Blockchain-Based Solution for Detecting and Preventing Fake Check Scams

Badis Hammi, Sherali Zeadally, Yves Christian Elloh Adja, Manlio Del Giudice, and Jamel Nebhen

***Abstract*—Fake check scam is one of the most common attacks used to commit fraud against consumers. This fraud is particularly costly for victims because they generally lose thousands of dollars as well as being exposed to judicial proceedings. Currently, there is no existing solution to authenticate checks and detect fake ones instantly. Instead, banks must wait for a period of more than 48 h to detect the scam. In this context, we propose a blockchain-based scheme to authenticate checks and detect fake check scams. Moreover, our approach allows the revocation of used checks. More precisely, our approach helps the banks to share information about provided checks and used ones, without exposing the banks' customers' personal data. We demonstrate a proof of concept of our proposed approach using *Namecoin* and *Hyperledger* blockchain technologies.**

***Index Terms*—Authentication, blockchain, fake check, hyperledger fabric, integrity, namecoin, security.**

I. Introduction

IN OUR current society, checks represent one of the dominant payment methods. A check is an order written by a depositor instructing the bank to pay a specific amount to a recipient from the depositor's bank account. Unfortunately, numerous malicious scammers exploit some flaws in the banking system to commit frauds. Indeed, frauds employing fake checks are growing rapidly and cost billions of dollars. The number of complaints received by the Federal Trade Commission's (FTC) Consumer Sentinel Database (Sentinel) and the Internet Crime Complaint Center (IC3) more than doubled between 2014 and 2017, rising from 12 781 to 31 980 [1], [2]. In this article, we focus on fake check scams. This fraud is achieved by: 1) targeting people mainly through some email scam; 2) establishing a relationship (a business relationship most of the time); 3) sending them overpaid counterfeit paycheck; and finally 4) asking for the overpayment.

Fake check scam has more disastrous consequences on the victims than many other attacks (e.g., phishing, malware spread, and so on). According to the U.S. Better Business Bureau (BBB) [1], the Postal Inspection Service reports stopping fake checks with a face value of $62 billion from entering the USA in fiscal year 2017 and another 13 724 counterfeit postal money orders totaling over $14 million in 2017 alone [1]. The survey for 2016 saw the first increase in check fraud losses since 2008. During 2016, check frauds cost the banks (worldwide) $789 million, an increase of more than 25% from the $615 million reported losses in 2014 [1]. The average loss to consumers in general from counterfeit checks was $1008 in 2017, and the loss to a victim in the military was $2200 on average [3]. The FTC reports that consumers lost more than $28 million to fake check scams in 2019 alone. The median loss reported was $1988, which is more than six times the median loss on all frauds tracked by the FTC [4]. The FBI's IC3 database reports 16 368 victims of advanced fee scams that lost a collective $57.8 million in 2017 [2], [5]. Besides, check fraud occurrences are likely vastly underreported. Only an estimated 29% of fraud victims report to any sort of authority such as the FTC or BBB [6] and less than one in ten victims ever report to law enforcement [1].

In addition to the financial and psychological harm, the victims are, most of the time, exposed to judicial proceedings because in the eyes of the law, they tried to scam the bank [2], [7].

To the best of our knowledge, there is no method designed to specifically protect users from fake check scams. In this context, we believe that the best solution to protect users is the detection of fake checks well before they are cashed. Certainly, there are some measures to detect the authenticity of the physical checks[1,2], (e.g., check's edges, magnetic ink character recognition line, bank logo, and paper quality). Nonetheless, con artists excel in the art of trickery and create very realistic checks especially today when numerous scammers use professional printers and magnetic ink. Moreover, currently, numerous banks propose to users to print their checks themselves, which removes the physical protections from the checks. Consequently, the check authentication solution implemented must be more effective, and each bank must ensure that the submitted check is provided by

Manuscript received 29 September 2020; revised 15 February 2021; accepted 3 June 2021. Date of publication 1 July 2021; date of current version 1 November 2022. Review of this manuscript was arranged by Department Editor D. Sarpong. *(Corresponding author: Badis Hammi.)*

Badis Hammi is with the EPITA School of Engineering and Computer Science, 94270 Le Kremlin-Bicêtre, France (e-mail: badis.hammi@epita.fr).

Sherali Zeadally is with the University of Kentucky, Lexington, KY 40506 USA (e-mail: szeadally@uky.edu).

Yves Christian Elloh Adja is with Telecom Paris, 91120 Palaiseau, France (e-mail: ello.adja@telecom-paristech.fr).

Manlio Del Giudice is with the University of Rome, Link Campus, 00162 Roma, Italy, and also with the Paris School of Business, 75013 Paris, France (e-mail: m.delgiudice@unilink.it).

Jamel Nebhen is with the Prince Sattam bin Abdulaziz University, Al-Kharj 16278, Saudi Arabia (e-mail: j.nebhen@psau.edu.sa).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TEM.2021.3087112.

Digital Object Identifier 10.1109/TEM.2021.3087112

[1] [Online]. Available: https://www.consumer.ftc.gov/articles/how-spot-avoid-and-report-fake-check-scams

[2] [Online]. Available: https://www.aarp.org/money/scams-fraud/info-04-2011/scam-alert-fake-checks.html

a real trusted authority before cashing it. Nevertheless, designing such a solution is very challenging due to the following difficulties.[3]

1) *Data sharing between banks:* Before paying a check, each bank (Cashing-Bank) must ensure that the check was really provided by a trusted authority (another bank). This verification is possible if each bank shares information about its provided checks. In other words, when a bank provides a checkbook to a customer, it shares the information about the customer and the provided checks. But no bank will share such information with other banks, mainly because of: a) user privacy: since the users have engaged with this bank and not with another bank; and b) commercial competition: if users' information are accessible freely by other banks, nothing prevents any bank from contacting these people and offering them its services. *Thus, it is necessary to design a sharing system, which ensures that the customers' data are not revealed to third parties.*
2) *Nonmodification of existing protocols:* Any proposal that requires any modification in the existing bank protocols such as the modification of the check format to add some data or a modification in the payment procedure will have a chain reaction on numerous parts or protocols of the banking ecosystem. Such consequences make banks resist the adoption of such proposals. *It is, therefore, mandatory that the proposed mechanism does not modify any of the existing protocols, solutions, or procedures and must seamlessly integrate into the current banking ecosystem.*
3) *Management of the sharing mechanism:* If a data sharing mechanism is deployed and used by banks to ensure the authenticity of checks, several issues will arise: a) Who will maintain and manage the mechanism (infrastructure, protocol, and so on)? b) How will the participating banks share the management fees? c) Who decides on the evolution of the mechanism? d) Where will the stored data be kept? e) Who can access the stored data? f) How can these data be accessed? g) If one bank decides not to use the mechanism anymore, how will this affect the other banks. *Consequently, it is mandatory to design a lightweight and low-cost sharing system that does not impose a burden on third parties that deploy it, and the system should be adaptable to handle different conditions.*
4) *Scalability:* Considering the number of banks as well as the large number of customers, *we need to propose a highly scalable mechanism that can handle such a load.*
5) *Authentication:* In the proposed system, third parties must ensure that the shared information is provided by the corresponding trusted bank. More precisely, the Cashing-Bank must ensure that the shared data were provided by the Providing-Bank, and it must also ensure that the Providing-Bank is trustworthy. *Accordingly, it is necessary to equip the sharing system with an effective authentication method.*

Contributions of This Article

We believe, like many other researchers [8]–[10], that blockchain represents a very promising technology for the development of decentralized and resilient security solutions. Therefore, in this article, we propose an effective blockchain-based mechanism that helps the banks to share information about provided checks. More specifically, our approach helps to verify the authenticity of a given check, without exposing the banks' customers' personal data. Following this verification, the Cashing-Bank can decide to continue the transaction or to abort it. Moreover, our proposed approach is cost-efficient, and it does not affect the existing bank's procedures while checking the authenticity of checks. The proposed approach should also not need any additional infrastructure management. We implemented our approach using the public blockchain *Namecoin*. Its evaluation demonstrates its ability in meeting the necessary requirements. To evaluate the performance of our proposed approach, we also deployed our check's authentication scheme based on the private blockchain *Hyperledger*.

The rest of this article is organized as follows. Section II describes the fake check scam. In Section III, we describe our approach for detecting fake checks. Then, Section IV discusses and analyzes our proposed approach. Section V discusses the *Hyperledger* implementation and analyzes the results obtained. Finally, Section VI concludes this article and identifies future research perspectives.

II. Fake Check Scam

In a fake check scam, a con artist asks a victim to deposit a check, which is usually for more than what the victim is owed, and then asks the victim to wire some of the money back. The scammers always have a good story to explain the overpayment [2], [7], [11]. We cite a typical real scenario[4] depicted in Fig. 1: a person that we call Alice sends an advertisement informing that she is available for giving math courses for secondary-school level, through an advertisements website such as *craigslist.org*. A scammer contacts her pretending that he is interested for his child. Both parties exchange by e-mail or even by phone in order to agree on the place, the amount (e.g., $500) and the dates (which are often not confirmed straightaway). Afterward, the scammer pretends to act in good faith by paying Alice in advance, sends her a fake check, but with an amount much higher than the agreed one (e.g., $2000). The scammer explains the check overpayment by being outside the country and by being his last check, explaining that he owes his child's nanny the overpayment (e.g., $1500) and asks gently Alice to make a money order or a wire transfer to the nanny. Consequently, Alice cashes the check and sends the overpayment to the scammer thinking she is doing it for the nanny. Legally, a person is responsible for the check he/she is depositing. Hence, it is Alice which will refund the check as well as the bank's fine and fees.

[3] In the remainder of this article, we use the term Providing-Bank to refer to the bank that provides the check and the term Cashing-Bank to refer to the bank that receives and pays the check.

