Securely Migrating Digital Identities from a Class PKI to a Blockchain

Bastian Fredriksson[฿]
Royal Institute of Technology
114 28, Lindstedtsvägen 3
Stockholm, Sweden
bastianf@kth.se

Keywords

Certificate authority, Digital identity management, PKI, Blockchain

ABSTRACT

The current hierarchical public key infrastructure used to authenticate machines on the internet is deeply flawed. A single certificate authority can compromise the security of the whole system. A possible way of resolving this problem would be to keep track of identities using a blockchain. In this paper we propose a protocol for migrating an identity in the current class PKI to a blockchain, by pinning a certificate issued by a certificate authority on the blockchain, and signing the transaction with the client's private key. The protocol is executed between a client and a node administering a virtualchain, operating on top of an existing blockchain. In order to validate transactions on the blockchain, we introduce the notion of a blockchain truststore. We also show how security improvements to TLS, such as Certificate Transparency, Public Key-pinning and OCSP stapling can be used to strengthen our protocol. To ensure that a migration is performed with the consent of the owner, an additional step for extended validation certificates is proposed, where a certificate authority needs to attest to a registration being performed correctly.

1. INTRODUCTION

1.1 Blockchains

Blockchains was originally introduced as a way of storing transactions for the Bitcoin cryptocurrency[15]. A blockchain is a public ledger shared among all nodes, called miners, in a large P2P-network. For miners to append a new block to the Bitcoin blockchain, one needs to provide a proof-of-work $\texttt{SHA2}^2(s|c)$ where s is a service string containing the information encoded in the block header and c is a 32-bit counter such that $\texttt{SHA2}^2(s|c) < 2^{(n-k)}$ 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[2]. Other proof systems such as proof of stake, proof of burn and proof of storage has been proposed. A list of transactions is linked to a block by putting a Merkle root hash in the block header. This makes it possible to validate a blockchain using only the block headers, while the actual transactions can be stored elsewhere.

Miners participating in building the blockchain will always try to continue on the longest chain in terms of difficulty. If two miners find a new block at approximately the same time, two chains of equal difficulty are created. Only one of these two blocks will survive, meaning that blockchains does not need to be consistent across the network. Blocks are *verified* by consecutive blocks being appended. The probability of a block to remain in the blockchain increases exponentially with the number of consecutive blocks, and a block is typically considered *confirmed* after about six consecutive blocks have been appended.

There are several systems based on blockchains, including Namecoin[12] a replacement for DNS running on its own blockchain, Blockstack[1] a blockchain agnostic, decentralized DNS and PKI, and Certcoin[4] a PKI forked off from Namecoin, Storj[18] a decentralized cloud storage solution, and Ethereum[6] a system which powers *smart contracts*, programs excecuted on a blockchain.

1.2 Class PKI

Identity on the internet today, is largely[1] administered by a hierarchical PKI (or *class PKI*) consisting of certificate authorities (CA) which issues X.509 certificates. The information contained in these certificates are signed by the CA, using a public-key cryptosystem (typically RSA). Certificates no longer in use, or otherwise compromised through leakage of the client's private key, must be revoked. Current revocation mechanism involves use of certificate revocation lists (CRL)[5] and validation of a certificate through a protocol called *Online Certificate Status Protocol* OCSP[16].

The highly centralized nature of the current PKI system has several disadvantages. First and foremost, a CA constitutes a single point of failure, which makes it an obvious target for attacks. One compromised CA will endanger the trust of the whole system, as shown by

[1]Another mechanism of trust which is commonly used, e.g by Linux packet managers, is Web of Trusts which is described in the OpenPGP standard[3].

ANSS '17 Jan 1–31, 2017, Stockholm, SE
© 2017 ACM. ISBN 000-0000-00-000/00/00...$13.37
DOI: 00.000/000_0

the DigiNotar accident[11], the Comodo hack[8] and the TürkTrust incident[14]. Thus, the current PKI system is susceptible to invisible mitm-attacks, where a third party intercepts and decrypts traffic en route. Secondly, it is possible for an identity to be linked to several public keys, which complicates the revocation process and leads to poor identity retention.

