FAKE CHECK SCAMS: A BLOCK CHAIN BASED DETECTION SOLUTION

Badis HAMMI[1] and Yves Christian Elloh Adja[2]

[1]PSB School, Paris, France and [2]Telecom Paris Tech, Paris, France

ABSTRACT

Fake checks are one of the most common instruments used to commit fraud against consumers. This fraud is particularly costly for victims, since they generally loose 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 hours to detect the scam. In this context, we propose a block chain based scheme to authenticate checks. More precisely, our approach helps the banks to share information about provided checks without exposing the banks' customers' personal data. A proof of concept of our scheme was realized using Python language and relying on Name coin block chain.

KEYWORDS

Block chain, Security, Fake check scam, Spam, Fraud detection

1. INTRODUCTION

Electronic mails (emails) play a very important role in our society. Indeed, emailing represents one of the most used communication tools especially for institutions (administration, business, etc.). According to [1], there were more than 3.8 billion email users before the start of 2019, which represents over 100 million more than the previous year (2017). In other words, over half of the entire planet is using emailing as a communication mean. This number is expected to grow to over 4.2 billion by the end of 2022 [1]. In 2018, the total number of business and consumer emails sent and received per day exceeded 281 billion, and is forecast to grow to over 333 billion by the year end of 2022 [1][2].

Unfortunately, more than the half of the email activity are Spams (54% in 2018) [3]. Spams represent unwanted communication intended to be delivered to an indiscriminate target, directly or indirectly [4]. More specifically, Spam is a prospecting technique consisting of massively disseminating information by email, often of an Advertising nature, unsolicited by recipient Internet users. According to [5], the most prevalent type of spam is advertising-related email; this type of spam accounts for approximately 36% of all spam messages. The second most common category of spam is adult-related and makes up roughly 31.7% of all spam. The remaining 32.3% of the spam mass intends to harm users through phishing, spear phishing, malware propagation, scams and frauds. Spam damages and losses are estimated to be around $198 billion annually and will balloon to $257 billion per year if spam continues to flourish at its current rate [5].

Natarajan Meghanathan. (Eds) : CCSIT, NCWMC, DaKM - 2019
pp. 81-97, 2019. © CS & IT-CSCP 2019 DOI: 10.5121/csit.2019.90808

In this work, we focus on scams and more precisely on fake check scams. This fraud consists in (1) targeting people through email scam; (2) establishing a relationship (a business relationship most of the time); (3) sending them overpaid counterfeit paycheck, then, (4) asking for the overpayment. In comparison to the other scam types such as phishing or malware spreading, this fraud has more disastrous consequences on the victims. Indeed, in addition to the financial harm, the victims are most of the time exposed to judicial proceedings, since, in the eyes of the law; they tried to scam the bank.

There have been numerous solutions against spams. However, current solutions are not completely effective given the huge number of spams that land on users' mailboxes. Moreover, these solutions are ineffective against fake check scams, since the email represents only the first step. Even worse, the scammers target their victims on specialized ad websites, newspaper, specialized customers sites, and others, then, send them emails directly without using a massive spam tool.

Existing solutions were designed with the purpose of stopping spams. However, the majority of them treat spams as a whole (e.g. according to domain names) and do not handle each case apart. To the best of our knowledge, there is no method designed specifically to protect users from fake check scams. In this context, and considering that the victim can be reached directly via email and not through massive spam, we believe that the best solution to protect users is the detection of fake checks before their cashing by victims. Certainly, numerous works that aim to detect materially fake checks have been realized. Nonetheless, con artists excel in the art of trickery and create very realistic checks, besides, numerous scammers use professional printers and magnetic ink. Consequently, the designed check authentication solution must be more effective and each bank must be able to be completely sure that the treated check is provided by a real trusted authority before cashing it. Nevertheless, designing such a solution is very challenging due to the following difficulties:

- **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 would be feasible if each bank share 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 about the provided checks. Howbeit, no bank will share such information, mainly for (1) users privacy: since the users have engaged with this bank and not with another; and (2) for commercial competition: if users' information are accessible, nothing prevents any bank from contacting these people and offering them its services**. Thus, it is necessary to design a sharing system that ensures the non exposure of customers data to other third parties.**

- **Management of the sharing mechanism:** if a data sharing mechanism is deployed and used by banks to ensure the authenticity of checks, numerous questions will rise such as (1) who will maintain and manage the mechanism (infrastructure, protocol, etc.) ? (2) How the participating banks will share the management fees? (3) Who decides about the evolution of the mechanism? (4) Where the stored data will be kept? (5) Who can access these data? (6) How these data can be accessed? (7) If one bank decides not to use the mechanism anymore, how its retirement will affect the other actors. **Consequently, it is mandatory to design a lightweight and low-cost sharing system that does not**

represent a burden for the third parties that deploy it and which adapt to all possible situations.

