public key and a digital signature created by a trusted Certificate Authority (CA). If the communication is facilitated over the Internet, the identifier is usually an IP-address or a domain name which can be resolved through DNS. The validity of the information in the certificate can be checked by inspecting the signature of the CA using the CA's public key. The CA's public key is stored in a file (which is typically bundled with the operating system or the web browser) containing a list of trusted CAs and their corresponding public keys.

2 Background

2.1 Public key infrastructure

In this section we cover the basics of an X.509 public key infrastructure (X.509 PKI), a PKI with one or more trusted certificate authorities who are binding names to keys by issuing certificates in the X.509 format.

Definition 1 (Public key infrastructure). A public key infrastructure is a set of entities, policies and procedures used to issue, manage and revoke (name, key) pairs used for authentication.

Definition 2 (Certificate). A certificate is a signed digital document which binds a name to a public key.

2.1.1 Certificate Authorities

The certificate authority (CA) is the entity responsible for issuing certificates. These certificates are signed by the CA, and relying parties can check the validity of the signature using the corresponding public key stored in a file, called the truststore. Certificate authorities can usually be either a root CA or an intermediary CA. The root CA issues a certificate to the intermediary CA, which allows the intermediary CA to sign certificates on behalf of the root CA. This is a common practice, and is done both for practical and security reasons. There are typically many different intermediary certificate authorities, with different trust levels, issuing different types of certificates. For all intermediary CAs acting on behalf of the same root CA, only one certificate needs to be stored in the client's truststore, namely the certificate of the root CA, called the root certificate. The root CA is typically kept offline, unless it is needed to create or revoke an intermediary CA. This adds an extra layer of security, since if the intermediary CA gets compromised, which could happen if an adversary gets hold of their private key, the root CA can go online and revoke trust in the intermediary CA, without each client having to switch out a certificate in their truststore as shown in figure 1.

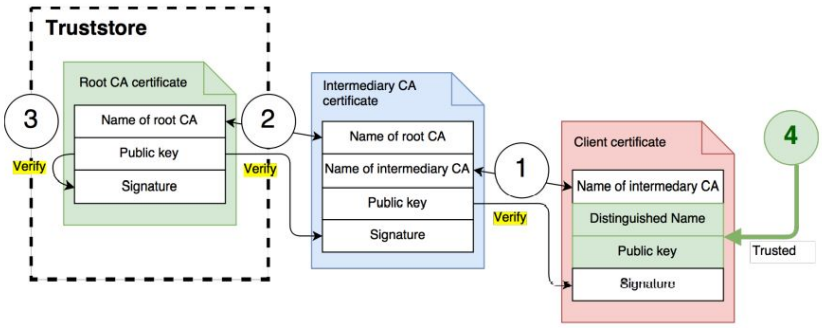

Figure 1: A certificate chain with three certificates consisting of a root certificate (green), an intermediary certificate (blue) and a leaf certificate (red). The public key for the Distinguished Name (DN) in the certificate is trusted 4 if the signatures in the chain are valid. To validate the certificate chain, the client has to check the signature of the leaf certificate using the public key of the intermediary CA 1 , and check the signature of the intermediary CA using the public key of the root CA 2 . The root certificate is self-signed 3 .

2.1.2 Certificate Issuance

When a certificate is to be issued, the client creates a Certificate Signing Request (CSR) containing the information which is to be signed by the CA, such as Common Name (CN), email address, company name, department, address and public key, and sends this information to a Registration Authority (RA). The RA is an entity approved by the CA, which helps to apply for, approve, reject and revoke certificates [1].

Once the RA has asserted the validity of the information in the CSR, it contacts the CA which in turn stamps the certificate with a date of expiry and signs the certificate with its private key. The certificate is then sent back to the client [2].

