BlockCACert – A Blockchain-Based Novel Concept for Automatic Deployment of X.509 Digital Certificates

Adam Mihai Gergely(✉) and Bogdan Crainicu

University of Medicine, Pharmacy, Science and Technology "G.E. Palade", Târgu-Mureș, Romania
{adam.gergely,bogdan.crainicu}@umfst.ro

Abstract. Digital certificates (also known as X509 public-key certificates) are used to prove the identity of a client or of a server to the communicating partner and to establish an encrypted channel for secure communication. The most prominent use of digital certificates is found in the HTTPS protocol, for securing websites and clients by means of digital signatures and public-key encryption. Because digital certificates are signed by Certificate Authorities that supposedly everyone trusts, it is implicitly said that these certificates provide some degree of trustworthiness. But history shows us that Certificate Authorities can also make mistakes, issue false certificates or be compromised in some form or another. Also, currently there are multiple types of digital certificates that can be issued: DV, OV and EV, which cost differently based on the grade of trust. We believe that real trust should not be quantifiable in degrees of trust: we either trust a certificate or we don't. We propose BlockCACert, a blockchain-based solution for CAs for issuing all types of X509 certificates using the ring signatures concept, which fixes the problem of general trust and multiple-level trust by using a global procedure for issuing all types of digital public-key certificates.

Keywords: Blockchain · X509 · Digital signatures · Digital certificates · Public-key · Private-key · Certification authorities · PKI · Ring signatures

1 Introduction

Within a typical PKI (Public Key Infrastructure) deployment, digital certificates were used to prove the identity of clients and servers by using a tertiary entity known as a Certification Authority (CA).

An entity X that wanted to have a certificate issued for its identity would generate a PKI cryptographic key pair (private key and public key) and a Certificate Signing Request (CSR) using its own identification data bonded to the public-key. In order to obtain a valid signed certificate, this entity would contact a CA and request certification (usually for a fee). The CA would analyze this request, perform a validation on the requesting entity's data, and after the fee is paid, a certificate would be issued. This certificate contains the binding of the CA's digital signature with the CSR resulting in a

L. Moldovan and A. Gligor (Eds.): Inter-Eng 2021, LNNS 386, pp. 820–832, 2022.
https://doi.org/10.1007/978-3-030-93817-8_73

final certificate. The entity would use the certificate throughout the certificate's validity period. After the certificate expires, entity X can apply for recertification by applying for a certificate renewal. If there are no validation or identity proofing problems, the issuing CA reissues a new certificate for the same entity. At any given time, if the entity requests a certificate revocation or the CA finds out that the entity is not eligible for certification (in case of major problems or frauds), the CA can revoke the entity's certificate.

The procedure described above is known as the Standard or Traditional PKI. This form of traditional PKI gives a certain amount of control to the Certification Authorities and assumes that these CAs are honest and do not cheat. However, history shows us that there have been successful attacks on CAs and certificates were issued by fraud.

Another type of issue which we consider a problem is the multiple level trust. We believe that trust exists fully, not partially. Currently, there are multiple main levels of trust: DV (domain validation), OV (organization validation) and EV (extended validation). We believe that having a single type of trust eliminates the need for multiple-level trust mechanisms and helps to avoid confusion regarding trust.

This paper presents a theoretical proof-of-concept and is organized as follows:

In Sect. 2 we briefly discuss about the main components of the PKI ecosystem. Section 3 summarizes the current problems of trust regarding Certificate Authorities. Section 4 argues the problem of multiple-type validations. Section 5 proposes a novel concept called BlockCACert for automatic deployment of X.509 digital certificates and Sect. 6 analyzes a challenging and disruptive approach by which the CA uses ring signatures mechanism in the signing process of the request before publishing the certificate to the blockchain. We conclude the paper with our view regarding the positive impact of our concept on the world of digital certificates.