- **Non modification of existing protocols:** any proposal that require any modification in the existing bank protocols such as the modification of the check format to add some data or in the payment procedure, will have a chain reaction on numerous parts or protocols of the banking ecosystems. Such consequences persuade banks to refuse the adoption of the proposal. **Ergo, the proposed mechanism must not modify any of the existing protocols, solutions or procedures and must be transplantable on the current banking ecosystem.**

- **Scalability:** considering the number of banks as well as the huge number of customers**, it is mandatory to propose a highly scalable mechanism that can handle such a load.**

- **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 was provided by the Providing-Bank, and must make sure that the Providing-Bank is trustworthy**. Accordingly, it is necessary to equip the sharing system with an effective authentication method.**

Contribution of this work

We believe, such as many researchers [6][7][8] that block chain represents a very promising technology for the development of decentralized and resilient security solutions. Therefore, in this paper we propose an effective block chain 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, the proposed approach is financially low-cost, does not affect the existing protocols and release the banks from any additional infrastructure management.

This paper is organized as follows: Section 2 depicts the details of the fake check scam. Afterwards, Section 3 exhibits our proposal. Then, Section 4 discuss and analyzes our evaluation campaign. Finally, Section 5 concludes the paper and points out future research perspectives.

2. 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. Then, asks the victim to wire some of the money back. The scammers always have a good story to explain the overpayment. We cite a typical real scenario: 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 email or even by phone in order to agree on the place, the amount (e.g. 500$) and the date (which is often not for straightaway). Afterwards, the scammer, pretending acting 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 his child's nanny owe him the overpayment (e.g. 1500 $), and asks gently Alice to make a money order or a wire transfer to the nanny. Consequently Alice cash the check and sends the overpayment to the scammer thinking she is doing it for the nanny. Legally, persons are responsible for the checks they deposit. Hence, it is Alice which will refund the check as well as the bank's fine and fees.

This fraud is possible because of the check payment protocol. Indeed, before being credited to an account, 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 in a period of 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 of 3000e. If the check exceeds this amount, a first part of the check's amount is credited meanwhile the next step. (2) the Cashing-Bank sends the check to the Providing-Bank for money collection. (3) If all goes well, the customer can receive the amount of the check. But if the check is unpaid, bounced, irregular or fake, the Cashing-Bank will re-issue the corresponding amount from the customer's account which also pays additional fees (according to the bank and the country policy, a fine or judicial proceedings can be considered). The Float is the amount of time it takes for money to move from one account to another. It can goes from 48 hours until several weeks. 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 realize his fraud.

Fake checks drive many types of scams like 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:

Mystery shopping scam: con artists lure victims by sending spams or posting ads for mystery shoppers in job classifieds. 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, 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[3]. Another form of this fraud, is a scammer that sends spam emails informing being the inheritor of an amount of money but he cannot cash it by himself because of family (or other) problems, and hires the victim to cash it for him while keeping a compensation.

Fake job scam: as it was depicted in the beginning of this section, typically this scam starts with a victim responding to an online posting, to a spam, or the victim may have posted information online, to propose a job. Either way, the victim eventually gets "hired" by the con artist and receives emails 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.

Unexpected check scam: typically this fraud starts following a spam email inviting victims to participate to 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 reversed as fees.

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[4].

3. PROPOSED APPROACH

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

3.1. BACKGROUND

Our approach relies mainly on (1) a block chain and (2) Lagrange Interpolating Polynomial. In this section we provide a quick summary of these concepts.

3.1.1. BLOCK CHAIN

A block chain is defined as a distributed database (ledger) that maintains a permanent and tamper-proof record of transactional data. A block chain 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 prevent a single point of failure. All copies are updated and validated simultaneously [8].

