In Proceedings of the First Italian Conference on Cybersecurity (ITASEC17), Venice, Italy.

Certificate Validation through Public Ledgers and Blockchains

Marco Baldi[1], Franco Chiaraluce[1], Emanuele Frontoni[1], Giuseppe Gottardi[2], Daniele Sciarroni[1], and Luca Spalazzi[1]

[1] DII, Università Politecnica delle Marche, Ancona, Italy
{m.baldi, f.chiaraluce, e.frontoni, l.spalazzi}@univpm.it, S1070022@studenti.univpm.it
[2] Namirial S.p.A., Senigallia, Italy
g.gottardi@namirial.com

Abstract

Public key infrastructures (PKIs) are of crucial importance for the life of online services relying on certificate-based authentication, like e-commerce, e-government, online banking, as well as e-mail, social networking, cloud services and many others. One of the main points of failure (POFs) of modern PKIs concerns reliability and security of certificate revocation lists (CRLs), that must be available and authentic any time a certificate is used. Classically, the CRL for a set of certificates is maintained by the same (and sole) certification authority (CA) that issued the certificates, and this introduces a single POF in the system. We address this issue by proposing a solution in which multiple CAs share a public, decentralized and robust ledger where CRLs are collected. For this purpose, we consider the model of public ledgers based on blockchains, introduced for the use in cryptocurrencies, that is becoming a widespread solution for many online applications with stringent security and reliability requirements.

1 Introduction

Security and reliability of public key infrastructures (PKIs) is a critical aspect for the provision of many remote services that are now widespread. According to the well-known X.509 standard [1], modern PKIs rely on public key certificates that are signed by the issuing certification authorities (CAs) and can be openly exchanged over the Internet. Each certificate contains the public key of a user (often a service provider (SP)) that can be used for performing encryption through some asymmetric cryptosystem. The owner of the corresponding private key will be the sole user able to decrypt the encrypted message. Besides the user's public key, each certificate contains some other information, including an expiry date. However, it may happen that for some reason a certificate must be revoked before its expiry date, and for this purpose the standard provides the use of certificate revocation lists (CRLs). A CRL is a list containing identifiers of certificates revoked by the CA that issued them, and it should always be checked before using a certificate. The continuous availability of CRLs and their updating is therefore of crucial importance for ensuring security of online services.

In fact, there is a long record of attacks involving certificates and CAs. For example, on March 2011 an attack hit a Comodo[1] affiliate registration authority and 9 fraudulent Secure Sockets Layer (SSL) certificates were issued, apparently with the aim of gathering information about some government activities [2]. On July 2011 the Dutch CA DigiNotar was successfully attacked, and numerous fraudulent digital certificates were issued for a number of Internet domains operated by famous companies (like Google, Microsoft, Facebook), as well as intelligence agencies [3]. On November 2011, the DigiCert Sdn. Bhd. CA has issued 22 fraudulent certificates concerning Malaysian government services, and these certificates did not contain

[1] https://ssl.comodo.com/

any revocation information [4]. During the same month, the Dutch CA KPN has stopped issuing digital certificates after suffering a DDoS attack [5], and the security company F-Secure discovered a malware that uses a digital certificate stolen from the government of Malaysia [6]. On December 2011, the GemNET security firm suffered a data breach including administrative information. The parent company KPN has stopped signing operations of certificates by GemNET [7]. All these episodes point out the need for reliable, timely and highly available information about revoked certificates. However, in current PKIs, CRLs are commonly made available in a single point over the Internet, known as certificate distribution point (CDP), and this obviously introduces a single point of failure (POF). There are a few classical approaches aimed at solving this problem, with their merits and their defects, but none of them is able to fully address this issue.

In this paper, we present a new solution based on public ledgers and blockchains that is actually able to remove any single POF and to achieve high reliability and security in the distribution of certificate revocation information. The paper is organized as follows: in Section 2 we describe the use of CRLs and the main issues affecting existing solutions. in Section 3 we briefly introduce blockchains. in Section 4 we describe the proposed solution and its main procedures, and Section 5 concludes the paper.

