A Covert Communication Method Using Special Bitcoin Addresses Generated by Vanitygen

Lejun Zhang[1, 2], Zhijie Zhang[1], Weizheng Wang[3], Rasheed Waqas[1], Chunhui Zhao[1, 4], Seokhoon Kim[5] and Huiling Chen[6, *]

Abstract: As an extension of the traditional encryption technology, information hiding has been increasingly used in the fields of communication and network media, and the covert communication technology has gradually developed. The blockchain technology that has emerged in recent years has the characteristics of decentralization and tamper resistance, which can effectively alleviate the disadvantages and problems of traditional covert communication. However, its combination with covert communication thus far has been mostly at the theoretical level. The BLOCCE method, as an early result of the combination of blockchain and covert communication technology, has the problems of low information embedding efficiency, the use of too many Bitcoin addresses, low communication efficiency, and high costs. The present research improved on this method, designed the V-BLOCCE which uses base58 to encrypt the plaintext and reuses the addresses generated by Vanitygen multiple times to embed information. This greatly improves the efficiency of information embedding and decreases the number of Bitcoin addresses used. Under the premise of ensuring the order, the Bitcoin transaction OP_RETURN field is used to store the information required to restore the plaintext and the transactions are issued at the same time to improve the information transmission efficiency. Thus, a more efficient and feasible method for the application of covert communication on the blockchain is proposed. In addition, this paper also provides a more feasible scheme and theoretical support for covert communication in blockchain.

Keywords: Covert communication, blockchain, Bitcoin address.

[1] College of Information Engineering, Yangzhou University, Yangzhou, 225127, China.

[2] School Math & Computer Science, Quanzhou Normal University, Quanzhou, 362000, China.

[3] University of Aizu, Aizu Wakamatsu, 9658580, Japan.

[4] College of Information and Communication Engineering, Harbin Engineering University, Harbin, 150001, China.

[5] Department of Computer Software Engineering, Soonchunhyang University, Asan, Korea.

[6] Department of Computer Science and Artificial Intelligence, Wenzhou University, Wenzhou, 325035, China.

[*] Corresponding Author: Huiling Chen. Email: chenhuiling.jlu@gmail.com.

Received: 16 May 2020; Accepted: 29 May 2020.

1 Introduction

Information theft and interception technologies have developed significantly since the 21st century due to the development of the communications industry, thereby increasing threats to users' data and privacy. At present, information encryption technology [Pérez-Cabré, Abril, Millán et al. (2011); Zou, Li and Su (2015); Chen, Lu and Hsu (2015)] and information hiding technology [Gobbo and Benini (2013); Kaneda, Hirano, Iwamura et al. (2008); Ono (2012); Bash, Goeckel, Guha et al. (2015)] are the main methods used to effectively address the issue of information theft. In contrast to traditional encryption technology, which focuses on the information itself and its readability, covert communication is targeted more at the existence of information. It uses the redundancy of the information to embed the processed message into communication and media information for transmission. [Huang, Xiao and Xiao (2008)] Traditional covert communication methods can greatly reduce the existence of the information, increase the imperceptibility of information, and decrease the degree of change in data information. However, the channels are easily detected and interfered with, the participants are easily exposed, and group transmission is difficult.

Blockchain [Nakamoto (2019)] is a new technology gradually developed in recent years, which breaks the mainstream mechanism of relying on third parties for information exchange or trade transfers. It changes the traditional service applications [Xue, Han, Wang et al. (2016)] and solutions for solving complex optimization problems [Chen, Heidari, Zhao et al. (2020)]. The production of bitcoin and its mining machine has also impacted the traditional social manufacturing industry and its related models and frameworks [Xue, Wang, Zhang et al. (2018)]. Blockchain has the characteristics of decentralization, tamper resistance, and anonymization, among others. These characteristics give the application of blockchain in covert communication the advantages of difficulty of monitoring, anonymization of communication receivers, high tamper resistance, and anti-interference. In addition, the blockchain is an open platform in which each node can obtain data information in the block from the chain. Some new technologies, such as the microgrid selection optimization method [Li, Li, Cao et al. (2018)], can use the alliance chain for secure storage and transmission. In Fig. 1, the message sender adds information to the chain, and multiple receivers in the group can filter the information from the data of the chain based on the transaction ID or other conditions and then achieve the group transmission.

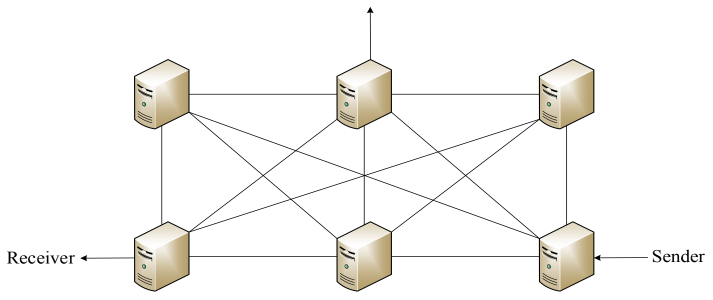

Figure 1: Group covert communication

However, the combination of blockchain and covert communication has thus far been rare and has been limited mostly to the theoretical level. There are no practical systems or methods that can be applied. Designing a feasible and efficient blockchain-based covert communication method is urgent and significant. The present research improved on BLOCCE, a method of hidden communication based on blockchain, and designed a more efficient and practical technique called V-BLOCCE. This paper aims to provide a more feasible scheme and theoretical support for covert communication in blockchain.

2 Related work

Covert communication can be traced back to the steganography proposed in "Steganographia," a work published in the 16th century. The technology gradually developed with the emergence of the Internet at the end of the 20th century and has been widely used in the fields of digital communication and network security. In most encoding methods, error correction codes are frequently used to improve the error detection ability and reliability. Due to their ubiquitous redundancy, error correction codes are often used as carriers for storing secret information in shortwave and satellite communications. These can be combined with integer wavelet transform to reduce its own robustness and effectively decrease the error of the information hiding algorithm. [Wang and Ma (2007)] In addition to encoding, images [Luo, Qin, Xiang et al. (2020); Peng, Lin, Zhang et al. (2019); Kang, Liu, Yang et al. (2019)] and videos [Long, Peng and Li (2018)] are often used for encryption and information hiding. Coverless information hiding can effectively resist steganalysis algorithm. Cao et al. [Cao, Zhou, Sun et al. (2018)] have found that it has better security and information capacity when combined with molecular structure images of materials. Digital watermarking [Cox, Miller and Bloom (2002)] is also an effective method often used to hide marks and protect copyrights. It is often enhanced in combination with encryption [Babar, Kendule and Shinde (2018)] and applied in areas such as IP protection and deep learning [Rouhani, Chen and Koushanfar (2018)].