2.1.3 Revoking certificates

A certificate can be revoked by the CA who issued the certificate after a request from an authorised person, such as the domain owner. A certificate can be revoked for multiple reasons, for example if the certificate belongs to a company which has gone out of business, or if the private key of the certificate has leaked. A certificate can also be revoked without authorisation from the owner of the certificate, which might happen if the certificate was issued by accident. When a certificate has been revoked, it is important to notify clients to no longer trust the certificate. This is usually done through one of two mechanisms: a Certificate Revocation List (CRL) [3] or through the Online Certificate Status Protocol (OCSP) [4]. Revocation services are provided directly by the CA or by contacting an external entity called a Validation Authority (VA), authorised to inform about the revocation status of certificates on behalf of the CA

2.1.4 X.509 Certificates

The most common format for certificates is defined in the X.509 standard and certificates following this format are called X.509 certificates [5]. The format is shown in Figure 2. The fields of a certificate are:

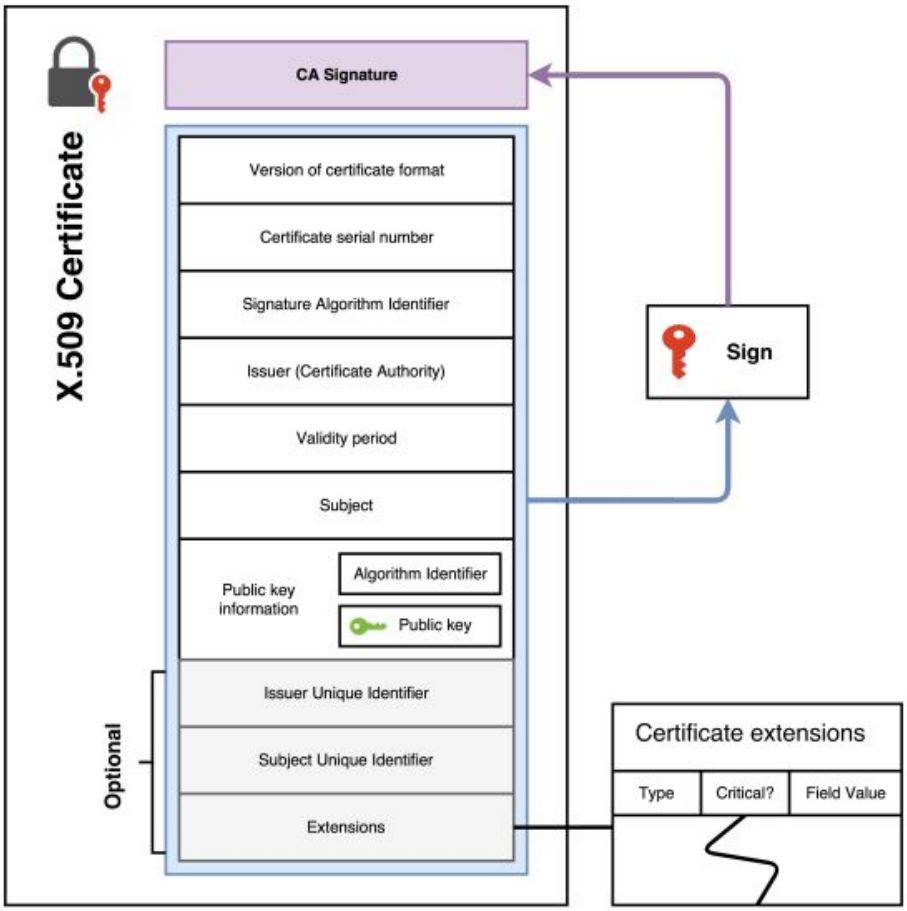

Figure 2: An image depicting version 3 of the X.509 certificate format. The public key information is trusted for the specified subject (certificate holder) if the CA signature is valid under the public key belonging to the issuer specified in the certificate, the certificate is presented within the specified validity period, and certificate has not been revoked.