2 PKI Ecosystem

2.1 Digital Signatures and Digital Certificates

Digital Signatures are small pieces of data that contain authentication information which can be verified using a public key. Digital signatures are used to prove that a message or content belongs to a person or an entity. A digital signature, if implemented properly, is much more secure than a standard hand-written signature and harder to falsify. Of course, the private key used to generate the signature must not be compromised.

Digital Certificates are also known as Public-key certificates, or standard X.509 type documents (the standard which denotes the family of certificates). A digital certificate associates a digital signature to a public key with an identity. The certificate is used to verify if a public key belongs to an entity.

In a general PKI platform, the digital signature from within the digital certificate belongs to the Certification Authority (CA). Regardless of the nature of the digital signature used, the signature proves that the signer trusts the fact that the public key belongs to the certificate owner (information extracted from within the certificate).

2.2 Public PKI Systems

When we talk about the management of digital certificates or about public or private PKI systems, basically we talk about the processes that take place at the Certification

Authority. The CA acts like a third party, which is trusted by all the other communicating parties. The CA certifies that at least one of the communicating parties is genuine and is authorized to execute secure communications in its name.

A typical illustration of a traditional public PKI systems along with the respective workflow are detailed in Fig. 1:

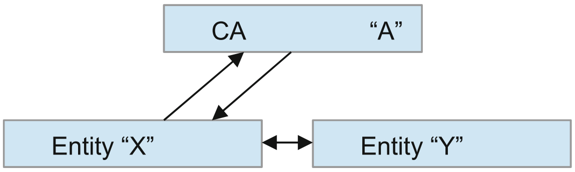

Fig. 1. A typical PKI CA workflow

Step 1: Entity X generates it's keypairs (private and public keys).
Step 2: Entity X creates a CSR binding its public key with its own information.
Step 3: Entity X contacts CA A and proves its identity to the CA. (After paying the certification fee,) The CA signs the CSR with its own private key producing a Certificate (CRT) which is given to Entity X.
Step 4: Entity X gives the certificate to any communicating partner. When entity Y wishes to authenticate Entity X, it validates the certificate against its CA stored public keys. Using the public keys, entity Y can verify the signature of the CA "A" applied to Entity X's certificate.
Step 5: After authentication is performed, secure communication may now commence.

This scenario, however, introduces some security risks which are described later in this paper.

3 Current Problems of Certificate Authorities

CA-based PKI contains numerous single points-of-failure. The centralized trust model of CA-based PKI means that the security of the system is based on single points-of-failure: CAs. Also, the structure of an X509 certificate may rely on a single root certificate for validation. If the Root CA in the chain is compromised, it can compromise the entire chain of trust [4].

There are online services, like Google's Certificate Transparency Project [8] which tries to detect rogue or forged CAs and their certificates. While Certificate Transparency (CT) acts similar to a blockchain, by providing a public logging of digital certificates information, using the blockchain for PKI offers an alternative solution to storing certificates securely and detecting any possible fraudulent activity.

Opsmate Inc. aggregates a list of Certificate Authority failures, in which false CAs or certificates were involved and other attacks which were performed against the PKI system, because of negligence or distrustful practices of CAs [9]. As the list points out, a

CA can suffer from both technical and practice habits problems. While we believe that the technical problems can arise like any other computer software bug, they can also be fixed and a CA cannot be held liable for this if it implemented software guidelines correctly. We can only hope that improved technology, best practices and further regulations in this industry are developed, to try to secure and align CAs to the best practices, highest security and high-quality service grid.

4 Current Shortcomings of Different Validation Types

All X.509 certificates, while having a similar structure and purpose, they differ in the type of validation used to issue them. Any certificate that is issued to a requesting entity is issued by a Certification Authority (CA) after the CA performs some form of validation to ensure that the requesting party proves its identity and its authorization to use the certificate.

In the PKI Digital Certification industry there are multiple types of validations:

- DV – Domain Validation
- OV – Organization Validation
- EV – Extended Validation

Below we present a summary of these validation types:

4.1 Domain Validation (DV)

DV certificates are the most common type of SSL certificates. They are verified using only the domain name as element of verification. Informally the workflow is as follows: the entity that requests a DV certificate sends the request to the CA. The CA then passes on a Challenge of HTTP or DNS type which the client must resolve. After the challenge is resolved and proof of domain control is validated, the certificate can be issued [10].

