Computers and Electrical Engineering 81 (2020) 106523

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Long-term verification of signatures based on a blockchain[☆]

Tomasz Hyla[*], Jerzy Pejaś

West Pomeranian University of Technology in Szczecin, Faculty of Computer Science and Information Technology, ul. Żołnierska 52, Szczecin 71-210, Poland

ARTICLE INFO

Article history:
Received 23 October 2018
Revised 6 September 2019
Accepted 25 November 2019
Available online 2 December 2019

Keywords:
Digital signature
Signature verification
Validity model
Blockchain
Timestamp

ABSTRACT

Digitally signed documents must remain stored for many years. In this paper, a scheme that would allow maintaining signature validity without the necessity to use timestamps from trusted third parties is proposed. According to the scheme, after inserting data into a blockchain, a user can store a signed document in his storage without the need to perform any maintenance actions in the future. The Round-based Blockchain Time-stamping Scheme is proposed that is scalable, i.e., it requires embedding a constant number of bytes into a blockchain independent from a number of input documents. The scheme allows to prove that a document existed not only before a certain date, but after a certain date as well. Moreover, the purpose of the scheme is to meet non-repudiation requirements for digitally signed documents. The scheme allows verifying signature validity using a chain model and under some conditions using a modified shell model.

1. Introduction

Many signed digitally documents must remain stored for many years. Digital signatures are valid as long as the certificate of a signer's public key is valid (usually two years). Any additional actions must be taken to enable verification in a longer period (20 or more years). The simplest one is to add a trusted timestamp immediately after signing a document. Several trusted authorities offer such on-line services. A timestamp is a digital signature of a hash of a signed document and current time that is created by a Trusted Authority (TA). Therefore, it will also expire. Hence, any long-term archive that uses trusted timestamps to maintain validity of digital signatures must periodically re-timestamp all documents.

Nowadays, blockchain technology has become very popular, what is mainly caused by introducing Bitcoin cryptocurrency [1]. In practice, cryptographic techniques which are used to build a blockchain, i.e. hash functions, hash linking, hash trees and authenticated dictionaries have been known for many years. The basic property of a blockchain is immutability, i.e., the blockchain ledger contains permanent, non-editable and unchangeable transactions' history.

The blockchain does not require cryptocurrency to exists [2]. In general, there are two types of blockchains: permissioned and permissionless [3]. The cryptocurrencies are based on permissionless blockchains, where anyone can anonymously participate in block mining. Such blockchains are usually based on Proof-of-Work (PoW) consensus. In the PoW approach, creating (mining) a new block requires solving a computational problem using brute force. The computational problem, for example in Bitcoin, is finding a hash with a given number of zeros at the beginning. In practice, this approach causes that

[☆] This paper is for CAEE special section SI-bciot.Reviews processed and recommended for publication to the Editor-in-Chief by Guest Editor Dr. Shaohua Wan.

[*] Corresponding author.
E-mail addresses: thyla@zut.edu.pl (T. Hyla), jpejas@zut.edu.pl (J. Pejaś).

https://doi.org/10.1016/j.compeleceng.2019.106523
0045-7906/© 2019 Elsevier Ltd. All rights reserved.

it is computationally impossible to change a blockchain used by a popular cryptocurrency under the condition that no one has more than 50% of the computing power.

In contrary, the permissioned blockchains work in private networks, where participants are whitelisted [2]. The permissioned blockchains usually use Practical Byzantine Fault Tolerance (PBFT) algorithm as a consensus mechanism. A good example of a permissioned blockchain implementation and evaluation was described in 2019 by Knirsch et al. [4]. Moreover, the data in a blockchain can be stored using two approaches. In first, data are embedded into a blockchain (on-chain storage). In second, hashes of records are embedded into a blockchain (off-chain storage).

The first solution that allows eliminating re-timestamping operation was to publish periodically a hash from a current block on a trusted third-party website or print in a daily newspaper [5]. A few years after Bitcoin was introduced several solutions that use Bitcoin to permanently store a hash were presented. Mainly, because it is possible to read the time when each block in a blockchain was mined. Because of that, when it is possible to insert a hash from a document into a cryptocurrency blockchain, then block mining time can act as (trusted) timestamp. In 2012, Clark and Essex [6] showed a general approach to carbon dating and proposed CommitCoin. Stavrou and Voas [7] analysed different approaches for a blockchain that can be used for timestamping purposes.

Several techniques of data insertion into Bitcoin were proposed and reviewed by Sward et al. [8], e.g. inserting a hash as a Bitcoin address in a transaction (what in fact is a hash of a user public key). Gao and Nobuhara [9] also proposed storing data in a blockchain by encoding it into Bitcoin address. In their experiments, the transactions were recorded in approximately 24 min with a minimum cost of 0.0005BTC. Currently, a few commercial services that allow to embed a hash into Bitcoin exist, e.g. *BTProof, po.et*, or *OrginStamp*. One of the most advanced projects that allow timestamping using Bitcoin is *OpenTimestamps* project [10] that uses Merkle hash trees to aggregate data before embedding it into Bitcoin. The purpose of the *OpenTimestamps* is to provide a scalable timestamping service.

Moreover, other cryptocurrencies can be used to embed a hash into a blockchain using a simpler approach, e.g., smart contracts in Ethereum. Such an application of smart contracts is used as an example in Ethereum tutorials. Execution of a smart contract function that changes it state requires payments using Ethereum gas. Additionally, there are several possible attacks and problems related to smart contracts [11] that have to be considered.

One of the main drawbacks of solutions mentioned above is that PoW consensus do not fulfil one of the integrity requirements, i.e. *Finality* property [12]. The *Finality* property states that no transaction will be rejected after it is approved and recorded in the ledger. Simplifying, in Bitcoin it is usually assumed that a transaction is considered as accepted when about six consecutive blocks are accepted, because with a high probability (but not with certainty) this chain containing the transaction will be the main one. That property in PoW consensus is called *Probabilistic Finality* property. In *Probabilistic Finality*, the risk that a transaction is rejected is mitigated by operational actions, e.g., waiting for some time after a transaction is recorded in a block and several new blocks are added after it [12].

Except using a blockchain as timestamping service, a blockchain can be also used to establish a public key infrastructure. For example, Fromknecht et al. [13] proposed *Certcoin* that has no central authority and uses distributed data structures stored in a blockchain. They assumed that cryptocurrency like Bitcoin or Namecoin is a permanent, public ledger into which information can be inserted. Because these cryptocurrencies are based on PoW consensus, as mentioned above and described in details in [12] cannot fulfil all integrity requirements, i.e. *Finality* property. Therefore, in such application also a permissioned blockchain should be used like HyperLedger Fabric v0.6. In recent years, several applications, especially in healthcare domain, based on permissioned blockchains were proposed. For example, a private blockchain (Hyperledger Fabric) was used by Ichikawa et al. [14] to ensure the integrity and availability of stored personal health records.

