The ISC Int'l Journal of Information Security

January 2023, Volume 15, Number 1 (pp. 17–26)

http://www.isecure-journal.org

Hierarchical Deterministic Wallets for Secure Steganography in Blockchain

Omid Torki [1], Maede Ashouri-Talouki [1,*], and Mojtaba Mahdavi [1]

[1] *Department of Information Technology Engineering, Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran.*

ARTICLE INFO.

Article history:
Received: –
Revised: –
Accepted: –
Published Online: –

Keywords:
Steganography, Blockchain, Bitcoin, Ethereum, Transaction Address, Hierarchical Deterministic Wallets

Type: –

doi: --

dor: --

ABSTRACT

Steganography is a solution for covert communication, and blockchain is a p2p network for data transmission, so the benefits of blockchain can be used in steganography. In this paper, we discuss the advantages of blockchain in steganography, which include the ability to embed hidden data without a manual change in the original data and the readiness of the blockchain platform for data transmission and storage. By reviewing the previous four steganography schemes in blockchain, we have examined their drawback and shown that most of them are non-practical schemes for steganography in the blockchain. We have proposed two algorithms for steganography in blockchain, the first one is a high-capacity algorithm for the key and the steganography algorithm exchange and switching, and the second is a medium-capacity algorithm for embedding hidden data. The proposed method is a general method for steganography in each blockchain, and we investigate how it can be implemented in the two most popular blockchains, Bitcoin and Ethereum. Experimental result shows the efficiency and practicality of the proposed method in terms of execution time, latency, and steganography fee. Finally, we have explained the challenges of steganography in blockchain from the steganographers' and steganalyzers' points of view.

1 Introduction

In addition to cryptography, which is aimed to hide the exchanged information, steganography is created to hide the principle of the existence of communication between two persons [1]. In steganography, if the adversary even doubts the existence of communications (while he has not even been able to prove it), the steganographer has failed. In some steganography schemes, the message is encrypted and embedded to improve security, contrary to steganography's nature. On the other side, steganalysis is used to discover the existence of communication. Any media can be used for steganography, but media with a high degree of redundancy are more suitable [1]. For this reason, photo, audio, and video are often used for steganography [2–4].

Blockchain is a p2p network used for digital currency [5]. Due to its unique features, researchers in various fields have taken advantage of the Blockchain [6–11]. Bitcoin is the first and the most widely used digital currency. In Bitcoin, the distributed consensus achieved between miners ensures that the information sent to the blockchain remains unchanged and

* Corresponding author.
Email addresses: o.torki@eng.ui.ac.ir, m.ashouri@eng.ui.ac.ir, m.mahdavi@eng.ui.ac.ir
ISSN: 2008-2045

permanent. In Bitcoin, the sender signs the transaction and sends it to the blockchain. There are also different payment models in Bitcoin, the three most popular of which are payable to the public key (p2pk), pay to public key hash (p2pkh), and pay to script hash(p2sh). In p2pk, the sender deposits the money into the receiver's public key, and the receiver can receive it by signing his public key and sending it to the blockchain. In p2pkh, the sender deposits the money into the receiver's public key hash, and the receiver can receive the money by signing the public key and sending it to the blockchain. In p2sh, the sender and the receiver agree on a script and send its hash to the blockchain, and when they want to execute this script, they perform it by sending it to the blockchain. Each Bitcoin transaction contains one or more input and output addresses, meaning that a Bitcoin amount is collected from one or more input addresses in a transaction and credited to one or more output addresses.

Ethereum is the second most important digital currency. In Ethereum, unlike Bitcoin, which is based on transactions, the main component is the accounts [12]. Each account has a 20-byte address and a value that is the Ether balance of that account. There are two types of accounts in Ethereum: accounts that are managed by individuals and accounts that are managed by a code (smart contracts). Each transaction in Ethereum includes an input address (the address of an account) and an output address, which can be the address of another account managed by one individual or the address of a smart contract to invoke that contract code. Moreover, each transaction has two other values: gas limit and gas price. Every operation in the Ethereum network (either transactions or smart contracts) requires a certain amount of gas. The gas limit is equal to the minimum amount required to perform the transaction sent in the Ethereum network, while the gas price is the amount the user intends to pay for each gas unit. The most crucial concept that Ethereum introduced is smart contracts [13]. A smart contract is a code that can be executed by sending the required parameters and invoking its functions. The smart contract remains unchanged when sent to the Ethereum blockchain, and its execution is transparent. This has led to the advent of distributed apps with the help of smart contracts. In this paper, we introduce the advantages of blockchain for embedded hidden data(steganography) and propose two algorithms for this purpose: high-capacity and medium-capacity algorithms for data embedding. The unique feature of these algorithms is that they do not manually change the original data during the embedding process, which makes the steganalysis very difficult. In addition, we will show that there are no practical schemes for this purpose by analyzing previous steganography schemes in the blockchain.