Blockchain technology has been developed in recent years. It is widely used in the design of some mechanisms, such as incentive mechanism of data storage [Ren, Liu, Ji et al. (2018)] and fair trading mechanism of surplus power [Xia, Tan, Wang et al. (2019)]. In addition, it is also used to design some frameworks, such as a blockchain enabled distributed security framework using edge-cloud and software defined networking (SDN) [Medhane, Sangaiah, Hossain et al. (2020)].

There are several methods of covert communication; however, there are few applications and achievements combined with blockchain, and most of them only stay on the argument. Exploring data insertion methods on the blockchain, some researchers analyzed the covert communication on the blockchain but also pointed out the threat of illegal content insertion on the chain [Matzutt, Hiller, Henze et al. (2018)]. Theoretically, there are two methods to achieve covert communication through blockchain: by using time attributes, such as the transaction time interval, and by using storage attributes, such as a Bitcoin address and its hash value. BLOCCE [Partala (2018)] is a relatively complete and feasible system proposed by Juha Partala, who simultaneously put forward a complete model of hidden communication under the blockchain that uses the least significant bit of the public key address to store information, indicating the direction for related research. However, whereas the method contributes mainly by providing some theoretical support, many problems in its actual implementation remain. In the present work, a more effective covert communication method based on the BLOCCE system is proposed, called V-BLOCCE, and some existing problems are addressed.

3 Design foundation of V-BLOCCE

3.1 BLOCCE, a covert communication method based on the Bitcoin public key address

The BLOCCE system pioneers the use of Bitcoin addresses for information storage. Fig. 2 shows the system flow, with the main process as follows:

1) Asymmetric encryption and other processing are carried out on the plaintext to ensure the security of the information.
2) The encrypted information is converted into binary form to combine the Bitcoin public key address.
3) The public key is generated in a loop to find the address with the least significant bit that meets the requirements, which is then used to store one bit of the binary string obtained in the previous step. This process is repeated until all the messages are stored.
4) After the information is stored, some methods, such as identifier setting, are used to mark the beginning of covert communication.
5) The addresses are used to carry out transactions, and the generation of a new block is awaited to initiate the next transaction. Each block carries out one transaction to ensure the address sequence.
6) The receiver queries the transaction information sent by the transaction initiator and reads the identifiers to obtain the addresses of the transaction in order.
7) The least significant bit of the address is extracted and integrated into a binary string.
8) The binary information is inverse transformed and asymmetrically decrypted to obtain the original information.

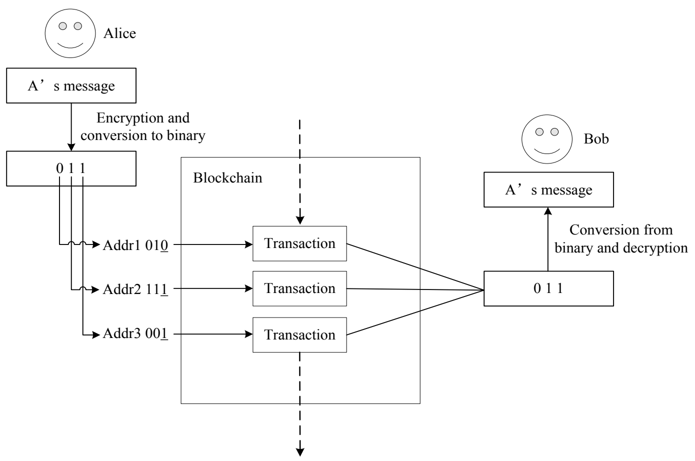

Figure 2: System process of BLOCCE

In the above image, the processed message that the sender Alice wants to send is “011.” The BLOCCE method first divides the binary information into 3 bits, which are “0,” “1,” and “1.” Then the public key is cyclically generated, and the “Addr1” whose least significant bit is “0” is found to store the first bit of binary information. This information will be recorded in the blockchain after the transaction is completed. Next, the subsequent addresses “Addr2” and “Addr3” are generated in the same way to store the last two digits of information. Then the system uses these two addresses to initiate two transactions, which are recorded in different blocks that have been generated successively to ensure order. The receiver Bob obtains this transaction information by constantly searching for transactions issued by Alice in the blockchain, extracts the least significant bits of these transaction addresses to restore the binary string, and finally carries out reverse processing to restore the original information.

The BLOCCE method cleverly distributes hidden information by using the least significant bits of the address for distributed storage. Combining the characteristics of the blockchain itself can prevent information from being tampered with maliciously and makes the channel invulnerable to attacks. In addition, the public key is specified by a unique private key. The use of public keys to store information also guarantees the authenticity of the sender of transactions and communications and prevents the appearance of other people impersonating the sender.

However, there are still some problems with this method:

1) The information embedding rate is very low because only one binary number, that is, one bit of information, is saved for each address, and other resources in the address are wasted.
2) Too many addresses are used. Due to the low embedding rate of information, several addresses are needed to store the complete information, and the number of addresses increases linearly with the number of binary characters.
3) The system efficiency is low. When there is a lot of information to be transmitted, the method needs to continuously generate many addresses to store the information. Regardless of the transaction confirmation, just the embedding of information takes a lot of time, which makes the system inefficient.
4) The communication efficiency is low. Each time the BLOCCE method generates an address, the transaction is carried out and recorded in the block after storing 1 bit of binary information. Moreover, after a transaction is completed, the system has to wait for the generation of a new block before initiating the next transaction. The entire communication process has low efficiency in transferring all information, and the use of multiple transactions to transfer information is not suitable for networks with slow block generation speeds.
5) The system entails high costs. Although Alice controls the key pair generation such that she can transfer bitcoins to herself without losing them, the use of a large number of transactions for hidden transmission still requires a lot of miner fees. Therefore, this system is not applicable under certain models, such as proof of work.

3.2 Bitcoin address generation software Vanitygen

Vanitygen is a Bitcoin address generation software written in C that supports regular Bitcoin addresses. The software is more than an order of magnitude faster than the official vanity address patch of the Bitcoin client Vanitygen can accurately search for addresses with specific prefixes, suffixes, or regular matches. The basic principle of Vanitygen is to continuously generate addresses randomly to obtain the address that meets the requirements. Vanitygen implements address generation through commands and can use different parameters to achieve different functions.

It is worth noting that, as previously mentioned, Bitcoin addresses use base58 encoding, with the capital "O," capital "I," lowercase "l," and the number "0" not appearing in the address [Nakamoto (2019)]. If the specified string contains the above characters, the software will continue to run but will not generate an address that meets the requirements. The proposed method uses this software to generate better addresses for storing information, with the goal of improving the efficiency of information embedding and decreasing the number of addresses used.