4.2 Organization Validation (OV)

OV certificates require more validation than DV ones but provide more trust. For this type of certificates, the CA verifies the actual entity or organization that wants to get a certificate. OVs are used by corporations, governments and other entities that want to provide the extra layer of identity information, alongside the technical domain element. OV is also used for code signing, document signing, or client authentication [10].

4.3 Extended Validation (EV)

It is said that EV certificates provide the maximum amount of trust because of the extra verifications being performed, even more than in the case of OV certificates.

The guidelines for issuing EV-type certificates were laid down by the CA/Browser Forum [10] and extra documentation must be provided to issue an EV certificate. As with OV, the EV lists the company/organization name in the certificate itself. EV certificates are the only type of certificates which cannot be locally issued (by creating a local-owend home CA), because it is said that modern browsers hardcode the IDs of the CAs which are authorized to issue EV certificates. While these types of validations offer different grades of trust, obviously for different fees, we believe that real trust must be expressed by a single type of validation, similar to an OV-type validation.

We believe that OV-type validation has enough fields to assure anyone of who the domain/CN is and who operates this certificate (O & OU fields). But, in our opinion, that EV-type validation is nothing more than an extension to OV-type validation, which used to turn the web-browser bar to green color or offer a special marker indicating an EV-type validation – the "O" (Organization) filed is sufficient to denote who is the owner of the certificate in question. A single type of validation is sufficient to ensure trust in a resource (Certificate – "CN" field) ownerd by a Person or an Orgzanization (Certificate - "O" field) [10].

5 BlockCACert Concept

We can use blockchain, as a public ledger, to register any kind of PKI operations, like issuing new certificates, renewing existing certificates or revoking certificates. Our new concept combines the blockchain public ledgering technology with the traditional PKI concept, adding the ring signatures component.

The BlockCACert concept only uses Blockchain technology for securely storing the digital certificates. It does not operate like a cryptocoin's blockchain, hence there is no proof of work involved. There are no miners involved and the transaction is processed automatically without anyone receiving a reward or a monetary value (as opposed to Bitcoin or other cryptocoins).

Implementation details about how BlockCACert is built, how the agents join/leave the system or the protocols involved are not part of the BlockCACert concept.

5.1 Blockchain Technology Basics

Blockchain [2, 3] is an innovative technology which aims to provide a secure, anonymous, based on consensus and decentralized public-ledger solution for transactions between users, without a central authority overseeing any of the transactions and without the possibility of linking the users to their actual identities. Blockchain was created with the concept of security and anonimity in mind [1, 4].

The non-financial aspect of blockchain technology relies in the mechanism used to operate the blockchain. On the blockchain, an easy verification can be made to verify the chain of ownership (Fig. 2).

5.2 Indexes Used by BlockCACert in the Concept Demonstration

C – Client (person, company, institution, etc.);
B – Multilayer Blockchain;
B_e – Blockchain $\rightarrow$ Entities' Layer;
B_c – Blockchain $\rightarrow$ Certificates' Layer;