The organization of this paper is as follows: Section 2 introduces the benefits of blockchain in steganography. In Section 3 we review the previous schemes for steganography in blockchain and analyze them. In Section 4 we review the HDW algorithm, which is a building block for our proposed algorithms. In Section 5 we propose two algorithms for steganography in the blockchain. In Section 6 we evaluate the proposed algorithms. Section 7 describes the open challenges of steganography in blockchain and finally Section 8 concludes the paper.

2 Blockchain As a Platform for Steganography

Since blockchain is a p2p network available from all over the world, it will be an excellent platform for steganography. Specifically, the blockchain has two major advantages over its previous platform for steganography, such as image and video. The first advantage is that for steganography in an image or video, we need to change parts of it to get the photo or video we want. However, in blockchain, obtaining transactions that, for example, the recipient's address contains the information we want is possible. To this end, by repeating and creating different addresses, we can achieve some addresses that contain the desired hidden message. An alternative way is to permutation addresses with no specific order and embeds hidden data in the permutation of output addresses of a transaction. Moreover, we can combine the two approaches to have more embedding capacity, and in both approaches, we have embedded our data while not making any manual changes. The second advantage is that in the old steganography methods, it is necessary to send an image and a video or a place to store it. The repeated sending of these files is self-doubtful, while in the blockchain, sending and receiving transactions are common due to their inherent nature. The sender and receiver of data do not need to design and implement a platform to send and store their data and use the ready-made blockchain platform.

3 Review and Analysis of Previous Steganography Schemes in Blockchain

In this section, we review and analyze the proposed schemes of Partala *et al.* [14], Xu *et al.* [15], Zhang *et al.* [16] and Gimenez-Aguilar [17].

In the Partala *et al.* scheme [14], the sender puts one bit of the data in the least significant bit (LSB) of the transaction address (the output address) and sends it to the blockchain. This process repeats until

embedding all bits of the data. To ensure the correct retrieval of information at the receiver side, the sender should wait for each transaction to be placed in the blockchain, and then he can send the next bit to the blockchain through the next transaction. So, this scheme needs one hour and twenty minutes to send a byte of information through the Bitcoin network. The receiver must also have the sender's addresses to detect information entered into the blockchain. Therefore, the sender needs to use a small number of addresses that the receiver is aware of, making tracking easier.

In the Xu *et al.* scheme [15], the sender must be a miner. In this scheme, the miner, by using a key, firstly selects several transactions of a block that he intends to create and publish in the blockchain; afterward, he uses the permutation of these transactions to embed the hidden data in the block. Nowadays, many miners are working together to create a pool, such as BTC or AntPool, so a single miner has no power in the blockchain and cannot produce new blocks. On the other hand, only a pool manager determines the order of block transactions. Due to the small number of pools, only a few pool managers in the world can use this method for steganography; however, since pool managers have a lot of capital and power, they do not need to use steganography, so this scheme is entirely impractical.

In the Zhang *et al.* scheme [16], the sender firstly encrypts his data and then encodes it by the base-58 encoding system to have a similar encoding as Bitcoin addresses. Then, he selects a number of its data characters and generates an address that contains these characters using the Vanitygen software. He repeats this process until he obtains several addresses containing hidden data. Then, using an indexing algorithm, it generates an index for the correct retrieval of information at the receiver side and encrypts and places it in the op-return command. The op-return is a command that prevents the miners' subsequent data from processing, similar to the comments in the programming languages. The first challenge in this scheme is that it does not explicitly state how the receiver detects the presence of new information in the blockchain. Most likely, the receiver should check and decrypt all transactions with an op-return command and see if there is new data. However, the second and most important challenge is the use of op-return command; op-return is a low-usage command and putting the encrypted data in it is highly questionable. Also, since the adversary knows the algorithm, the steganalysis is not very complicated.