Block chain technology was created to solve the double spending problem in crypto-currency [9]. However, currently, numerous works explore block chain 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 [10][11][12][13][14].

The block chain ledger is composed of multiple blocks; each block is composed of two parts. The first 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, etc. The second is called the header and contains information about its block e.g. timestamp, hash of its transaction, etc. as well as the hash of the previous block. Thus, the set of the existing blocks forms a chain of linked and ordered blocks. The longer is the chain, the harder is to falsify it. Indeed, if a malicious user wants to modify or swap a transaction on a block, (1) it must modify all the following blocks, since they are linked with their hashs. (2) Then, it must change the version of the block chain that each participating node stores [8], which is very hard to achieve.

3.1.2. LAGRANGE POLYNOMIAL INTERPOLATION

Lagrange polynomials allow to interpolate a series of points by a polynomial which passes exactly by 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 at each $x_j$value

the corresponding valuey_j. Thus, the function coincide at each point [15][16]. Equation 1 defines the Lagrange polynomial associated with these points.

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

Equation 1 can also be written as described by Equation 2 [16].

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

3.2. SYSTEM OPERATION

Our approach aims helping the banks to share information about provided checks in order to verify their authenticity during the payment and without exposing any of the customer's personal data. The proposed scheme relies on a public blockchain. Also, the choice of a hash algorithm as well as a signature algorithm is required. In the remaining of this paper we consider using (1) *Name coin* blockchain, (2) *SHA-256* as a hash algorithm and (3) *Elliptic Curve Digital Signature Algorithm (ECDSA)*.

Our system intervene during 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 which public key is certified by a trusted authority, i.e. each bank must own a certificate, accessible for any third party.

Table 1. Checkbook's Authentication Information data structure.

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

3.2.1. 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 block chain. More precisely, for each provided checkbook, the bank creates a data structure related to this check book called Checkbook's Authentication Information (CAI) and adds it to the public block chain through a transaction. The CAI data structure is composed of three fields (1) a Lagrange polynomial (2) a hash and (3) a cryptographic signature, as depicted by Table 1.

Algorithm 1: Basic operations provided by the bank's entity which executes the CAI creation

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

We assume that the bank's entity which executes this task, can provide basic protocol primitives in order to recover needed information to create the CAI structure and to send it to the block chain. Such an API is depicted in Algorithm 1, while Algorithm 2 describes the whole process.

3.2.1. A. 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 regard, considering a solution where each check is registered individually in the block chain will be more than heavy, since it requires as much block chain transactions as existing checks. Ergo, we use Lagrange polynomials as an aggregation for all the checks that the checkbook owns. For example, considering a checkbook that contains four checks having the following numbers:$E = \{2,3,4,5\}$. Considering Equation 2, the Lagrange polynomial built according to this set will be:

$$L(x) = (x-2)(x-3)(x-4)(x-5)$$

$$= x^4 - 14x^3 + 71x^2 - 154x + 120 \quad (3)$$

All the elements of the set E are roots to the computed polynomial L(x) described by Equation 3. Consequently, following this logic, our scheme must build a Lagrange polynomial for each provided checkbook using its check numbers. However, it implies to build Lagrange polynomials of degree 100 (or 50), which is time and CPU cycles consuming, and particularly requires a large space on the CAI structure[5]. . To optimize this step, especially knowing 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 will serve for the verification phase by testing if the check number is in the interval composed by the two roots of the Lagrange polynomial. More thoroughly, if we consider the last example of the set E. The Lagrange polynomial created according to its upper and lower bounds ([2, 5]) is as described by Equation 4:

$$(x)=(x-2)(x-5)$$
$$=x^2-7x+10 \quad (4)$$

The two roots of L(x) are 2 and 5. Thus, all the elements of E will be in the interval composed by 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 in the interval built by the two roots of the polynomial.

The Lagrange polynomial field is structured as depicted by Equation 5[7].

$$\langle Polynomial\ degree\ (n), [x^n](L(x)), [x^{(n-1)}](L(x)), \dots, [x](L(x)), [x^0](L(x))\rangle \quad (5)$$

For example, if we consider the polynomial described by Equation 4, the Lagrange polynomial field of the CAI data structure will be ⟨2, 1, −7, 10⟩.