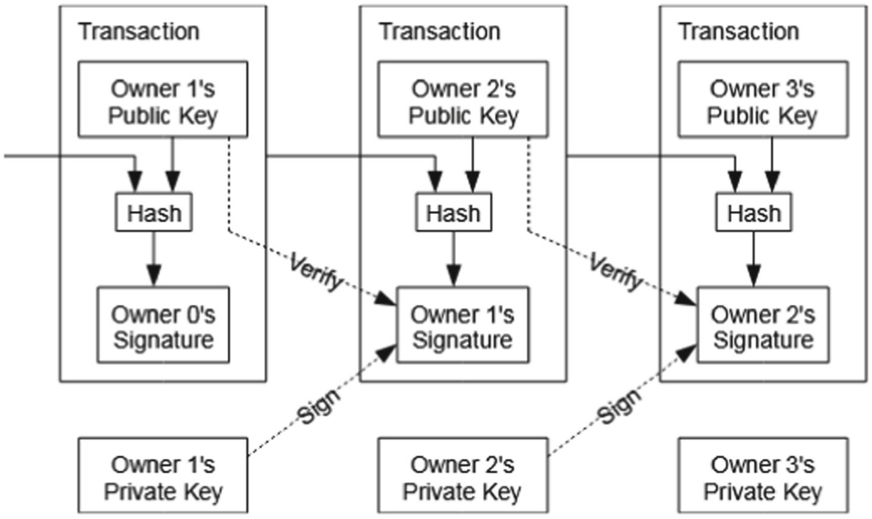

Fig. 2. Original blockchain verification scheme (Bitcoin) [2]

CA_x – Certificate Authorities;
K_{priv} – Private Key;
K_{pub} – Public Key;
CSR – Certificate Signing Request;
CSR_m – Modified Certificate Signing Request (modified standard CSR header);
CRT – Final (signed) Digital Certificate;
ACode – Secret Authorization Code;
EACode – Encrypted Secret Authorization Code;
$\mathrm{K_{priv}CA_x}$ – CAx's Private Key.
$\mathrm{K_{pub}CA_x}$ – CAx's Public Key.

5.3 PKI Component

BlockCACert concept uses the traditional PKI and X.509 standards for issuing, administering and revoking certificates. The only modification of BlockCACert in the PKI area is the introduction of a new type of CSR: CSRm.

The CSRm is the same as a regular CSR, except it contains one field that does the linking between the PKI and the blockchain: The Client Secret Authorization Code (ACode).

5.4 Blockchain Structure Component

Starting from the observation made by Axon in [5] and Axon and Goldsmith in [6], who consider the suitability of a blockchain-based PKI for contexts in which PKI is required, but in which linking of identity with public key is undesirable, our novel concept uses the blockchain technology only for its properties as an unmodifiable public ledger, where no financial elements are present in this implementation. Moreover, according to [7] privacy is very often neglected in most state-of-the-art blockchain-based PKI implementations, topic that we also followed in the process of designing our solution.

Our blockchain architecture does not use proof-of-* concepts and neither does reward users for transactions with some cryptocoins, like the traditional application of blockchain technologies. Our version of blockchain has an abstract multi-layer concept. Technically, there are no multiple layers, as every block comes one after the other. But, semantically, there are 2 layers of blocks: the Entity layer (Be) and the Certificate Layer (Bc).

The Entity Layer stores the blocks regarding to new and retired entities (persons, companies, institutions, and so on.). The Certificate Layer stores the blocks regarding to certificate operations: issuance, revocation, renewals, etc.

The following figure summarizes our blockchain structure (Fig. 3):

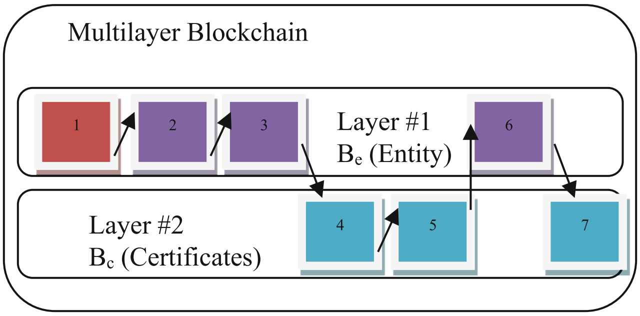

Fig. 3. BlockCACert – the multi-layer blockchain structure

Legend:

1. – Genesis
2. – P.Hash; ID1; Type1
3. – P.Hash; ID2; Type1
4. – P.Hash; ID3; Type2
5. – P.Hash; ID4; Type2
6. – P.Hash; ID5; Type1
7. – P.Hash; ID6; Type2

The contents of the Entity Layer: (Be) (Table 1):