1.3 Issuing certificates in a class PKI

A X.509 certificate, associated with a cryptographic keypair and a *common name* (CN), is typically issued by a certificate authority once a user proves that she controls a specific domain, usually by responding to a mail sent to the domain owner. These certificates will be referred to as domain validated certificates (*DV Certificates*). Another type of certificate, called extended validation certificates (*EV Certificates*) are typically harder to obtain since the CA has strict procedures in place which requires them to verify the physical identity of the person or organization who receives the certificate.

While issuing certificates mostly is an ad hoc process, some CAs offers a more streamlined process where DV certificates are issued and installed automatically using a web client. One such solution is Certbot which implements the Automated Certificate Management (ACME) protocol, used by the Let's Encrypt certificate authority.

The ACME client creates a CSR which is sent to the certificate authority. Messages are encoded in JSON and protected by HTTPS. The client executes a challenge response protocol to prove knowledge of the secret key corresponding to the public key in the CSR. Upon successful response, the certificate authority answers with a JSON message containing a link to the signed certificate[9].

1.4 Blockchain-based PKI

Blockchains looks like a possible replacement for the current class PKI. It would solve the problem of CAs being a single point of failure since the blockchain is distributed making it more resilient against denial of service. Man in the middle-attacks would be efficiently mitigated since the public key is pinned on the blockchain, which in turn would lead to better identity retention.

A blockchain-based approach also leads to new challenges. For example, although the distributed nature of the blockchain should make it scale better than a corresponding class PKI - the number of transactions which can be processed will be limited to the average network bandwidth of the nodes participating in the network. The block size is currently fixed to at most 1 MB in the Bitcoin network, effectively limiting the number of transactions which can be included in a block. In practice, the Bitcoin network is able to process about 3 transactions every second.

Another problem is the *endless ledger problem* which arises from the nodes in the network being forced to keep track of all previous blocks, making it expensive to run a full node. For example, the current size of the Bitcoin blockchain (including transactions) is about 96 GB, growing at a rate of about 10 GB/month[2].

Anyone who designs a blockchain-based PKI which intend to replace or coexist with an existing PKI must also deal with the problem of an adversary maliciously registering an identity on the blockchain which is already registered in the class PKI. This is the issue which will be investigated in this paper.

2. RELATED WORK

The most notable blockchain-based PKI systems are Blockstack[1] and Certcoin[4]. Blockstack is a combined PKI and DNS currently running on top of the Bitcoin blockchain, although it is possible to migrate to another blockchain, since the system itself is blockchain-agnostic. In order to provide this kind of agility, Blockstack features provides a *virtualchain* on top of the underlying blockchain being used. The virtualchain is built by nodes running the Blockstack software, which are parsing and filtering the transactions of the Bitcoin blockchain.

Certcoin is a fork of the Namecoin blockchain which features its own format for identity and transactions. Both Certcoin and Blockstack assumes the legitimate owner of an identity being the first person who registers that identity, which is similar to how DNS works.

An interesting feature of Namecoin and derivatives thereof, is that it requires a two-step registration-phase[1]. When a new domain is registered, a user first needs to *pre-order* the domain by posting $H(domain|key)$ to the blockchain. This prevents a race condition between a malicious user monitoring new domains being registered and a legitimate user trying to register a domain. After the pre-order transaction has been confirmed by the network, a user *reveals* the previously pre-ordered domain by posting the actual domain and the corresponding key to the blockchain.

Namecoin also requires a user to post two public keys to the blockchain, namely one *online key* K^1_{pub} and one *offline key* K^2_{pub}. The online key is deployed on the server or client machine and is used to encrypt data and create signatures. The offline key is kept secure offline, and is only used to revoke or transfer to a new keypair if the online key gets lost or compromised.