1. Certificate Serial Number Uniquely identifies a certificate issued by the CA.
2. Signature Algorithm Identifier Contains the name and parameters of the signature algorithm used by the CA to construct the CA signature.
3. Issuer The X.500 name of the certificate authority who has signed the certificate.
4. Validity period Contains a start and expiry date which defines the period where the certificate should be considered valid.
5. Subject The distinguished name (DN) of the entity who owns the private key corresponding to the public key in the certificate.
6. Public key information Contains the public key of the subject together with the algorithm and parameters used to construct the key.
7. Extensions Added in X.509 version 3 and contains a list of certificate extensions.

2.2 Blockchains

A blockchain is an append-only public ledger replicated among all nodes in a large P2P-network, originally designed to store financial transactions for the Bitcoin cryptocurrency. The design of the Bitcoin blockchain was proposed by a pseudonym named Satoshi Nakamoto in 2009 [6], and since then several blockchains designed for different purposes have emerged. The blockchain offers consensus among entities in a decentralised network, effectively solving the problem of double spending explained in Section 2.4.5. Physical money, such as bank notes, cannot be spent twice since they can only be in one place at a time. With the introduction of digital money in the form of credit cards, the double spending problem was solved by a clearing house, a central authority which approves transactions and keeps track of the balances for each account. Blockchains solves the double spending problem for digital money without a central authority. Instead, the central authority is replaced by an open, dynamic and decentralised P2P-network, where each node in the network keeps its own copy of the blockchain. Transactions are broadcast on this network and recorded on the blockchain, which can be thought of as a digital billboard, used by each participating party, to independently verify that a person has coverage for their expenditures. Due to the way blocks are added to the blockchain, once a block has been added it cannot easily be removed, hence it is hard to revert or change a transaction after it has been sent to the network. A blockchain, in its original design, is a ledger which consists of blocks, each block contains a block header and a list of transactions. The list of transactions is linked to the block header through the root hash of a Merkle tree with the transactions as leaves. Each block header also contains a hash of the previous block, which links blocks to its predecessor in an ever-growing chain. Transactions are packaged and included in a block which is appended to the chain by a node in the network selected according to a consensus algorithm. When a new block is created it is broadcast over the network, and participating nodes update their own local copy of the blockchain with the new block. Transactions are confirmed by consecutive blocks being appended, and a fork may be created if two blocks are added at approximately the same time. How to resolve a fork is determined by the underlying consensus algorithm.

2.2.1 Financial transactions on a blockchain

Bitcoin features a sequential transaction model which is used to transfer money between users in the system. Each transaction contains of a list of inputs $\mathbb{I}$, and a list of outputs $\mathbb{O}$. An output $o \in \mathbb{O}$ consists of an amount $|o|$ of coins bundled with a locking script σ_L which puts an encumbrance on the output which has to be fulfilled in order to spend it. The most common encumbrance is to present a public key and signature with the corresponding private key, referred by hash in the locking script, which is called a *Pay to Public Key Hash* (P2PKH) *transaction*. An input $i \in \mathbb{I}$ consists of a reference to an output, and an unlocking script σ_U which fulfills the encumbrance defined in the referred locking script, more formally $\sigma_L(\sigma_L) = \text{TRUE}$. The difference between the sum of coins

in the input and the sum of coins in the output

$$\sum_{i \in \mathbb{I}} |i| - \sum_{o \in \mathbb{O}} |o|$$

is indirectly interpreted as a *transaction fee* collected by the miner who includes the transaction into the blockchain. An output is considered spent if it has been referred by a valid input a subsequent transaction. A node who keeps track of all transactions on the blockchain maintains a set of outputs which has not been spent yet, called *Unspent Transaction Outputs* (UTXO), determining the distribution of money in the system

2.2.1 Simple Payment Verification

Simple Payment Verification (SPV) is a method used to verify transactions without storing the whole blockchain. Clients relying on SPV are called SPV *clients* or *thin clients*. SPV is important for low powered devices with limited processing and storage capabilities such as smartphones and laptops. A thin client typically only downloads and verifies the block headers, which are small in size and can be verified quickly. Block headers belonging to the longest chain are assumed to be hard to fabricate since they require Proof of Work, and are used as a trusted source of information, which is used to verify transactions. A thin client verifies the existence of a set of transactions $\mathbb{T}$, by submitting a *Bloom filter* containing $\mathbb{T}$ to the network. The network answers with a set of transactions matching the Bloom filter, together with a set of Merkle proofs, which proves that a transaction was included in a specific block [7].