Table 1. Entity layer fields

Field no	Blockchain-registered data for a given ACode(C)
0 × 1	Previous Hash
0 × 2	ID Block
0 × 3	Block Type
0 × 4	Registration Date
0 × 5	Country
0 × 6	State
0 × 7	City
0 × 8	O
0 × 9	OU
0xA	CN
0xB	ACode
0xC	Notes

The contents of the Certificates Layer (Bc) (Table 2):

Table 2. Certificate layer fields

Field no	Blockchain-registered data for a given *ACode(C)*
0 × 1	Previous Hash
0 × 2	ID Block
0 × 3	Block Type
0 × 4	Registration Date
0 × 5	CN
0 × 6	Validity
0 × 7	X509 CRT Encoded (DER/PEM)
0 × 8	Notes

The number of bytes per each field is subject to open discussion, remaining to be established by each implementing entity. There are no fix limits on how much space should the blockchain header occupy. This is a theoretical concept and implementation data may vary based on specific implementation.

5.5 Workflow of PKI-Blockchain Transactions

Let's assume we have a client C which registers on the blockchain as an entity. This entity can be a real person, a company, an institution, and so on. The method by which the client C registers on the blockchain is beyond of this paper's scope. Ideally, it can register on the blockchain via an official authority, via a registering company, etc.

Technically, when registering on the blockchain, C registers on the Entities' Layer, by creating a new block with the following elements:

- (the checksum of the previous block)
- New Block ID
- Block Type: 1
- Registration date (YYYY-MM-DD format)
- Country
- City
- Organization (O)
- Organization Unit (OU)
- Common Name (CN)
- Secret Authentication Code
- Notes

After the registration, C gets a Secret Authentication Code which defined the respective C as being unique and authorizes C to perform blockchain-based PKI operations.

We define the code as follows:

$$ACode(C) = Hash\big(CN + date\big(''U''\big)\big); \quad (1)$$

where,

ACode(C) = The Secret Authorization Code of Client C;

Hash() = A cryptographic hash function like SHA;*

CN = X.509 Standard's Common Name field.

date("U") = a pseudo-code representation of the current date and time, in standard Unix epoch format.

When C decides that it wants to begin cryptographic operations, C will generate its CSR via traditional PKI method:

- It generates its K_{priv}
- It generates its CSR (which contains the K_{pub}) with the following fields, actually generating a CSR_m:
- Country
- State
- Cirty
- (Organization)
- OU (Organization Unit)
- CN (Client Name or Server URL)
- EACode – this is the new field in the $\mathrm{CSR_m}$ format;

The following formula describes the *EACode* field:

$$EACode(C) = ENCRYPT\big(ACode(C),\ K_{pub}CA_x\big); \tag{2}$$

As the formula describes, before inserting the ACode into the EACode field, this code must be encrypted with the Public Key of the Certificate Authority which is going to be contacted in order to perform the certification operation. This encryption assures that no one can steal the client's secret authorization code and only the respective CA can decrypt it using its private key. Modifying the standard CSR to include an encrypted authorization code yields a new CSR fomat: CSRm (CSR modified). The following formula describes the new CSRm format:

$$CSRm = CSR + ACode\,Field; \tag{3}$$

- After generating the CSR_m, C contacts an arbitrary CA (CA_x) and applies for a new certificate by supplying some registration data and the CSR_m.
- CA_x decrypts the *EACode* using its *KprivCA*:

$$ACode(C) = DECRYPT\big(EACode(C),\ K_{priv}CA_x\big); \tag{4}$$

- The CA contacts the blockchain (B) and begins validating the data against the Decrypted Authorization Code found in the Entity's Layer of the blockchain (B_e) and supplied in the CSR_m (Table 3);

Table 3. Registered vs. supplied fields comparison table

Field no	Blockchain-registered data for a given *ACode(C)*	Validation Condition	Data supplied in the CSRm for a given ACode
1	Country	=/≠	Country
2	State	=/≠	State
3	City	=/≠	City
4	Organization (O)	=/≠	Organization (O)
5	Organization Unit (OU)	=/≠	Organization Unit (OU)
6	Common Name (CN)	=/≠	Common Name (CN)
7	*ACode(C)* (created at registration on B_e)	=/≠	*ACode(C)* (decrypted from CSR_m)