[4] True story

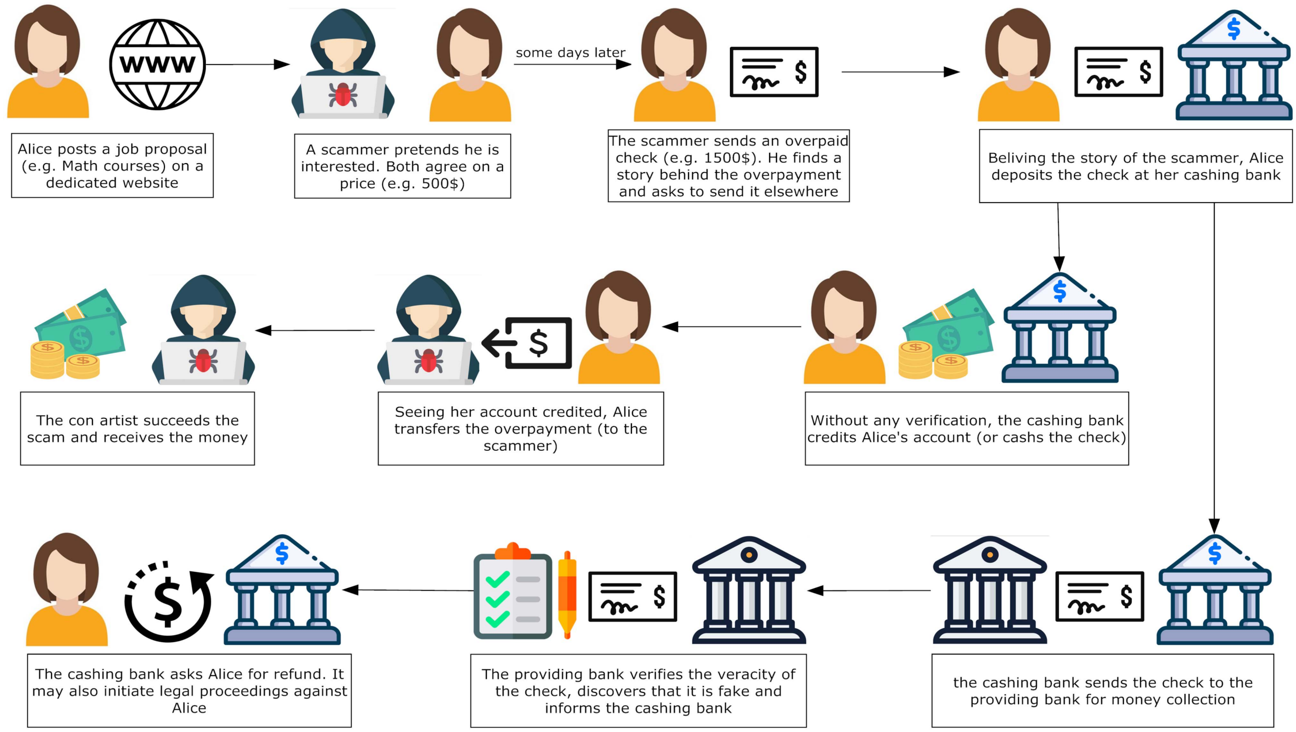

WWW
Alice posts a job proposal (e.g. Math courses) on a dedicated website
A scammer pretends he is interested. Both agree on a price (e.g. 500$)
some days later
$
The scammer sends an overpaid check (e.g. 1500$). He finds a story behind the overpayment and asks to send it elsewhere
$
$
Beliving the story of the scammer, Alice deposits the check at her cashing bank
$
Without any verification, the cashing bank credits Alice's account (or cashs the check)
$
Seeing her account credited, Alice transfers the overpayment (to the scammer)
The con artist succeeds the scam and receives the money
$
$
the cashing bank sends the check to the providing bank for money collection
$
The providing bank verifies the veracity of the check, discovers that it is fake and informs the cashing bank
$
$
The cashing bank asks Alice for refund. It may also initiate legal proceedings against Alice

Fig. 1. Fake check scam scenario.

This fraud is possible because of the check payment protocol used in the banking ecosystem. Indeed, the deposited check goes through several steps: 1) without verification of the check, the Cashing-Bank credits the account of the customer that deposits the check within one working day from the date of deposit. In some countries, this credit is provided only if the amount of the check does not exceed a known threshold. For example, in France, this threshold is € 3000. If the check exceeds this amount, a portion of the check's amount is credited while waiting for the next step to be executed; 2) the Cashing-Bank sends the check to the Providing-Bank for money collection; and 3) if all goes well, the customer can receive the amount stated on the check. But if the check is unpaid, bounced, irregular, or fake, the Cashing-Bank will reissue the corresponding amount from the customer's account who also pays additional fees (according to the bank and the country policy, a fine or judicial proceedings can be considered) [1]. The *Float* is the amount of time it takes for money to move from one account to another. It ranges from 48 h until several weeks, depending on the banks involved and their mutual agreements, e.g., in the USA. The processing of the check through the Federal Reserve System may take up to three or more business days. When the check is presented, the countdown for the midnight rule begins [2], [12]. A check deposited on a Friday may not be returned until the following Wednesday or even later, which is in compliance with the Uniform Commercial Code [2]. In the case of a fake check scam, it is only when the amount of the transaction is claimed from the Providing-Bank that the fraud is discovered, which gives the scammer all the time to do the fraud.

Fake checks drive many types of scams such as those involving phony prize wins, fake jobs, mystery shoppers, online classified ad sales, payment for a sold item, and many others. We describe some of the most common ones in the following [2], [13].

1) *Mystery shopping scam:* Con artists lure victims by sending spams or posting ads for mystery shoppers in classified job advertisements.[5] When victims respond to the ads, they are led to believe that they have been hired as mystery shoppers to evaluate the services of money transfer companies (e.g., *MoneyGram*). Victims are then sent checks that appear to be from legitimate companies and instructed to deposit the checks in their bank accounts and then withdraw most of the money and wire it to someone else (often a purported fellow mystery shopper). Victims are told to keep several hundred dollars of the money as payment. When the checks are later discovered to be phony, the banks reverse the deposit, and the victims are left liable for the money withdrawn, usually several thousand dollars.[6] This occurs in addition to a potential fee and judicial proceedings. Another form of this fraud is a scammer who sends spam e-mails informing the potential victim that he/she has inherited a large amount of money but he/she cannot cash it by himself/herself because of family (or other) problems and hires the victim to cash it for him while keeping a compensation.
2) *Fake job scam:* As we have described earlier in this section, this scam typically starts with a victim responding to an online posting (spam message), or the victim may

[5] [Online]. Available: https://www.consumer.ftc.gov/articles/0053-mystery-shopper-scams

[6] [Online]. Available: Financial Industry Regulatory Authority: www.finra.org/

have posted information online, to seek a job. Either way, the victim eventually gets "hired" by the con artist and receives e-mails or phone calls with instructions. Similar to the mystery shopping scam, the victim then receives a legitimate looking check and is told to cash it, wire some portion of the proceeds to a third party, and keep the remainder as payment [1], [14].

3) *Unexpected check scam:* Typically, this fraud starts with a spam e-mail inviting victims to participate in a fake lottery or to play a simple online game. This event triggers the delivery of a "surprise" check to the victims' door. The scammers inform that a part of the prize must be used as fees [1].

   The checks are fake, but they look real especially considering that there is no physical protection on the check. They look so real that even bank tellers may be fooled. The companies whose names appear may be real, but someone has dummied up the checks without their knowledge. Moreover, for money savings, numerous people print their own checks.[7]

Fraud is a major problem in business [15], and bank fraud is extremely dangerous for the organizational development of a bank [16]. In particular, fake checks is one of the biggest challenges faced by financial institutions. The main reason is that technology has made it progressively easy for criminals to create realistic counterfeit and false checks. According to Chhabra *et al.* [17], "technological advancements enable criminal actors to perpetrate innovative frauds that are very difficult to detect. Since most banking systems accept scanned copies of checks for clearance, identifying erasable ink alterations and printed signatures on digital images can be very challenging."

To the best of our knowledge, there is no existing check authentication method that relies solely on information technology (IT) resources. Rose [2] provided generic hints on how machine learning can be applied to detect fake checks. They discussed concerns in the detection of physical errors on the check's shape/design (e.g., logo), which makes it unreliable if the adversary prints good-quality checks. Similarity, Kumar and Gupta [18] proposed an automated methodology for the forensic authentication of bank checks. To determine check authenticity, a support vector machine was used to verify the color and texture characteristics extracted from images of genuine and fake checks.

The Official Gazette of the U.S. Patent Office reported several advanced methods for effectively blocking the counterfeiters and preventing continuing check fraud. For example, U.S. Patent no. 3 829 133 [19] explains a type of check, which integrates a masked individual code recognized only to the authorized drawer of the check who has advance knowledge of the key, by which the individual code is determined. U.S. Patent no. 4 231 593 [20] defines a check having first and second coatings: the first one is electrically conductive, and the other is electrically nonconductive. The main advantage of this method is to avoid any tentative to alter checks. U.S. Patent no. 5 371 798 [21] defines a method of making a check by dividing the clear band of the check into two parallel portions: one portion printed with ferrous beaded ink and the other portion printed with nonferrous inks. The goal of this method is to distinguish the authentic checks.

The analysis of the aforementioned patents leads to the conclusion that although each method may be useful in detecting fake checks, they are outdated and represent only a physical security, which make them noneffective currently, where banks' customers can print their own checks. Therefore, there is a need to develop a fake check detection method that can meet the needed requirements and that can be easily integrated into existing bank equipment to be effective and, therefore, adoptable.

III. Proposed Approach

The main goal of our approach is to provide banks with a powerful mechanism that allows the instant authenticity verification of a given check and, hence, avoid the current *float* period of more than 48 h.

A scammer can create a fake check according to the following two methods:

1) by considering random information;
2) by considering real information of an already cashed check.

This article is an extension of our previous work [22], where we presented a method to detect fake checks. However, our previous work did not consider the second case, where the scammer uses real data to create a fake check. Thus, the detection approach cannot detect the scam, which makes it ineffective. To address this drawback in our previous approach, we propose an extension to it so that the extended approach could detect all possible check scams.

A. Background

Our approach relies mainly on 1) a blockchain and 2) Lagrange interpolating polynomial. In this section, we briefly describe these concepts.

Blockchain is not a new concept to banks. Indeed, numerous studies [23]–[26] have described the challenges and opportunities of implementing blockchain technology in the banking sector [e.g., central bank digital currency, payment clearing and settlement systems operated by central banks, assets transfer and ownership, audit trail, and regulatory compliance (regulation)]. Industry participants see an opportunity to apply blockchain to their products and services and develop coordinated solutions that could help overcome existing industry challenges by providing greater transparency and improving conduct [27]. Nonetheless, to the best of our knowledge, no works on the detection of fake checks have been proposed (blockchain-based or IT-based in general).

1) Blockchain: A blockchain is defined as a distributed database (ledger) that maintains a permanent and tamper-proof record of transactional data. A blockchain is completely decentralized by relying on a peer-to-peer network. More precisely, each node of the network maintains a copy of the ledger to

[7] [Online]. Available: https://www.thebalance.com/before-you-print-your-own-checks-315315

prevent a single point of failure. All copies are updated and validated simultaneously [10].

Blockchain technology was created to solve the double spending problem in cryptocurrency [28]. However, currently, numerous works explore blockchain applications in multiple use cases and use them as a secure way to create and manage a distributed database and maintain records for digital transactions of all types [29]–[35].

The blockchain ledger is composed of multiple blocks, and each block is composed of two parts. The first part represents the transactions or facts (that the database must store), which can be of any type such as monetary transactions, health data, system logs, traffic information, and so on. The second part is called the header and contains information about its block (e.g., timestamp, hash of its transaction, as well as the hash of the previous block). Thus, the set of existing blocks forms a chain of linked and ordered blocks. The longer the chain, the harder is to falsify it. Indeed, if a malicious user wants to modify or swap a transaction on a block, first, the user must modify all the following blocks because they are linked with their hashes. Second, the user must change the version of the blockchain that each participating node stores [10], which is very hard to achieve.