1.1. Motivation and contributions

The motivation for this paper is to describe a complete solution that would allow to maintain validity of a large number of digital signatures using a blockchain without the need to do maintenance operations and analyse related security problems. Individual users or small firms usually do not have specialised IT infrastructure that includes a long-term archive, which maintain validity of digital signatures by periodically re-timestamping digital signatures. Hence, it is a great advantage to store a digitally signed file together with archival data on a disc and there is no need to do anything more with it.

The main contributions of the paper are as follows. First, Round-based Blockchain Time-stamping Scheme (RBTS scheme) is proposed that allow maintaining signature validity (preserving non-repudiation of origin property) without the necessity to use timestamps from trusted third parties. The scheme is based on Haber-Storrneta scheme [15] and OpenTimestamps scheme [10]. In addition to OpenTimestamps, RBTS scheme contains new algorithm for signatures verification according to modified shell model and an algorithm (*Update-Round*) for adding new signed documents which is extended by instructions specific to signed digitally documents. The paper also analyses and discusses several security issues related to a blockchain that must be considered. In contrary to OpenTimestamps, the RBTS scheme requires a permissioned blockchain, because only those blockchains fulfil all integrity requirements (*Agreement on Transaction Validity, Tamper Evidence, Finality*). Alternatively, the scheme can be used with permissionless blockchain (cryptocurrency), but it will provide lower overall security level. Instead of *Finality* property the *Probabilistic Finality* property can only be assumed and there are no guarantees that current cryptocurrencies will be available in the long-term. The schemes fulfil also the proof-of-existence property. Similar to OpenTimestamps, although one might argue if OpenTimestamps fulfil that property as it is based only on Bitcoin that has only *Probabilistic Finality* property.

The main advantage of RBTS scheme, comparing to XML Advanced Electronic Signatures (XAdES) and RFC 4998 Evidence Record Syntax (ERS) that are current mainly used solutions to the problem of prolonging signature validity, is that after inserting data into a blockchain, a user can store a signed document in his storage without the need to do any maintenance actions in the future, except monitoring security of cryptographic algorithms (like in current Public Key Infrastructure (PKI)). The scheme is scalable, i.e., it requires embedding a constant number of bytes into a blockchain, independent from a number of input documents. The scheme allows verifying signature validity using *Chain Model* and under some conditions using *Modified Shell Model*. Moreover, the scheme satisfies non-repudiation requirement for digitally signed documents.

1.2. Paper structure

The rest of this paper is organized as follows. In the following section, models of signature verification in the long period are briefly reviewed. The Section 3 contains informative description of the proposed scheme, a scheme definition, its security model, and detailed scheme description. In Section 4, a security analysis and a performance evaluation are described. Finally, the study's conclusions are presented.

2. Preliminaries

Nowadays, three validity models can be used to verify digital signature [16]: *shell model, modified shell model* and *chain model*. In all models a signature must be mathematically correct; they differentiate in a way how a chain of certificates is validated. In the *shell model*, verification time is a basis for the validation decision. In the *modified shell model*, the signing time is a basis for the validation decision (if a signature is valid at signing time, it remains valid for all time). In the *chain model* all certificates in a certificate chain must be valid at a time when a subordinate certificate was issued. In a situation when in a subordinate certificate validity period is shorter or equal to a root certificate validity period, the validation in both models will be identical. The formal definitions for the models are [16]:

Definition 1. (Modified Shell Model) A digital signature is valid at verification time T_v if all certificates in the certification chain are valid at T_s: $T_i(k) \leq T_s \leq T_e(k)$ for all $1 \leq k \leq N$ and no certificate is revoked at T_s.

Definition 2. (Chain Model) A digital signature is valid at verification time T_v if:

1. The end-entity certificate $Cer(N)$ is valid at the signing time T_s: $T_i(N) \leq T_s \leq T_e(N)$ and $Cer(N)$ is not revoked at T_s.
2. Every CA certificate in the chain is valid at the issuance time of the subordinate certificate in this chain: $T_i(k-1) \leq T_i(k) \leq T_e(k-1)$ and the certificate $Cer(k-1)$ is not revoked at $T_i(k)$ for all $2 \leq k \leq N$.

Notation: T_v - verification time, T_s - signing time, T_i issue time, T_e - expiration time, Cer - a certificate, $Cer(N)$ - a certificate chain with N certificates, $Cer(k)$ - kth certificate from a certificate chain, CA - Certificate Authority.

Only models mentioned above, i.e., *Chain Model* and *Modified Shell Model*, can be used for long-term signature validation. The basic variant of the scheme presented in the next section works in *Modified Shell Model*, because certificates' policies of most of PKIs do not allow issuing certificates with validity longer than a parent (issuer) certificate.

3. Signature verification based on a blockchain

In this section, a scheme that allows verifying signatures in long-term and uses a blockchain instead of timestamps is proposed. The scheme is designed to be efficient and scalable. Hence, the amount of data that must be inserted into a blockchain is constant and does not depend on the number of signed documents on input. The linking with a previous block is a similar approach to Haber and Stornetta scheme [15] and data is inserted to a blockchain similar to OpenTimestamps [10]. The schemes satisfies the proof-of-existence property (similar to OpenTimestamps). Furthermore, the scheme also satisfies the non-repudiation-of-origin property of digitally signed documents. The scheme is designed to allow verifying digital signatures using *Modified Shell Model*.

3.1. Informative description

The user signs a document and then sends it to Long-Term Authority (LTA) in one of commonly used formats, e.g. XAdES. The signature includes declared signing time and a signer's certificate. The LTA collects data sent by users during a round which lasts P_r seconds (the P_r should be equal to an average period P_{B_i} between blocks). At the end of round, LTA verifies signatures from all documents and rejects invalid ones. Afterwards, LTA creates a hash tree, where leaves of the tree are documents and a root of the tree is the hash that is inserted into a blockchain (Fig. 1). The root of the tree is also linked with a last block from the blockchain and with current time (the time is taken from LTA clock synchronised with reliable time signal, e.g. from Global Positioning System).