Gimenez-Aguilar *et al.* [17] discovered various features in Ethereum that can embed hidden data. Specifically, they presented their schemes in three categories: embedding in addresses, embedding in transaction information, and embedding in smart contracts. Each feature has a different embedding capacity, but the maximum embedding capacity is 512 bits, related to embedding in the sender address. The proposed by Gimenez-Aguilar *et al.* can only be implemented in Ethereum. At the same time, our schemes are a general framework for blockchain steganography and can be implemented in all blockchains and digital currencies.

$$
\begin{aligned}
&\textit{private key generation info}: k, y\\
&\textit{address generation info}: k, g^{y}\\
&i-\textit{th private key}: x_i = y + H(k||i)\\
&i-\textit{th public key}: g^{x_i} = g^{H(k||i)}g^{y}\\
&i-\textit{th address}: H(g^{x_i})
\end{aligned}
$$

Figure 1. HDW algorithm [18]

4 Hierarchical Deterministic Wallets (HDW)

Hierarchical Deterministic Wallets (HDW) is a method for creating a public key, a private key, and a transaction address in Bitcoin and other digital currency. In HDW, instead of having a private key and an address, we have a piece of private key generation info, and address generation info that can be used to generate an unlimited number of private keys and addresses [18]. The advantage of this method is that it is possible to generate an unlimited number of private keys, public keys, and addresses, which, although generated for us regularly using an algorithm, are completely random and unrelated to those unaware of the key. This algorithm works as follows [18]: the private key generation info is k, y and the address generation info is k, g^y. The i-th private key is obtained as $x_i = y + H(k||i)$ and the i-th public key is obtained as $g^{x_i} = g^{H(k||i)}g^y$, where H is a hash function and "||" means concatenation. For the p2pk payment model, the i-th public key is sufficient, but for the p2pkh payment model, the i-th address is the hash of i-th public key. The algorithm is shown in Figure 1.

5 SSB: New Secure Algorithms for Steganography in Blockchain

In this section, we introduce two algorithms for steganography in the blockchain. The first algorithm is a high-capacity embedding algorithm that is used for key agreement, key exchanging, steganography algorithm agreement, and so on. In contrast, the

second algorithm is a medium-capacity embedding algorithm for hidden data embedding and transmission. The proposed methods are a general framework for steganography in the blockchain. To show their feasibility, we show how to implement the proposed algorithms in the two digital currencies, Bitcoin and Ethereum. The main idea of the proposed framework is to use HDW to exchange a secret message between sender and receiver and inform the receiver of the existence of a new message in the blockchain.

5.1 High-Capacity Embedding Algorithm

This algorithm has high embedding capacity and is used for key agreement, key exchange, steganography algorithm agreement, and similar cases that require more embedding capacity. In this algorithm, the user will lose some coins (for example, Bitcoin or Ether) that were entered in the transaction. In this algorithm, the sender encrypts his message with a symmetric encryption algorithm such as AES and sends it to the blockchain as an output address of the transaction (of course, before sending it to the blockchain, the sender should use random padding to match the bit length of the encrypted message with the output bit length of the output address of the transaction). In other words, the output address of the transaction sent to the blockchain is equal to the encrypted and padded message. Since the output of the symmetric encryption and the transaction and account addresses that usually the output of a hash function is both random and indistinguishable from each other, it is impossible for the adversary and people. They do not know the key to recognize the transaction that is not the hash function's output. The transaction input address must be created with the HDW algorithm to notify the receiver that a new message will be sent to the blockchain. In other words, by testing different values for i in the ordered manner and generating consecutive addresses using the HDW algorithm, the receiver can compare these values with the input addresses of blockchain transactions and will be notified of the existence of new data. Clearly, the sender loses the coin entered in the transaction in this algorithm and cannot use it in future transactions. It's because the transaction's output address is an encrypted message and doesn't relate to a valid private key. Algorithms 1 and 2 describe the high-capacity embedding algorithm in sender and receiver, respectively. In the following algorithms, $Trans(inaddress, b, outaddress)$ transfers b coin from $inaddress$ to $outaddress$.

Implementation of high-capacity embedding algorithm in Bitcoin and Ethereum: The main difference between Bitcoin and Ethereum is that Bitcoin uses unspent transaction output (UTXO) while Ethereum has accounts. This allows multiple output addresses per transaction in Bitcoin, while the output of each transaction in Ethereum is just the address of one account. Since we need one output address in a high-capacity embedding algorithm, implementing the high-capacity embedding algorithm is no different in Bitcoin and Ethereum. In other words, the sender must send the hidden message to the blockchain through the p2pkh or p2sh payment model in Bitcoin and account address in Ethereum.