The core task of a blockchain network is to ensure that the trustless nodes in the network agree on a single tamper-proof record of transactions. Thus, to jointly address the problems of trust, anonymity, scalability, poor synchronization, and to prove the honest validation of blocks, consensus mechanisms are deployed [36]. There are numerous consensus algorithms that have been proposed in the past few years such as Byzantine fault tolerant (BFT) [37]–[39], state-machine-replication-based BFT [40]–[42], Dwork, Lynch, and Stockmeyer [39], View-stamped Replication [43], Paxos [44], [45], and many others.

In this article, we use *Namecoin*[8] blockchain to implement our approach. *Namecoin* uses the Nakamoto consensus. Nakamoto [28] proposed a permissionless consensus protocol based on a framework of cryptographic block-discovery racing game also known as proof of work (PoW). From a single node's perspective, the Nakamoto consensus protocol defines three major procedures [46].

1) *Chain validation* provides a Boolean judgment on whether a given chain of blocks has the valid structural properties. It checks if each block in the chain provides a valid PoW solution, and no conflict between transactions as well as the historical records exists.
2) *Chain comparison and extension* compares the length of a set of chains, which may be either received from peer nodes or locally proposed. It guarantees that an honest node only adopts the longest proposal among the candidates' views of the blockchain.
3) *PoW solution searching* defines a cryptographic puzzle-solving procedure in a computation-intensive manner, which is hard to compute but easy to verify. A PoW is requested for each block validation. The difficulty of the mathematical challenge can be adapted according to the time needed to validate a block and to the miners' computation power [10].

TABLE I
CHECKBOOK'S AUTHENTICATION INFORMATION DATA STRUCTURE

Field	Size (bytes)
Lagrange Polynomial	Variable
Hash(*Lagrange polynomial*‖*Full name*‖*Providing-Bank*‖*Account number*)	32
Signature(*Full name*‖*Full Address*‖*Providing-Bank*‖*Routing number*‖*Account number*‖*Lagrange polynomial*)	64

2) Lagrange Polynomial Interpolation: Lagrange polynomials allow us to interpolate a series of points by a polynomial, which passes exactly through these points. More thoroughly, given a set of points (x_j, y_j) with no two x_j values equal, the Lagrange polynomial is the polynomial of lowest degree that assumes for each value x_j the corresponding value y_j. Thus, the function coincides at each point [47], [48]. The following equation defines the Lagrange polynomial associated with these points:

$$L(X) = \sum_{j=0}^{n} y_j l_j(x), \ \ l_j(x) = \prod_{j=0, j\neq i}^{n} \frac{X - x_j}{x_i - x_j}. \qquad (1)$$

Equation (1) can also be written as follows [48]:

$$L(x) = \prod_{j=0}^{n} (x - x_j). \qquad (2)$$

B. System Functioning

Our approach helps the banks to share information about provided checks in order to verify their authenticity during the payment all without exposing any of the customer's personal data. The proposed scheme relies on a public blockchain. Also, we need to choose a hash algorithm as well as a signature algorithm. In this article, we consider using 1) *Namecoin* blockchain; 2) *SHA-256* as a hash algorithm; and 3) *Elliptic Curve Digital Signature Algorithm (ECDSA)*.

Our system uses two phases of the check's lifecycle: 1) the check provision and 2) the withdrawal operation. Our approach requires that each Providing-Bank owns a key pair with the public key certified by a trusted authority, i.e., each bank must own a certificate, accessible by any third party.

1) Check Provision Phase: When a bank creates a checkbook for a customer, it must share the information related to the customer and the checkbook through a public blockchain. More precisely, for each provided checkbook, the bank creates a data structure related to this checkbook called Checkbook's Authentication Information (CAI) and adds it to the public blockchain through a transaction. The CAI data structure is composed of three fields, as shown in Table I: 1) a Lagrange polynomial; 2) a hash; and 3) a cryptographic signature.

We assume that the bank's entity, which executes the check's provision task can provide basic protocol primitives in order to recover the information required to create the CAI structure and

[8] [Online]. Available: https://namecoin.org

send it to the blockchain. Algorithm 1 shows such an API, while Algorithm 2 describes the whole process.

Algorithm 1: Basic Operations Provided by the Bank's Entity, Which Executes the CAI Creation.

```
Function GETFIRST(Checkbook $chBook$) : Integer
    //returns the number of the first
    check in the checkbook
Function GETLAST(Checkbook $chBook$) :Integer
    //returns the number of the last
    check in the checkbook
Function GETCHECKNB(Check $ch$) : Integer
    //returns the check's number
Function GETNAME(CustomerProfile $customer$) :
    String //returns the customer's full
    name recorded by the Providing-Bank
Function GETADDRESS(CustomerProfile $customer$) :
    String //returns the customer's full
    address recorded by the
    Providing-Bank
Function GETACCOUNTNB(CustomerProfile $customer$)
    : Integer //returns the customer's
    account number recorded by the
    Providing-Bank
Function LAGRANGEINTERPOLATIONFCT(Integer
    $Coef1$, Integer $Coef2$) : [ ] Integer //returns the
    Lagrange polynomial created using the
    coefficients: $Coef1$ and $Coef2$
Function SENDTRANSACTION(Blockchain $bc$,
    DataStructure $CAI$) //send a blockchain
    transaction containing the CAI data
    structure
```

Lagrange polynomial field: The checkbooks provided by banks contain generally either 50 or 100 checks. Besides, the customer uses only one check for cashing at a time. Therefore, our scheme must be able to verify the authenticity of each check individually. In this context, considering a solution where each check is registered individually in the blockchain will be costly because it requires as many blockchain transactions as existing checks. To address this issue, we use Lagrange polynomials as an aggregation for all the checks that the checkbook owns. For example, we consider a checkbook that contains four checks having the following numbers: $E = \{2, 3, 4, 5\}$. Considering (2), the Lagrange polynomial built according to this set will be

$$\begin{aligned} L(x) &= (x-2)(x-3)(x-4)(x-5) \\ &= x^4 - 14x^3 + 71x^2 - 154x + 120. \end{aligned} \tag{3}$$

All the elements of the set E are roots of the computed polynomial $L(x)$ described by (3). Consequently, following this logic, our scheme must build a Lagrange polynomial for each provided checkbook using its check numbers. However, we need to build Lagrange polynomials of degree 100 (or 50), which takes time and consumes CPU resources and requires a large space on the CAI structure.[9] To optimize this step, especially when we know that the check numbers of a checkbook are always consecutive, we compute the Lagrange polynomial considering only the upper and lower bounds of the interval composed by the check numbers. The resulting polynomial[10] will be used in the verification phase by testing if the check number is in the interval composed of the two roots of the Lagrange polynomial. If we consider the last example of the set E, the Lagrange polynomial created according to its upper and lower bounds ([2,5]) is

$$\begin{aligned} L(x) &= (x-2)(x-5) \\ &= x^2 - 7x + 10. \end{aligned} \tag{4}$$

The two roots of $L(x)$ are 2 and 5. Thus, all the elements of E will be in the interval composed of the polynomial roots ([2,5]). Hence, if a Lagrange polynomial is built according to the numbers of the first and last checks of a checkbook, all the checks' numbers of the corresponding checkbook will fit into the interval built by the two roots of the polynomial.

The Lagrange polynomial field of the CAI is structured as follows[11]:

$$\begin{aligned} &\langle \text{Polynomial degree } (n), [x^n](L(x)), [x^{n-1}](L(x)), \\ &\ldots, [x](L(x)), [x^0](L(x))\rangle. \end{aligned} \tag{5}$$

For example, if we consider the polynomial described by (4), the Lagrange polynomial field of the CAI data structure is $\langle 2, 1, -7, 10\rangle$.

Most programing languages (e.g., C, C++, Java) require an integer to be stored in four bytes. In this context, the Lagrange polynomial field of the CAI will have a size of 16 bytes.[12] Other programming languages such as Python use more space to represent integers. Therefore, the size of the Lagrange polynomial field of the CAI will depend on the programming language used.

Hash field: The hash field contains a hash computed on the following fields:

1) *Lagrange polynomial:* the computed polynomial;
2) *Full Name:* the customer's full name;
3) *Providing-Bank:* the bank that provided the checkbook;
4) *Account number:* the customer's account number.

The check contains the data used to compute the hash. This hash field serves as the landmark to find the block containing the corresponding CAI on the blockchain.

Signature field: The signature is provided on the information shared by all the checks of the checkbook. Fig. 2 illustrates the various types of information:

1) *Full Name:* the customer's full name;
2) *Full Address:* the customer's address;
3) *Providing-Bank:* the bank that provided the checkbook;
4) *Routing number:* generally composed of nine digits. It identifies the location where the account was opened;

[9] Each coefficient must be stored as an integer. Considering the polynomial degree (100), this field will have at least 404 bytes).

[10] The resulting polynomial will always be of degree 2 because it is a Lagrange Interpolation using two coefficients (the upper and lower bounds).

[11] $[x^n](L(x))$ indicates the coefficient corresponding to x^n, thus the first coefficient of the polynomial.

[12] (3 integer coefficients × 4 bytes per integer) + 4 bytes for the integer representing the polynomial's degree.

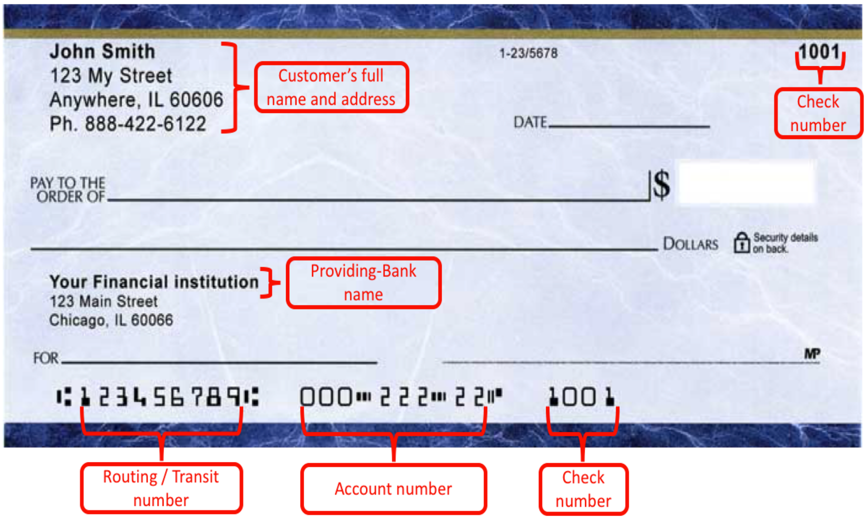


Fig. 2. Check example.

5) *Account number:* the customer's account number;

6) *Lagrange polynomial:* the computed polynomial.

The signature is performed using the private key corresponding to the certificate of the Providing-Bank.