2 Certificate revocation lists and relevant issues

In a PKI, a trusted CA issues a certificate to ensure integrity of a user's public key. The certificate is digitally signed by the issuing CA, and therefore can be smoothly exchanged among users, without the need of resorting to a central distribution point. Each certificate has a prefixed time of validity, typically between 3 and 5 years. However, it may happen that a certificate must be revoked before its expiry date, for example due to security incidents. For this purpose, the certificate usually contains an extension known as CDP. The CDP provides a pointer (for example a uniform resource locator (URL) or another kind of reference) to a server containing a digitally signed and timestamped list of the serial numbers of revoked certificates. Such a list is also known as CRL. Therefore, in order to know whether a certificate shall be considered valid or not, a client must gather the most recent CRL published by the CA that issued the certificate and verify that the serial number of the certificate is not included in such a list. This verification is of crucial importance to ensure that the client uses a legitimate certificate, and to guarantee integrity of the associated public key.

A first issue concerns the speed of CRL updates. In fact, CRLs do not operate in real time: they are commonly updated periodically by the issuing CA, and there may be a delay between a security breach and the subsequent CRL update, resulting in a temporary use of compromised certificates. An even more serious issue is in the fact that each CA is responsible for updating and maintaining its own CRLs. Therefore, if the CDP of a CA is temporarily unreachable over the Internet, then integrity of all certificates issued by that CA cannot be verified, resulting in a dangerous single POF. Another issue concerns traceability of certificate revocations. In fact, when a CA updates its CRL, this is done by overwriting the previous file, therefore no historical data are kept. These issues are partly addressed by some existing solutions that are briefly reviewed next.

2.1 Existing solutions

The most common solutions adopted until now for making CRLs verification easier and more robust are

- Online certificate status protocol (OCSP) [8]
- Simple certificate validation protocol (SCVP) [9]
- Proprietary solutions (PKI toolkits) [10]
- Middle-ware solutions [11]

The OCSP, described in [8], is based on a client-server architecture in which clients request whether a certificate is valid or not and servers (or OCSP responders, that are part of PKI products or independent packages) inform them of this. The signer of the response must not necessarily be the emitter of the certificate subject of the request. The main advantage of OCSP is that it avoids clients having to download the CRL, and allows them to get real-time responses from OCSP responders. This also reduces the requirements in terms of bandwidth and increases scalability. The OCSP also supports hierarchical requests: a client can query an OCSP responder that in turn forwards the request up to the root CA.

Based on these features, OCSP represents a good solution to the availability problems of CRLs. However, there may be situations in which having multiple OCSP responders may expose the system to security flaws [12]. This in general happens when there is the possibility of a man-in-the-middle attack. The OCSP is also vulnerable to replay attacks. In fact, a positive response can be captured by an attacker and forwarded to the client with some delay, and such a delayed reply could allow keeping a client in the dark about the revocation of a certificate occurred in the meantime. Obviously, this kind of flaw could be prevented by using a nonce and limiting the validity of the response. However, many OCSP responders and many clients do not support the use of nonces and CAs send responses that are often valid for many days. Another security flaw may arise from the fact that, in some cases, when a client does not receive a response from an OCSP responder within a reasonable time, it neglects the protocol and accepts the certificate, that may have been compromised in the meantime. A further issue concerning OCSP that we aim at overcoming with the proposed solution concerns the case in which a CDP becomes temporarily unavailable. In such a case, although many OCSP responders may be available, the CRL could not be retrieved.

The SCVP [9] can be seen as an extension of the OCSP, and it also delegates the management of certificates to a server. The latter informs the client if a certificate is valid and provides a path to a trusted certificate. The SCVP supports two types of servers: trusted servers and untrusted servers. The untrusted server can provide clients with the certificate path and revocation information such as CRLs or OCSP responses. Trusted servers instead run the whole validation on behalf of the client, and represent a useful solution for companies that want to centralize and standardize the management of PKIs. Although the SCVP improves over the OCSP, the SCVP may still represent a single POF, as well as a bottleneck in case of overload [13]. In addition, with SCVP the initial setup is rather complex and expensive.