2.2.2 Proof of Work

Proof of Work, also called *Nakamoto consensus* is the consensus algorithm which is used by a number of blockchains including Bitcoin, Lightcoin and Ethereum. In Proof of Work, there are machines working on solving a time puzzle, called *miners*. The next *block leader* becomes the machine who first solves the puzzle. The puzzle is constructed in such a way that it is hard to solve but easy to verify, which in cryptography is called a *trapdoor function*. Bitcoin uses a variant of hashcash, which originally was intended to be used to limit spam in email systems. In Bitcoin, the puzzle to solve is to find a SHA2 hash value SHA2^2 (block header) such that SHA2^2 (block header) $< 2^{(\text{n-k})}$, where n = 256 is the number of output bits in the SHA2 hash function and k is a difficulty factor, collectively determined by the nodes in the network every 2016 blocks, such that on average a new block in appended to the blockchain every 10 minutes [7].

The *longest chain* in Bitcoin's Proof of Work is considered to be the blockchain with the most accumulated difficulty $\sum k$. When a fork is created, a copy of both chains are kept, until the fork is resolved which typically happens when the next block is added, making one of the chains longer than the other [8]. Miners are incentivised to add new blocks to the blockchain by collecting transaction costs, and by the *block reward* which is paid to the address found in the first transaction in the block, called the *coinbase transaction*. The coinbase transaction does not transfer money from anyone, but creates

new money which can be spent after the block has enough confirmations [4].

3 Methodology

This chapter contains the design goals we attempt to achieve, explains our approach for solving the problem and defines the entities in our proposed PKI.

3.1 Design Goals

In choosing how to proceed, we need to consider the design goals we want to achieve.

1. **Identity retention** An identity should, to the furthest extent possible, only be able to be issued, changed or revoked after permission from the person or organisation owning the identity. The current PKI achieves this through strict vetting rules where a CA requires proof in form of paperwork, or physical presence of an employee acting on behalf of the organisation before a certificate is produced. These procedures occasionally fail, due to not being carried out properly or not at all, which can result in impersonation. We believe a stronger guarantee for identity retention of already existing identities can be enforced by public key cryptography, where new certificates need to be signed by the organisation's private key.
2. **Expiration of old identities** There should be a mechanism where identities are renewed on a regular basis, and old identities are purged. This reduces storage requirements, since identities which are no longer active can be forgotten, and avoids "locking up" names in our naming system.
3. **Transparency** Anyone should be able to connect and retrieve a "global" view of the all identities in the system, and being able to verify every transaction. This is important to be able to detect fraudulent behavior, without relying on a central authority.
4. **Scalability** The system needs to scale in a reasonable way when more and more identities are registered. Namely, storage and bandwidth requirements should be met by consumer off-theshelf hardware, and the system should be able to handle enough transactions to scale globally.
5. **Backward compatible** Processes for issuance, revocation and verification of names and public keys should be compatible with the current PKI if possible.
6. **Thin client support** The system should support thin clients, to permit certificate verification for low-powered devices such as smartphones.

3.2 System Model

The entities involved in our distributed PKI are clients and servers trying to establish secure connections among themselves, and a set of blockchain nodes operating a large P2P-network as described in Figure 3. Some of these blockchain nodes are stakeholders, and may be chosen to approve transactions sent to the network.