The data structure is stored in a public blockchain, which means that any third party can access these data. However, only a hash and a signature are stored. Since the hash function behind the signature is not reversible, it is impossible to recover the customer's data (data that were hashed/signed).

2) Revocation/Usage of a Check: When a deposited check is cashed, the latter must be tagged as "no more valid" because it has already been used. In the same context, if the bank wishes to revoke a check or a checkbook following some reason such as check theft, the corresponding check must be tagged as "no more valid" because it has been revoked.

In contrast to the checkbook logic, where all the checks have consecutive numbers and are provided at the same time, the usage of checks for payment is completely random. Indeed, some customers will use their checks regularly, while others will spend numerous months or years to exhaust a checkbook. Thus, it is mandatory to keep track of each used check individually in order to avoid it being used again or, more thoroughly, in order to avoid the valid data of a used check from being used again for one or more fake checks.

Consequently, for each cashed check, the Providing-Bank must share the information related to the customer and the check through the public blockchain. More precisely, for each cashed/revoked check, the bank creates a data structure called check validity information (CVI) and adds it to the public blockchain through a transaction. The CVI data structure is composed of two fields, as shown in Table II: 1) a hash and 2) a cryptographic signature.

Since multiple checks belonging to different checkbooks can have the same number, the hash field serves as the landmark (feature) to find the block containing the corresponding CVI on the blockchain. It contains a hash computed on the following fields: 1) *Check number:* the cashed/revoked check number; 2) *Full Name*; 3) *Providing-Bank*; and 4) *Account number*.

The signature field ensures that the CVI was created by the Providing-Bank, since it can be verified using the Providing-Bank's public key. The signature of the CVI covers the same fields considered by the CAI's signature. In addition, it also covers the *Check number* field. Algorithm 3 describes the CVI creation and sharing operation.[13]

3) Withdrawal Operation Phase: When a customer deposits a check in a Cashing-Bank, a few verifications are done before triggering any operation in order to ensure the payment. These verification operations can be achieved 1) by the human agent who handles the check through a dedicated human–machine interface available on the system or 2) by the ATM machine since it is designed to work through text/image recognition. The program that runs the described verifications can be deployed on 1) the bank's terminals (computers and ATM machines) through a software update or 2) on a server managed by the bank, and thus, the bank's terminals will simply be used as input interfaces. Hence, the bank infrastructure must host at least one up-to-date copy of the used blockchain. Moreover, we assume that the bank's entity, which executes the verification scheme, can provide basic protocol primitives in order to recover any information required from the blockchain and from the deposited check. Algorithm 4 presents the API, Algorithm 5 describes the whole verification process, and Fig. 3 describes a check's lifecycle when our approach is used (we did not consider the step of the payment between banks).

Our proposed scheme must ensure that the deposited check is authentic by relying on the CAI data structure. The scheme must also ensure that it has not been revoked or cashed by relying on the CVI data structure. Considering that browsing the blockchain is a costly operation regarding the execution time, for optimization purposes, when browsing the blockchain, for each treated block, we simultaneously verify if it contains the CAI or the CVI of the deposited check.

[13] Due to space limitations, we do not redefine the parameters used by Algorithm 2.

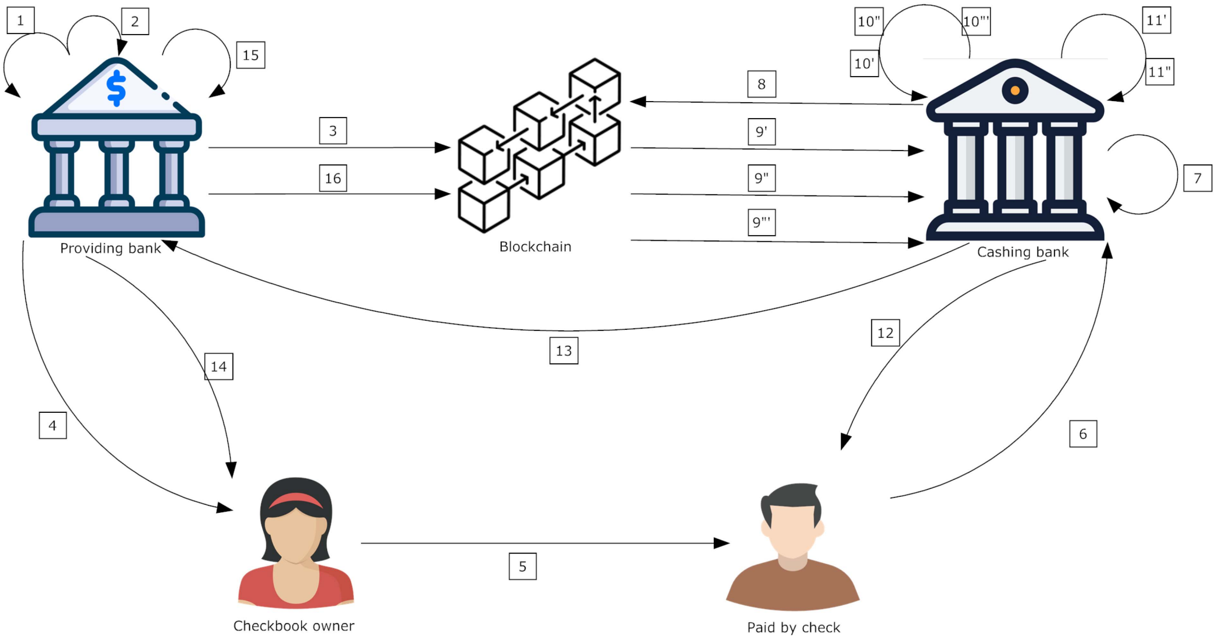


Fig. 3. Check's lifecycle when our detection approach is applied: (1) creates the checkbook, (2) creates checkbook's CAI, (3) stores CAI, (4) provides the checkbook, (5) pays with a check, (6) deposits the check, (7) creates the hash field relying on the check's data, (8) searches for CVI or CAI, (9′) CVI found, (10′) verifies CVI signature, (11′) if CVI is valid, then the payment procedure stops, else back to (8), (9″) CAI found, (10″) verifies CAI signature and check number, (11″) if CAI is valid, then go to (12), else go back to (8), (9‴) No CVI and no CAI found, (10‴) payment procedure stops, (12) pays the check, (13) sends the cashed check, (14) retrieves the amount of the paid check, (15) creates CVI for the cashed check, and (16) stores CVI.

When we add an information to a blockchain, the information is always added to the end of the blockchain (chained to the last block). Accordingly, the blockchain's block that contains the CAI of a checkbook is always situated before the block that contains the CVI of a check belonging to the respective checkbook. In other words, if a check has been already cashed/revoked, the CVI position is always after the CAI. Hence, knowing that the blockchain browsing goes from the end toward the beginning, while browsing the blockchain, if our verification scheme finds the CAI first, it means that the check is authentic and that has not been used or revoked. But if it finds the CVI first, it means that the check is no longer valid because it was used or revoked.

The verification operations are provided as follows: *first*, the information corresponding to the customer's full name, the Providing-Bank's name, and the customer's account number, which are available on the check, is concatenated and then hashed. We call this *hashCAI*. *Second*, the information corresponding to the check's number, the customer's full name, the Providing-Bank's name, and the customer's account number is concatenated and then hashed. We call this *hashCVI*. *Third*, our scheme browses the blockchain to find the block containing the CAI whose hash field is equivalent to the computed *hashCAI* or the block containing the CVI whose hash field is equivalent to the computed *hashCVI*. As described above, for optimization purposes, the CAI and CVI search are done simultaneously for each searched block of the blockchain.

1) If no block is found (no CAI nor a CVI), it means that no transaction was performed by the check's Providing-Bank to record the checkbook; thus, the check is fake and the payment operation aborts.
2) If the CVI is found first, the system must authenticate it beforehand. Indeed, since it is a public blockchain, the system must ensure that it is the Providing-Bank that provided that transaction and not another entity. In this case, our verification scheme uses the data (check's number, customer's full name, customer's full address, Providing-Bank, check's routing number, and customer's account number) that exists in the deposited check to verify the CVI's signature relying on the bank's public key (available on the bank's published certificate). If the CVI's signature is verified, then it indicates that the check is not fake, but that it has already been cashed or revoked. If the CVI's signature verification fails, then it means that it is not the Providing-Bank that added this CVI structure and the CAI/CVI research process must continue.
3) If the CAI is found first, the system must authenticate it beforehand (must ensure that the CAI was created and added to the blockchain by the Providing-Bank). To achieve this, the verification scheme concatenates the data (customer's full name, customer's full address, Providing-Bank, check's routing number, and customer's account number) obtained from the check with the Lagrange polynomial, which is in the retrieved CAI structure and uses this set of data (check information + the polynomial) to verify the CAI's signature using the bank's public key. If the signature verification fails, the payment operation aborts.

Algorithm 2: CAI Creation and Sharing Through the Blockchain.

```
Declaration:
Const $bankName$: String//Providing-Bank's name
Const $routingNb$: Integer//routing/transit number used by the bank
$chBook$: Checkbook//the newly created checkbook
$customerX$: CustomerProfile//the profile of the customer owner of the newly created checkbook
$bc$: Blockchain//the used blockchain
SendTransaction$bc$, CAI_Creation$chBook$, $customerX$//send a blockchain transaction containing the CAI data structure
functionCAI_Creation$chBook$, $customerX$
    $lPolynomial \leftarrow$ LagrangeInterpolationFct(getfirst$chBook$, GETLAST$chBook$;
    $hash \leftarrow$ SHA-256(Concatenate(getName($customerX$), $bankName$, getAccountNb($customerX$)))//applies SHA-256 hash algorithm on the set of defined parameters
    $signature \leftarrow$ ECDSA(Concatenate(getName($customerX$), getAddress$customerX$, $bankName$, $routingNb$, getAccountNb$customerX$), $lPolynomial$), $bankPrivateKey$)//applies ECDSA signature algorithm on the set of defined parameters using the bank's private key
    $CAI \leftarrow$ makeArray($lPolynomial$, $hash$, $signature$)//creates the CAI data structure
    return $CAI$
end function
```

TABLE II
CVI Data Structure

Field	Size (bytes)
Hash(*Check number*‖*Full name*‖*Providing-Bank*‖*Account number*)	32
Signature(*Check number*‖*Full name*‖*Full Address*‖*Providing-Bank*‖*Routing number*‖*Account number*)	64

Fourth, once the signature has been verified, considering that each check has been verified individually, the verification scheme resolves the Lagrange polynomial of the CAI. Then, the scheme verifies if the number of the deposited check fits into the interval represented by the polynomial's roots (as described in Section III-B1 a). If the verification succeeds, the payment operation is initiated.