3. SYSTEM MODEL

The entities involved in our protocol is the client who wants to migrate a certificate from a class PKI to a blockchain, a certificate authority who is responsible for signing any certificate requests created by the client, a set of nodes operating a virtualchain, which will be called *v-nodes* for brevity, and a blockchain network, operating any sort of blockchain which allows for data to be appended to a transaction (such as the Bitcoin blockchain). The assuption of a virtualchain is not strictly required but simplifies protocol design and provides agility since no particular blockchain is assumed.

Define a registration object as a document with the following fields:

```
RegistrationObject {
    $id = I$
    $proof_{K^1K^2} = Sig_{K^1_{priv}K^2_{priv}}(I)$
    $cert = (Cert_{K'}, \Re_{OCSP})$
    $Sig_{K'_{priv}}(id|proof|cert)$
}
```

With *identity* I we will assume something like a tuple (`CN`, K^1, K^2), but we are not going to argue in favor of

[2]See https://blockchain.info/charts/blocks-size

any particular format, nor what content can be put therein, an identity could potentially incorporate anything from addresses to biometric information.

The v-nodes support the following virtualchain operations:

- `PreOrder`($H(I)$) Pre-order an identity I, bound to the two keypairs (K^1_{pub}, K^1_{priv}) and (K^2_{pub}, K^2_{priv}).

- `Register(RegistrationObject)` Registers an identity I whose CN has not been previously registered on the blockchain.

- `Release`($H(I)$) Releases an identity I back to the pool of unregistered identities.

- `Attest`($\mathbb{L} = \{H(I_X)|X \hookleftarrow [1 \dots N]\}$, $Sig_{CA_{priv}}(\mathbb{L})$) Confirm that the identities listed was correctly transferred to the blockchain.

- `Edit(CN,` Sig_1, Sig_2`...)` Change or add a public key for a CA in the blockchain truststore.

- `Remove(CN,` Sig_1, Sig_2`...)` Remove a CA from the blockchain truststore.

Identities, certificates and signatures are typically too large to post directly in the blockchain. Hence, will will assume that the v-nodes maintain a distributed hash table, for example Chord[17] or Kademlia[13], which makes it possible to store only the hash of the data in the blockchain.

4. PROTOCOL DESCRIPTION

4.1 Overview

This section describes a protocol executed between a client or a CA and network of v-nodes. The purpose of the protocol is to securely migrate an existing identity, consisting of a certificate signed by a CA, to a blockchain.

Consider a scenario where you can have your identity confirmed either by a certificate signed by a CA or by a transaction on the blockchain. A company, say Google, who already have a certificate issued by a CA, want to migrate to a blockchain due to security reasons. Since the first person who registers an identity on the blockchain is considered the owner of said identity, it would be possible to hijack Google's trademark if an adversary registers its own public key on the blockchain in Google's name before Google does. Thus there is a risk of *identity retention* being violated if a class PKI is allowed to coexist with a blockchain-based PKI.

Our protocol solves this problem by forcing v-nodes to assert that a client has a certificate issued, and that the client has knowledge of the secret key corresponding to the public key stamped in the certificate. The certificate together with the proof of the private key is pinned on the blockchain for verification. To provide extra security for companies with EV certificates, an extra step is required after identity registration where a CA is *attesting* to a certain registration being performed correctly.

We have decided to skip the pre-registration phase required by Namecoin and its derivatives, since the use of certificates makes such measures unnecessary.

4.2 The blockchain truststore

Before the blockchain is opened for write-access, a CA, for example Symantec, posts a signed document to the blockchain containing a list of all CAs currently in operation and their corresponding public keys, a list of pinned keys[7] and a list of domains which has an EV certificate issued according to a certificate transparency log[10]. We call this the *blockchain truststore*, which will be available at a well-known block number.

The truststore can be edited through the virtualchain operations `Edit` and `Remove`. Each such transaction must be signed by multiple CAs currently in the truststore to be considered valid.

The purpose of the blockchain truststore is to act as an authority for which CAs that were in operation at a certain point in time, their public keys, pinned keys and which domains that require extended validation. This information could be kept off the chain, but it would undermine the security of the system since any such information could be tampered with.