LTA for every signed document creates a reduced hash tree, gets a certificate chain associated with user's certificate, gets revocation objects (i.e., Certificate Revocation List (CRL) lists, Online Certificate Status Protocol (OCSP) responses) and creates

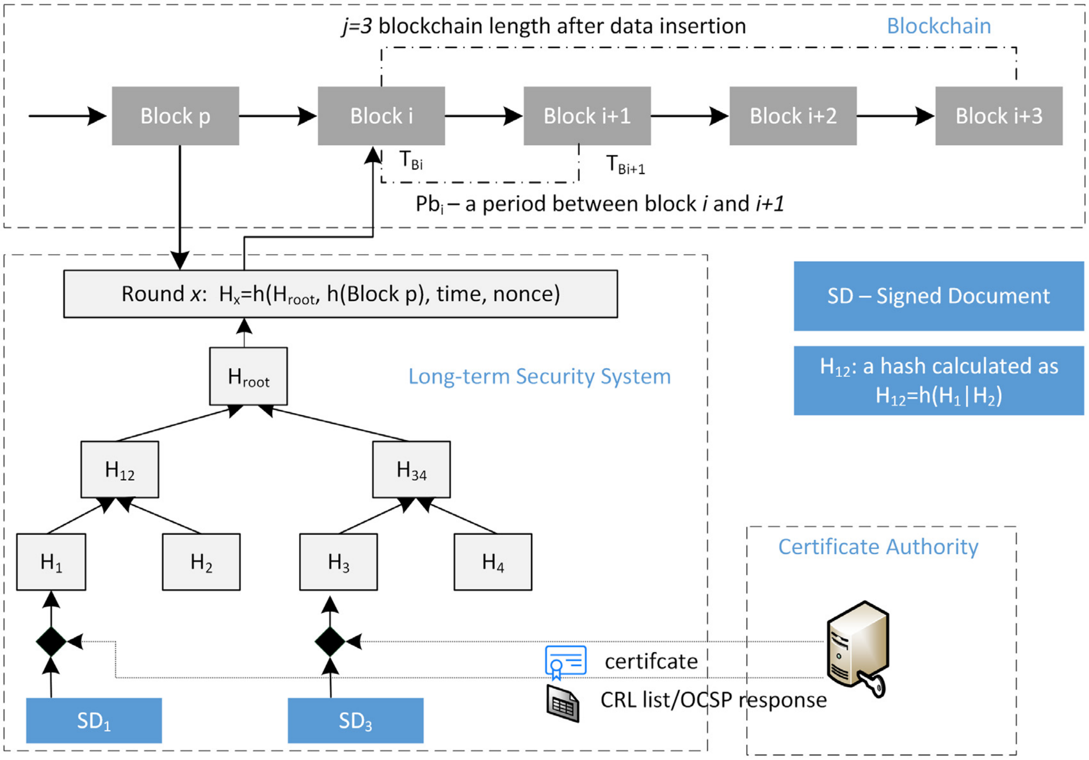
j=3 blockchain length after data insertion
Blockchain
Block p
Block i
Block i+1
Block i+2
Block i+3
T_{Bi}
T_{Bi+1}
Pb_i – a period between block i and i+1
Round x: H_x=h(H_{root}, h(Block p), time, nonce)
H_{root}
Long-term Security System
H_{12}
H_{34}
H_1
H_2
H_3
H_4
SD_1
SD_3
SD – Signed Document
H_{12}: a hash calculated as H_{12}=h(H_1|H_2)
Certificate Authority
certifcate
CRL list/OCSP response

Fig. 1. A diagram of a round-based blockchain timestamping.

Blockchain Archival Object (BAO) that is send back to a user or a storage system. As long as a blockchain is considered secure, no further security maintenance actions are required. The system is easy scalable as a number of transactions send to a blockchain is constant.

The insertion to a blockchain could be done in several ways. In permissioned blockchains, the data are inserted according to the blockchain specification. In a cryptocurrency (only when the cryptocurrency fulfils all security requirements), a hash can be inserted as a hash of a public key of a transaction' recipient. The cost of a transaction is a commission for miners and burned coins that are sent to a non-existing address. The LTA monitors the transaction until it is inserted by miners into a block and *j* following blocks are mined. If the transaction is rejected, it creates another one with a higher transaction fee.

The verification algorithm consists of three steps. In the first step, a verifier computes a hash of the round and verifies if it is inserted into the blockchain. In the second step, the mathematical correctness of the signature and claimed signature time are verified based on the time when a block, which contains the hash from the round, was mined. In the final step, every certificate from the certificate chain, based on the data stored in BAO, is verified using revocation objects.

3.2. Scheme definition

In the proposed Round-based Blockchain Time-Stamping scheme (RBTS scheme) the following entities take part: Long-Term Authority (LTA), Verifier - a user who verify a signature, Signature Creator - a user who create a signature and sends a signed document to LTA.

In this paper, the following notation is used (similar to [16]): B - Blockchain; B_i current ith block of the blockchain B, SD_k - kth signed document, BAO_k - Blockchain Archival Object for SD_k, ADO_k - Archival Document Object - partially filled BAO_k, h - a cryptographic hash function, T_s - claimed time when a document was signed, T_c - current time; T_x - time when a round was computed, T_i - certificate issue time, T_e - certificate expiration time, T_{B_i} - time when B_i was mined, Cer - a certificate, $CerCh_k$ - a certificate chain for SD_k, RI_k - revocation information related to $CerCh_k$, HT_x - a hash tree for round x, RH_k - a reduced hash tree for BAO_k.

Definition 3. The Round-based Blockchain Time-Stamping scheme consists of four polynomial time algorithms:

1. **Setup** *()* → *params; Run by LTA;* On the output of this algorithm parameters of the scheme are fixed, i.e.: supported digital signature algorithms, supported hash functions, supported digital signature formats, and the scheme parameters: blockchain B, cryptographic hash function h, j_{tr} - a minimum blockchain length that is required for the transaction F in

B_i to be considered valid; t_{int} - the maximum period between a previous block and a block with an embedded hash from a round.

2. **Create-Archival-Document-Object**. $SD_k \rightarrow ADO_k$, *for* $k = 1 \ldots n$*; Run by LTA;* This algorithm takes all signed documents as input and outputs archive document objects for each signed document SD_k.
3. **Update-Round**. $(ADO_1, \ldots, ADO_n, x) \rightarrow (BAO_1, \ldots, BAO_n, H_x, B_i)$*; Run by LTA.* This algorithm takes archival document objects as input and outputs these objects with additional data necessary to verify a signature in long-term and a hash from a round x (H_x) is inserted into a block B_i of a blockchain.
4. **Long-Term-Verify**. $(BAO_k, B_i) \rightarrow$ *true/false; Run by Verifier;* This algorithm takes the blockchain archival object and the blockchain as input and returns *true*, when the signature and claimed signature time are valid. Otherwise returns *false*.

The RBTS scheme can be configured (in *Setup* algorithm) to support a different digital signature' algorithms set. Therefore, algorithms like *Generate-Key, Sign*, and *Verify* are not in the scope of the scheme. However, algorithm *Verify* is internally used by *Long-Term-Verify* algorithm.