Algorithm 3: CVI Creation and Sharing Through the Blockchain.

```
Declaration:
$check$: Check//the cashed/revoked check
SendTransaction($bc$, CVI_CREATION($check$, $customerX$))//send a blockchain transaction containing the CVI data structure
functionCVI_Creation($check$, $customerX$)
    $hash \leftarrow$ SHA-256(Concatenate(getCheckNb(check), GETNAME$customerX$, $bankName$, GETACCOUNTNB$customerX$//applies SHA-256 hash algorithm on the set of defined parameters
    $signature \leftarrow$ ECDSA(Concatenate(getCheckNb(check), getName$customerX$), getAddress$customerX$, $bankName$, $routingNb$, getAccountNb$customerX$)), $bankPrivateKey$)//applies ECDSA signature algorithm on the set of defined parameters using the bank's private key
    $CVI \leftarrow$ makeArray$hash$, $signature$//creates the CVI data structure
    return $CVI$
end function
```

IV. Implementation, Evaluation, and Discussion

A. Implementation

To implement our approach, we opted for *Namecoin* blockchain [49]. *Namecoin* is a fork of Bitcoin which aims to provide a decentralized Domain Name Service. Indeed, it implements the top level domain. bit, which is independent of the Internet Corporation for Assigned Names and Numbers.[14] Table III presents the main features of *Namecoin*.

We opted for *Namecoin* for three main reasons.

1) It allows data storage in the form of key/value pair. Users have the possibility to store values up to 520 bytes in size, which is more than sufficient to host the CAI/CVI structures.
2) The daily volume of transactions is relatively weak, which facilitates the data search in the blockchain.
3) Transactions fees are very low cost—the average transaction fee is about \$0.00032 (accessed on January 19, 2021).[15]

For the hash function, we use *SHA-256* because it represents one of the recommended hash algorithms by the *National Institute of Standards and Technology*[16] [50].

[14] [Online]. Available: https://www.icann.org

[15] [Online]. Available: https://bitinfocharts.com/namecoin/

[16] [Online]. Available: https://csrc.nist.gov/Projects/Hash-Functions

Algorithm 4: Basic Operations Provided by the Bank's Entity, Which Executes the Check's Verification Process.

```
Function GETCHECKNB(Check $ch$) : Integer//returns the check's number
Function GETNAME(check $ch$) : String//returns the customer's full name from the deposited check
Function GETADDRESS(Check $ch$) : String//returns the customer's full address from the deposited check
Function GETACCOUNTNB(Check $ch$) : Integer//returns the customer's account number from the deposited check
Function GETROUTINGNBCheck $ch$) : Integer//returns the bank's routing number from the deposited check
Function GETBANKNAME(Check $ch$ : String//returns the bank's name from the deposited check
Function GETLAGRANGEPOLYNOMIAL(DataStructure $CAI$) : [ ] Integer//returns the Lagrange polynomial from the CAI
Function RESOLVEDEG2EQUATION[ ] Integer $polynomial$ : [ ] Integer//resolves an equation of the second degree and returns an array with the found solutions sorted in ascending order
Function ISEXISTINGTRANSACTION(Blockchain $bc$, String $hashCAI$, String $hashCVI$) : Boolean//verifies if the blockchain at least have a block which contains a transaction which in turn contains the needed CAI or CVI
Function GETCAIORCVI(Blockchain $bc$, String $hashCAI$, String $hashCVI$) : DataStructure//Browses the blockchain and verifies simultaneously if the browsed block contains the needed CAI or the CVI. It returns the first structure found (CAI or CVI)
Function ERROR(String $error Message$)//returns and error message
```

For the signature algorithm, we opted for *ECDSA* [51], [52]. *ECDSA* has several advantages over traditional signature algorithms such as *Rivest Shamir Adleman* especially concerning key sizes and signature time [53]–[55].

B. Evaluation Framework and Scenarios

Regarding the evaluation framework, we used *Multichain*[17] to simulate the *Namecoin* blockchain. *Multichain* is an open-source blockchain platform, which helps in the design and deployment of blockchain applications. It is fully configurable according to the user's needs, and it can, therefore, be set up to reproduce the same functions as any other blockchain. We used this feature to simulate a *Namecoin* blockchain. Currently,[18] the *Namecoin* blockchain includes 542 258 blocks. To simulate the *Namecoin* blockchain, for our experiments, we used a blockchain with 500 000 blocks.

The program that shares the new issued checkbook's data (applicable to the Providing-Bank) and the authentication verification program (applicable to the Cashing-Bank) were developed using *Python* language, version *2.7*. For cryptographic operations, we used *OpenSSL* library, version *1.1.1a*.

To the best of our knowledge, there is no other check authentication method that relies solely on IT resources. *Thus, we cannot compare the efficiency of our method with another existing method.* Moreover, since our approach verifies if a check's record already exists in the blockchain, there are no false positives or false negatives.

Knowing that searching a block in the blockchain is made in a sequential method (block by block), the authentication time of a check depends on the position of the block, including its corresponding CAI (or CVI) in the blockchain. Accordingly, we were interested in measuring this time for different cases: 1) the needed block is at the beginning of the blockchain; 2) the needed block is in the middle of the blockchain; 3) the needed block is at the end of the blockchain; and 4) the check's record does not exist in the blockchain (fake check). For each scenario, we executed 100 tests, where we measured the time needed to find the block. Each test is applied on a different block. More specifically, for the first scenario, the needed blocks were between blocks 1 and 1000. For the second scenario, the needed blocks were between blocks 225 000 and 226 000. For the third scenario, the needed blocks were between blocks 499 000 and 500 000.

We are aware that the search time depends mainly on the processing power of the machine used as well as the programing language (e.g., C is faster than *Python*) used. However, we wanted to compare the different discussed cases using the same language and host. We performed all tests using the following testbed: the host system has an Intel(R) Quad-Core i7 CPU 3.80 GHz with 16 GB of RAM. It executes an up-to-date version of the *KALI Linux 4.12.0* distribution.

C. Evaluation Results

1) Adversary Model: Any probabilistic polynomial time adversary $\mathcal{A}$ can create a fake check with a high physical quality. $\mathcal{A}$ can create the check using random data or using real data of other real checks. Moreover, $\mathcal{A}$ can get access to the transactions' data stored in the blockchain because it is a public blockchain. Finally, since we use a public blockchain, $\mathcal{A}$ can also send transactions to the blockchain about false checks.

2) Formal Validation: To evaluate the safety and robustness of our proposed approach, we have provided a formal validation. We used *Scyther* [56], a tool for the automatic verification of security protocols. In the latter, a security protocol is defined as

[17] [Online]. Available: https://www.multichain.com

[18] January 19, 2021.

Algorithm 5: Check Verification Process.

```
Declaration:
$ch$: Check // the deposited check
$bc$: Blockchain // the used blockchain
procedure CheckVerification($ch$, $bc$)
    $hashCAI \leftarrow$ SHA-256(Concatenate(getName($ch$), getBankName($ch$), getAccountNb($ch$)));
    $hashCVI \leftarrow$ SHA-256(Concatenate(getCheckNb($ch$),getName($ch$), getBankName($ch$), getAccountNb($ch$)));
    if (IsExistingTransaction($bc$, $hashCAI$, $hashCVI$)) then
        $CI \leftarrow$ getCAIorCVI($bc$, $hashCAI$, $hashCVI$)
        if $CI.hash == hashCVI$ // the CVI found first
        then
            $signatureData \leftarrow$ Concatenate(getCheckNb($ch$),getName($ch$) getAddress($ch$), getBankName($ch$),
            getRoutingNb($ch$), getAccountNb($ch$));
            if ECDSA_Verify($signatureData$, $BankPublicKey$) then
                // applies ECDSA signature verification algorithm on the set of defined parameters using
                the bank's public key
                Error("Check has already been cashed or revoked ")
        else
            if $CI.hash == hashCAI$ // the CAI found first
            then
                $signatureData \leftarrow$ Concatenate(getName($ch$) getAddress($ch$), getBankName($ch$), getRoutingNb($ch$),
                getAccountNb($ch$), getLagrangePlynomial($CAI$));
                if ECDSA_Verify($signatureData$, $BankPublicKey$) then
                    $polynomialRoots \leftarrow$ resolveDeg2Equation(getLagrangePlynomial($CAI$));
                    if (getCheckNb($ch$) $\geq polynomialSolutions[0]$ AND getCheckNb($ch$) $\leq polynomialSolutions[1]$) then
                        CheckPaymentOperation($ch$) // triggers the check payment operation
                    else
                        Error("Failure in the check's authentication ")
                else
                    Error("Failure in the bank's authentication")

    else
        Error("Check does not exist") // no CAI or CVI was found
end procedure
```

TABLE III
NAMECOIN FEATURES (JANUARY 19, 2021)

Data field	Feature
Type	Public blockchain
Feature	Fork of Bitcoin
Average transaction fee	0.00032 $
Block time	9 min 40 sec
Transaction avg /h	21
Blockchain dimension	6.34 GB

an interaction among different roles. Each role is played by an agent and described by a sequence of events (send, receive, and so on).

The *Scyther* code relies on three roles: the Providing-Bank, the blockchain, and the Cashing-Bank. To comply with the *Scyther* operation mode, we model our approach as follows: 1) the Providing-Bank sends the CAI to the blockchain; and 2) the Cashing-Bank receives the CAI from the blockchain. The check is physically sent from the Cashing-Bank to the Providing-Bank for money collection. Thus, the information on the check is accessible only by these two banks. In our formal validation, we model this step as a secure exchange using symmetric cryptography between the two banks.

Listing 1 describes the *Scyther* code of the Providing-Bank's role. The claim event types are the goals of the formal validation. For the authentication of the Providing-Bank by the blockchain, we used three authentication claim types, which are *"Alive," "Weakagree," and "Niagree."* To explain these claims, we assume that A is the initiator and B is the responder.

1) *Alive claim:* We consider that a protocol guarantees to A aliveness of B if whenever A completes a run of the protocol, apparently with B, then the latter has previously been running the protocol [57].
2) *Weakagree:* We consider that a protocol guarantees to A weak agreement with B if whenever A completes a run of the protocol, apparently with B, then the latter has previously been running the protocol, apparently with A [57].
3) *Niagree:* We consider that a protocol guarantees to A noninjective agreement with B on a set of data items (variables) if whenever A completes a run of the protocol, apparently with B, then the latter has previously been running the protocol, apparently with A, and B was acting as the responder in its run, and the two agents agreed on the data values corresponding to all the data items [57].