Algorithm1: high-capacity embedding algorithm (sender)
Input:
hm /* hidden message
y, k, i /* parameters of HDW Alg
sk /* shared secret key between sender and receiver
Output:
$Trans(inaddress, b, outaddress)$ /* Transaction that should be sent to blockchain.
$C \leftarrow Enc_{sk}(hm)$ /* symmetric encryption
$C' \leftarrow RandPad(C)$ /* Random Padding
$inaddress \leftarrow HDW(y, k, i)$
$create\ Trans(inaddresss, b, C')$
$send\ Trans\ to\ Blockchain$

Table 1.

Algorithm2: high-capacity embedding algorithm (receiver)
Input:
y, k, i /* parameters of HDW Alg
sk /* shared secret key between sender and receiver
Output:
hm /* hidden message
$loop(.)$
$inaddress \leftarrow HDW(y, k, i)$
$check\ the\ transaction\ of\ new\ block$
$Trans(inaddress, b, C') \leftarrow find(inaddress)$
$C \leftarrow remove\ RandomPad(C')$
$hm \leftarrow Dec_{sk}(C)$ /* symmetric decryption

Table 2.

5.2 Medium-Capacity Embedding Algorithm

This algorithm has less capacity for embedding and is used for hidden data embedding. The details of this algorithm are as follows: the sender first decides in how many outputs he intends to embed his hidden data (how many output addresses per block) and how many bits he intends to embed in each address, then selects the desired bits of each address to embed the hidden data. Afterward, using the HDW algorithm

and testing different values for i, the sender obtains the desired address containing the embedded bits. To obtain the address that contains m bits of desired data, we need an average of 2^m effort(run HDW algorithm). Once all addresses have been created, more bits can be embedded by permuting these addresses. For this purpose, n addresses have $n!$ permutations, and we must assign a number from zero to $n!-1$ to each permutation. It is enough to have a solution to compare two permutations of addresses to achieve this goal. We use the same method as [15] for this purpose, so that if we have two permutations $(a_1, a_2, \ldots, a_n)$ and $(a'_1, a'_2, \ldots, a'_n)$, we say $(a_1, a_2, \ldots, a_n)$ is greater than $(a'_1, a'_2, \ldots, a'_n)$, if we have $i \in [1, n]$ that:

$$\begin{cases} a_j = a'_j & j < i \\ a_j > a'_j & j = i \end{cases}$$

For instance and without loss of generality, suppose that we have three addresses $a_1 = 110$, $a_2 = 010$, $a_3 = 101$. These addresses have six, 3!, permutation include $P_1 = \{010, 101, 110\}$, $P_2 = \{010, 110, 101\}$, $P_3 = \{101, 010, 110\}$, $P_4 = \{101, 110, 010\}$, $P_5 = \{110, 010, 101\}$ and $P_6 = \{110, 101, 010\}$. With respect to the following definition, we can easily see that $P_1 < P_2 < P_3 < P_4 < P_5 < P_6$, and we can code P_i as $i - 1$ (e.g., P_3 represents the value of 2). For example, if the sender wants to send "100" as the hidden data for the receiver, it is enough to send a transaction to the blockchain with output addresses as $\{110, 010, 101\}$ in order.

Suppose that sender generates n output addresses and embeds m bits in each address. Using this algorithm, the number of embedded bits in the addresses is equal to nm bits, the number of embedded bits arising from the permutation of the addresses is $\log_2 n!$, and the capacity of this algorithm is equal to $nm + \log_2 n!$ per block. In this algorithm, the coin entered in the transaction can be used in subsequent transactions, and the user, except for the transaction fee, does not lose money. It's because, unlike the high-capacity embedding algorithm, the output address of the transaction is achieved from the HDW algorithm and relates to a valid private key. Moreover, to inform the receiver of a new message, both the sender and receiver know the private key and can expend this coin. This coin can be expended for sending new hidden messages or usual transactions. It is worth noting that at the beginning of the data transmission between sender and receiver, the value of the variable i in the HDW algorithm is equal to one and increased in order. The receiver can run the HDW algorithm and test different values of i, starting with one, to notify the presence of new hidden data. Also, the sender and receiver can change the keys and restart the value of i to one at appropriate intervals through the high-capacity algorithm. Algorithms 3 and 4 describe the medium-capacity embedding algorithm in sender and receiver, respectively. In these algorithms, $index[]$ is a vector of size m, representing the locations of output addresses where hidden data must be embedded.