3.3 OP_RETURN script

OP_RETURN [Bartoletti and Pompianu (2017)] is a script operation code used to carry additional information in Bitcoin transactions, similarly to the remark information in daily transfers. Once the content is recorded in the chain with the transaction, it cannot be changed or deleted. OP_RETURN has been widely used in recent years. For example, the Omni Layer platform used to issue coins relies heavily on it. The monthly usage of OP_RETURN continues to increase, even reaching millions of times. The length of this field in Bitcoin is 80 bytes, and Bitcoin Cash (BCH) extends it to 220 bytes. When transmitting, the remarked information needs to be converted into a hexadecimal, and the string "0x" has to be added in front. V-BLOCCE uses this field to store the address order and index information and then sends it along with the transaction. After receiving this information, the receiver can obtain the correct address sequence and index information to ensure that each index corresponds to the correct address during the information restoration process so that the cipher text can be correctly restored.

4 The covert communication method V-BLOCCE using special Bitcoin addresses generated by Vanitygen

4.1 System model of V-BLOCCE

The technique proposed in this work is a modification of the BLOCCE system. Because the original method converts each character in the ciphertext into an 8 bits binary number and embeds only one bit of binary information in the least significant bit of the public key address, the embedding rate of the information is 1 bit per address. Eight addresses are used to effectively transmit one byte of information, and the efficiency of information transmission and embedding is extremely low. V-BLOCCE changes the fundamental method of information embedding, cancels the binarization of ciphertext, and encodes it by base58. Through such encoding, the ciphertext can have the same encoding dictionary as the Bitcoin address. This enables the proposed method to directly embed characters in bytes, which increases the size of the transmission unit and improves the embedding efficiency. Eq. (1) can be used to calculate the amount of information, where $p(x)$ is the probability of x appearing in the dictionary:

$$h(x)=-\log_2(p(x)) \tag{1}$$

In the character string encoded by base58, each character is selected from 58 characters so the amount of information represented by each character is $\log_2 58$ bits. Even if only one character is stored in each address by using V-BLOCCE, the amount of embedded information is still $\log_2 58$ times that of the original method. In addition, the proposed method applies circular embedding to repeatedly use the character information in a Bitcoin address; hence, the embedding efficiency can be greatly improved.

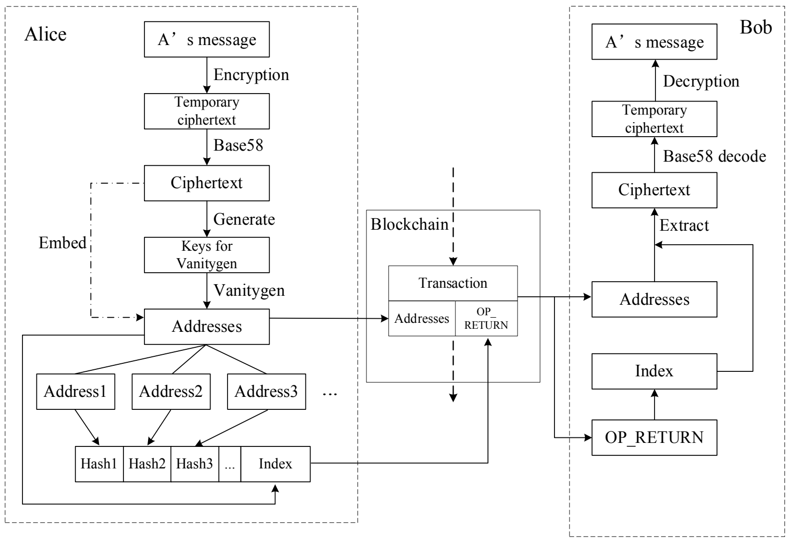

Figure 3: The system model of the V-BLOCCE method

Fig. 3 shows the overall model of the system. Alice initially encrypts the information to generate a temporary ciphertext, which is then base58 encoded to obtain the final ciphertext. Next, the system identifies several different characters in the ciphertext (the specific number is given in later experiments). These characters are stitched into a string, which is used as a key required by the Vanitygen software to generate a Bitcoin address. Then the system iterates through these addresses to match the characters appearing in the ciphertext. When a successful match is obtained, the characters are used to store the ciphertext, and the indexes are recorded.

After the information embedding, the index list of the message and the address can be generated. Then the address sequence information arranged in the form of the address hash value and the index information are encrypted. The result is filled into the OP_RETURN field to be carried out. Because the address sequence is generated based on the content of the address, there is no requirement as to which addresses are used in each transaction. If the length of information is short and a field can put down the information completely, then all transactions are integrated into a one-to-many transaction, and all the information on the address sequence and the index are sent together. However, if the number of OP_RETURN needed is more than the actual number of addresses used, multiple transactions are made to the same address to ensure that each transaction is valid; an address is used only once when the receiver restores the information. The sender can also use some irrelevant addresses for transactions, whereas the receiver determines the addresses and the order used according to the hash value in the OP_RETURN field, filtering out irrelevant addresses at the same time.

When multiple transactions are used to transfer information in the BLOCCE method, the system must wait for a new block to be generated before proceeding to the next transaction to sort the information by timestamp and ensure the order of the transferred information. The proposed method greatly reduces the efficiency of transactions and the timeliness of information transmission. In the proposed method, these transactions are done simultaneously to improve the efficiency of information transmission. The premise is that the order of the address sequence and index information stored in each OP_RETURN must be guaranteed; otherwise, it would be difficult for the receiver to recover the address sequence information and index information completely and correctly.

After the transaction is completed, the receiver Bob obtains the transaction information, including the Bitcoin address used, by querying the transaction sent by Alice. Next, he obtains the order of the addresses and the index information by the content in OP_RETURN. Bob can then combine the indexes and addresses to extract the ciphertext, which is decoded by base58 and decrypted by the previous encryption algorithm to obtain the plaintext information.

Based on the system model, an address sequence information generation algorithm is applied to represent the order in which addresses are used when hash collisions are allowed. Then the OP_RETURN fields order guarantee algorithm is designed for the segmentation and generation of OP_RETURN fields.

4.2 Address sequence information generation algorithm

When the indexing of information and addresses is generated, the order of the addresses needs to be guaranteed to ensure that the receiver can correctly restore the information. Here, the hash value of each address is calculated and sorted instead of the address. To decrease the correlation between the address and the hash value, only the part of the address of length *x* is used to calculate its hash.