PKI toolkits are concerned with making PKI-enabled applications, that is, to provide a software application with the ability to check validity of certificates. This is a well suited solution for commercial applications and to perform tests and experiments. However, it does not always provide support for CRLs and requires high competence, resulting long and expensive. The main drawback of these solutions with respect to the approach we propose is in that they are not standardized, and depend on the toolkit vendor. This indeed limits their interoperability.

A different approach is that followed by middle-ware solutions (like ValiCert, CertValidator e KyberPASS) that, differently from PKI toolkits, provide a centralized solution for the validation of certificates by applications. These are able to reduce costs with respect to PKI toolkits and ensure compatibility with many verification paradigms (CRLs, OCSP, SCVP, etc.). Middle-ware solutions exploit validation authorities (VAs) that may implement replication mechanisms

to ensure high availability and may support multiple PKIs and real-time validation through OCSP. This solution is actually able to address most issues affecting certificate validation by providing a robust and scalable infrastructure. However, it relies on a middle-ware that can be still seen as a single POF.

3 Blockchains and cryptocurrencies

The idea of public, decentralized ledgers has been introduced with the notion of cryptocurrency, first appeared in 1998 [14] and now become widespread through several implementations. Among them, Bitcoin is the first and most widespread solution [15, 16]. These schemes are based on a completely decentralized paradigm, removing the risks due to the presence of a single or a few POFs. Moreover, using a decentralized system with a large number of users increases resilience against natural and malicious disasters, and also reduces costs.

The main innovation is in the use of a public virtual ledger, where all transactions are registered in sequential order and are made publicly available. In fact, a copy of the whole ledger may be owned by any user registered to the network. However, only those transactions which have been registered in the public ledger for not less than some time are considered as valid by the users. Public-key encryption is used to generate a pair of keys for each user. The private key is kept secret by its owner, while the public key is used to generate a unique public address that identifies its owner in the network.

The public address is used as the starting and ending point of transactions. One user wishing to transfer some asset to another user begins a transaction with its address as the source address and the other user's address as the destination address. When such a transaction has been written and validated in the public ledger, then the user who received the asset can transfer it to a third user by starting a new transaction. In order to originate a transaction for alienating some asset, the user needs the private key corresponding to the public address that previously received that asset, and this ensures that a user who does not own the asset cannot transfer it to another user. However, a user owning a given asset could try to spend it twice (double spending), and this is prevented by using a transaction verification mechanism.

In fact, each transaction must be verified in order to be appended to the chain contained in the public ledger, and such a verification can be made by any other user in the network. However, in order to verify a transaction, the user must make an effort. In the classical proof-of-work scheme [17] used in Bitcoin, the verification effort is represented by the solution of a cryptographic puzzle, that has a significant computational cost. When this effort is undertaken by many users in the network, the solution can be found in a reasonable time, and the transaction verified. If a malicious user wishes to verify his own transactions, he must be able to take control of a very large computing power. Moreover, a fake transaction creates a fork in the blockchain that is resolved, after a few verifications, by pruning the shortest branch. Therefore, in order to validate a fake transaction, a malicious user should command more than half of the whole network computing power. The users providing computing power for the verification of transactions are called miners [18], and receive a reward any time they verify a new block of transactions. Rewards can be generated as newly coined money or as transactions fees. In order to limit the total amount of circulating money, the amount of newly coined money is decreasing over years, hence transaction fees are the only rewarding mechanism in the long term. An alternative to proof-of-work schemes are proof-of-stake schemes, in which for a user to verify transactions it is needed to own some amount of currency. This avoids the cost in terms of power consumption following from classical proof-of-work schemes, but requires some round-robin mechanism to ensure rotation of miners.

This paradigm represents a breakthrough in the context of electronic cash systems, and of collaborative and peer-to-peer networking in general. In fact, the existence of a public ledger that cannot be maliciously altered and is characterized by a very high level of availability has paved the way for many possible applications. For example, this paradigm has already been considered for developing decentralized domain name systems resistant to censorship [19] and anonymous credential systems [20]. Many other applications, like decentralized electronic voting systems, can also be envisaged [21].