Implementation of medium-capacity embedding algorithm in Bitcoin and Ethereum: Since each Bitcoin transaction can have multiple output addresses, the implementation of this algorithm in Bitcoin is straightforward. In other words, the sender can put n output addresses in a transaction and apply the desired permutation to the order of these output addresses. However, the main challenge of implementing this algorithm in Ethereum is that in each transaction, there is only one account address as the output address, and by sending several transactions to a block, the order of these transactions (i.e., the order of output addresses) determined by the miners. Therefore, the possibility of applying the desired permutation and embedding the $\log n!$ bit by the permutation of these transactions is taken from the sender. To meet this challenge, we offer two solutions, one based on transactions and another smart contract.

- Solution based on transactions: In this method, the sender implicitly uses the Ether value to inform the receiver about the order of transactions in the block. In other words, suppose that in (a_i, e_i), a_i is the transaction output address (generated using the HDW algorithm), and e_i is the Ether value deposited in the a_i account. If the permutation desired by sender is $\{a_1, a_2, \ldots, a_n\}$, he must send the transactions $(a_1, e_1), (a_2, e_2), \ldots, (a_n, e_n)$, to the block such that $e_1 < e_2 < \ldots < e_n$.
- Smart contract solution: In this solution, the sender sends a smart contract to the blockchain. The content of the code of this contract does not matter, but this contract must get n numbers as input in each invocation. The contract's content can be different, including that these n numbers are n account addresses and deposit an amount to the account of these addresses upon invoking, or take n numbers as input and calculate their sum or average of them. Another critical point is that when deploying a contract in blockchain, this must be done through an account whose address is generated by the HDW algorithm to inform the receiver of the existence of such a contract. After the appropriate interval, for more security, the sender and receiver can use another contract.

6 Evaluation

In [1, 19], four criteria for evaluating steganographic algorithms are introduced, and we evaluate our algo-

Algorithm3: medium-capacity embedding algorithm (sender)

Input:

$hm \leftarrow hm_1||hm_2$ /* hidden message: hm_1: mn bits, represent as $n * m$ matrix, hm_2 $\log n!$ bits

$index[]$ /* vector of size m

y, k, i /* parameters of HDW Alg

Output:

$Trans(inaddress, b, outaddress)$ /* Transaction that should be sent to blockchain.

$j = 1$

repeat n times

loop(.)

$outaddress_j \leftarrow HDW(y, k, i)$

if for all $m : hm_1[j, m] == outaddress_j[index[m]]$

break

$i \leftarrow i + 1$

end loop

$j = j + 1$

select appropriate permutation of $\{outaddress_j\}_{j=1}^{j=n}$ *such that value of* $\{outaddress_j\} = hm_2$:

$\{P_l\}_{l=0}^{\log n!} \leftarrow$ *all permutations of* $\{outaddress_j\}_{j=1}^{j=n}$ *in ascending order*

$v \leftarrow Binary2Decimal(hm_2)$

set P_v *as an appropriate permutation*

create $Trans(inaddress, \{b_j\}_{j=1}^{j=n}, \{outaddress_j\}_{j=1}^{j=n})$

send Trans to Blockchain

Table 3.

rithm with these criteria.

1) Visibility: This means that the information that contains hidden data is indistinguishable from the information without hidden data by the eye. In our scheme, this feature is maintained due to the use of the HDW algorithm. In addition, in our algorithm, none of the address bits are changed manually; instead, by repeating the HDW algorithm, we reach the desired address, so the address containing hidden data is not different from the address without hidden data, even with statistical analysis it is indistinguishable.
2) Robustness: This means that the embedded information is not lost due to accidental or intentional changes and can be retrieved. Due to the distributed consensus protocol and the signature on the transactions in the proposed algorithms, the data remains permanently unchanged in the blockchain.
3) Security: No one is aware of hidden data. As stated in the proposed algorithms, the data is not embedded manually, and the address containing the hidden data is indistinguishable from the address without hidden data.
4) Capacity: The maximum data that can be embedded. As stated in Section 5, the capacity of the high-capacity algorithm is equal to the output bit length of the hash function, and the capacity of the medium-capacity algorithm is equal to $nm + \log_2 n!$ bits.