3.3. Security model

A document-centric threat modelling approach was used to model security of RBTS scheme (similarly to [17]). The secure archiving process is a subject to several threats. During the design process of the scheme, the following threats were identified:

- accidental or intentional unauthorised modification of a document;
- accidental or intentional unauthorised modification of a digital signature attached to a document;
- unauthorised submission of a backdated document;
- submission of a document with an invalid digital signature;
- loss of validity of a digital signature (e.g. disclosure of a private key, end of a certification validity period);
- loss of trust in authorities that issue a certificate used to sign a document;

The scheme should be resistant to attacks that implement above threats. The RBTS scheme satisfies the following security requirements:

- R1: *Non-repudiation of origin* - according to ISO/IEC 13888-1:2009: *service intended to protect against an entity's false denial of having created the content of a message (i.e. being responsible for the content of a message)*;
- R2: *Proof-of-existence* - it is possible to proof that a document existed after certain date and before another date (i.e., a document existed between two points in time);

The RBTS scheme works with the following security assumptions:

1. Digital signature scheme should be proven secure against existential forgery under an adaptive chosen message attack (for definition see [18]).
2. The digital signature scheme is independent from a signature scheme used in a blockchain.
3. The blockchain fulfil the following integrity requirements (for full definitions see [12]):
   (a) *Agreement on Transaction Validity* - only a legitimate transaction can be recorded in the ledger, depending on the transaction semantics.
   (b) *Tamper Evidence* - the ledger be not tampered with and consistent among participants.
   (c) *Finality* - no transaction will be rejected after it is approved and recorded in the ledger.
4. The cryptographic primitives, i.e. digital signature schemes, hash functions are considered to be secure.
5. Additionally, certificates from a certificate chain are not revoked in the period between calculation of a hash from a round and insertion of the hash into a blockchain.

Notice, that there are no confidentiality assumptions related to a blockchain. This results from the fact that RBTS stores in the blockchain only hashes that can be publicly posted.

There is no requirement for a specific type of blockchain, except that it must fulfil the integrity requirements from assumption no. 3. Mainly, the consensus mechanism used in the blockchain has the greatest impact on the fulfilment of the integrity requirements. Currently, all three integrity requirements, i.e., *Agreement on Transaction Validity, Tamper Evidence*, and *Finality* are satisfied by the PBFT consensus. PoW and PoS (Proof-of-Stake) used in cryptocurrencies do not satisfy *Finality* [12], and PoA does not satisfy *Tamper Evidence* [19]. Alternatively, the assumption 3c) can be replaced with *Probabilistic Finality* requirement. Such assumption lower overall security level but might be acceptable in some cases and allows using permissionless blockchains, i.e. cryptocurrencies.

Long-term Authority does not have to be a trusted authority. LTA is expected to execute algorithms without deviation, since any deviation can be easily verified by users using *Long-Term-Verify* algorithm. LTA does not possess any secret keys. Users are sending only a hash of a document and a signature. If full privacy is required (i.e., LTA does not know identity of signature creators), then users should run *Create-Archival-Document-Object* algorithm and steps 1 to 9 and step 30 from *Update-Round* algorithm.

Three types of adversaries A_1, A_2 and A_3 are defined to describe formally the security model:

1. Adversary A_1 simulates a malicious user who want to cause the situation where a signed document with an invalid date will return true when verified using *Long-Term-Verify* algorithm. A_1 possess a valid certificate and a signing (private) key or possess currently invalid ones that were valid in the past. In other words, the A_1 performs the following attacks:
   (a) A_1 creates a document with invalid date, but with a valid signature and finds a collision with previously embedded hash H_x in blockchain B.
   (b) A_1 creates a document with invalid date, but with a valid signature and embeds forged hash H_x from round x into block B_i of blockchain B that mining time match the invalid date.
2. Adversary A_2 simulates an adversary who want to invalidate a signature by invalidating at least one certificate from a certificate chain by any possible way and causing *Long-Term-Verify* algorithm to return *false* for a correct signature, i.e. for a BAO_k. In other words, A_2 has the following abilities:
   (a) A_2 tricks CA that issued a given certificate, to invalidate that certificate;
   (b) A_2 steels user's private key forcing the user to invalidate his certificate;
   (c) A_2 causes CA to stop functioning or to lose trust.
   A_2 executes the attack in which he or she replaces transaction F, where $H_k \in H_x \in F$, from B_i to $F^{'}$ of his choice, where $F \neq F'$.
3. Adversary A_3 simulates a malicious user that cooperates with a malicious LTA. The A_3 wants to create a signature for any given user without possessing a private key and sends it to LTA to be embedded in blockchain B, so *Long-Term-Verify* will return *true*. The A_3 can perform two attacks:
   (a) A_3 creates a valid signature without possessing a private key and sends it to LTA. LTA proceed normally, because the signature is correct, embeds H_k into B and creates BAO_k.
   (b) A_3 creates an invalid signature and sends it to LTA. LTA behaves maliciously and accepts the invalid signature. Then LTA creates valid certificate chain $CertCh_k$ and revocation information object RI_k without possessing a CA private key. Finally, LTA embeds H_k into a blockchain B and creates BAO_k.

Theorem 1 (RBTS scheme security)**.** *The RBTS scheme achieves security goals (non-repudiation of origin and proof-of-existence) when adversaries A_1, A_2, and A_3 have negligible probability of success under the Assumptions 1–5.*

3.4. Round-based blockchain timestamping scheme

The RBTS scheme consists of four algorithms. The *Setup* (Figure Algorithm 1) in executed upon an initialisation of LTA system. *Create-Archival-Document-Object* (Figure Algorithm 2) is executed whenever LTA receives a new signed document and *Update-Round* (Figure Algorithm 3) is executed at the end of a round. The round should last an average period between blocks, but it always finishes when a new block in a blockchain is mined. The *Long-Term-Verify* (Figure Algorithm 4) can be executed in any moment by the verifier to check if a digital signature is valid.

Notice that *RI* are retrieved at the *Update-Round* execution time to eliminate the possibility that *Cer* will be revoked in the period between the SD_k is received and the *Update-Round* is executed. The SD_k can be written in one of common formats (e.g. XAdES). Also, the ADO_k can be stored in an XML formal compliant with XAdES.

Algorithm 1: Setup.

```
Input  : none
Output: params
Initiate empty params structure.
Choose and insert IDs of the following algorithms to params:
    Alg.B.hash - a cryptographic hash function used in a blockchain;
    Alg.tree - an algorithm for creating Merkle tree;
    Alg.rtree - an algorithm for creation of a reduced Merkle tree;
    Alg.Sig - a digital signature scheme (e.g. Rivest-Shamir-Adleman (RSA) or Elliptic Curve Digital Signature Algorithm (ECDSA));
    Alg.Sig.key - key parameters of Alg.Sig.
Assign to $j_{tr}$ the value of a blockchain length that will mean that $B_i$ is successfully inserted into $B$ ;   ▷ when a permissioned blockchain is used, then $j_{tr} = 1$
Assign to $t_{int}$ the maximum allowed period between a previous block and a block in which a hash from a round is embedded.
```

In step 12 in *Update-Round*, when Bitcoin is used instead of a permissioned blockchain, the H_x can be embedded as a recipient public address (a recipient public address is a hash of his public key). As long as *Update-Round* uses in step 11 *Alg.B.hash*, the H_x is undistinguishable from a real public address and therefore cannot be rejected because of this. If F is not inserted into B_i, then in B_{i+1} a transaction fee should be increased.