The majority of programming languages (e.g. C, C++, Java ...) considers that an integer needs four bytes to be represented. Along these lines, the Lagrange polynomial field of the CAI will have a size of 16 bytes. 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 used language.

Algorithm 2: CAI creation and dissemination on the block chain

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
1. SENDTRANSACTION(*bc*, CAI_CREATION(*chBook*, *customerX*)) // send a blockchain transaction containing the CAI data structure

FunctionCAI_Creation(*chBook*, *customerX*)
2. *lPolynomial*← LAGRANGEINTERPOLATIONFCT(GETFIRST(*chBook*), GETLAST(*chBook*));
3. *hash*← SHA-256(CONCATENATE(GETNAME(*customerX*), *bankName*, GETACCOUNTNB(*customerX*))) // applies SHA-256 hash algorithm on the set of defined parameters
4. *signature*← 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 ← MAKEARRAY(*lPolynomial*, *hash*, *signature*) // creates the CAI data structurereturn CAI
end function

3.2.1. B. HASH FIELD

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

- Full Name: the customer's full name;
- Providing-Bank: the bank that provided the checkbook;
- Account number: the customer's account number.

The data considered in the hash are available on the check. This hash field will serve as the landmark to find the block on the blockchain level.

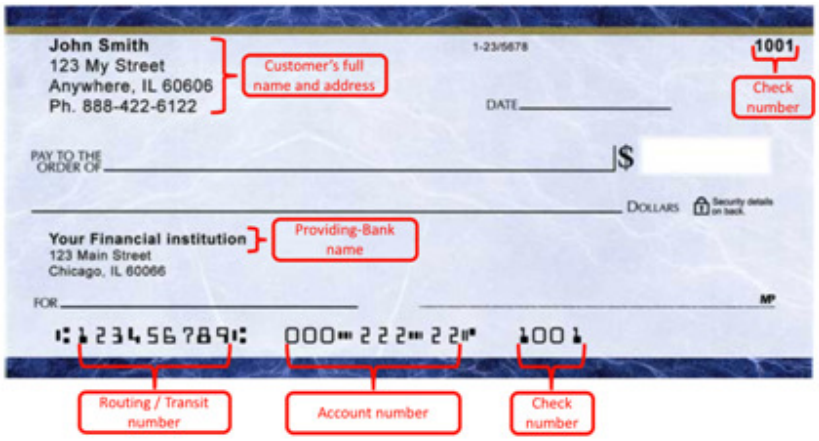

Figure 1: Check example

3.2.1. C. SIGNATURE FIELD

The signature is provided on the information that all the checks of the checkbook contain. Figure 1 exhibits these information:

- Full Name: the customer's full name;
- Full Address: the customer's address;
- Providing-Bank: the bank that provided the checkbook;
- Routing number: generally composed of nine digits. It identifies the location where the account was opened;
- Account number: the customer's account number;
- Hash: the hash contained in the second field of the data structure;
- 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 this data. However, there are only a hash and a signature that are stored. Since the hash function behind the signature is not reversible, it is impossible to recover the customer's data (data that was hashed).

3.2.1. WITHDRAWAL OPERATION PHASE

When a customer deposits a check in a Cashing-Bank, few verifications are provided before triggering the payment. These verification operations can be realized (1) by the human agent that treats the check through a dedicated Human Machine Interface (HMI) 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 be just 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 needed information from the blockchain and from the deposited check. Such an API is depicted in Algorithm 3, while Algorithm 4 describes the whole verification process.

Algorithm 3: Basic operations provided by the bank's entity which executes the check's verification process

1. **Function** GETCHECKNB(Check *ch*) : Integer // returns the check's number
2. Function GETNAME(Check *ch*) : String // returns the customer's full name from the deposited check
3. **Function** GETADDRESS(Check *ch*) : String // returns the customer's full address from the deposited check
4. **Function** GETACCOUNTNB(Check *ch*) : Integer // returns the customer's account number from the deposited check
5. **Function** GETROUTINGNB(Check *ch*) : Integer // returns the bank's routing number from the deposited check
6. **Function** GETBANKNAME(Check *ch*) : String // returns the bank's name from the deposited check
7. **Function** GETLAGRANGEPOLYNOMIAL(DataStructure *CAI*) : [ ] Integer // returns the Lagrange polynomial from the CAI
8. **Function** RESOLVEDEG2EQUATION([ ] Integer *polynomial*) : [ ] Integer // resolves an equation of the second degree and returns an array with the found roots sorted in ascending order
9. **Function** ISEXISTINGTRANSACTION(Blockchain *bc*, String *hash*) : Boolean // verifies if the blockchain have a block which contains a transaction which in turn contains the needed hash field
10. **Function** GETCAI(Blockchain *bc*, String *hash*) : DataStructure // returns the needed CAI from the blockchain
11. **Function** ERROR(String *errorMessage*) // returns and error message