The capacity of embedding due to permutation of output addresses is shown in Figure 2. As shown in Figure 2, the embedding capacity increases exponentially with increasing the number of output addresses, (n). Based on selecting 30 random transactions from 10 randomly chosen blocks from [20], the number of output addresses of each transaction has an average of 3.45 and a standard deviation of 1.2. However, five output addresses are common in each transaction. Also, in the review of 30 transactions, 12 transactions had five output addresses. There are also transactions in the blockchain with even more than 30 output addresses, and the number of output addresses of transactions can be different at different times. One of the reasons is the difference in the transaction fee at different times. The time of embedding the hidden data in the output address of the transactions by using the HDW algorithm (obtaining the desired address) in terms of the number of bits that we intend to embed m is shown in Figure 3. Implementation performed

Algorithm4: medium-capacity embedding algorithm (receiver)

Input:

$hm \leftarrow hm_1||hm_2$ /* hm_1: $n * m$ empty matrix, hm_2 $\log n!$ empty bit

$index[]$ /* vector of size m

y, k, i /* parameters of HDW Alg

Output: hm /* hidden message

$loop(.)$

$outaddress \leftarrow HDW(k, y, i)$

$check\ the\ transactions\ of\ new\ block$

$Trans(inaddress, b_i, \{outaddress_i\} \bigcup outaddress) \leftarrow find(outaddress)$

$if\ find(outaddress) = True:$

$break$

$i = i + 1$

$j = 1$

$repeat\ n\ times$

$hm_1[j, l] == outaddress_j[index[l]],\ for\ l = 1\ to\ m$

$hm_2 \leftarrow value\ of\ permutation of\ \{outaddress_j\}_{j=1}^{j=n}:$

$\{P_l\}_{l=0}^{\log n!} \leftarrow all\ permutations\ of\ \{outaddress_j\}_{j=1}^{j=n}\ in\ ascending\ order$

$if\ P_v == \{outaddress_j\}_{j=1}^{j=n}:$

$hm_2 \leftarrow Decimal2Binary(v)$

$hm \leftarrow hm_1||hm_2$

Table 4.

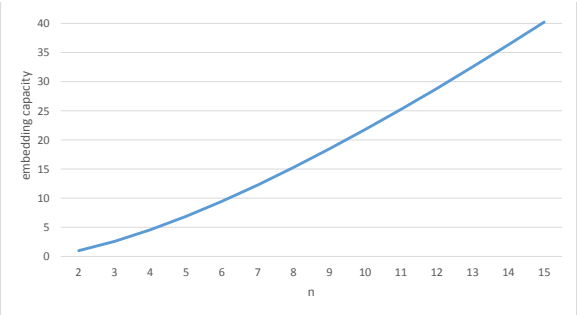

Figure 2. Embedding capacity based on permutation of output addresses

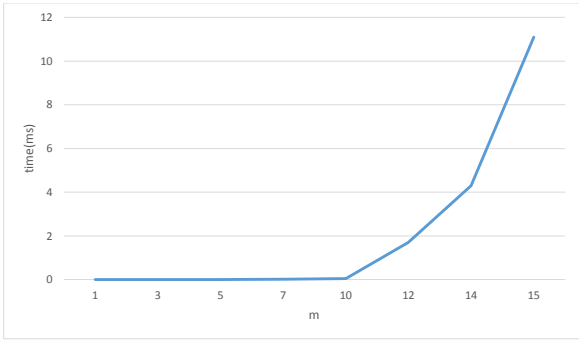

Figure 3. Execution time of embedding m bits in output address by using of HDW

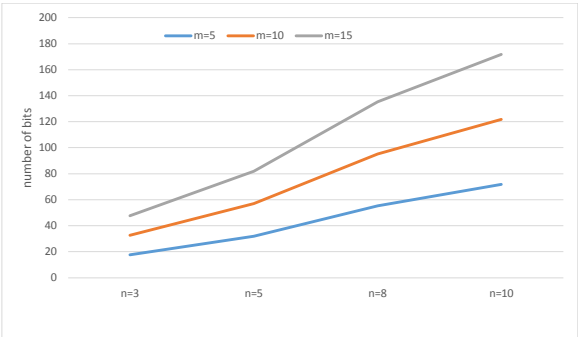

Figure 4. Capacity of medium-capacity algorithm

on an Intel (R) Core (TM) i5-6200U processor with 8GB memory, running Windows 7 and Python programming language. The number of bits that can be embedded in a transaction(capacity of the medium-capacity algorithm) for different values of the number of bits embedded in each transaction, m, and the number of output addresses of each transaction, n, is shown in Figure 4. Therefore, 81.9 bits can be embedded in a transaction with five output addresses and 15 embed bits per address for security reasons.