Algorithm name: Address sequence information generation algorithm

Input: The address set consisting of *n* addresses used *Addr*=$\{a_1, a_2,\dots, a_n\}$ and the length of the calculated part in the hash calculation rule *x*

Output: Array containing the address sequence information *HA*

```
1) int $s$=1;                         //Parameter for cyclic control
2) char $HA$=[];                      //Stores the address order
3) for ($s$=1; $s$ <=$n$; $s$++) {    //$n$ is the number of addresses
4)        char $H_s$[]="";            // Stores the hash values of the address currently in use
5)        int $L_s$=0;
```

```
6)  Calculate the length of address $a_s$ as $L_s$;
7)  Calculate the hash value of $a_{s_{L_s-x}}$ to $a_{s_{L_s-1}}$ as $H_s$;
8)  Concatenate $H_s$ and "/";// Indicating the hash value of $L_s$-x to $L_s$-1 bytes
9)  for (i=1; i<s; i++) {
10)     if ($H_s$==$H_i$){
11)         Calculate the hash value of $a_{s_{L_s-2x}}$ to $a_{s_{L_s-x-1}}$ as $H_s$;
12)         Calculate the hash value of $a_{i_{L_i-2x}}$ to $a_{i_{L_i-x-1}}$ as $H_i$;
13)         Concatenate $H_s$ and "//"; concatenate $H_i$ and "//";
14)         for (k=1; k<s; k++) {
15)             if($H_s$ == $H_k$){
16)             Calculate the hash value of $a_{s_{L_s-3x}}$ to $a_{s_{L_s-2x-1}}$ as $H_s$;
17)             Calculate the hash value of $a_{k_{L_k-3x}}$ to $a_{k_{L_k-2x-1}}$ as $H_k$;
18)             Concatenate $H_s$ and "///"; concatenate $H_k$ and "///";}
19)             if($H_i$==$H_k$){
20)             Calculate the hash value of $a_{i_{L_i-3x}}$ to $a_{i_{L_i-2x-1}}$ as $H_i$;
21)             Calculate the hash value of $a_{k_{L_k-3x}}$ to $a_{k_{L_k-2x-1}}$ as $H_k$;
22)             Concatenate $H_i$ and "///"; concatenate $H_k$ and "///";}}}}}
23) Concatenate $H_1$, $H_2$,…, $H_n$ as HA;
24) return HA;
```

In this algorithm, the last *x* bytes of each address are taken, and their hash values are calculated when a "/" is stitched after the hash value, indicating that it is the value of the last *x* bytes. If no hash collision occurs, the hash values are stitched into the character string *HA* in the order in which the addresses are used. If a hash collision occurs, the parts taken by the corresponding addresses are moved forward by *x* bytes and the hash values of the new parts are recalculated, with the values being concatenated with "//" to indicate that they are from the last 2*x* to *x* bytes of these addresses. If there is a collision between the recalculated hash values, the corresponding addresses are fetched and calculated for the third time. The new hash values and "///" are concatenated to indicate that the part of the corresponding addresses that are used to calculate the hash value are again shifted forward by *x* bytes. For completely random strings, the probability of taking three parts with exactly the same hash values is slim. Finally, the string obtained by concatenating the obtained final hash value according to the order of address used is the *HA* storing address order information.

4.3 OP_RETURN fields order guarantee algorithm

Due to the maximum length of OP_RETURN field, multiple transactions may be required for transmission. Therefore, the order of the OP_RETURN field in each transaction needs to be guaranteed to help the receiver correctly restore the address sequence and index. During the transmission process, multiple transactions are issued at the same time, and a one-byte space *N* is set in the OP_RETURN field to store the sequence information. The algorithm for storing S_{all} in each OP_RETURN field is as follows:

Algorithm name: OP_RETURN fields order guarantee algorithm

Input: The string S_{all} obtained by concatenating address sequence information *HA* and the index information, and the OP_RETURN field length limit *l*

Output: Several OP_RETURN fields OP_1, OP_2, …, OP_w

```
1) int $t$=0;                 // Number of required OP_RETURN fields
2) int $n$=0;                 // Parameter for cyclic control, which is the serial number of current OP_RETURN
3) int $w$=1;                 // OP_RETURN fragment index
4) char $N$ []=""; // Converts $n$ into a string for splicing into information and ensure the fragment order
5) char $S_{all}'$=[];         //Stores the encrypted $S_{all}$
6) char $H_S$[]="";           //Stores the hexadecimal numbers converted from $S_{all}'$
7) Encrypt $S_{all}$ to $S_{all}'$;
8) $t = \lfloor L_{S_{all}'}/(l-1) \rfloor$; //$L_{S_{all}'}$ is the length of $S_{all}'$. Each segment OP_RETURN uses 1 byte $N$ to store the fragment order
9) for ($n$=0; $n$<$t$; $n$++) {
10) Convert $n$ into a string and record it in $N$;
11) if ($L_N < 2$){ Concatenate "0" and $N$ as new $N$;}        //$L_N$ is the length of $N$
12) if($L_{S_{all}'}$<=$l$-1) {      // One OP_RETURN can store all ciphertext
13)     Convert $S_{all}'$ into a hexadecimal as $H_S$;
14)     Concatenate "0x", $N$, and $H_S$ as $OP_w$;}
15) else {
16)     Convert ${S_{all}'}_0$ to ${S_{all}'}_{l-2}$ into a hexadecimal as $H_S$;
17)     Concatenate "0x", $N$, and $H_S$ as $OP_w$;
18)     $w$++;
19)     Update $S_{all}'$ by removing ${S_{all}'}_0$ to ${S_{all}'}_{l-2}$ from it;}}
20) return $OP_1$, $OP_2$, …, $OP_w$;
```

The system first determines the length of S_{all}'. If one OP_RETURN can store all the ciphertext, the system uses one transaction to transfer multiple addresses at the same time and fills all S_{all}' in the OP_RETURN field after converting it into a hexadecimal. If multiple fields are required to contain the information, the system first calculates the number of transactions specifically requiring *t*, which is also the number of OP_RETURN fields. When the information is filled into the OP_RETURN fields, the order of the content in each OP_RETURN is also filled in front at the same time, such as "01" "02" to "0*t*." This method can guarantee the order of the fields, and has no effect on the conversion between information and hexadecimals.

5 Experimental testing and analysis

There are basically three operating networks for cryptocurrencies such as Bitcoin: Mainnet, Testnet, and the private network Regtest. Among them, Regtest is a network that runs in a local environment, which is equivalent to a local test network not connected to the global P2P environment. It is suitable for development and self-testing [Zola, Pérez-Sola, Zubia et al. (2019)] In the Bitcoin public network, the average block generation time is 10 minutes, which effectively improves the network security and reduces the occurrence of stale blocks [Gervais, Karame, Wüst et al. (2016)] However, this increases the transaction confirmation delay and decreases the transaction efficiency. To simplify the model and facilitate the experiments, the Bitcoin Regtest network is used for testing in this chapter.

In addition, the encryption process is simplified to symmetric encryption, and the classic and fast DES algorithm is used for encryption, reducing the influence of irrelevant factors on the experimental results.