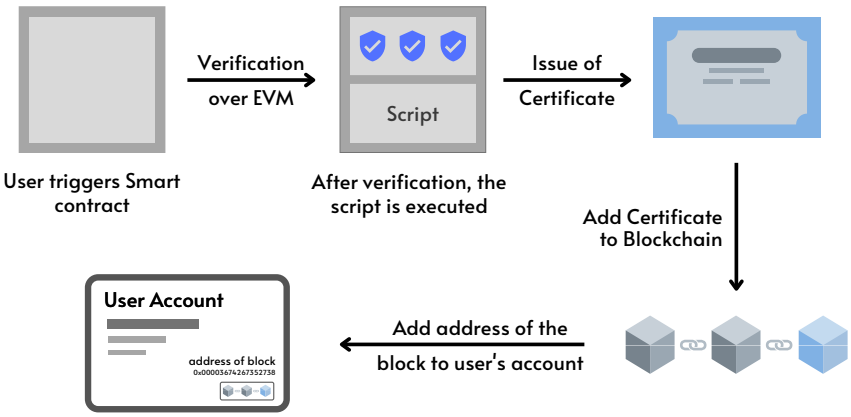

Figure 3: System Model

1. **Stakeholder** A semi-trusted entity such as a government, CA or browser vendor authorised to maintain the blockchain.
2. **Blockchain CA** A CA approved by the stakeholders eligible to approve new transactions processed by the stakeholders.
3. **Validator** A third party who monitors the blockchain network to ensure everyone follows the protocol.
4. **Thin client** The client of an end-user (such as a web browser) with limited processing, storage and bandwidth abilities.
5. **Domain owner** An owner of a DNS domain name, running a web service.

4 Blockchain Design

This section covers the design of our blockchain scheme. This includes a description of the format of transactions, blocks and accounts.

4.1 Root CA

A parent contract named the CertificatePKI is the Root CA of the system. Certificate PKI is a secured contract which can not be modified by anyone once it is deployed.

This smart contract contains the mapping of all the users and their respective certificates using the selenium mapping. This parent smart contract has access to all the certificate creation and deletion algorithms and all the deployments in the system happens through this smart contract.

4.2 Validation Authority

The smart contract of selenium (v0.4.17) used for creating the certificates acts as the validation authority. These contracts are responsible for verifying the authenticity of the user/device and to make sure that the entered public key belongs to the user himself.

To verify the public key, the system asks the user to encrypted hash of their account number encrypted using his private key. Then the system takes the users public key and decrypts the hash. If the result thus received is actually equal to the account number of the user, we can safely say that the public key the user has provided actually belongs to him.

4.3 Certificates Authorities

The miners of the ethereum blockchain act as the certification authority in the system. Their task is to create the certificates using the information provided by the users and then add the certificates this generated to the blockchain itself. Once the contract is added to the blockchain, the miner then adds the certificates address on the blockchain to the mapping of the parent smart contract to keep a track of all the smart contracts thus generated.

4.3.1 Certificate Issuance

For issuing a certificate in his name, the user first generates a CIR, i.e, a Certificate Issuance Request. Once this request is verified by the smart contract, along with the details of the user, the contract is then sent to the ethereum miners for generating the certificates.

The miner takes the information the user has provided and creates a certificate using that information and adds the certificate to the blockchain. In order to add this block to the blockchain, the miner adds a certain code, called the "block verification code" which enables the block to be added to the blockchain. Since this code can only be generated by the miners, it also acts as the miners signature on the certificate.

Once all of this is done, the address of the certificate this created is tied to the users account address and this pairing is added to the mapping to the Certificate PKI contract.

4.3.2 Revoking Certificate

A certificate can be revoked by the root CA which is a smart contract in our model. A certificate can be revoked upon request of the user or when the validity of the certificate has expired. In blockchain, we can not delete or remove the pre-existing certificates. To mitigate this, we are using a "delayed smart contract". This smart contract runs when any of the above condition is met. In practice, this smart contract will change the state of an existing certificate on the as expired and change its timestamp to 0. Later when the user will try to issue another certificate on their account the smart contract will reset the time stamp.