A comparison of the proposed schemes and state of the art is shown in Table 5. The latency and fee criteria are given for embedding 250 bits hidden data,

	latency	fee	capacity	security	generality
Partala [14]	BTC: 208.3 h ETH: 4.5 h	BTC: $1100 ETH: $2358	1	secure	general
Xu [15]	BTC: 10 min ETH: 13 sec	$125 M	$\log N!$	secure	general
Zhang [16]	50 min	$4.4	80	non-secure	only Bitcoin
Gimenez-Aguilar [17]	65 sec	$9.4	512	secure	only Ethereum
Ours	BTC: 50 min ETH: 65 sec	BTC-HC: $8.8 BTC-MC: $13.2 ETH-HC: $18.8 ETH-MC-transaction: $141 ETH-MC-smart contract: $51.4	81.9	secure	general

Table 5. Comparison of proposed scheme and state of the art schemes. BTC: Bitcoin, ETH: Ethereum, HC: high-capacity embedding algorithm, MC: medium-capacity embedding algorithm N: number of transactions in a block

and the capacity is maximum embedding capacity per block. The latency criterion is depend on the blockchain that is used, therefore we consider 50 minutes for Bitcoin and 65 seconds for Ethereum (5 block latency to accept publishing transaction in a block). The average transaction fee in Ethereum (based on 10 random-selected transactions and 10 smart contracts [21]) is $9.4 per transaction and $51.4 for invocation of a smart contract. Moreover, the average transaction fee for Bitcoin transaction is $4.4 [20]. As it is shown in Table 5, the Partala [14] scheme in terms of latency and fee, and the Xu [15] scheme in term of fee are impractical. For the Xu scheme, the value of fee is calculated based on the least current hash-rate of a mining pool (i.e. 3 EH/s) and fee of a miner device [20] and based on the fact that the sender must be a mining pool (i.e. pool manager). As mentioned in Section 3, the Zhang scheme [16] is not secure, because of using encrypted data in op-return command. The schemes of Zhang and Gimenez-Aguilar [17] are blockchain-specific methods and Gimenez-Aguilar's scheme is subjected to statistical analysis, because of manual change in Ethereum transaction fields. In overall, our proposed scheme, is practical in terms of latency and fee, and secure and general-propose to implement steganography in each blockchain.

7 Open Challenges

As mentioned earlier, blockchain is a very suitable platform with unique features for steganography. However, this field is just beginning, and many challenges can be addressed in various ways. More specifically, there are two major challenges, one for steganographers and one for steganalyzers:

– Finding blockchain features (in all digital currencies, not just Bitcoin) that can embed data and provide high-capacity algorithms is an open challenge for steganographers, especially embedding algorithms, as stated earlier, should be able to embed the original data without manual change and to repeat until the desired data is reached.
– Finding methods to discover steganography in blockchain will be an open challenge for steganalyzers, especially since the new steganography algorithms embed the data (like the algorithm presented in this paper) without manually changing the original data and only by repeating the process until reaching the data they want.

8 Conclusion

In this paper, we describe the advantages of blockchain for steganography and prove that blockchain can serve as a perfect steganography platform. More significantly, the first one is that the steganography can be carried out in blockchain without altering the original data and can only be done by repeating the HDW algorithm until the data has been embedded. The second one is that blockchain is a ready-made platform for data transmission and storage, and Steganographer does not need to design a new platform for data transmission and storage. We propose two general algorithms for steganography in blockchain, the first is a high-capacity algorithm for exchanging keys and algorithms, and the second is a medium-capacity algorithm for embedding hidden data. The proposed method is a general method for steganography in the blockchain. To show the feasibility of the proposed method, we investigate how this method can be implemented in Bitcoin and Ethereum. Evaluating results show that the proposed method is efficient and practical in terms of execution time, latency, and embedding fee. Finally, the challenges posed to steganography in blockchain for steganographers and steganalyzes are raised.

References

[1] Tayana Morkel, Jan HP Eloff, and Martin S Olivier. An overview of image steganography. In *ISSA*, volume 1, 2005.

[2] Eko Hari Rachmawanto, Christy Atika Sari, et al. Secure image steganography algorithm based on dct with otp encryption. *Journal of Applied Intelligent System*, 2(1):1–11, 2017.

[3] Said E El-Khamy, Noha O Korany, and Marwa H El-Sherif. A security enhanced robust audio steganography algorithm for image hiding using sample comparison in discrete wavelet transform domain and rsa encryption. *Multimedia Tools and Applications*, 76(22):24091–24106, 2017.

[4] Ramadhan J Mstafa and Khaled M Elleithy. A video steganography algorithm based on kanade-lucas-tomasi tracking algorithm and error correcting codes. *Multimedia Tools and Applications*, 75(17):10311–10333, 2016.

[5] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. bitcoin. org. *URL: https://bitcoin. org/bitcoin. pdf (accessed: 24.02. 2020)*, 2008.