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, are concatenated and then hashed. **Second**, the program browse the block chain to find the block containing the CAI which hash field is equivalent to the computed hash: (1) if the block is not found, 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 abort. (2) if the block is found, the authentication of the bank is necessary. In other words, since it is a public blockchain, the system must ensure that it is the bank that provided that transaction and not another entity. For that aim, the verification scheme concatenates the data existing in the check (customer's full name, customer's full Address, Providing-Bank, Check's routing number, customer's account number) as well as the Lagrange polynomial existing on the retrieved CAI

structure, and uses it to verify the CAI's signature relying on the Banks public Key (available on the published certificate). If the signature verification fails, the payment operation abort. **Third**, once the signature is verified, considering the fact of verifying each check individually in the perspective of creating a revocation system for the used and revoked checks, the verification scheme resolves the Lagrange polynomial of the CAI. Then, extracts the check's number and verifies if it fits in the interval composed by the polynomial's roots (as detailed in Section 3.2.1.a). If the verification succeeds, the payment operation can be triggered.

4. IMPLEMENTATION, EVALUATION AND DISCUSSION

4.1. IMPLEMENTATION

To implement our approach, we opted for *Name coin*[1]blockchain [17]. *Name coin* is a fork of Bit coin which aims to provide a decentralized DNS. Indeed, it implements the top level domain .bit, which is independent of Internet Corporation for Assigned Names and Numbers (ICANN)[2]. Table 2 describes the main features of *Name coin*.

Algorithm 4: Check verification process

```
Declaration:
ch: Check          // the deposited check
bc: Blockchain     // the used blockchain
Procedure CHECKVERIFICATION(ch, bc)
   customerData ← CONCATENATE(GETNAME(ch), GETBANKNAME(ch), GETACCOUNTNB(ch))
  IF (ISEXISTINGTRANSACTION(bc, SHA-256(customerData))) THEN
       CAI ← GETCAI(bc, SHA-256(customerData))
       signatureData ← CONCATENATE(getName(ch), GETADDRESS(ch), GETBANKNAME(ch),
      GETROUTINGNB(ch), GETACCOUNTNB(ch), GETLAGRANGEPLYNOMIAL(CAI));
      IF ECDSA_VERIFY(signatureData, BankPublicKey) THEN // applies ECDSA signature
      verification algorithm on the set of defined parameters using the bank's public key
            polynomialRoots ← RESOLVEDEG2EQUATION(GETLAGRANGEPLYNOMIAL(CAI ));
         IF ( GETCHECKNB(ch) ≥ polynomialRoots[0] AND GETCHECKNB(ch) ≤  polynomialRoots[1])
          THEN
                CHECKPAYMENTOPERATION(ch) // triggers the check payment operation
          ELSE
                ERROR("Failure in the check's authentication ")
       ELSE
          ERROR("Failure in the bank authentication ")
  ELSE
Error("Failure in the check's owner authentication")
End procedure
```

[1]https://namecoin.org

[2]https://www.icann.org

Table 2: *Namecoin* features and statistics

Data Field	Feature
Type	public block chain
Feature	Fork of Bit coin
Transaction fee average	$0.00037 USD
Block time	11 minutes
Blocks avg/h	6
Transaction avg /h	22
Block chain dimension	5.63 GB

We opted for *Name coin* for three main reasons:

1. It allows data storage in the form of key/value pair. Users have the possibility to store values of 520 bytes of size which is more than sufficient to host the CAI structure;
2. The daily volume of transactions is relatively weak, which facilitates the data search in the blockchain;
3. Transactions fees are low cost (the average transaction fee is about $0.00037 USD[3]).