Algorithm 2: Create-archival-document-object.

```
   Input  : $SD_k$
   Output: $ADO_k$
 1 The algorithm creates an empty archival document object $ADO_k \in SD_k, CerCh_k, RI_k, RH_k, Info$, where $Info$ are other
   technical information necessary to read and verify $ADO_k$;
 2 Insert $SD_k$ into $ADO_k$, $Ts \in SD_k$, also $SD_k$ contains the signature value, $Alg.Sig$ and $Alg.Sig.key$ information;
 3 if $SD_k \notin CerCh$ then
 4     find and download $CerCh_k$;
 5 end
 6 Insert $CerCh$ into $ADO_k$;
 7 Insert $Info$ into $ADO_k$, i.e., other technical metadata (e.g. LTA identification data, blockchain $B$ reference);
```

Algorithm 3: Update-round.

```
    Input  : $\{ADO_1, \dots, ADO_n\}$, $x$ - round number, $n$-number of SD in round $x$;
    Output: $\{BAO_1, \dots, BAO_n\}$, $H_x \in B_i$
 1  for $k \leftarrow 1$ to $n$ ;                                    ▷ verify if all signatures are valid
 2  do
 3      Verify if a signature from $SD_k$ is mathematically correct using verification algorithm from $Alg.Sig$;
 4      Verify if claimed signature time $T_s$ is earlier or equal to current time and not older than time $T_{B_{i-1}}$ when a last
        block in $B$ was mined ($T_{B_{i-1}} \le T_s \le T_c$);
 5      Get CRL lists or query OSCP service and get revocation information about all certificates $RI_k$ from $CerCh_k$;
 6      Verify if all certificates in $CerCh_k$ are valid and are not revoked using information from $RI_k$;
 7      If one of the verifications fails, reject $ADO_k$;
 8      Else insert $RI_k$ to $ADO_k$ which results in $BAO_k$;
 9  end
 10 Create a root hash $H_{root}$ using $Alg.tree$ and $BAO_k, k = 1.n$ as an input;
 11 Create a hash for the round $x$, i.e. $H_x = h(H_{root}, h(B_{last}), T_c, r)$, where $r$ is a random value, $h$ is $Alg.B.hash$, and
    $B_{last} = B_{i-1}$;
 12 Prepare a transaction $F$ for $B$ which has $H_x$ built in as one of its fields, i.e. $H_x \in F$;
 13 while $F \notin B_i$ ;                                           ▷ insert transaction into blockchain
 14 do
 15     send $F$ to be integrated in $B_i$;
 16     wait until $B_i$ is mined;
 17     if $H_x \in F \in B_i$ then
 18         break;
 19     end
 20     $i = i + 1$;
 21 end
 22 wait until $j_{tr}$ following blocks are mined;
 23 if a chain with the block $B_i$ is rejected then
 24     repeate step 14;
 25 end
 26 for $k \leftarrow 1$ to $n$ ;                                    ▷ prepare the archival objects for storage
 27 do
 28     Create $RH_k$ from $HT_x$ using $Alg.rtree$;
 29     Insert $RH_k$, $h(B_{last})$ into $BAO_k$;
 30     Insert $T_x = T_c$, $i$ and $r$ into $Info_k$ from $BAO_k$;
 31 end
```

Long-Term-Verify algorithm verifies signatures using *Modified Shell Model*. In case the verification using *Chain Model* is required, the certificates from a certificate chain can be timestamped using *Update-Round* before their expiration date and *Long-Term-Verify* must additionally verify if each certificate' timestamp is correct.

4. Results and discussion

In this section, firstly performance evaluation of RBTS scheme is presented. Next, security of the scheme is analysed, followed by a discussion on long-term security aspects of blockchains.

```
Algorithm 4: Long-term-verify.
   Input  : $BAO_k$
   Output: $true/false$
 1 Calculate $H_{root}$ using $RH_k$ and $BAO_k$ as an input to $Alg.tree$;
 2 Calculate $H_x = h(H_{root}, h(B_{i-1}), r)$, where $h = Alg.B.hash$;
 3 Get $B_i$ which contains $H_x$ from $B$;
 4 Find block $B_{last}$:
 5     $p = i - 1$;
 6     while $h(B_p)! = h(B_{last})$ do $p = p - 1$
 7     $B_{last} = B_p$;
 8 Verify if $j_{tr} > j$, i.e. if blockchain length $j$ after $B_i$ is more than required threshold $j_{tr}$;
 9 Verify if a signature from $SD_k \in BAO_k$ is mathematically correct using a verification algorithm from $Alg.Sig$;
10 Verify claimed signature time $T_s : T_{B_{last}} \le T_s \le T_x < T_{B_i}$;
11 Verify if $t'_{int} = T_{B_i} - T_{B_{last}} < t_{int}$
12 foreach $Cer$ in $CerCh_k$ do
13     Verify if $Cer$ is mathematically correct;
14     Verify if $Cer$ is not revoked based on information from $RI_k$;
15     Verify if $Cer$ was valid in period between $B_{last}$ and $B_i$: $T_i \le T_{B_{last}} < T_{B_i} \le T_e$;
16 end
17 If one of the verifications fail or search in the step 3 fail, return $false$, else return $true$;
```

Table 1
Performance comparison.

Property	Newspaper based timestamp (historical)	XAdES	ERS	Proposed RBTS scheme
creation algorithm for n documents	$n \cdot t_{verify}$ $+n \cdot t_{archive}$ $+t_{paper}$	$n \cdot (t_{verify}$ $+t_{timestamp}$ $+t_{archive})$	$n \cdot t_{verify}$ $+t_{round}$ $+t_{timestamp}$ $+n \cdot t_{archive}$	$n \cdot t_{verify}$ $+t_{round}$ $+t_{bchain}$ $+n \cdot t_{archive}$
maintenance algorithm for n documents	not required	$n \cdot t_{timestamp}$ periodical time-stamping	$t_{timestamp}$ periodical time-stamping	not required
verification algorithm for a document after p re-timestamping periods	t_{verify} requires access to a newspaper	$t_{verify}+$ $p \cdot t_{ts\_verify}$	t_{verify} $+t_{tree}+$ $p \cdot t_{ts\_verify}$	t_{verify} $+t_{tree}$ $+t_{b\_verify}$
long-term trust source	newspaper: inability to modify all copies	Trusted Timestamps' Authority	Trusted Timestamps' Authority	blockchain: integrity assumptions

4.1. Performance evaluation

Similar security goals can be achieved through three solutions: (1) newspaper based timestamping (historical) - a hash from a signed document is published in a daily newspaper, i.e. security is based on an assumption that no one is able to change all copies of the newspaper; (2) re-timestamping every n years, e.g., XAdES with archival timestamps, or (3) re-timestamping every n years with usage of hash trees to group documents (e.g., Evidence Record Syntax (ERS) RFC 4998).

In that section, the performance of two algorithms from RBTS scheme is analysed, i.e., *Update-Round* and *Long-Term-Verify*, that have the biggest impact on the performance. The execution of *Update-Round* algorithm can be divided into a few phases (n - number of input documents), i.e. phase 1: $n \cdot t_{verify}$ (time required to verify all signatures), phase 2: t_{round} (time required to calculate a hash for a round, i.e., a set of n documents), phase 3: t_{bchain} (time required to embed a hash for a round into a blockchain), phase 4: $n \cdot t_{archive}$ (time required to create an archival form for the signature).