- The condition for data matching is an AND-ing operation which verifies the matching of the values:

$$Field1(B_e)\,AND\,Field1(CSR_m) = 1;$$
$$Field2(B_e)ANDField2(CSR_m) = 1;$$

$$\begin{aligned}
Field3(B_e)ANDField3(CSR_m) &= 1; \\
Field4(B_e)ANDField4(CSR_m); &= 1; \\
Field5(B_e)ANDField5(CSR_m) &= 1; \\
Field6(B_e)ANDField6(CSR_m) &= 1; \\
Field7(B_e)ANDField7(CSR_m) &= 1; \qquad (5)
\end{aligned}$$

- If all the fields match, then the client is validated.
- If, at least one of the fields do not match, the client is invalidated and the transaction is rejected.
- When CA_x validates C on B_e, it proceeds to sign the CSR_m with the $K_{priv}CA_x$, using the ring signatures scheme described below resulting a final signed digital certificate belonging to the C:

$$CRT\ (C) = CASIGN[RINGS]\big(CSRm(C),\ K_{priv}CA_x\big);\ ) \qquad (6)$$

- This certificate is issued to the client and is registered on the blockchain (B) as a new transaction on the B_c Layer, by creating a new block with the following elements:

  - (the checksum of the previous block)
  - New Block ID
  - Block Type: 2 (certificate block)
  - Registration date (YYYY-MM-DD format)
  - Common Name (CN)
  - Validity
  - Encoded X.509 Certificate (DER/PEM Format)
  - Notes

Anyone wishing to verity the certificate, can do it by the following means:

- Traditionally, via PKI tools: checking the Signature of the CA within the CRT against a Certificate Root Store;
- Interogating the blockchain (BC of B) by querying the validity of the certificate or the act of certification from CAx who digitally signed the CSRm with its KprivCAx.

6 Ring Signatures in BlockCACert

A ring signature is a type of digital signature that allows any member of a group of users (ring) to sign a message with its own key in an anonymous way. Within the process, no one, but except actual signer, knows which member signed the message - this means it is computationally infeasible to determine which of the group members' keys was used to produce the digital signature. Ring signatures must satisfy two independent notions of security: anonymity and unforgeability.

The notion of ring signature was formalized by Rivest, Shamir and Tauman in [11]. In contrast to group signatures, the authors designed a signature scheme with no managers involved, and which do not require any central coordinator, setup and revocation procedures. Moreover, according to authors, there is no way to distribute specialized keys and to revoke the anonymity of the actual signer, and there are no prearranged groups of users.

A ring signature scheme is defined by two procedures [11]:

- ***ring-sign***(m, P_1, P_2,…,P_n, s, S_s) – produces a ring signature σ for the message m, where P_1, P_2,…,P_n are the public keys of the n ring members, S_s is the secret key of the s-th member, who is the actual signer.
- ***ring-verify***(m, σ) – takes a message m and a signature σ (which includes the public keys of all the possible signers), and outputs either *true* or *false*.

Ring Signatures is fundamental concept used in the CryptoNote/Monero cryptocurrency [13, 14].

In order to preserve the privacy between BlockCACert layer's communication, we propose a formal ring signature mechanism starting from the functional definition of a ring signature presented by Bender, Katz, Morselli in [12]:

- key generation: $Gen(1^k)$, where k is a security parameter, outputs a public key PK and secret key SK. Entity layer (B_e) is responsible for key generation. The keys are not included into the block structure, but each block has an associated pair of keys, which can be changed/regenerated at any given time (due to length restrictions, we are unable to detail the way the keys are regenerated and relinked to the blocks).
- message signature: $Sign_{s,SK}(M, R)$ outputs a signature σ on the message M with respect to the ring $R = (PK_1,..., PK_s)$, where ($R[s]$, SK) is a valid key-pair output by *Gen*. Here it is important to note that in the signing process we have the secret keys associated with the blocks chained to the current block s at which the signature is applied.
- signature verification: $Vrfy_R(M, \sigma)$ verifies a signature σ on a message M with respect to the ring of public keys R. The verification process takes place only at the Certificate Layer (B_c). If the signature verification fails, that block at the Certificate Layer does not perform any additional analysis to check if a new public key has appeared in the blockchain – the method, although it seems quite rigid, solves quickly two issues: double spending prevention and no more computational resources.