Listing 2 depicts the *Scyther* code of the blockchain's role. We define the same three authentication claims (Alive, Weakagree, and Niagree). Fig. 4 shows a screenshot of the execution of our formal validation code, where only the authentication claims were activated. In the screenshot presented, the first, second, and third columns represent the protocol name, the role concerned

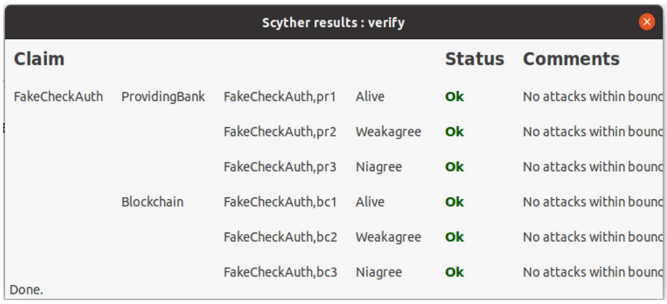
Listing 1: Scyther code of the Providing-Bank role.

```
usertype String;
usertype Polynomial;

protocol FakeCheckAuth(ProvidingBank, Blockchain,
  CashingBank) {

const bankName: String;
const routingNum: String;

hashfunction sha256;
function Concatenate;

role ProvidingBank {
fresh lPolynomial: Polynomial;
const costumerName;
const costumerAccountNB;
const costumerAddress;

macro h = sha256(lPolynomial, bankName,
  costumerName, costumerAccountNB);
macro signedDataHash = sha256(lPolynomial,
  bankName, costumerName, costumerAccountNB,
  routingNum, costumerAddress);
macro ecdsa = {signedDataHash}sk(ProvidingBank);

send_1(ProvidingBank, Blockchain, (lPolynomial, h,
  ecdsa));

claim_pb1(ProvidingBank, Alive);
claim_pb2(ProvidingBank, Weakagree);
claim_pb3(ProvidingBank, Niagree);

send_3(ProvidingBank, CashingBank,
  {ProvidingBank,(costumerName, costumerAddress,
  costumerAccountNB)}k
  (ProvidingBank,CashingBank));

claim_pb4(ProvidingBank, Secret, costumerName);
claim_pb5(ProvidingBank, Secret, costumerAddress);
claim_pb6(ProvidingBank, Secret,
  costumerAccountNB);
}
```

Listing 2: Scyther code of the blockchain role.

```
role Blockchain {
// Parameters declaration{\ldots}..

recv_1(ProvidingBank, Blockchain, (lPolynomial, h,
  ecdsa));
send_2(Blockchain, CashingBank, (lPolynomial, h,
  ecdsa));

claim_bc1(Blockchain, Alive);
claim_bc1(Blockchain, Weakagree);
claim_bc3(Blockchain, Niagree);
}
```

(Providing-Bank and blockchain), and a unique identifier of the claim, respectively. The fourth column represents the claim type with the parameters. The two last columns (status and comments) show the result of the verification process (Fail or Ok) and a short description. The “No attack within bounds“ should be interpreted as: *“Scyther did not find any attacks by reaching the bound”* [57]. As we can see, the validation proves that the tested part of our protocol is safe and secure.

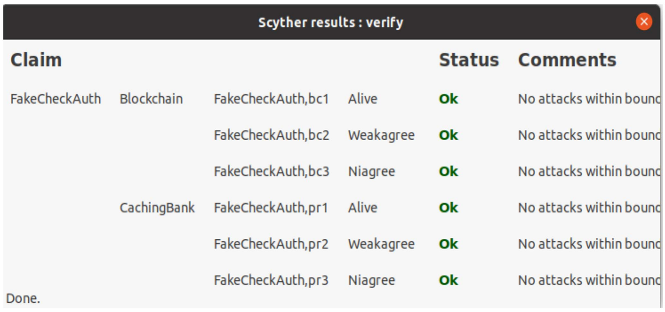


Scyther results : verify

Claim				Status	Comments
FakeCheckAuth	ProvidingBank	FakeCheckAuth,pr1	Alive	Ok	No attacks within bounc
		FakeCheckAuth,pr2	Weakagree	Ok	No attacks within bounc
		FakeCheckAuth,pr3	Niagree	Ok	No attacks within bounc
	Blockchain	FakeCheckAuth,bc1	Alive	Ok	No attacks within bounc
		FakeCheckAuth,bc2	Weakagree	Ok	No attacks within bounc
		FakeCheckAuth,bc3	Niagree	Ok	No attacks within bounc

Done.

Fig. 4. Formal validation results of the authentication claims between the Providing-Bank and the blockchain.



Scyther results : verify

Claim				Status	Comments
FakeCheckAuth	Blockchain	FakeCheckAuth,bc1	Alive	Ok	No attacks within bounc
		FakeCheckAuth,bc2	Weakagree	Ok	No attacks within bounc
		FakeCheckAuth,bc3	Niagree	Ok	No attacks within bounc
	CachingBank	FakeCheckAuth,pr1	Alive	Ok	No attacks within bounc
		FakeCheckAuth,pr2	Weakagree	Ok	No attacks within bounc
		FakeCheckAuth,pr3	Niagree	Ok	No attacks within bounc

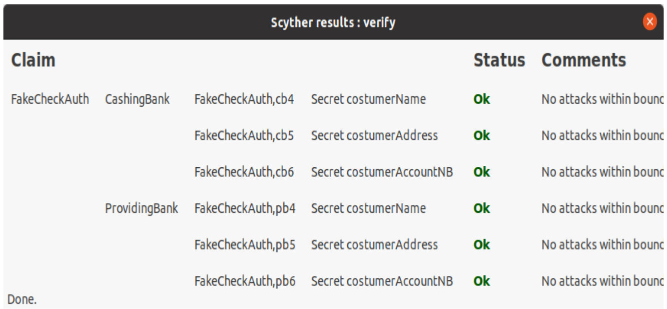
Done.

Fig. 5. Formal validation results of the authentication claims between the Cashing-Bank and the blockchain.

Listing 3: Scyther code of the Cashing-Bank role.

```
role CashingBank {
// Parameters declaration.....

recv_2(Blockchain, CashingBank, (lPolynomial, h,
  ecdsa));

claim_cb1(CashingBank, Alive);
claim_cb2(CashingBank, Weakagree);
claim_cb3(CashingBank, Niagree);

recv_3(ProvidingBank, CashingBank,
  {ProvidingBank,(costumerName, costumerAddress,
  costumerAccountNB)}k
  (ProvidingBank,CashingBank));

claim_cb4(CashingBank, Secret, costumerName);
claim_cb5(CashingBank, Secret, costumerAddress);
claim_cb6(CashingBank, Secret, costumerAccountNB);
}
}
```

Listing 3 describes the *Scyther* code of the Cashing-Bank’s role. First, we define the claims of the authentication with the blockchain. Fig. 5 shows a screenshot of the execution of the formal validation code, where we verify the authentication claims between the blockchain and the Providing-Bank. The results highlight the security and robustness of this phase.



Claim				Status	Comments
FakeCheckAuth	CashingBank	FakeCheckAuth,cb4	Secret costumerName	Ok	No attacks within bounc
		FakeCheckAuth,cb5	Secret costumerAddress	Ok	No attacks within bounc
		FakeCheckAuth,cb6	Secret costumerAccountNB	Ok	No attacks within bounc
	ProvidingBank	FakeCheckAuth,pb4	Secret costumerName	Ok	No attacks within bounc
		FakeCheckAuth,pb5	Secret costumerAddress	Ok	No attacks within bounc
		FakeCheckAuth,pb6	Secret costumerAccountNB	Ok	No attacks within bounc

Done.

Fig. 6. Formal validation results of the secrecy claims between the Cashing-Bank and the Providing-Bank.

Moreover, Listings 1 and 3 define the interaction of the check sent from the Cashing-Bank to the Providing-Bank. We define the Secret claim on the client's personal data that the check contains. Fig. 6 shows a screenshot of the execution of the formal validation code, where we verify the secrecy claims between the Cashing-Bank and the Providing-Bank. The results highlight the security and robustness of this phase.

3) Security and Performance Requirement Evaluation: In this section, we focus on the evaluation of the security features, performance features, and the design challenges discussed earlier (see Section I) of our proposed check authentication approach.

a) Data sharing between banks: For each checkbook, a CAI is shared. The CAI structure protects the users' data from exposure because it only includes a hash and a cryptographic signature, which are nonreversible. The Cashing-Bank is aware of the check's owner data through the deposited check. However, this is not unique to our approach because it is also the case with existing bank's protocols today.

b) Management of the sharing mechanism: Our approach relies on a public blockchain, which is fully autonomous. The bank does not need to manage any additional infrastructure. Furthermore, sending transactions or reading them from a blockchain represents a process that can be easily integrated and handled.

c) Nonmodification of existing protocols: Our approach does not require any modification of the existing banking protocols such as modifying the check's format, adding some physical security feature on the checks or the ATMs, or modifying the communication protocol between the different banks. It only requires an additional verification before executing the usual payment protocol and an additional action after providing a new checkbook or cashing a check.

d) Scalability: Our system relies on a public blockchain, which, in turn, relies on a peer-to-peer network. It is known that peer-to-peer networks are one of the best solutions to achieve high scalability [58]. Furthermore, numerous studies [59]–[62] have demonstrated the scalability of blockchains. Other works [63]–[65] focused on Bitcoin and showed its highly scalability. In our article, we used Namecoin, a Bitcoin fork. Hence, it inherits its high scalability.

e) Availability: The totally decentralized architecture of blockchains makes them robust against denial-of-service/distributed-denial-of-service attacks. Indeed, services are duplicated and distributed over different network nodes. That is to say, even if an attacker manages to block a node, it cannot block all the other nodes.

f) Authentication: Our approach provides an authentication of the deposited check after browsing a blockchain to find the check's record. This removes the possibility of false negatives and false positives from our scheme and makes it highly reliable. It allows the authentication of a legitimate check and the detection of a fake one. Moreover, even if the adversary stores transactions related to a fake check, the authentication process is fully reliable because it only authenticates the checks that are signed using the Providing-Bank's private key. Finally, if the adversary uses a fake check that contains the information of another real check, the detection system will detect it, thanks to the CVI structure that is signed by the Providing-Bank's private key. However, if the check was not used by the legitimate user yet, then the detection system authenticates it as a good check, and the scammer can use it for one time. But this case have a low probability to happen.

4) Financial Cost: Regarding the financial cost, we believe that each security service provided needs a cost, as long as it remains lower than the potential damages (a fake check scam generally costs several hundreds of dollars of damages). In our approach, for each created checkbook, a blockchain transaction is needed. Then, one transaction for each deposited (or revoked) check. For example, if we consider the example of a checkbook that has 100 checks, when all these checks are used, the checkbook costs 101 transactions. The transaction cost depends on the blockchain used. However, this cost remains negligible compared to the potential damages. We recommended the use of *Namecoin* whose transaction's fees are around \$0.00077. Thus, if we consider the last example, one checkbook of 100 checks costs $101 \times \$0.00077 = \$0.077$, which is a negligible fee. Moreover, the time needed to consume all the checks of a checkbook in order to reach this fee is highly variable according to the consumers and can vary from a few days to several years.

We are aware that the evolution of cryptocurrency rate represents an issue. However, according to studies such as [66] and [67], the evolution of the cryptocurrencies rates will become more stable over time [10]. Finally, the transaction fees can be added to the account maintenance cost each time the customer asks for a new checkbook.