The Table 1 contains comparison of RBTS scheme with solutions (1)–(3). All algorithms in other solutions, that were used to create an archival signature, have similar parts for verification of an input signature and for creation of an archival object. Therefore, t_{verify} is similar in all schemes (in practice one might omit this part of an algorithm when there is low probability of incorrect signatures). The preparation of an archival object is a simple part of the algorithm with practically no computations, so the $t_{archive}$ in total time is negligible. The algorithms differ in the part related to measures used to ensure long-term validity. The XAdES must get a timestamp from TA for every signature. Therefore, when n is large, the XAdES performance depends on TA capability of simultaneous timestamps' generation. ERS requires calculating a hash tree and timestamping only the root of that tree for n signed documents. RBTS in contrast, requires calculating a hash tree and embedding the root of that tree into the blockchain. Therefore, for n documents ($n \geq 2$) the ERS is the fastest solution. The

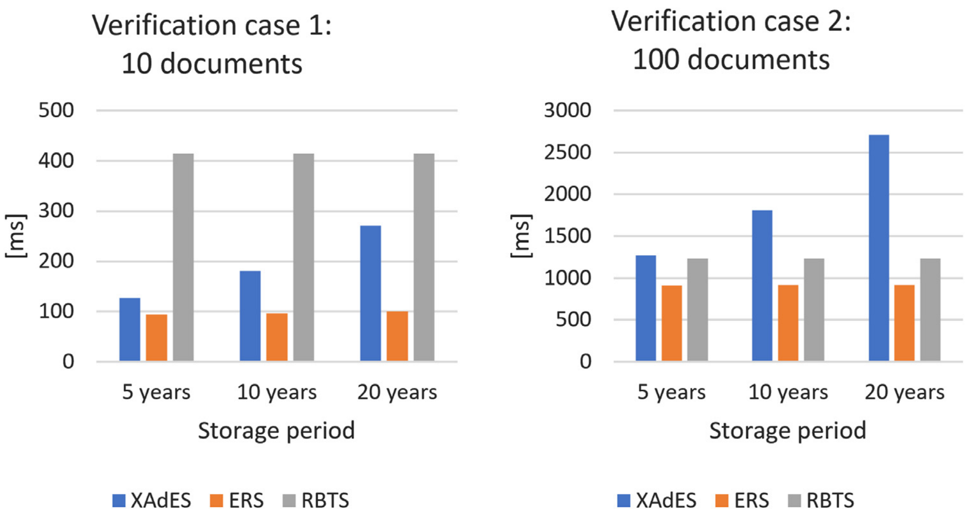

Fig. 2. Comparison of verification time.

blockchain embedding time mostly depends on average time between blocks and is around 60 min (6 × 10 min) in Bitcoin and 90 s (6 × 15 s) in Ethereum.

The XAdES and ERS require maintenance, i.e., re-timestamping usually every 2 or 3 years. RBTS scheme similarly to a historical approach that publish timestamps in daily newspapers does not require any future actions (with the assumptions, like in other solutions, that used cryptographic algorithms will not be broken).

The verification procedure is generally fast in all solutions including proposed RBTS scheme when one document is verified. All solutions require to verify the initial signature based on stored revocation information. XAdES verification has similar time as ERS solution only requires to additionally recreate (from a reduced hash tree) and verify a root of a hash tree. That operation is fast, and its execution time is negligible in contrast to a timestamp verification.

The verification algorithm was implemented and compared with XAdES and ERS. The algorithm was implemented using C#, and two sets of documents were created. The first set consisted of 10 files with average size equal to 1 MB. The second one consisted of 100 files. The ECDSA was used as a signature scheme. The test was carried out on a test computer with an Intel Core i7 7700 K @4,2 GHz processor, 32 GB RAM, and an SSD drive. The test included verification of the two sets of documents after 5, 10, and 20 years. The test assumes that all files were created in one point in time.

The tests' results presented on Fig. 2 show that ERS and RBTS verification time is practically constant and do not depend on storage period (ERS in fact requires verifying one more timestamp for each two years of storage, but the timestamp verification time is below 2 ms). The RBTS is slower than ERS, because to simulate a blockchain in the test was used Bitcoin and *blokchain.info* service that can return a block with a given number. The time required to confirm that a hash is embedded in a blockchain (t_{b_verify} - steps 1 to 11 of the algorithm) was 37 ms plus additional 286 ms to download two blocks (B_i, B_{last}). Possibly, this time might be reduced, when a copy of a blockchain is stored locally.

The addition (creation) algorithms of ERS and RBTS schemes have similar characteristic as verification. This is mostly due to the fact that the schemes differ in one computationally intensive operation. ERS requires timestamp and RBTS embedding data in blockchain (other operations are a few orders of magnitude faster). The RBTS embedding time into a permissioned blockchain is around 10–15 s, in contrast to below 1s time to get timestamp in ERS.

To sum up, when comparing ERS to RBTS, the RBTS is slower, but good enough for practical application and requires no maintenance actions (re-timestamping), which is the biggest advantage of the scheme.

4.2. Security analysis

4.2.1. Sketch of the proof of the theorem

Adversary. A_1 to succeed proceed as follow. Adversary A_1 obtains revocation information object RI_k for his certificate Cer_k for given time T. Then A_1 signs a document in time $T_2 > T_1$ using invalid date T_1 and creates BAO_k. Next, A_1 simulates *Update-Round* algorithm. A_1 has two options if he or she want to succeed. Firstly, he or she can find some hash H_x embedded in blockchain B that lies in block B_i where $T_{B_{last}} \leq T_1 \leq T_x < T_{B_i}$. Afterwards, A_1 has to generate such H'_{root}, which satisfy the equation $H_x = h(H'_{root}, h(B_{last}), T_x, r)$, where $h(B_{last})$, T_x are known to him and constant and r is a random value. A_1 to generate H'_{root} must find a preimage of a hash function h by changing r or adding some hashes from random documents into the hash tree. Because h is assumed to be a secure cryptographic hash function the probability of success $\xi_{A1.1}$ is negligible. Secondly, A_1 prepares a forged transaction F' that contains H_x and embeds it into a block B_i where $T_{B_{last}} \leq T_1 \leq T_x < T_{B_i}$. To do that, adversary A_1 must break *Tamper Evidence* property of a blockchain *B*. The probability of success $\xi_{A1.2}$ of such action is negligible. Therefore, the probability of success of A_1 is equal to a maximum of $\xi_{A1.1}$ and $\xi_{A1.2}$ which is negligible.