5.1 Experiment and analysis of information embedding based on dictionary coding

BLOCCE converts the information into binary form and stores it by using the least significant bits of the public key address. The method can be understood as converting information into a binary string, with each address storing one character of the string. To improve the efficiency of information embedding, V-BLOCCE encodes the information by using the same base58 encoding as the Bitcoin address. In this case, the characters in the address can directly represent characters in the encoded ciphertext. That is, this method converts the information into a string of the form base58 and uses the Bitcoin address to store the characters in this string. To simplify the model and more intuitively represent the gap between the two methods, V-BLOCCE is assumed to transmit only one character in the base58 string for each address in the experiment, similarly to BLOCCE. Therefore, when transmitting the same information, the lengths of the BLOCCE binary string and the V-BLOCCE base58 string determine the number of addresses that need to be used, as well as the information embedding efficiency, system efficiency, and cost. In the present experiment, the length of the plaintext encrypted message is assumed to be 1 byte to 10 bytes and 10 bytes to 100 bytes, respectively, and the lengths of the strings after binary and base58 encoding are calculated. Fig. 4 shows the experimental results:

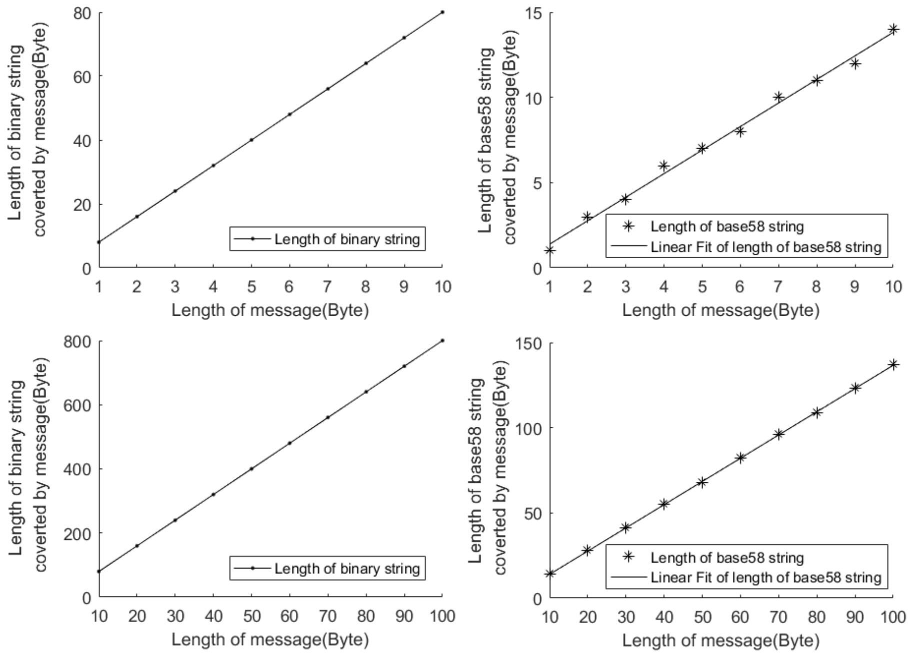

Figure 4: String length after information of different lengths is converted into binary form and base58 encoding

The lines in the figure correspond to the binary conversion and base58 encoding, respectively. The length of the string after binary conversion increases linearly with the length of the original character, and the result of base58 encoding is also basically linear. When the original strings are the same, the length of the string after base58 encoding is much shorter than that after binary conversion. Assuming that both methods use one address to transmit one character, the number of addresses used by V-BLOCCE can be much fewer compared with BLOCCE. In fact, V-BLOCCE uses special Bitcoin addresses for information embedding, and several characters in each address are reused instead of embedding only one character in an address. This can greatly limit the number of addresses required and further improves the rate of information embedding. Consequently, the dictionary-based encoding method adopted by V-BLOCCE has more advantages than that shown in the above comparison results and can effectively improve the BLOCCE disadvantage of a low information embedding rate.

5.2 Experiments and analysis of Vanitygen

The test results show that the time it takes for Vanitygen to generate special Bitcoin addresses using keys of different lengths differ. To make the method more efficient, the Vanitygen software is first tested to find the key that achieves the highest overall efficiency.

First, the time only for Vanitygen to generate addresses using keys of different lengths is tested, without considering other factors. The left figure of Fig. 5 shows the results:

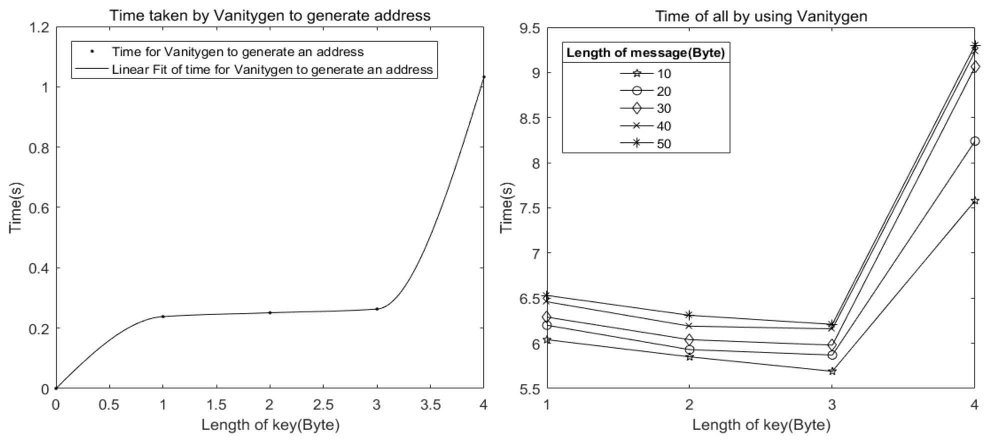

Figure 5: The address generation time for Vanitygen and total system time when keys of different lengths are used

When the length of the key is 1 byte to 3 bytes, the time it takes to generate a Bitcoin address is only about 0.2 seconds to 0.3 seconds; this increases to 1 second when the length of the key is 4 bytes. Longer keys are also tested, with the results indicating that when the length of the key reaches 5 bytes, Vanitygen takes up to about 60 seconds to generate a Bitcoin address. When the key is longer, the software takes several hours or more. Regarding the efficiency of the method, an address generation time of about 1 second or below is acceptable, whereas up to tens of seconds or even hours is clearly not ideal. The optimal solution range of the key is tentatively set between 1 byte and 4 bytes.

The above process considers only the address generation time of the Vanitygen software itself. In actual applications, the ciphertext embedding time and the transaction time after obtaining the address are also considered. Based on the optimal solution range obtained above, the overall time consumption of the system when the keys are of different lengths is calculated. The right figure of Fig. 5 shows the relationship between the length of the key and the overall time consumption. The range of the key value is set as the optimal range to determine the best solution. As shown in the figure, the total time consumptions do not change much when the length of the key is 1 byte to 3 bytes. The times basically range between 5 seconds and 7 seconds and is shortest when the length is equal to 3 bytes. Therefore, under the same network environment and hardware conditions, setting the length of the key to 3 could result in optimal efficiency for this method. In subsequent experimental comparisons, the default key length in the proposed method is set to 3.