4.4 Security Analysis

5.1.1 51% Attack

A 51% attack is when a cryptocurrency miner or group of miners gains control of more

than 50% of a network's blockchain. The 51% attack scenario is rare — especially for more established cryptocurrencies — mainly because of the logistics, hardware, and costs required to carry one out. But a successful block attack may give the attacker complete control of the network and allows them to double-spend coins, prevent other transactions from confirming, and block other miners from mining.

Even if this kind of attack occurs only the user itself or the parent contract i.e. can modify a contract. The contract in itself cannot be modified.

5.1.2 Double Spending Attack

Double spending means the expenditure of the same digital currency twice or more to avail the multiple services. It is a technical flaw that allows users to duplicate money. Since digital currencies are nothing but files, a malicious user can create multiple copies of the same currency file and can use it in multiple places. This issue can also occur if there is an alteration in the network or copies of the currency are only used and not the original one. There are also double spends that allow hackers to reverse transactions so that transaction happens two times.

The contracts built in our model prevents such attacks. When any attacker tries to duplicate any certificate, the old certificate will get renewed under the same account holder. The old certificate will get expired.

4.5 Limitation

The current system only enables one user to issue one certificate in their name. Even if someone tries to issue a second certificate to their name, the first certificate is lost in the process. Apart from this, the system in its current form is only compatible with RSA type key pairs and further algorithms need to be added to the system.

4.6 Future Work

One possible Avenue of future work can be adding the support for issuing multiple contracts to a single user. This can increase the scope of the PKI system as it would then be applicable across the board for any kind of verification certificate. This can be done using some elements of web2 where we can store different objects containing details of all the certificates issued to a particular user.

Finally, improvements can be done to shift the PKI from the *Proof of Work* (POW) based blockchain to the Proof of Stake based blockchain. The major hindrance in this would be the low cost of creating a certificate. Since the cost of certificate creation cannot be too high due to practical reasons, we need to come up with other incentives for the miners to mine these certificates instead of other smart contracts on the blockchain. This means that we might have to create a separate blockchain system for these certificates and add other incentives for the miners which are simply not there in the ethereum based system.

5 Conclusion

Although the use of a blockchain as an authoritative source of information for the web sounds tempting, mainly due to its strong security guarantees, it is not without drawbacks. The current system only enables one user to issue one certificate in their name. Even if someone tries to issue a second certificate to their name, the first certificate is lost in the process. The system in its current form is only compatible with RSA type key pairs and further algorithms need to be added to the system.

References

[1] I. Eyal, A. E. Gencer, E. G. Sirer, and R. van Renesse, "Bitcoin-ng: A scalable blockchain protocol," 2015.

[2] K. Davarpanah, D. Kaufman, and O. Pubellier, "Neucoin: the first secure, cost-efficient and decentralized cryptocurrency," 2015.

[3] L. coronado garcía, "On the security and the eciency of the merkle signature scheme," *IACR Cryptology ePrint Archive*, vol. 2005, p. 192, 01 2005.

[4] R. Dahlberg, T. Pulls, and R. Peeters, "Efficient sparse merkle trees - caching strategies and secure (non-)membership proofs," in *NordSec*, 2016.

[5] C. Evans, C. Palmer, and R. Sleevi, "Public Key Pinning Extension for HTTP," RFC 7469, Apr. 2015.

[6] R. Housley, W. T. Polk, W. S. Ford, and D. Solo, "Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile," *RFC*, vol. 3280, pp. 1–129, 2002.

[7] I. Bentov, C.-T. Lee, A. Mizrahi, and M. Rosenfeld, "Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract]y," *SIGMETRICS Perform. Evaluation Rev.*, vol. 42, pp. 34–37, 2014.

[8] J. D. Bruce, "The mini-blockchain scheme," 2014.