5) Numerical Results: In this section, we present the numerical results related to the time consumption of our approach. As described in Section IV-B, we were interested in measuring the time taken for the check's authentication process described by Algorithm 5 (including all the algorithm's steps), for different cases of the needed block's position. Since we are interested in measuring the time needed for our approach to provide a response regarding the needed block position on the blockchain, we did not consider the case for a revoked bank certificate. Fig. 7 shows the average and standard deviation over the 100 tests for each scenario.

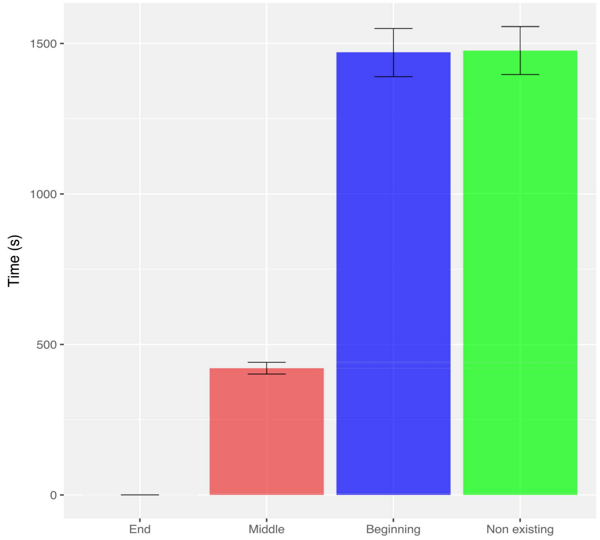


Fig. 7. Execution time of the check's authentication process according to the position in the blockchain of the block that owns the needed CAI (*Namecoin* implementation).

Regarding the scenario where the needed CAI is in a block located at the end of the blockchain (between the positions [499 000, 500 000]), the average authentication time is 0.30 s with a standard deviation of 0.15 s. This time was expected considering that the search in a blockchain starts with the last block. For the second scenario, where the needed block is in the middle of the blockchain (between the positions [225 000, 226 000]), the average authentication time is 420.84 s with a standard deviation of 20.03 s. We note the costly nature of the blockchain search operation, especially when our approach does not look for transactions' blocks' identifiers (ID), but browse the data of each transaction of each block. This cost is more important for the next scenario, where the needed block is at the beginning of the blockchain (between the positions [1, 1000]), since the average authentication time is 1469.68 s with a standard deviation of 80.40 s. Finally, for the scenario where the needed CAI does not exist in the blockchain, the time it takes to obtain a response is 1476.21 s, which is almost exactly the same time as in the last scenario, with a similar standard deviation of 75.50 s. Nonetheless, even if the execution of the check's authentication process can spend a few minutes in some cases, it remains very far from the current *float* time of more than 48 h. Furthermore, it protects the banks and especially the customers from being scammed.

In this article, we have developed a proof of concept and evaluated it on a simple machine with a high-level interpreted language (*Python*) known to have execution times higher than many other programming languages such as C and C++. A bank is likely to have more powerful machines with a better implementation (for instance, we did not apply any advanced or optimized block searching method).

V. Implementation on a Private Blockchain

The design of our approach requires a public (permissionless) blockchain. In this section, we analyze the impact of a private (permissioned) blockchain adoption. Even if our approach detects the fake checks before they are cashed and in a record time in comparison to the *float* time of at least 48 h, in some cases, the response is still being obtained in few minutes, according to the needed block position. Knowing that the search operation in the permissioned blockchain *Hyperledger Fabric* is optimized, we analyze its impact on our approach. Later on, we discuss how this implementation affects the security and performance requirements discussed above.

Fabric is a modular and extensible open-source system for deploying and operating permissioned blockchains, and it is one of the *Hyperledger* projects hosted by the Linux Foundation[19] [68]. It has a highly modular and configurable architecture, enabling innovation, versatility, and optimization [69].

In a public or permissionless blockchain, peers make part of the network anonymously. Private or permissioned blockchains, on the other hand, run a blockchain among a set of known identified participants. Hence, this type of blockchain provides a way to secure the interactions among a group of entities that have a common goal, but they do not fully trust each other [68] (e.g., the participating banks in our approach).

We used the same testbed described in Section IV-B to implement our approach relying on the *Hyperledger Fabric* blockchain. We implemented *Hyperledger Fabric version 1.4*. We implemented our proposed detection and verification algorithms using *JavaScript* because *Fabric* offers (among other) a ready-to-use *JavaScript* API to manage the blockchain. We re-executed the same scenarios described in Section IV-B: 1) the needed block is at the beginning of the blockchain; 2) the needed block is in the middle of the blockchain; 3) the needed block is at the end of the blockchain; and 4) the check's record does not exist in the blockchain (fake check). Also, for each scenario, we executed 100 tests, and we measured the time needed to find the block and to execute the detection algorithm for each test when applied on a different block. Fig. 8 shows the average and standard deviation over the 100 tests conducted for each scenario.

When CAI is in a block located at the end of the blockchain (between the positions [499 000, 500 000]), the average authentication time is 755.59 ms with a standard deviation of 9.43 ms. For the second scenario, when the needed block is in the middle of the blockchain (between the positions [225 000, 226 000]), the average authentication time is 754.62 ms with a standard deviation of 7.51 ms. For the scenario where the needed block is at the beginning of the blockchain (between the positions [1, 1000]), the average authentication time is 766.06 ms with a standard deviation of 27.32 ms. Finally, for the scenario where the needed CAI does not exist in the blockchain, the time taken to obtain a response is 752.31 ms with a standard deviation of 9.62 ms. It is worth noting that, in contrast with the *Namecoin* implementation, the position of the block has no impact on the searching time. Moreover, the results obtained are in the order of milliseconds. The reason for the better results compared with those obtained before (through the public blockchain implementation) is explained by the architecture of

[19] [Online]. Available: www.hyperledger.org

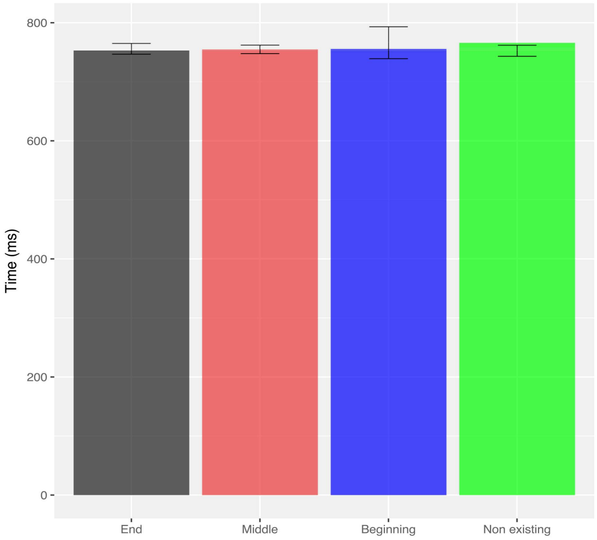


Fig. 8. Execution time of the check's authentication process according to the position in the blockchain of the block that owns the needed CAI (*Hyperledger Fabric* implementation).

Hyperledger Fabric, which implements an additional database layer. Indeed, each peer locally maintains the ledger in the form of the append-only blockchain and as a snapshot of the most recent state in a key–value store [68]. More precisely, it stores one tuple of the form (key, value, version) for each unique entry of the blockchain in a state database. Hence, the state database is simply an indexed view into the chain's transaction log [69]. Accordingly, searching for blocks' transactions in *Hyperledger Fabric* blockchain is as optimized as the search process in a database.

State database options include *LevelDB* and *CouchDB*. *LevelDB* is the default state database embedded in the peer process. *CouchDB* is an optional alternative external state database that provides additional query support permitting rich queries [69]. In our implementation, we used *CouchDB* as a state database.

However, implementing such a private blockchain will not satisfy the needed requirements, especially the infrastructure management requirement. Indeed, we stated that the proposed approach must be lightweight and low-cost and must not represent a burden for third parties that deploy it. However, the deployment of the *Hyperledger Fabric* by all the participating banks will involve additional work. Moreover, this approach will also require banks to deploy additional human resources in order to set up and maintain such infrastructures.

VI. Conclusion

Fake checks continue to be one of the most common instruments used to commit fraud against consumers. This fraud is very costly for victims because they generally lose thousands of dollars as well as being liable to judicial proceedings. Fake check scam continues to exist because of the existing check payment protocol, which credits the customers' accounts before verifying the authenticity of the deposited checks and their owners.

To the best of our knowledge, currently, there is no IT authentication scheme, which helps in the authentication of legitimate checks as well as the detection of fake ones. In this context, in this article, we propose a blockchain-based scheme, which allows the authentication of checks almost instantly after their deposit, thus avoiding the current *float* time of more than 48 h as well as the bilateral procedure initiation between the banks involved, making them saving time and resources. Our proposed scheme is low cost and easy to implement, and it satisfies all the needed requirements as well as overcome the challenges we have discussed.

In our future works, we will focus on reducing the CAI/CVI searching time. We will investigate how to use more advanced membership query techniques such as Bloom filters as well as advanced blockchain searching methods such as parallel processing.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable comments, which helped improve the content, organization, and presentation of this article.

References

[1] S. Baker, "Don't cash that check: BBB study shows how fake check scams bait consumers," Tech. Rep., Better Bus. Bureau, Arlington County, VA, USA, Sep. 2018.
[2] L. M. Rose, "Modernizing check fraud detection with machine learning," Ph.D. dissertation, Dept. Financial Crime Compliance Manage., Utica College, Utica, NY, USA, 2018.
[3] *Federal Trade Commission*, "Consumer sentinel network data book 2017," Federal Trade Commission, Washington, DC, USA, Tech. Rep., Mar. 2018.
[4] C. Tressler, "FTC: The bottom-line on fake checks scams," Federal Trade Commission, Washington, DC, USA, Tech. Rep., Feb. 2020.
[5] "2017 Internet crime report," Federal Bureau of Investigation/Internet Crime Complaint Center, Washington, DC, USA, Tech. Rep., 2018.
[6] K. Pak and D. Shadel, "AARP Foundation national fraud victim study," AARP Foundation, Washinton, DC, USA, Tech. Rep., 2011.
[7] C.-D. Chen and L.-T. Huang, "Online deception investigation: Content analysis and cross-cultural comparison," *Int. J. Bus. Inf.*, vol. 6, no. 1, pp. 91–111, 2011.
[8] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the Internet of Things, *IEEE Access*, vol. 4, pp. 2292–2303, 2016.
[9] A. Reyna, C. Martín, J. Chen, E. Soler, and M. Díaz, "On blockchain and its integration with IoT: Challenges and opportunities," *Future Gener. Comput. Syst.*, vol. 88, pp. 173–190, 2018.
[10] M. T. Hammi, B. Hammi, P. Bellot, and A. Serhrouchni, "Bubbles of trust: A decentralized blockchain-based authentication system for IoT," *Comput. Secur.*, vol. 78, pp. 126–142, 2018.
[11] R. M. Factora, "Financial and legal methods to protect individuals from financial exploitation," in *Aging and Money*. New York, NY, USA: Springer, 2014, pp. 109–122.
[12] C. W. Smith, "Defense to a payor bank's liability for late returns," *CCH Deposit Law Notes*, vol. 2, no. 6, p. 8, 2001.
[13] A. T. Riggs and P. M. Podrazik, "Financial exploitation of the elderly: Review of the epidemic—Its victims, national impact, and legislative solutions," in *Aging and Money*. New York, NY, USA: Springer, 2014, pp. 1–18.
[14] J. Jones and D. McCoy, "The check is in the mail: Monetization of Craigslist buyer scams," in *Proc. APWG Symp. Electron. Crime Res.*, 2014, pp. 25–35.
[15] I. Abiola, "An assessment of fraud and its management in nigeria commercial banks," *Eur. J. Social Sci.*, vol. 10, no. 4, pp. 628–640, 2009.
[16] J. A. Ojo, "Effect of bank frauds on banking operations in Nigeria," *Int. J. Investment Finance*, vol. 1, no. 1, p. 103, 2008.
[17] S. Chhabra, G. Gupta, M. Gupta, and G. Gupta, "Detecting fraudulent bank checks," in *Proc. IFIP Int. Conf. Digit. Forensics*, 2017, pp. 245–266.
[18] R. Kumar and G. Gupta, "Forensic authentication of bank checks," in *Proc. IFIP Int. Conf. Digit. Forensics*, 2016, pp. 311–322.