The blockchain technology has several advantages over classical systems with loose replication consistency. The latter in fact require a centralized orchestration server, while the blockchain is completely decentralized and therefore has no central POF. This also results in higher scalability, since peers can freely leave or join the network without incurring overheads. Moreover, it has recently been shown that the blockchain provides strong consistency [22], that is an important advantage since transactions are consistent over the network and also time-stamped and ordered. Such a feature is particularly important for certificate revocation data, since inconsistent or outdated information may lead to the use of an invalid certificate.

4 Proposed solution

In the solution we propose, CRLs are distributed through the use of a private blockchain, shared among CAs. We use a private blockchain instead of some existing public blockchain because certificate and CRL writing privileges are given only to a subset of trusted nodes (like CAs) and not to all users. CRLs are constantly updated and available to everyone who can access the blockchain. Those users who cannot access it directly, can query any CA to obtain the CRLs, since all of them will have the same information.

For this purpose we need to build an architecture made up of three types of entities: the first and most important are the CAs, that are responsible for issuing certificates to requestors who meet the requirements and maintain CRLs. Certificate requestor are represented by SPs, that require issuing of certificates for running the services they provide to users. The third role is played by users that just need to read certificates and the relevant CRLs. This type of system can be realized by using an infrastructure based on a private blockchain: this chain contains a set of transactions showing any activity related to the certificates issued by the authorities themselves. The blockchain is shared among the various CAs that, as in public blockchains used for cryptocurrency, can download a copy of the current chain when they join the network, and then can update it with new issues or withdrawals of certificates. Obviously - and this is at the basis of this technology - every update must be immediately transmitted to each node of the network so that everyone has a constantly updated copy. Certificates and CRLs are included in marked and recognizable transactions, that can be easily tracked. The blockchain writing and reading operations are described in the following sections.

4.1 Certificate issuing

Certificates may be issued only by CAs, that are authorized entities compliant with some security and reliability standards. A CA issues a certificate to a requesting SP on condition that it meets some set of requirements.

In the system we propose, all issued certificates are stored in the private blockchain, as shown in Fig. 1. Upon the request of a SP of issuing a new certificate, the CA produces a special kind of “certificate” asset and releases a token corresponding to the new certificate. All certificates issued by the same CA have a certain ID (in the figure each ID is associated to a

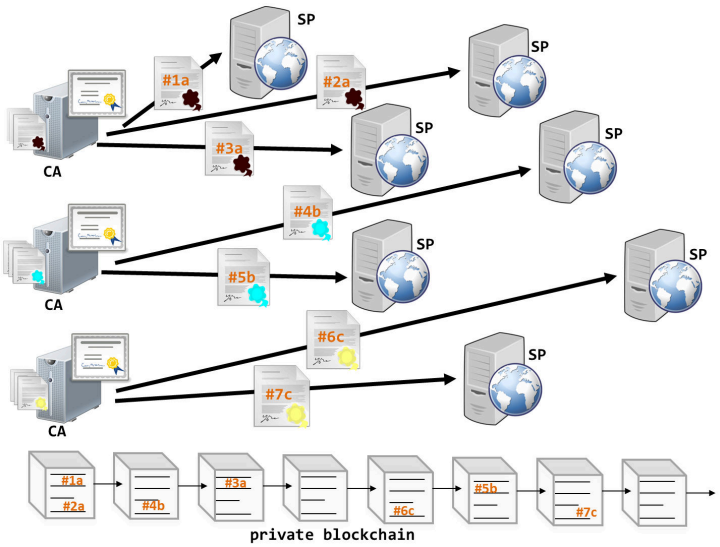

Figure 1: Certificate issuing by multiple CAs.

different color). The SP can register with the private blockchain through one of the existing nodes, and once it enters the network it acquires a wallet address that is needed to communicate with the CA. In fact, the CA must be informed of the SP wallet address in order to release the certificate. To send the certificate to the requesting SP, the CA makes a transaction sending the certificate token to the SP wallet address. The transaction ID can be reused as a reference of the issued certificate (e.g., for certificate revocation and suspension).