[6] Seyoung Huh, Sangrae Cho, and Soohyung Kim. Managing iot devices using blockchain platform. In *2017 19th international conference on advanced communication technology (ICACT)*, pages 464–467. IEEE, 2017.

[7] Asaph Azaria, Ariel Ekblaw, Thiago Vieira, and Andrew Lippman. Medrec: Using blockchain for medical data access and permission management. In *2016 2nd International Conference on Open and Big Data (OBD)*, pages 25–30. IEEE, 2016.

[8] Damiano Di Francesco Maesa, Paolo Mori, and Laura Ricci. Blockchain based access control. In *IFIP international conference on distributed applications and interoperable systems*, pages 206–220. Springer, 2017.

[9] Lanxiang Chen, Wai-Kong Lee, Chin-Chen Chang, Kim-Kwang Raymond Choo, and Nan Zhang. Blockchain based searchable encryption for electronic health record sharing. *Future Generation Computer Systems*, 95:420–429, 2019.

[10] Yang Xu, Guojun Wang, Jidian Yang, Ju Ren, Yaoxue Zhang, and Cheng Zhang. Towards secure network computing services for lightweight clients using blockchain. *Wireless Communications and Mobile Computing*, 2018, 2018.

[11] Patrick McCorry, Siamak F Shahandashti, and Feng Hao. A smart contract for boardroom voting with maximum voter privacy. In *International Conference on Financial Cryptography and Data Security*, pages 357–375. Springer, 2017.

[12] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

[13] Huashan Chen, Marcus Pendleton, Laurent Njilla, and Shouhuai Xu. A survey on ethereum systems security: Vulnerabilities, attacks, and defenses. *ACM Computing Surveys (CSUR)*, 53(3):1–43, 2020.

[14] Juha Partala. Provably secure covert communication on blockchain. *Cryptography*, 2(3):18, 2018.

[15] Mengtian Xu, Hanzhou Wu, Guorui Feng, Xinpeng Zhang, and Feng Ding. Broadcasting steganography in the blockchain. In *International Workshop on Digital Watermarking*, pages 256–267. Springer, 2019.

[16] Lejun Zhang, Zhijie Zhang, Weizheng Wang, Rasheed Waqas, Chunhui Zhao, Seokhoon Kim, and Huiling Chen. A covert communication method using special bitcoin addresses generated by vanitygen. *Computers, Materials & Continua*, 65(1):597–616, 2020.

[17] Mar Gimenez-Aguilar, Jose M De Fuentes, Lorena González-Manzano, and Carmen Camara. Zephyrus: An information hiding mechanism leveraging ethereum data fields. *IEEE Access*, 9:118553–118570, 2021.

[18] Arvind Narayanan, Andrew Miller, Steven Goldfeder Joseph Bonneau, and Edward Felten. Bitcoin and cryptocurrency technologies. 2021.

[19] Fabien AP Petitcolas, Ross J Anderson, and Markus G Kuhn. Information hiding-a survey. *Proceedings of the IEEE*, 87(7):1062–1078, 1999.

[20] *BTC.com vebsite*. https://btc.com, Desember 2020.

[21] *EtherScan vebsite*. https://etherscan.io/, February 2022.

Omid Torki received his B.Sc. degree in Computer Engineering from the University of Isfahan (Shahreza Campus), in 2016 and M.Sc. degree in Information Technology Engineering (Information Security) from University of Isfahan, in 2018. He is currently a Ph.D. candidate in Computer Engineering (Information Security) at University of Isfahan. During the 18th International ISC Conference on Information Security and Cryptology (ISCISC2021) and for the first time in Iran, he designed and implemented an event to teach 15-18 years old students the concepts of information security. His research interests are secure multiparty computation, outsourcing, blockchain and smart contract, security protocols and steganography.

Maede Ashouri-Talouki is an Assistant Professor in the IT Engineering department of the University of Isfahan (UI). She received her B.Sc., M.Sc., and Ph.D. degrees from the University of Isfahan in 2004, 2007, and 2012, respectively. In 2013, she joined the University of Isfahan. Her research interests include mobile networks security, user privacy and anonymity, cryptographic protocols, and network security.

Mojtaba Mahdavi received the B.Sc., M.Sc., and Ph.D. degree in computer engineering from Isfahan University of Technology, Isfahan, Iran, respectively, in 1999, 2002, and 2011. His research interests are network security, data hiding (steganography and watermarking), and wireless networks. He is currently with the department of information technology engineering at the University Isfahan, Isfahan, Iran.