[19] R. E. Smagala, "Coded checks and in methods of coding," U.S. Patent 3 829 133, Aug. 1974.

[20] V. G. Bell Jr, T. P. Burke, G. D. Margolin, and V. V. Vurpillat, "Check with electrically conductive layer," U.S. Patent 4 231 593, Nov. 1980.

[21] W. F. McWhortor, "System and method for enhancing detection of counterfeit financial transaction documents," U.S. Patent 5 371 798, Dec. 1994.

[22] H. Badis and E. A. Y. Christian, "Fake check scams: A blockchain based detection solution," in *Proc. 9th Int. Conf. Comput. Sci. Inf. Technol.*, 2019, pp. 81–97.

[23] N. Dashkevich, S. Counsell, and G. Destefanis, "Blockchain application for central banks: A systematic mapping study," *IEEE Access*, vol. 8, pp. 139918–139952, 2020.

[24] R. O'Shields, "Smart contracts: Legal agreements for the blockchain," *NC Banking Inst.*, vol. 21, p. 177, 2017.

[25] L. Cocco, A. Pinna, and M. Marchesi, "Banking on blockchain: Costs savings thanks to the blockchain technology," *Future Internet*, vol. 9, no. 3, 2017, Art. no. 25.

[26] Y. Guo and C. Liang, "Blockchain application and outlook in the banking industry," *Financial Innov.*, vol. 2, no. 1, pp. 1–12, 2016.

[27] E. Ducas and A. Wilner, "The security and financial implications of blockchain technologies: Regulating emerging technologies in Canada," *Int. J.*, vol. 72, no. 4, pp. 538–562, 2017.

[28] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Working Paper, p. 9, 2008.

[29] M. A.Khan and K. Salah, "IoT security: Review, blockchain solutions, and open challenges," *Future Gener. Comput. Syst.*, vol. 82, pp. 395–411, 2018.

[30] A. Bahga and V. K. Madisetti, "Blockchain platform for industrial Internet of Things," *J. Softw. Eng. Appl.*, vol. 9, no. 10, pp. 533–546, 2016.

[31] A. Fayad, B. Hammi, and R. Khatoun, "An adaptive authentication and authorization scheme for IoT's gateways: A blockchain based approach," in *Proc. 3rd Int. Conf. Secur. Smart Cities, Ind. Control System Commun.*, 2018, pp. 1–7.

[32] S. Huh, S. Cho, and S. Kim, "Managing IoT devices using blockchain platform," in *Proc. 19th Int. Conf. Adv. Commun. Technol.*, 2017, pp. 464–467.

[33] M. T. Hammi, P. Bellot, and A. Serhrouchni, "BCTrust: A decentralized authentication blockchain-based mechanism," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2018, pp. 1–6.

[34] S. Zeadally and J. B. Abdo, "Blockchain: Trends and future opportunities," *Internet Technol. Lett.*, vol. 2, no. 6, 2019, Art. no. e130.

[35] J. V. Hacker, F. Bodendorf, and P. Lorenz, "A framework to identify knowledge actor roles in enterprise social networks," *J. Knowl. Manage.*, vol. 21, pp. 817–838, 2017.

[36] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "A survey of distributed consensus protocols for blockchain networks," *IEEE Commun. Surv. Tut.*, vol. 22, no. 2, pp. 1432–1465, Apr./Jun. 2020.

[37] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," in *Concurrency: The Works of Leslie Lamport*. New York, NY, USA: ACM, 2019, pp. 203–226.

[38] M. Castro and B. Liskov, "Practical Byzantine fault tolerance and proactive recovery," *ACM Trans. Comput. Syst.*, vol. 20, no. 4, pp. 398–461, 2002.

[39] C. Dwork, N. Lynch, and L. Stockmeyer, "Consensus in the presence of partial synchrony," *J. ACM*, vol. 35, no. 2, pp. 288–323, 1988.

[40] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," in *Concurrency: The Works of Leslie Lamport.*New York, NY, USA: ACM, 2019, pp. 179–196.

[41] L. Lamport, "Using time instead of timeout for fault-tolerant distributed systems," *ACM Trans. Program. Lang. Syst.*, vol. 6, no. 2, pp. 254–280, 1984.

[42] F. B. Schneider, "Implementing fault-tolerant services using the state machine approach: A tutorial," *ACM Comput. Surv.*, vol. 22, no. 4, pp. 299–319, 1990.

[43] B. Liskov and J. Cowling, "Viewstamped replication revisited," Massachusetts Inst. Technol., Cambridge, MA, USA, Tech. Rep. MIT-CSAIL-TR-2012-021, 2012.

[44] L. Lamport, "Paxos made simple," *ACM Sigact News*, vol. 35, pp. 1–11, 2001.

[45] L. Lamport, "The part-time parliament," in *Concurrency: The Works of Leslie Lamport*. New York, NY, USA: ACM, 2019, pp. 277–317.

[46] W. Wang *et al.*, "A survey on consensus mechanisms and mining strategy management in blockchain networks, *IEEE Access*, vol. 7, pp. 22328–22370, 2019.

[47] P. G. Ciarlet and P. A. Raviart, "General Lagrange and Hermite interpolation in R^n with applications to finite element methods," *Arch. Rational Mech. Anal.*, vol. 46, no. 3, pp. 177–199, 1972.

[48] J.- P. Berrut and L. N. Trefethen, "Barycentric lagrange interpolation," *SIAM Rev.*, vol. 46, no. 3, pp. 501–517, 2004.

[49] H. A. Kalodner, M. Carlsten, P. Ellenbogen, J.Bonneau, and A. Narayanan, "An empirical study of Namecoin and lessons for decentralized Namespace design," in *Proc. Workshop Econ. Inf. Secur.*, 2015, pp. 1–23.

[50] *Secure Hash Standard (SHS)*, Federal Information Processing Standards Publication, FIPS PUB 180-4, Aug. 2015, p. 31.

[51] *Digital Signature Standard (DSS), National Institute of Standards and Technology*, Gaithersburg, MD, USA, FIPS PUB 186-4, 2013, p. 130.

[52] *Public Key Cryptography for the Financial Services Industry: Elliptic Curve Digital Signature Algorithm (ECDSA)*, ANSI, X9.62:2005, 2005, p. 128.

[53] K. Lauter, "The advantages of elliptic curve cryptography for wireless security," *IEEE Wireless Commun.*, vol. 11, no. 1, pp. 62–67, Feb. 2004.

[54] E. D. Win, S. Mister, B. Preneel, and M. Wiener, "On the performance of signature schemes based on elliptic curves," in *Proc. Int. Algorithmic Number Theory Symp.*, 1998, pp. 252–266.

[55] H. Badis, F. Achraf, K. Rida, Z. Sherali, and B. Youcef, "A lightweight ECC-based authentication scheme for Internet of things (IoT)," *IEEE Syst. J.*, vol. 14, no. 3, pp. 3440–3450, Sep. 2020.

[56] C. J. F. Cremers, *Scyther: Semantics and Verification of Security Protocols*. Eindhoven, The Netherlands: Eindhoven Univ. Technol., 2006.

[57] G. Lowe, "A hierarchy of authentication specifications," in *Proc. 10th Comput. Secur. Found. Workshop*, 1997, pp. 31–43.

[58] E.K. Lua, J. Crowcroft, M. Pias, R.Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *IEEE Commun. Surv. Tut.*, vol. 7, no. 2, pp. 72–93, Apr./Jun. 2005.

[59] A. Chauhan, O. P. Malviya, M. Verma, and T. S. Mor, "Blockchain and scalability," in *Proc. IEEE Int. Conf. Softw. Qual., Rel. Secur. Companion*, 2018, pp. 122–128.

[60] S. Kim, Y. Kwon, and S. Cho, "A survey of scalability solutions on blockchain," in *Proc. Int. Conf. Inf. Commun. Technol. Convergence*, 2018, pp. 1204–1207.

[61] Q. Zhou, H. Huang, Z. Zheng, and J. Bian, "Solutions to scalability of blockchain: A survey, *IEEE Access*, vol. 8, pp. 16440–16455, 2020.

[62] S. Goswami, "Scalability analysis of blockchains through blockchain simulation," Master's thesis, Dept. Comput. Sci., University of Nevada, Las Vegas, NV, USA, 2017.

[63] G. Karame, "On the security and scalability of bitcoin's blockchain," in *Proc. 2016 ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 1861–1862.

[64] R. Dennis and J. P. Disso, "An analysis into the scalability of bitcoin and ethereum," in *Proc. 3rd Int. Congr. Inf. Commun. Technol.*, 2019, pp. 619–627.

[65] I. Eyal, A. E. Gencer, E. G. Sirer, and R.V. Renesse, "Bitcoin-NG: A scalable blockchain protocol," in *Proc. 13th Symp. Netw. Syst. Des. Implementation*, 2016, pp. 45–59.

[66] K. Saito and M. Iwamura, "How to make a digital currency on a blockchain stable," *Future Gener. Comput. Syst.*, vol. 100, pp. 58–69, 2019.

[67] O. J.Mandeng, "Cryptocurrencies, monetary stability and regulation," LSE Institute of Global Affairs, London School of Economics, 2018.

[68] E. Androulaki *et al.* "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proc. 13th EuroSys Conf.*, 2018, Art. no. 30.

[69] *A Blockchain Platform for the Enterprise*, Hyperledger Fabric, 2019.