4.2 Certificate validation

In the proposed scheme, certificate validation is built upon reading of an updated CRL from the shared ledger. The CRL can be written only by CAs. The writing procedure of a new CRL into the blockchain can be initiated by any CA. The initiating CA creates a stream into the blockchain and all nodes into the network are registered to that stream. However, only CAs are allowed to write into such a stream, while the remaining nodes have read-only access. When a CA wishes to revoke a certificate before its expiration time, it just needs to write into the CRL stream the identifier of the transaction associated with the issuance of that certificate. Alternatively, for legacy purposes and compliance with existing PKIs, a CRL in a classical format (e.g., according to X.509) can be written into the CRL stream.

Then, all network users (i.e., CAs, SPs, clients) are able to read the CRL from the public ledger in order to validate certificates. In fact, any of them may need to validate a certificate: a CA may inquire about revoked certificates from other CAs, a SP may check whether its certificate (or that of another SP) has been revoked or not, and clients need to check the reliability of websites or other web services. The CRL writing and reading procedures are schematically described in Fig. 2. In addition, from time to time CAs and SPs can save the whole CRL stream locally, and make it available to users that do not take part in the blockchain.

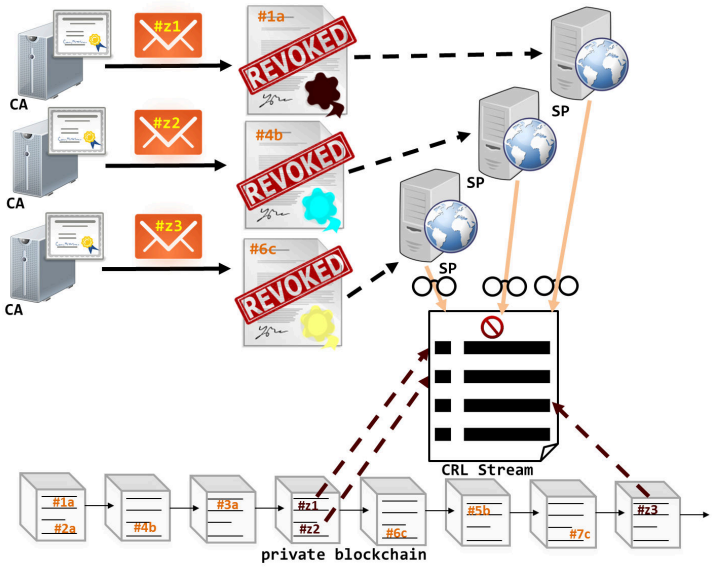

Figure 2: Certificate revocation and validation through the blockchain.

4.3 Implementation

A valuable software toolkit to implement the proposed scheme is Multichain[2], that provides an open source private blockchain platform. Multichain currently exploits a proof-of-work scheme, but its extension to a proof-of-stake model is expected in the near future. By using Multichain, a private blockchain can be created by a first server (that generates the first block of the chain) and then populated by including other servers. The following commands allow to create and initialize a new blockchain with Multichain.

```
multichain-util create chain1
multichaind chain1 -daemon
```

The new chain can be accessed and shared by any other node using the IP address and the listening port of the first server through the following command:

```
multichaind chain1@[serverIP]:[serverPort]
```

that initializes the same blockchain on the new node and makes it obtain a wallet address. The new node may become an active network node by obtaining the relevant rights from the first server (which by default has administrator privileges), through the following command:

```
multichain-cli chain1 grant [NewNodeWalletAddress] connect
```

Other nodes can also receive the same privileges as the first server by using the same command, but with the string "admin" instead of "connect" as its last argument. Other privileges that may be granted are: "send" and "receive" (to participate in transactions), "issue" (to issue assets), "create" (to create streams), "mine" (to mine blocks), "activate" (to grant privileges only for the actions "connect", "send" and "receive").