Adversary. A_2 to succeed proceed as follow. Adversary A_2 invalidates at least one of *Cert* from $CerCh_k$ for a given BAO_k. However, A_2 does not have access to all copies of BAO_k and therefore the only way for him to cause *Long-Term-Verify* algorithm to return *false* is to modify blockchain *B*. A_2 prepares random transaction $F^{'}$. Then A_2 replaces transaction *F* with $F^{'}$, where $H_k \in H_x \in F$. Similarly to A_1, A_2 has to break *Tamper Evidence* property of a blockchain *B*. The probability of success ξ_{A2} of such action is negligible.

Adversary A_3 to succeed has to break the security of a digital signature scheme, so he or she can sign a message of his choice as any user. Another option for A_3 is to cooperate with LTA that creates false certificate chain $CerCh_k$ and revocation information object RI_k. Also, LTA has to break the security of a signature scheme to create false certificates, i.e., to impersonate a Certificate Authority (CA). The probability of success ξ_{A3} of such action is negligible as it is assumed that a signature scheme used in RBTS scheme must be proven secure against existential forgery under an adaptive chosen message attack.

Adversaries A_1, A_2 and A_3 have a negligible probability of success under the assumptions and therefore, according to Theorem 1, we can say that RBTS scheme achieves its security goals.

4.2.2. Long-term security of cryptographic algorithms

The cryptographic algorithms might become too weak to provide desirable security level due to development of new cryptanalysis techniques. It is also possible that in the next 20 years the general-purpose quantum computers will become a reality. The most popular signature schemes, like RSA or ECDSA are vulnerable to an attack using Shor's quantum algorithm.

In recent years, several quantum safe (resistant to an adversary using a quantum computer) signature schemes have been proposed, e.g., BLISS (Bimodal Lattice Signature Scheme) signature scheme based on lattices or hash-based XMSS (eXtended Merkle Signature Scheme) scheme. However, the schemes are not used in commercial applications yet. Moreover, it is possible that together with a development of quantum computers, the new quantum algorithms will be invented that can be used to break signature schemes currently recognized as quantum safe. The situation with hash functions is slightly better: the Grover's quantum algorithm can be used to reduce only by half the time required time to find a preimage of hash functions. In the long-term, the possibility that someone will break a hash function (popular hash functions, e.g. SHA256, are not provable secure) is a much bigger threat.

One of the simplest and obvious solutions to the problem of long-term security of digital signature schemes is adding redundancy by using two or more sets of cryptographic algorithms and two or more blockchains based on different cryptographic primitives. The signature is valid as long as one of the algorithms' sets is secure. When one sets of algorithms is broken, it is possible to send a document to some notary services for re-signing, so still two algorithms' sets will be valid. It is a conceptually easy solution, but it requires to monitor state of algorithms using off-line methods. It also adds another layer of complexity to a signature verification procedure, when several changes of algorithms have occurred. In case of blockchains like Bitcoin, the hard fork will be one of solutions when the quantum computer will become a viable threat or a hash function (i.e., SHA256 in case of Bitcoin) will be broken.

4.2.3. Long-term security of blockchains

Apart from the fact, that cryptocurrencies fulfil only *Probabilistic Finality* property, there are no guarantees that they will be available in the long-term. However, using a cryptocurrency based on a blockchain has one major advantage, it is possible to observe exchanges rates of a given cryptocurrency. Major fall of the exchange rate might indicate indirectly that a security of a given blockchain was compromised. The long-term security of blockchains is an active research area. Except risk related to cryptographic algorithms mentioned in the previous section, several other issues related to long-term security must be considered:

- The Bitcoin was proposed by Nakamoto [1] and launched without any formal proofs for PoW scheme and the backbone protocol. However, in case of Bitcoin a formal proof was provided later using black-box analysis by Garay et al. [20].
- The blockchain should be decentralised. However, recent observations about Bitcoin shows trends towards centralisation of mining pools. Centralisation of mining power is a threat [21] as it increases a chance for e.g. 51% attack or for a selfish mining attack. In Bitcoin history one mining pool (GHash.io) temporarily achieved more than 50% of mining power.
- The cryptocurrencies like Bitcoin are not coordinated, i.e. there is no authority that might force changes (like increasing the number of transactions in a block) in the protocol. This might in a long-term cause a cryptocurrency to not be able to adapt to requirements on the market and cause the cryptocurrency to be abandoned by users.
- The total energy consumption used by all Bitcoin miners is enormously high and is greater than energy consumption in Austria (in August 2019) [22]. This might cause a shift to Proof-of-Stake (PoS) consensus mechanism. In PoS miners run a process that randomly selects one of them proportionally to the stake that each possesses according to the current blockchain ledger. Ethereum is changing into PoS, but this process will require some coordination. It is not known how the change will influence Ethereum.
- In many countries around the world legal restrictions concerning cryptocurrencies are imposed by governments.
- The vulnerabilities in software might lead to unexpected situations in a blockchain, e.g., CVE-2010-5139 in Bitcoin in 2010 or a bug that caused problems with a change from version 0.7 to 0.8 in Bitcoin [23]. If a transaction is in a block with six subsequent successor blocks, then censoring this transaction has negligible probability. However, the probability of software errors denies this statement. It also shows that *Probabilistic Finality* is not a strong assumption.

- The implicit assumption that blockchains are trust-free is not clear, because blockchain users must have a certain amount of trust into the blockchain providers or software developers [24].

A hash H_x from round x in *Update-Round* algorithm can be embedded using a mechanism like Smart Contracts that are available in Ethereum. This simplifies the process, but the major disadvantage is that such approach adds another layer of security problems related to smart contracts [11]. The security analysis shows that currently Bitcoin (the most widely adopted cryptocurrency) is not mature enough to be used, because there are many unknowns related to its usage in economy and it is unknown what will happen with PoW concept that requires enormous amount of energy.

Nowadays, because of the above facts, it is recommended to use a permissioned blockchain implementing PBFT consensus with the RTBS scheme.

5. Conclusion

The archival document object created by RBTS scheme has a simple form, is easy to store for the long-period and does not require a secure storage to detect unauthorised changes. The main advantage of RBTS scheme is lack of maintenance operations like re-timestamping. Verification requires only blockchain archival object and a blockchain. The performance of the scheme is suitable for practical purposes and the scheme works well with a large number of input documents.

The security model assumes that a used blockchain fulfil integrity properties. Because of that requirement, a permissioned blockchain should be used, e.g. Hyper Ledger Fabric v0.6 based on Practical Byzantine Fault Tolerance consensus. Alternatively, when a lower security level is accepted, the blockchain fulfilling *Probabilistic Finality* property (based on Proof-of-Work consensus), i.e., cryptocurrencies can be used. In that case, two cryptocurrencies should be used at once, which is a standard approach for timestamps generated by trusted third parties.

The RBTS scheme implicitly assumes that users are sending signed documents (or only a digital signatures) immediately after a signature is created. Time stamping provided by the scheme confirms that the document was created in a time interval, not in a specific point of time. In case of permissioned blockchains that interval will be below 1 min, which is enough for most of the applications. Alternatively, if lower security assumptions are allowed, then, for example, in Bitcoin the interval will be around 60 min or and in Ethereum around 1 min.