5.3 Comparative experiment and analysis of the plaintext information embedding rate

To verify the concept that using special address generated by Vanitygen can improve the embedding rate, Vanitygen and completely random address generation methods are used to implement the entire process. The average amount of plaintext information embedded in each address is calculated in all cases. The length of the plaintext string is taken from 10 bytes to 100 bytes and the experimental results are as follows:

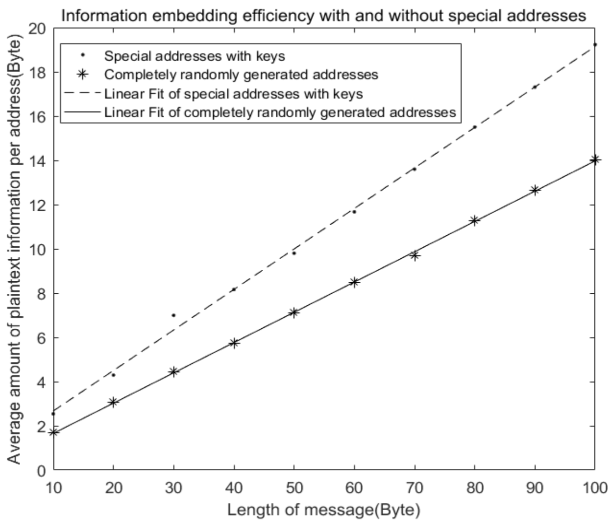

Figure 6: The average amount of plaintext information embedded in each address when Vanitygen-generated and randomly generated addresses are used

In both cases, the average amount of plaintext information contained in each address increases linearly with the length of the transmitted plaintext information. However, with the generation of special Bitcoin address, fewer addresses are needed each time, and each address contains more information on average. Therefore, the method of picking out three characters from the encrypted plaintext information to form a key and of using Vanitygen to generate a special address containing the key can improve the information embedding efficiency to a certain extent.

5.4 Experiment and analysis of the system time consumption, number of required addresses and transactions, and overall cost

After the optimal case of V-BLOCCE is obtained, comparison with the original BLOCCE method is carried out. In restoring the original method, to simplify the model as much as possible and highlight the improvements, the asymmetric encryption at the beginning of the original method is replaced with DES encryption, which is also used in the proposed method. This not only achieves the encryption effect, but also reduces the impact of the encryption process on the overall time consumption. Although the use of different encryption algorithms may lead to discrepancies between the experimental and the actual results, the disparity between two methods is unchanged. Then the encrypted information is converted into binary form and stored in the least significant bit of the address to embed the information. After the information is embedded, each address is traded separately.

In terms of time consumption, the two methods are first assumed to require the same transaction confirmation time to transmit the same plaintext information, and only the times of address generation and information embedding are compared between the two methods. Several tests are then run by using plaintext messages of different lengths for transmission. When the plaintext information is short and the length is only 1 byte to 5 bytes, the time consumption of the BLOCCE method is about 9 seconds to 10 seconds, whereas that for V-BLOCCE is only 5 seconds to 6 seconds. The length of plaintext information is then further increased. Fig. 7 shows the time consumption of the two methods when the test length is 10 bytes to 100 bytes.

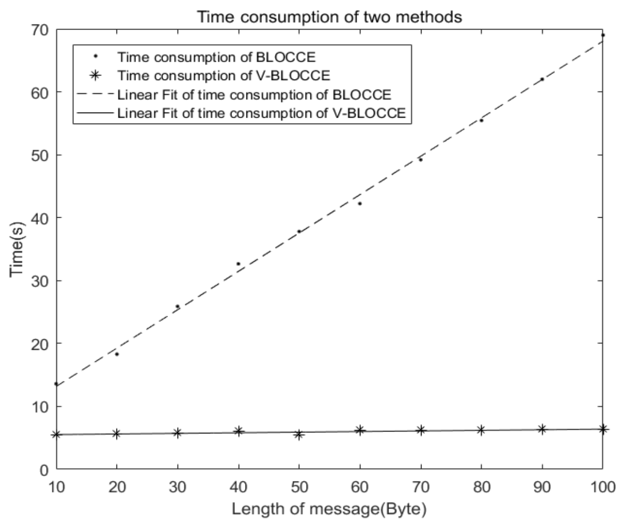

Figure 7: Comparison of the total system time when the two methods transmit information with a length of 10 bytes to 100 bytes

For the two methods, the time consumption tends to increase with the increase of the information length. However, V-BLOCCE takes less time, which increases slowly, with the overall time remaining stable. When the length of the information is continuously increased, the gap between the two becomes more obvious. This is because the information embedding efficiency of the original method is too low, and only 1 bit of data is stored per address, which causes this method to take a lot of time to generate and collide the addresses to embed the data. In contrast, V-BLOCCE greatly improves the efficiency of information embedding and repeatedly uses characters that appear in the address to match the ciphertext. In some cases, one address can embed all the information. Therefore, V-BLOCCE solves the problem of low efficiency in the BLOCCE system and greatly decreases the time required to implement the system process.

The transaction confirmation time required for each method is then considered separately. Assuming that the average confirmation time of each transaction is T_{check} and the number of transactions required by the system is *N*. Then the total transaction confirmation time of the system is equal to T_{check}×*N*. If T_{check} is unchanged, the total transaction confirmation time relationship of the two methods can be simplified to the relationship of the number of transactions required by the two methods. *N* in BLOCCE is the number of addresses used, whereas *N* in V-BLOCCE is the number of OP_RETURN fields that need to accommodate the address sequence information and index information. To ensure the security of the remarks, an encryption operation is required before the contents are converted into a hexadecimal. In the experiment, DES is selected to encrypt the string of the address sequence and index to generate ciphertext. The length of the OP_RETURN field is set to 220 bytes of the Bitcoin wallet, and the hash algorithm is SHA1. When the length of the plain text message is 10 bytes to 100 bytes, the relationship between the numbers of transactions required by the two methods is as follows:

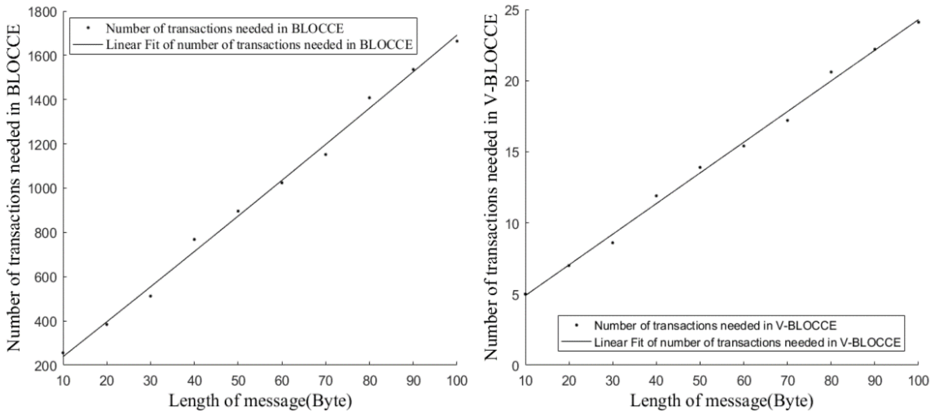

Figure 8: Number of transactions required to pass plain text messages of different lengths in the two methods

As shown in Fig. 8, the number of transactions required by both methods increases with the length of the plaintext; however, in transmitting the same information, the number of transactions required by V-BLOCCE is much smaller than that by BLOCCE. Therefore, V-BLOCCE has a shorter transaction confirmation time and higher information transmission efficiency.