For the hash function we use *SHA-256* since it represent one of the recommended hash algorithms by the *National Institute of Standards and Technology (NIST)*[4][18].

For the signature algorithm we opted for *Elliptic Curve Digital Signature Algorithm (ECDSA)* [19][20]. *ECDSA* represent multiple advantages over traditional signature algorithms such as *Rivest Shamir Adleman (RSA)* especially concerning key sizes and signature time [21][22].

4.2. EVALUATION FRAMEWORK AND SCENARIOS

Regarding the evaluation framework, we used *Multichain*[5]to simulate the *Namecoin* blockchain. *Multichain* is an open source blockchain platform which helps in the building and deployment of blockchain applications. It is fully configurable according to the user's needs, thus, can be setup to reproduce the same functioning as any other blockchain. We used this feature to simulate a *Namecoin* blockchain. Currently[6]the *Namecoin* blockchain includes 451444 blocks. To simulate the *Namecoin* blockchain, for our experiments, we used a blockchain owning 500000 blocks.
The program that shares the new issued checkbook's data (applicable by the Providing-Bank) as well as the authentication verification program (applicable by 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 rely solely on Information Technology (IT) resources. Thus, we cannot compare the efficiency of our method

[3] https://bitinfocharts.com/namecoin/
[4] https://csrc.nist.gov/Projects/Hash-Functions
[5] https://www.multichain.com
[6] 11th of May 2019

with another existing method. Moreover, since our approach verifies if a check's record is existing in the blockchain or not, there are no false positives nor false negatives.

knowing that the main mode of searching a block in the blockchain is feasible in a sequential method (block by block). The authentication time of a check will depend on the position of the block including its corresponding CAI in the blockchain. Accordingly, we were interested in measuring this time for different cases : (1) the needed block is in the beginning of the blockchain; (2) the needed block is in the middle of the blockchain; (3) the needed block is in 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 experimentations, where we measured the time needed to find the block. Each experimentation affects a different block. More specifically, for the first scenario, the needed blocks where between the blocks 1 and 1000. For the second scenario, the needed blocks were between the blocks 225000 and 226000. For the third scenario, the needed blocks where between the blocks 499000 and 500000.

We are aware that the search time depends mainly on the used machine as well as the used programing language (e.g. *C* is faster than *Python*). However, we wanted to compare the different discussed cases considering the same basis, language and material. The experiments have been performed on the following testbed: the host system features an *Intel(R) Quad-Core i7-7700k CPU 4.20 GHZ* with 16 GB of *RAM*. It executes an up-to-date version of the *KALI Linux 4.12.0* distribution.

4.3. EVALUATION RESULTS

4.3.1. SECURITY AND PERFORMANCE REQUIREMENTS EVALUATION

In this section we focus on the evaluation of the security features, performance features and the design challenges discussed earlier (see Section 1) and that must be satisfied by a check authentication approach:

Data sharing between banks: for each checkbook a CAI is shared. The CAI structure ensures the non exposure of the users data since it only includes a hash and a cryptographic signature, which are non reversible. The Cashing-Bank will be aware of the check's owner data through the deposited check. However, this is not unique to our approach, since it is also the case of current bank's protocol.

Management of the sharing mechanism: our approach rely on a public blockchain which is fully autonomous. Ergo, the bank will have no additional infrastructure to handle. Furthermore, sending transactions or reading them from a blockchain represent a process that can easily be integrated and handled. Regarding the financial cost, we believe that each security service provided needs a cost, as long as it remains lower than the potential damages. In our approach, for each created checkbook, a blockchain transaction is needed. The transaction's cost depends on the used blockchain. However, this cost remains negligible compared to the potential damages. We recommended the use of *Namecoin*, which transaction's fees are around $0.00037 USD. We are aware that the evolution of cryptocurrency rate represents an issue. However, according to studies like [23] and [24], the evolution of the cryptocurrencies rates will get more stable over time [8]. Finally, the transaction's fees can be added to the account maintenance cost each time the customer asks for a new checkbook.

Non modification of existing protocols: our approach does not require any modification of the existing banking protocols such as modifying the checks format, adding a physical security on the checks or the ATMs or modifying the communication protocol between the different banks. It only represents an additional verification before executing the usual payment protocol and an additional action after providing a new checkbook.

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 meet scalability at large scale [25].