Even when Long-Term Authority cooperates with a person who send backdated documents, it will have to use an older block from the blockchain. This will cause the interval between a last (previous) block and a block to which the hash from the round will be embedded to increase significantly. Such a case will be detected by *Long-Term-Verify* algorithm. Determination of precise maximum value of that interval that should be accepted is a subject of our future works.

The second specific to RBTS scheme assumption is that certificates are not revoked in the interval starting when the hash for the round is calculated and finished when the hash is integrated into the blockchain. This interval should be short enough to be acceptable. In permissioned blockchains the time between block is usually around 10–15 s (In Bitcoin 10 min, in Ethereum 10–15 s).

Funding statement

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors. The research was performed as part of the authors employment at West Pomeranian University of Technology in Szczecin.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Nakamoto S. Bitcoin: a peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf.
[2] Christidis K, Devetsikiotis M. Blockchains and smart contracts for the internet of things. IEEE Access 2016;4:2292–303. doi:10.1109/ACCESS.2016.2566339.
[3] Sousa J, Bessani A, Vukolic M. A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform. In: 2018 48th annual IEEE/IFIP international conference on dependable systems and networks (DSN); 2018. p. 51–8. doi:10.1109/DSN.2018.00018.
[4] Knirsch F, Unterweger A, Engel D. Implementing a blockchain from scratch: why, how, and what we learned. EURASIP J Inf Secur 2019;2019(1):2. doi:10.1186/s13635-019-0085-3.
[5] Massias H, Avila XS, Quisquater JJ. Design of a secure timestamping service with minimal trust requirement. The 20th symposium on information theory in the benelux; 1999.
[6] Clark J, Essex A. Commitcoin: carbon dating commitments with Bitcoin. In: Keromytis AD, editor. Financial cryptography and data security: 16th international conference, FC 2012, Kralendijk, Bonaire, Februray 27–March 2, 2012, revised selected papers. Berlin, Heidelberg: Springer Berlin Heidelberg; 2012. p. 390–8.
[7] Stavrou A, Voas J. Verified time. Computer 2017;50(3):78–82. doi:10.1109/MC.2017.63.
[8] Sward A, Vecna I, Stonedahl F. Data insertion in Bitcoin's blockchain. http://digitalcommons.augustana.edu/cgi/viewcontent.cgi?article=1000&context=cscfaculty.
[9] Gao Y, Nobuhara H. A decentralized trusted timestamping based on blockchains. IEEJ J Ind Appl 2017;6(4):252–7. doi:10.1541/ieejjia.6.252.
[10] Todd P. Opentimestamps: Scalable, trust-minimized, distributed timestamping with Bitcoin. https://petertodd.org/2016/opentimestamps-announcement.

[11] Atzei N, Bartoletti M, Cimoli T. A survey of attacks on ethereum smart contracts (SoK). In: Maffei M, Ryan M, editors. Principles of security and trust: 6th international conference, POST 2017, held as part of the European joint conferences on theory and practice of software, ETAPS 2017, Uppsala, Sweden, April 22–29, 2017, proceedings. Berlin, Heidelberg: Springer Berlin Heidelberg; 2017. p. 164–86.

[12] Yoshihama S, Saito S. Study on integrity and privacy requirements of distributed ledger technologies. In: 2018 IEEE confs on internet of things, green computing and communications, cyber, physical and social computing,smart data, blockchain, computer and information technology, congress on cybermatics; 2018. p. 1657–64. doi:10.1109/Cybermatics_2018.2018.00276.

[13] Fromknecht SYC, Velicanu D. A decentralized public key infrastructure with identity retention, cryptology eprint archive, report 2014/803. 2014. https://eprint.iacr.org/2014/803.

[14] Ichikawa D, Kashiyama M, Ueno T. Tamper-resistant mobile health using blockchain technology. JMIR Mhealth Uhealth 2017;5(7):e111. doi:10.2196/mhealth.7938. URL http://mhealth.jmir.org/2017/7/e111/

[15] Cattaneo G, Cilardo A, Mazzeo A, Romano L, Saggese GP. In: Proceedings of International Conference on Advances in Infrastructure for Electronic Business, Education, Science, Medicine and Mobile Technologies on the Internet, Scuola Superiore Guglielmo Reiss Romoli, L'Aquila, Italy. 2003.

[16] Baier H, Karatsiolis V. Validity models of electronic signatures and their enforcement in practice. In: Martinelli F, Preneel B, editors. Public key infrastructures, services and applications: 6th European workshop, EuroPKI 2009, Pisa, Italy, September 10–11, 2009, revised selected papers. Berlin, Heidelberg: Springer Berlin Heidelberg; 2010. p. 255–70.

[17] Hyla T, Fray IE, Maćków W, Pejaś J. Long-term preservation of digital signatures for multiple groups of related documents. IET Inf Secur 2012;6(8):219–27.

[18] Goldwasser S, Micali S, Rivest RL. A digital signature scheme secure against adaptive chosen-message attacks. SIAM J Comput 1988;17(2):281–308. doi:10.1137/0217017.

[19] Angelis SD, Aniello L, Baldoni R, Lombardi F, Margheri A, Sassone V. PBFT vs proof-of-authority: applying the cap theorem to permissioned blockchain. In: Italian conference on cyber security, 2018-02-06; 2018.

[20] Garay JA, Kiayias A, Panagiotakos G. Blockchain and consensus from proofs of work without random oracles, cryptology eprint archive. 2018. Report 2017/775. https://eprint.iacr.org/2017/775,

[21] Beikverdi A, Song J. Trend of centralization in Bitcoin's distributed network. In: 2015 IEEE/ACIS 16th international conference on software engineering, artificial intelligence, networking and parallel/distributed computing (SNPD); 2015. p. 1–6. doi:10.1109/SNPD.2015.7176229.

[22] Digiconomist. Bitcoin energy consumption index. https://digiconomist.net/bitcoin-energy-consumption, [Accessed: 2019-08-29].

[23] Park JH, Park JH. Blockchain security in cloud computing: use cases, challenges, and solutions. Symmetry 9 (8). doi: 10.3390/sym9080164

[24] Glaser F. Pervasive decentralisation of digital infrastructures: a framework for blockchain enabled system and use case analysis. In: 50th Hawaii international conference on system sciences, HICSS 2017, Hilton Waikoloa Village, Hawaii, USA; 2017. January 4-7, 2017

Tomasz Hyla received PhD degree in Computer Science, Cryptography in 2011. Currently, Dr Hyla is employed as Assistant Professor and Head of Information Security Research Group at West Pomeranian University of Technology in Szczecin, Poland. His main subjects of interest are cybersecurity, pairing-based cryptography and new digital signature schemes.

Jerzy Pejaś received PhD degree in Control Systems from Gdansk University of Technology. Currently, he is employed as Associate Professor and Dean of the Faculty of Computer Science and Information Technology, West Pomeranian University of Technology in Szczecin, Poland. His main subjects of interest are methods of secure electronic signatures as well as new trends in applied cryptography.