Consider a scenario where a person P has migrated a certificate signed by CA which later became compromised and declared bankrupt. A new v-node, trying to validate the blockchain after this event, would consider P's registration to be invalid since the CA is no longer in operation. However, using a blockchain truststore, new v-nodes could go back in time and determine if the CA existed at the time the block was inserted into the chain.

4.3 Migrating a DV certificate

A domain validated (DV) certificate has been issued by a certificate authority after verifying that an owner controls the specific domain. Most certificates are of this type, and they are cheaper than EV certificates. A migration process should arguably be as automated as possible to reduce costs and time for migration.

Our client protocol for migrating a DV certificate is as follows:

1. **Request a certificate** Request a certificate $Cert_{K'}$ from the CA if no certificate has been issued yet. This process can be automated using for example the ACME client mentioned in 1.3.

2. **Initialize** Generate the two keypairs (K^1_{pub}, K^1_{priv}) and (K^2_{pub}, K^2_{priv}) and create an identity I.

3. **Migrate** Compute the key signature $Sig_{K^1_{priv}K^2_{priv}}(I)$ of the identity I, under both the private online key K^1_{priv} and the private offline key K^2_{priv}. Request a time-stamped OCSP response $\Re_{OCSP}$ from the CA and bundle it with $Cert_{K'}$. Sign the identity, the key signature, the certificate and the OCSP response with the private key K' of the certificate. Create a registration object R_I from this information and post `Register(`R_I`)` to the blockchain.

When a v-node discovers a registration on the blockchain in step 4 above, it must check the following:

(R1) There must not be any valid `Register` transaction posted previously, with the CN found in I.

(R2) The signature $Sig_{K'_{priv}}(id|proof_{K^1K^2}|cert)$ must be valid.

(R3) The signature $Sig_{K^1_{priv}K^2_{priv}}(I)$ must be valid.

(R4) The CN of the certificate $Cert_{K'}$ matches the CN of the identity I. Any other information found in the certificate, such as address, which is also present in I should also match.

(R5) $Cert_{K'}$ has not expired, has a valid CA signature and is bundled with a correct OCSP response $\Re_{OCSP}$ according to the blockchain truststore.

(R6) **Additional requirement** If the client has a pin in the blockchain truststore it is imperative that the public key of the certificate matches one of the pinned keys.

(R7) **Additional requirement** If the client has an EV certificate issued according to the blockchain truststore, assert that $Cert_{K'}$ is an EV certificate as well.

The keypair (K^1_{pub}, K^1_{priv}) does not necessarily have to be the same as the key attached to the certificate. However, to avoid confusion or due to compatibility reasons, one might prefer to use the same keypair in both the class PKI and the blockchain.

4.4 Migrating an EV certificate

An extended validation (EV) certificate costs more than an ordinary DV certificate due to the increased scrutiny performed by the CA. Typically, an EV certificate validation process involves verifying that the company or organization exists, and that a request to issue the certificate was in fact made by, or on behalf of, the company in question. Thus, an EV certificate offers higher assurance that the certificate has not been issued to a fraudulent party. With this in mind, we propose an extra verification step for migration of such certificates.

The CA should monitor the blockchain for new EV certificates. The CA should ensure that the company or organization of an EV certificate posted to the blockchain wants their certificate migrated to the blockchain and that the public keys posted in the registration request are valid. On a regular basis, say every week, the CA creates a list $\mathbb{L} = \{H(I_X) | X \hookleftarrow [1 \dots N]\}$ of all vetted EV certificates, signs this list with their private key and posts `Attest(`$\mathbb{L}$`,` $Sig_{CA_{priv}}(\mathbb{L})$`)` to the blockchain.

4.5 Booting up a v-node

New v-nodes needs to build a database from virtualchain operations which maps public keys to identities before they can start processing requests. This can be achieved either by directly parsing and filtering the contents of the underlying blockchain, or through bootstrapping information from other nodes, which in Blockstack is called *Simple Name Verification* (SNV). SNV helps v-nodes figure out if they have the same view of the global state at any given block, using a consensus hash computed over a list of virtualchain operations[1].