[2] http://www.multichain.com/

The Multichain platform provides some features that are of particular interest for our purposes. For example, it is possible to issue new assets by specifying the number of tokens and the level of granularity with which they can be exchanged in transactions through the command:

```
multichain-cli chain1 issue [WalletAddress] [AssetName] [UnitsNumber]
[Granularity]
```

This feature can be used to implement the issuance of certificates by CAs by using the command:

```
multichain-cli chain1 sendwithmetadata [WalletAddress]
'{"[AssetName]":[Units]}' [MetadataHash]
```

Another useful feature is the ability to create streams through the command:

```
multichain-cli chain1 create stream [StreamName] false
```

Key-value pairs can be stored in streams, signed by the server that published them, by using the command:

```
multichain-cli chain1 publish [StreamName] [Key] [ValueHex]
```

This feature is actually useful for writing the CRL to the blockchain. In the following we describe some use examples of the proposed solution.

4.3.1 Basic use example

Let us consider two SPs (P1 and P2) that request the issuance of a certificate to a CA. The sequence of events is as follows.

1. The CA creates the blockchain (being the only authority) and then generates the initial block.
2. The CA issues a "CAcertificate" asset and generates (at least) two tokens.
3. The two peers (P1 and P2) connect to the blockchain (knowing the CA IP address and port) and are given the right to receive assets by the CA.
4. The CA sends a unit of the asset to P1 and another to P2, and stores the ID of the two transactions.
5. Upon expiry of the concession of the two certificates, the CA communicates their revocation on a CRL stream created by the CA, on which P1 and P2 have read-only privileges.
6. The CRL stream items are identified through the triple <publisher, key, data>, and these fields can be used to identify each revoked certificate (for example: <Publisher = CA, key = ID of the transaction that had established the granting of the certificate, data = wallet address of the node (P1 or P2) that was beneficiary of the concession>).
7. Periodically, the CA can take a snapshot of the stream, and can expose the list of revoked certificates to external users.

4.3.2 Intermediate use example

Let us consider two CAs, CA1 and CA2, three SPs (P1, P2 and P3) and one user U. The sequence of events is as follows.

1. CA1 creates the blockchain and authorizes CA2 to connect to it with write privileges (it also gives CA2 the right to allow other peers to connect to it).

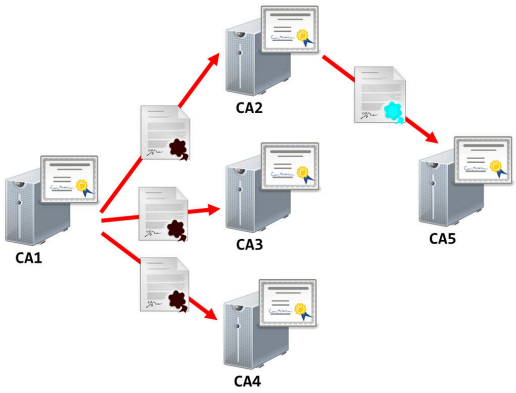

Figure 3: Example of hierarchical organization of certification authorities.

2. CA1 authorizes P1 and P2 to connect to the blockchain (and to receive assets), issues at least two units of the "CA1certificate" asset and delivers them to P1 and P2.
3. CA2 does the same with P3 (assigning it a unit of the "CA2certificate" asset).
4. Upon expiration, the certificates of P1, P2 and P3 are revoked and their value is stored in the CRL (i.e., in a CRL stream that can be accessed in read-write mode by CA1 and CA2, and in read-only mode by P1, P2 and P3).
5. The user U needs to read the CRL to validate P1, P2 and P3 certificates. For this purpose, it can request any CA to provide its own updated copy of the CRL stream.

4.3.3 Advanced use example

Let us consider many SPs, any number of users and five CAs with the hierarchical organization shown in Fig. 3. The sequence of events is as follows.

1. CA1, the root CA, creates the blockchain and the CRL stream.
2. As in previous cases, the other CAs connect to the blockchain through CA1 IP address.
3. Each of them requests and obtains a certificate from a higher hierarchical level CA.
4. Then each CA is authorized to carry out transactions (both as senders and as recipients) and to write to the CRL stream until its certificate expires.
5. Issuing of certificates to the SPs is carried out as in previous examples.
6. Reading the CRL instead is more complex. In fact, when a user wants to know if a certificate has been revoked, he must go back to the tree root.