Availability: the totally decentralized architecture of blockchains makes them robust against DoS/DDoS 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.

Authentication: our approach provides an authentication of the deposited check following a blockchain browsing to find the check's record. This exempts our scheme from false negatives and false positives detections and makes it completely reliable. It allows to authenticate a legitimate check and to detect a fake one.

4.3.2. NUMERICAL RESULTS

In this section we present the numerical results related to the time consumption evaluation of our approach. As described in Section 4.2, we were interested in measuring the needed time of the check's authentication process described by Algorithm 4 (including all the algorithm's steps), for different cases of the needed block's position. Knowing that searching in a blockchain is very costly in time, we used parallel programing mode for the execution of our verification program on the four cores of our host machine. Figure 2 exhibits the average and standard deviation over the 100 experimentations of each scenario.

Regarding the scenario where the needed CAI is in a block localized at the end of the blockchain (between the positions [499000,500000]), the average authentication time is 0.28 seconds (s) with a standard deviation of 0.15 s. This time scale was expected considering that the research 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 [225000,226000]), the average authentication time is 210.84 s with a standard deviation of 20.03 s. We can note the costly nature of the blockchain search operation, especially that our approach does not look for transactions' 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 in the beginning of the blockchain (between the positions [1,1000]), since the average authentication time is 735.21 s with a standard deviation of 80.40 s. Finally, for the scenario where the needed CAI does not exist on the blockchain, the needed time to obtain a response is 738.68 s which is almost the exact same time as the last scenario, with a similar standard deviation of 75.50 s. Nonetheless, even if the execution of the check's authentication can spend few minutes in some cases, it remains very far from the current float time of more than 48 hours. Furthermore, it protects the banks and especially the customers from being scammed.

We recall that we developed a proof of concept executed on a simple machine with a high level interpreted language (*Python*) known to have execution times more important than many other

languages such as *C*. A bank is surely in the capacity of using more powerful machines with a better conceived program.

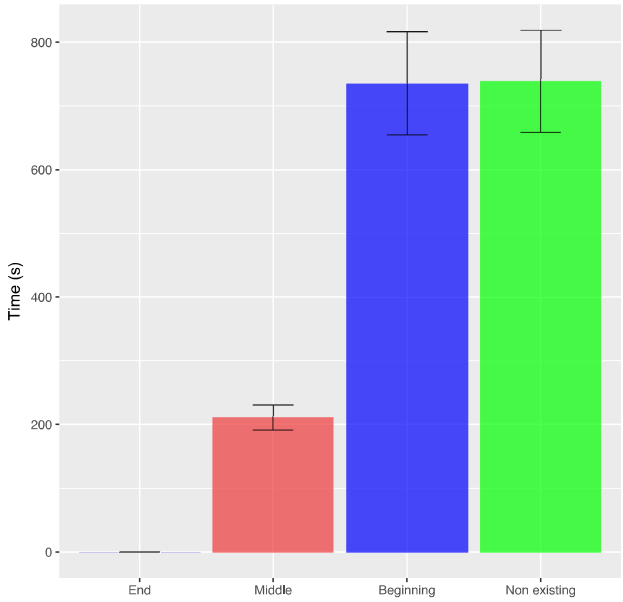

Figure 1 : Execution time of the check's authentication process according to the position in the blockchain of the block owning the needed CAI

5. CONCLUSIONS AND FUTURE WORKS

Fake checks continue to be one of the most common instruments used to commit fraud against consumers. This fraud is one of the most costly for victims, since they generally loose thousands of dollars as well as being exposed to judicial proceedings. Fake check scam continue to exist because of the current 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, we propose in this paper, 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 hours. Our proposed scheme satisfy all the needed requirements and overcome the discussed challenges. Even if our proposal can detect the fake checks in the great majority of cases, it remains non efficient against certain scenarios. Particularly, if a scammer creates a fake check by using the exact same information of an existing legitimate check (following a check theft or a social engineering hack), our system cannot detect the fraud. Also, if a scammer uses/reuses the information of an already deposited check, our current scheme cannot detect it. This can -in part- be solved by using a revocation system for used and revoked checks. Thereupon, in our future works we will focus on the design of a revocation system adapted to the provided authentication system. Furthermore, we will work on reducing the CAI searching time.