In fact, the system cost of both methods can also be expressed as the number of transactions. Regardless of the method used, because the sender of the information controls the generation of the private key and the public key pair, the addresses used in the entire process all actually belong to the sender. In other words, the transactions are transferred by the sender to himself. The transfer process does not cause the total amount of the sender to decrease; thus, the system cost consists mainly of the miner fee during the transaction. Fig. 8 also shows the system cost of the two methods, assuming that the miner fee for each transaction is the same. If the miner fee for each transaction is 0.1 bitcoins, it takes less than 3 bitcoins to pass 100 bytes of information in V-BLOCCE, compared to more than 100 bitcoins in BLOCCE. In contrast to BLOCCE, to improve the efficiency of the system, V-BLOCCE issues these transactions together when multiple transactions are required for information transfer. Because the transaction is successful only after it is confirmed by the Bitcoin network, the premise of a successful V-BLOCCE concealed communication is that the sender has a sufficient number of bitcoins that is greater than or equal to the sum of all transactions in all transfers. As previously mentioned, the sender controls each transaction such that the transaction amount of each transfer is small enough, or even negligible, to achieve low requirements on the amount of bitcoin owned by the sender. Hence V-BLOCCE has much lower cost compared to BLOCCE.

Through experiments, BLOCCE encrypts and binary converts the plaintext, resulting in the use of more than 100 addresses despite a message length of only 1 byte to 5 bytes. Nevertheless, the proposed method can repeatedly use characters in an address to embed information, effectively decreasing the number of addresses. When the plaintext length is 1 byte to 5 bytes, the number of addresses is between 3 and 4. The address usage under the two methods is also tested with a character length of 10 bytes to 100 bytes; Fig. 9 shows the results.

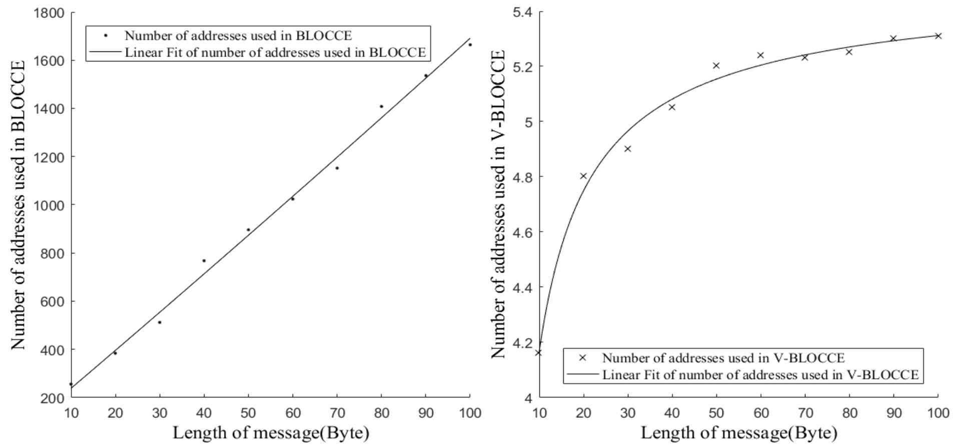

Figure 9: Comparison of the number of addresses used when passing messages 10 bytes to 50 bytes in length

The graph on the right in Fig. 9 represents the V-BLOCCE method. When the information length increases from 10 bytes to 30 bytes, the number of addresses used increases from 4 to 5 and stabilizes after the information length reaches 30 bytes. In the test, even if the character length continues to increase to hundreds or even a thousand, the number of addresses used remains low. The number of addresses used by the BLOCCE method, represented by the graph on the left, increases linearly with the length of the information. In contrast, the number of addresses used in the proposed method is much fewer than in the original method and has a smaller growth rate while maintaining a small value. The V-BLOCCE method uses fewer addresses than the BLOCCE method when the information length is very short, and this difference between the two methods becomes more obvious as the information length further increases. Therefore, V-BLOCCE can effectively solve the problem of having to use too many addresses in BLOCCE.

6 Conclusion

In this report, a new covert communication method that uses special Bitcoin addresses was constructed based on the BLOCCE method. Compared with the original method, base58 encoding was done on the information after encryption. The base58 dictionary was then used to repeatedly match the characters in the special address generated by Vanitygen in bytes, and the index in which these characters were located was recorded. After information embedding, the hash values corresponding to the addresses used were stitched in the order of address usage. Remarks consisting of the hash and index information were filled in the OP_RETURN fields and sent with the transactions. To improve the communication efficiency, after using a byte space to ensure the order of each OP_RETURN field, the transactions were carried out simultaneously without having to wait for the generation of a new block. The receiver obtained the transaction addresses and remark information by retrieving the transaction initiated by the sender and then decrypting it to obtain the correct address order and index information that can be combined to restore the secret information.

Compared with the BLOCCE method, the proposed method requires fewer transactions, has a higher information embedding rate, and uses fewer addresses. The overall time efficiency of the method is higher, and the total time for address generation, message embedding, and transaction generation is shorter. In addition, V-BLOCCE entails much less cost than the original method.

The present work improved the BLOCCE method and enhanced the feasibility and practicality of using Bitcoin transactions for covert communication, providing ideas and methods for the combination of blockchain and covert communication. Future works include expanding the presented set of methods and systems to other blockchain applications and platforms, such as Ethereum, toward achieving higher feasibility and a wider range of applications for hidden communication based on blockchain technology.

Acknowledgement: The authors would like to thank the reviewers for their detailed reviews and constructive comments, which have helped improve the quality of this paper.

Funding Statement: This work is sponsored by the Natural Science Foundation of Heilongjiang Province of China under Grant No. LC2016024. Natural Science Foundation of the Jiangsu Higher Education Institutions Grant No. 17KJB520044 and Six Talent Peaks Project in Jiangsu Province No. XYDXX-108.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

Babar, A. B.; Kendule, J. A.; Shinde, S. M. (2018): Enhanced digital image watermarking in combination with encryption. *Techno-Societal, International Conference on Advanced Technologies for Societal Applications*. Springer, Cham.