A v-node loads the blockchain truststore, and starts processing the virtualchain operations in a chronological manner, starting at the oldest operation. A registration operation can either be *native* if it belongs to the set of CNs not managed by a CA (for example the `.bit` top domain[12]), *domain validated* if it a migration performed with a DV certificate, or *CA validated* if it is a migration performed with an EV certificate.

For a registration of I to be accepted and inserted into the v-node's database, the following conditions apply:

- ✓ **Native registrations** R1, R3 and a registration must be preceded by a confirmed `PreOrder` transaction containing $H(I)$.
- ✓ **Domain validated registrations** R1-R7.
- ✓ **CA validated registrations** R1-R7 and the registration must be succeeded by a confirmed `Attest` operation which confirms the correctness of the registration.

Note It is possible to determine whether the OCSP response was valid and whether the certificate was expired at the time it was posted to the blockchain by looking at the timestamp of the block.

5. SECURITY ANALYSIS

Assuming the security of the underlying blockchain and that a majority of v-nodes is not malicious, a migration of a certificate should be secure as long as the CA is not compromised and as long as the client's private keys are kept secret. A compromised CA could generate its own keypairs and certificate and post it to the blockchain (unless the victim employs public key-pinning). In order to steal someone else's identity, it would have to be done after the person has her identity registered, but before the identity is being migrated to the blockchain. Furthermore, a CA should not be able to invalidate previous migrations. This property is guaranteed since there is no mechanism of for a CA to revoke an identity on the blockchain. Once a migration is complete, all changes to the identity must be signed with one of the private keys K^1_{priv}, K^2_{priv} only known to the owner.

Man in the middle A registration object being posted on the blockchain could be intercepted by an attacker, who would try to exchange the two keys K^1 and K^2 with her own keys before sending the transaction to the blockchain. However, this is not possible since the registration object is authenticated with the private key of the certificate, unknown to an adversary.

Signature leakage If the client has software installed which act as a signature oracle, for example due to a software bug, an adversary could register someone else's identity with his own private keys by tricking the victim into signing $id|proof_{K^1K^2}|cert$ with appropriate values. Such an attack is only mitigated if the victim has an EV certificate in the blockchain truststore, since a CA would not attest to such a fraudulent transaction.

Reuse of signatures An adversary must not be able to collect and reuse signatures, for example by interacting with an SSH client or web server administered by the victim. Our scheme protects against reuse of signatures by forcing the client to provide a signature of $id|proof_{K^1K^2}|cert$. The OCSP response and certificate contains data which is random and not controlled by an attacker which prevents signature reuse.

Using a revoked certificate This attack is mitigated by bundling a time-stamped OCSP response with the certificate, which proves that the certificate has not been revoked. The use of OCSP stapling also makes it possible for a v-node to validate a certificate without communication with a CA.

EV downgrade attack An adversary who has gained access to the victims private key could try to steal the identity of the victim by migrating it to the blockchain (assuming the victim has not done so previously). However, if a victim has an EV certificate issued, knowledge of the private key is not enough, since a CA must attest to a migration being performed correctly. An adversary could try to avoid the extra scrutiny by posting a DV certificate instead. We prevent this by incorporating a list of domains with EV certificates into the blockchain truststore. Due to R7, any such registration would be considered invalid by a v-node. We want to stress the fact that, in our design any newly registered identities will not be protected against this kind of attack, since their CNs will not be present in the blockchain truststore. Hence, it is important to migrate the identity to the blockchain as soon as possible, preferably at the time the domain is registered.