Regarding the implementation details of our proposed ring signature mechanism, one of the viable options would be to define a third layer for BlockCACert architecture, where the entire key management mechanism is modularized and integrated.

7 Conclusions

BlockCACert tries to improve the management of digital certificates by using blockchain technology as a proven security mechanism for CAs trustworthiness. This ensures that no fraudulent certificates can be issued, or even if, somehow, these certificates are issued, the fraud is immediately detectable on the blockchain.

This scheme eliminates the need for multiple types of validation. By using non-modifiable blockchain records, we can ensure that every entity who requests a certificate has a name, and some identification information, thus eliminating the need for multi-level trust. Our solution also introduces a degree of automatic deployment, to ease the task of authentication. As a future direction of research, we will investigate integration of BlockCACert with the privacy data storage protocol proposed in [15], that is based on the ring signature on the elliptic curve.

This concept was implemented in one of our labs, on 10 computers, and the results we got confirmed the desired workings of our concept. There are no numerical results, as this theoretical concept only proves that the workflow of storing digital certificates on blockchains works efficiently when implemented properly.

References

1. Gergely, A., Crainicu, B.: A theoretical study and review on blockchain's privacy mechanisms. In: IFIP SEC 2019. Lisbon, Portugal (2019)
2. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. Cryptography Mailing list at (2008). https://metzdowd.com
3. Xu, M., Chen, X., Kou, G.: A systematic review of blockchain. Financ. Innov. **5**(1), 1–14 (2019)
4. Guegan, D,: Public Blockchain versus Private blockchain, HAL ID: halshs-01524440, version 1 (2017)
5. Axon, L.: Private-awareness in Blockchain-based PKI, Centre for Doctoral Training in Cyber Security, CDT Technical Paper Series 21/15, University of Oxford (2015)
6. Axon, L., Goldsmith, M., PB-PKI: a privacy-aware blockchain-based PKI. In: Proceedings of the 14th International Joint Conference on e-Business and Telecommunications (ICETE 2017) – SECRYPT, vol. 4, pp. 311–318 (2017). ISBN: 978-989-758-259-2
7. Brunner, C., Knirsch, F., Unterweger, A., Engel, D.: A Comparison of blockchain-based PKI implementations. In: Proceedings of the 6th International Conference on Information Systems Security and Privacy – ICISSP, pp. 333–340. Valletta, Malta (2020). ISBN 978-989-758-399-5; ISSN 2184-4356
8. Google Certificate Transparency Project. https://certificate.transparency.dev
9. SSLMate: Timeline of Certificate Authority Failures, Opsmate Inc. https://sslmate.com/certspotter/failures
10. SSL Americas: DV, OV, IV, and EV Certificates. https://www.ssl.com/article/dv-ov-ev-certificates/
11. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd, C. (eds.) Advances in Cryptology—ASIACRYPT 2001. ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Berlin, Heidelberg (2001). https://doi.org/10.1007/3-540-45682-1_32
12. Bender, A., Katz, J., Morselli, R.: Ring signatures: stronger definitions, and constructions without random oracles. J. Cryptol. **22**, 114–138 (2009)
13. Shen, N.: Ring signature confidential transactions for monero. Cryptology ePrint Archive, Report 2015/1098 (2015)
14. Shen, N.: Ring confidential transaction. Ledger **1**, 1–18 (2016)
15. Li, X., Mei, Y., Gong, J., Xiang, F., Sun, Z.: A blockchain privacy protection scheme based on ring signature. IEEE Access **8**, 76765–76772 (2020)