REFERENCES

[1] Email Statistics Report, 2018-2022. *Technical report*, THE RADICATI GROUP, INC., March, 2018.

[2] Number of sent and received e-mails per day worldwide from 2017 to 2022 (in billions). *Technical report*, Statista, 2019.

[3] Spam: share of global email traffic 2014-2018. Technical report, Statista, 2019.

[4] Gordon V Cormack et al. Email spam filtering: A systematic review. *Foundations and Trennds ® in Information Retrieval*, 1(4):335–455, 2008.

[5] Spam Statistics and Facts. *Technical report*, spamlaws.com, 2019.

[6] Konstantinos Christidis and Michael Devetsikiotis. Blockchains and smart contracts for the internet of things. *IEEE Access*, 4:2292–2303, 2016.

[7] Ana Reyna, Cristian Martín, Jaime Chen, Enrique Soler, and Manuel Díaz. On blockchain and its integration with IoT. Challenges and opportunities. *Future Generation Computer Systems*, 88:173–190, 2018.

[8] Mohamed Tahar Hammi, Badis Hammi, Patrick Bellot, and Ahmed Serhrouchni. Bubbles of Trust: A decentralized blockchain-based authentication system for IoT. *Computers & Security*, 78:126–142, 2018.

[9] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. page 1–9, 2008.

[10] Minhaj Ahmad Khan and Khaled Salah. IoT security: Review, blockchain solutions, and open challenges. *Future Generation Computer Systems*, 82:395–411, 2018.

[11] ArshdeepBahgaandVijayKMadisetti.Blockchainplatformforindustrialinternetof things. *J. Softw. Eng. Appl*, 9(10):533, 2016.

[12] Achraf Fayad, Badis Hammi, and Rida Khatoun. An adaptive authentication and authorization scheme for iot's gateways: a blockchain based approach. *In 2018 Third International Conference on Security of Smart Cities, Industrial Control System and Communications (SSIC)*, pages 1–7. IEEE, 2018.

[13] Seyoung Huh, Sangrae Cho, and Soohyung Kim. Managing iot devices using blockchain platform. *In Advanced Communication Technology (ICACT), 2017 19th International Conference on*, pages 464–467. IEEE, 2017.

[14] Mohamed Tahar Hammi, Patrick Bellot, and Ahmed Serhrouchni. BCTrust: A decentralized authentication blockchain-based mechanism. *In Wireless Communications and Networking Conference (WCNC)*, 2018 IEEE, pages 1–6. IEEE, 2018.

[15] Philippe G Ciarlet and PA Raviart. General lagrange and hermite interpolation in Rn with applications to finite element methods. *Archive for Rational Mechanics and Analysis*, 46(3):177–199, 1972.

[16] Jean-Paul Berrut and Lloyd N Trefethen. Barycentric lagrange interpolation. *SIAM review*, 46(3):501–517, 2004.

[17] Harry A Kalodner, Miles Carlsten, Paul Ellenbogen, Joseph Bonneau, and Arvind Narayanan. An empirical study of namecoin and lessons for decentralized namespace design. *In WEIS*. Citeseer, 2015.

[18] FIPS PUB 180-4: Secure Hash Standard (SHS). *FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION*, page 31, August 2015.

[19] FIPS PUB 186-4: Digital signature standard (DSS). National Institute of Standards and Technology, page 130, 2013.

[20] ANSI, X9.62:2005. Public key cryptography for the financial services industry: Elliptic Curve Digital Signature Algorithm (ECDSA). page 128, 2005.

[21] Kristin Lauter. The advantages of elliptic curve cryptography for wireless security. I*EEE Wireless communications*, 11(1):62–67, 2004.

[22] Erik De Win, Serge Mister, Bart Preneel, and Michael Wiener. On the performance of signature schemes based on elliptic curves. *In International Algorithmic Number Theory Symposium*, pages 252–266. Springer, 1998.

[23] Kenji Saito and Mitsuru Iwamura. How to make a digital currency on a blockchain stable. arXiv preprint arXiv:1801.06771, pages 1–15, 2018.

[24] Ousmène Jacques Mandeng. Cryptocurrencies, monetary stability and regulation. *Technical report*, 2018.

[25] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, and Steven Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys & Tutorials*, 7(2):72–93, 2005.