6. CONCLUSION

We have proposed a protocol which allows a participant in a class PKI to register their identity on a blockchain, while ensuring identity retention. A fundamental building block of this protocol is to pin a certificate on the blockchain to prove that the person owns the identity, and the notion of a blockchain truststore which keeps track of CAs currently in operation. We have shown how security improvements to TLS including Certificate Transparency, Public Key-pinning and OCSP stapling can be used to guarantee the security of the migration protocol. To ensure a secure migration, even under circumstances where the private key has been exposed to an adversary, we have proposed an additional step where a CA attests to a migration being performed with the consent of the owner.

Our protocol allows an existing class PKI to coexist with a blockchain for improved security. Once a migration is complete, a web server could offer an additional way of authentication. Instead of trusting a CA, a client could look up the identity in the blockchain. In addition, our protocol could be used in a hybrid approach where a CA is used to connect a digital identity to a physical person, and transactions are performed on a blockchain.

The practicality of our approach to provide a more secure PKI depends on the underlying blockchain as well as the specific implementation. Although a blockchain offers built-in certificate transparency and public key pinning, while removing trust in third parties, it might not be a feasible solution on low-powered devices due to increased storage requirements. The blockchain approach might also allow an adversary to permanently seizure a domain from its rightful owner, if both the offline and online key leaks.

The author want to thank Mike Kushner for reviewing an earlier version of this paper. The most recent version can be found at `https://helix.stormhub.org/anss`.

References

[1] M. Ali, J. Nelson, R. Shea, and M. J. Freedman. Blockstack: A global naming and storage system secured by blockchains. In *2016 USENIX Annual Technical Conference (USENIX ATC 16)*, pages 181–194, Denver, CO, June 2016. USENIX Association.

[2] Bitcoin Wiki - Hashcash. https://en.bitcoin.it/wiki/Hashcash. Accessed: 2017-01-02.

[3] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer. Openpgp message format. RFC 4880, RFC Editor, November 2007. http://www.rfc-editor.org/rfc/rfc4880.txt.

[4] D. V. Conner Fromknecht and S. Yakoubov. A decentralized public key infrastructure with identity retention. Cryptology ePrint Archive, Report 2014/803, 2014. http://eprint.iacr.org/2014/803.

[5] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile. RFC 5280, RFC Editor, May 2008. http://www.rfc-editor.org/rfc/rfc5280.txt.

[6] Ethereum White Paper. https://github.com/ethereum/wiki/wiki/White-Paper. Accessed: 2017-01-02.

[7] C. Evans, C. Palmer, and R. Sleevi. Public key pinning extension for http. RFC 7469, RFC Editor, April 2015. http://www.rfc-editor.org/rfc/rfc7469.txt.

[8] D. Goodin. New hack on comodo reseller exposes private data. Newspaper article, The Register, 2011. http://www.theregister.co.uk/2011/05/24/comodo_reseller_hacked.

[9] K. J. Hoffman-Andrews, J. Automatic Certificate Management Environment (ACME). Internet draft, Internet Security Research Group, 2016.

[10] B. Laurie. Certificate transparency. *ACM Queue*, 12(8):10–19, 2014.

[11] J. Leyden. Inside 'operation black tulip': Diginotar hack analysed. Newspaper article, The Register, 2011. http://www.theregister.co.uk/2011/09/06/diginotar_audit_damning_fail.

[12] A. Loib. Namecoin. Technical report, Fakultät für Informatik, Technische Universität München, August 2014.

[13] P. Maymounkov and D. Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 53–65, London, UK, UK, 2002. Springer-Verlag.

[14] N. McAllister. Browser makers rush to block fake google.com security cert. Newspaper article, The Register, 2011. http://www.theregister.co.uk/2013/01/04/turkish_fake_google_site_certificate.

[15] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. May 2009.

[16] S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 internet public key infrastructure online certificate status protocol - ocsp. RFC 6960, RFC Editor, June 2013. http://www.rfc-editor.org/rfc/rfc6960.txt.

[17] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *SIGCOMM Comput. Commun. Rev.*, 31(4):149–160, Aug. 2001.

[18] S. Wilkinson et al. Storj - a peer-to-peer cloud storage network, 2014. https://storj.io/storj.pdf.