In order to explain the last step, let us consider a SP P1 and the hierarchical organization CA1 $\rightarrow$ CA2 $\rightarrow$ CA5 $\rightarrow$ P1. Let us suppose that the certificate issued by CA1 to CA2 expires. In this case, U shall be informed that the certificate issued to P1 is no longer valid. For this purpose, the user needs to explore the hierarchical structure up to CA1 and check the relevant CRLs in the blockchain.

5 Conclusion

We proposed a blockchain-based PKI solution for addressing the issues of reliability and security of certificate revocation information. Its main advantages are in removing any single POF and being relatively simple to implement by leveraging existing open source platforms. The distributed and robust nature of blockchains allows reducing the risk of security breaches coming from the lack of real-time and updated information concerning revoked certificates.

References

[1] Internet Engineering Task Force (IETF), "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 5280, 2008.

[2] R. A. Grimes, "The real security issue behind the Comodo hack," InfoWorld, 2011.

[3] C. Wisniewski, "Falsely issued Google SSL certificate in the wild for more than 5 weeks," Naked Security - Sophos, 2011.

[4] C. Wisniewski, "Another certificate authority issues dangerous certificates," Naked Security - Sophos, 2011.

[5] D. Goodin, "SSL authority stops issuing certificates following breach," The Register, UK, 2011.

[6] F-Secure, "Malware Signed With a Governmental Signing Key," News From The Lab, Nov. 2011.

[7] C. Wisniewski, "Second Dutch security firm hacked, unsecured phpMyAdmin implicated," Naked Security - Sophos, 2011.

[8] Internet Engineering Task Force (IETF), "X.509 Internet Public Key Infrastructure - Online Certificate Status Protocol - OCSP," RFC 6960, 2013.

[9] Internet Engineering Task Force (IETF), "Server-Based Certificate Validation Protocol (SCVP)," RFC 5055, 2007.

[10] Certicom Corporation, "Public-Key Infrastructure Technology and Concepts," 2001, online. Available: http://www.au-kbc.org/bpmain1/PKI/Trustpoint_PKI.pdf.

[11] R. Lendvai, "Accelerating Security Policies thru Middleware," SC Magazine, US edition, Jan. 2004.

[12] A. Langley, "No, don't enable revocation checking," 2014, online. Available: https://www.imperialviolet.org/2014/04/19/revchecking.html.

[13] N. A. Ghadiali, "Certificate Validation for PIV across the Federal Bridge using the Server-based Certificate Validation Protocol (SCVP) - RFC 5055," online. Available: http://www.electrosoft-inc.com/papers/.

[14] W. Dai, B-money proposal, 1998, online. Available: http://weidai.com/bmoney.txt.

[15] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008, online. Available: https://bitcoin.org/bitcoin.pdf.

[16] M. E. Peck, "The Bitcoin Arms Race is on!" IEEE Spectrum 50(6): 11-13, 2013.

[17] A. Back, "Hashcash - A Denial of Service Counter-Measure," Aug. 2002, online. Available: http://www.hashcash.org/papers/hashcash.pdf.

[18] M. B. Taylor, "Bitcoin and the age of Bespoke Silicon," Proc. CASES 2013, Montreal, QC, 2013.

[19] M. Wachs, M. Schanzenbach, C. Grotho, "On the Feasibility of a Censorship Resistant Decentralized Name System," Proc. FPS 2013, La Rochelle, France, 2013.

[20] C. Garman, M. Green, I. Miers, "Decentralized Anonymous Credentials," Cryptology ePrint Archive, report 2013/622.

[21] P. Noizat, "Blockchain Electronic Vote," In: Handbook of Digital Currency (D. Lee Kuo Chuen Ed.), chap. 22, Elsevier, 2015.

[22] E. G. Sirer, "Bitcoin Guarantees Strong, not Eventual, Consistency," Hacking, Distributed, Mar. 2016.