Bartoletti, M.; Pompianu, L. (2017): An analysis of bitcoin op_return metadata. *Lecture Notes in Computer Science*.

Bash, B. A.; Goeckel, D.; Towsley, D.; Guha, S. (2015): Hiding information in noise: fundamental limits of covert wireless communication. *Communications Magazine, IEEE*, vol. 53, no. 12, pp. 26-31.

Cao, Y.; Zhou, Z.; Sun, X.; Gao, C. (2018): Coverless information hiding based on the molecular structure images of material. *Computers, Materials & Continua*, vol. 54, no. 2, pp. 197-207.

Chen, H.; Heidari, A. A.; Zhao, X.; Zhang, L. J.; Chen, H. (2020): Advanced orthogonal learning-driven multi-swarm sine cosine optimization: Framework and case studies. *Expert Systems with Applications*, vol. 144, pp. 113113.

Chen, Y. H.; Lu, C. H.; Hsu, P. Y. (2015): Multilayered information encryption scheme with fine-grained authentication. *Asia-pacific Signal & Information Processing Association Summit & Conference, IEEE*, pp. 1126-1130.

Cox, I. J.; Miller, M. L.; Bloom, J. A.; Honsinger, C. (2002): *Digital Watermarking*. San Francisco: Morgan Kaufmann, USA.

Gervais, A.; Karame, G. O.; Wüst, K.; Glykantzis, V.; Ritzdorf, H. et al. (2016): On the security and performance of proof of work blockchains. *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pp. 3-16.

Gobbo, F.; Benini, M. (2013): From ancient to modern computing: a history of information hiding. *IEEE Annals of the History of Computing*, vol. 35, no. 3, pp. 33-39.

Huang, Y. F.; Xiao, B.; Xiao, H. H. (2008): Implementation of covert communication based on steganography. *International Conference on Intelligent Information Hiding & Multimedia Signal Processing, IEEE Computer Society*, pp. 1512-1515.

Kaneda, K.; Hirano, K.; Iwamura, K.; Hangai, S. (2008): Information hiding method utilizing low visible natural fiber pattern for printed documents. *International Conference on Intelligent Information Hiding & Multimedia Signal Processing, IEEE Computer Society*, pp. 319-322.

Kang, Y. H.; Liu, F. L.; Yang, C. F.; Xiang, L. Y.; Luo, X. Y. et al. (2019): Color image steganalysis based on channel gradient correlation. *International Journal of Distributed Sensor Networks*, vol. 15, no. 5, pp. 1550147719852031.

Li, P.; Li, R. X.; Cao, Y.; Li, D. Y.; Xie, G. (2018): Multiobjective sizing optimization for island microgrids using a triangular aggregation model and the levy-harmony algorithm. *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3495-3505.

Long, M.; Peng, F.; Li, H. Y. (2018): Separable reversible data hiding and encryption for HEVC video. *Journal of Real-Time Image Processing*, vol. 14, no. 1, pp. 171-182.

Luo, Y. J.; Qin, J. H.; Xiang, X. Y.; Tan, Y.; Liu, Q. et al. (2020): Coverless real-time image information hiding based on image block matching and dense convolutional network. *Journal of Real-Time Image Processing*, vol. 17, no. 1, pp. 125-135.

Matzutt, R.; Hiller, J.; Henze, M.; Ziegeldorf, J. H.; Müllmann, D. et al. (2018). A quantitative analysis of the impact of arbitrary blockchain content on bitcoin. *International Conference on Financial Cryptography and Data Security*, pp. 420-438.

Medhane, D. V.; Sangaiah, A. K.; Hossain, M. S.; Muhammad, G.; Wang, J. (2020): Blockchain-enabled distributed security framework for next generation IoT: an edge-cloud and software defined network integrated approach. *IEEE Internet of Things Journal*. https://doi.org/10.1109/JIOT.2020.2977196.

Nakamoto, S. (2019): Bitcoin: A peer-to-peer electronic cash system. *Manubot.*

Ono, N. (2012): Audio information hiding based on stereo phase difference in time-frequency domain. *Tenth International Conference on Intelligent Information Hiding & Multimedia Signal Processing, IEEE*, pp. 260-263.

Partala, J. (2018): Provably secure covert communication on blockchain. *Cryptography*, vol. 2, no. 3, pp. 18.

Peng, F.; Lin, Z. X.; Zhang, X.; Long, M. (2019): Reversible data hiding in encrypted 2D vector graphics based on reversible mapping model for real numbers. *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 9, pp. 2400-2411.

Pérez-Cabré, E.; Abril, H. C.; Millán, M. S.; Javidi, B. (2011): Photon-counting imaging based double-random-phase encryption for information security and verification. *Information Optics Euro American Workshop*, pp. 1-3.

Ren, Y. J.; Liu, Y. P.; Ji, S.; Sangaiah, A. K.; Wang, J. (2018): Incentive mechanism of data storage based on blockchain for wireless sensor networks. *Mobile Information Systems*, https://doi.org/10.1155/2018/6874158.

Rouhani, B. D.; Chen, H.; Koushanfar, F. (2018). Deepsigns: a generic watermarking framework for IP protection of deep learning models.

Wang, G. Y.; Ma, S. Y. (2008): Information hiding based on integer dwt and error correct coding. *Journal of Tianjin University of Technology*, vol. 18, no. 10, pp. 1248-1251.

Xia, Z. Q.; Tan, J. J.; Wang, J.; Zhu, R. L.; Xiao, H. G. et al. (2019): Research on fair trading mechanism of surplus power based on blockchain. *Journal of Universal Computer Science*, vol. 25, no. 10, pp. 1240-1260.

Xue, X.; Han, H. F.; Wang, S. F.; Qin, C. Z. (2016): Computational experiment-based evaluation on context-aware O2O service recommendation. *IEEE Transactions on Services Computing*, vol. 12, no. 6, pp. 910-924.

Xue, X.; Wang, S. F.; Zhang, L. J.; Feng, Z. Y.; Guo, Y. D. (2018): Social learning evolution (SLE): computational experiment-based modeling framework of social manufacturing. *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3343-3355.

Zola, F.; Pérez-Sola, C.; Zubia, J. E.; Eguimendia, M.; Herrera-Joancomartí, J. (2019): Kriptosare. gen, a dockerized Bitcoin testbed: analysis of server performance. *10th IFIP International Conference on New Technologies, Mobility and Security*, pp. 1-5.

Zou, C.; Li, G.; Su, W. (2015): A type encryption system of text information based on Chen-Mobius transformation. *IEEE International Conference on Anti-counterfeiting*, pp. 33-37.