Hindawi
Security and Communication Networks
Volume 2021, Article ID 6351144, 16 pages
https://doi.org/10.1155/2021/6351144

Research Article

High-Capacity Robust Behavioral Steganography Method Based on Timestamp Modulation across Social Internet of Things

Mingliang Zhang[1,2] **Xiangyang Luo**[1,2,3] **Pei Zhang**[1,2] **Hao Li**[1,2] **Yi Zhang**[1,2] **and Lingling Li**[4]

[1]*Zhengzhou Institute of Information Science and Technology, Zhengzhou 450001, China*
[2]*State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450001, China*
[3]*Henan Province Key Laboratory of Cyberspace Situation Awareness, Zhengzhou 450001, China*
[4]*School of Intelligent Engineering, Zhengzhou University of Aeronautics, Zhengzhou 450046, China*

Correspondence should be addressed to Xiangyang Luo; luoxy_ieu@sina.com

Received 17 September 2021; Accepted 2 December 2021; Published 31 December 2021

Academic Editor: Ding Wang

Social Internet of Things (SIoT) is an emerging field that combines IoT and Internet, which can provide many novel and convenient application scenarios but still faces challenges in data privacy protection. In this paper, we propose a robust behavioral steganography method with high embedding capacity across social networks based on timestamp modulation. Firstly, the IoT devices on the sending end modulate the secret message to be embedded into a timestamp by using the common property on social networks. Secondly, the accounts of multiple social networks are used as the vertices, and the timestamp mapping relationship generated by the interaction behaviors between them is used as the edges to construct a directed secret message graph across social networks. Then, the frequency of interaction behaviors generated by users of mainstream social networks is analyzed; the corresponding timestamps and social networks are used to implement interaction behaviors based on the secret message graph and the frequency of interaction behaviors. Next, we analyze the frequency of interaction behaviors generated by users in mainstream social networks, implement the interaction behaviors according to the secret message graph and the frequency of interaction behaviors in the corresponding timestamps and social networks, and combine the redundant mapping control to complete the embedding of secret message. Finally, the receiver constructs the timestamp mapping relationship through the shared account, key, and other parameters to achieve the extraction of secret message. The algorithm is robust and does not have the problem that existing multimedia-based steganography methods are difficult to extract the embedded messages completely. Compared with existing graph theory-based social network steganography methods, using timestamps and behaviors frequencies to hide message in multiple social networks increases the cost of detecting covert communication and improves concealment of steganography. At the same time, the algorithm uses a directed secret message graph to increase the number of bits carried by each behavior and improves the embedding capacity. A large number of tests have been conducted on mainstream social networks such as Facebook, Twitter, and Weibo. The results show that the proposed method successfully distributes secret message to multiple social networks and achieves complete extraction of embedded message at the receiving end. The embedding capacity is increased by 1.98–4.89 times compared with the existing methods SSN, NGTASS, and SGSIR.

1. Introduction

With the widespread use of IoT devices and social networks, people have combined them to form the SIoT [1], which build many interesting scenarios. For example, it allows users to convert data about plants, pets, and so forth acquired by sensors in their homes into posts, which are then posted to social networks to share their lives in real time. In this era of connected everything, users' privacy protection is always a hot topic being discussed. For edge devices involving personal safety, more covert means of communication are needed to transmit secret commands that prevent

communication data from being monitored and tampered within the link to avoid significant loss of life and property. Steganography is a privacy protection method that uses redundant methods such as human vision and hearing to conceal the existence of secret communication [2] and has received wide attention from scholars in the field of information security. It allows SIoT to deliver critical data with higher security requirements in an imperceptible manner while delivering ordinary data. Therefore, it is of great importance and practical value to study the covert communication of social IoT.

This paper will focus on the way how edge devices of SIoT can implement steganography on social networks. The traditional steganography method based on multimedia has high embedding capacity and superior performance in resisting statistical detection [3]. However, the generated steganography object is vulnerable to channel attack when it is sent through a lossy channel, which will result in the inability to extract secret message correctly. Meanwhile, once the carrier is modified during the embedding of secret message [4], there is a risk of being identified as abnormal data [5]. Social network behavioral steganography (SNBS) is a method, which uses user's comments, forwardings, and other behaviors on social networks to realize covert communication. It combines the rich interaction behaviors of social networks [6] with steganography to make up for the shortcomings of traditional steganography methods in terms of robustness and detection resistance. On lossy channel, it does not cause the loss of behavioral data but ensures the robustness of steganography methods. In addition, secret message is hidden in behaviors, which effectively resists the analysis and detection technology for multimedia data; the sender and receiver do not directly establish communication to avoid causing special attention of the third party [7] and avoid the possibility of the sender's exposure or betrayal to testify against the receiver. However, the embedding capacity of SNBS is very low. Although it avoids the detection of multimedia data, it also introduces behavioral abnormalities. This limits the practical use of such methods. How to improve the embedding capacity and correct the abnormal behaviors is an urgent problem to be solved in the practical application of SNBS technology. This paper will focus on the improvement of SNBS capacity and the correction of abnormal behaviors.

Recently, researchers have developed a series of studies in the field of SNBS. Existing social network steganography methods can be divided into two categories according to the difference of secret message carriers: social network multimedia steganography (SNMS) and social network behavioral steganography (SNBS).

SNMS refers to embedding secret message into multimedia such as posts and comments published by users. This kind of methods combines multimedia such as text [8–11], image [12–15], and video containing secret message with a social network and uses posts, comments, and other ways to realize covert communication.

SNBS uses user's interaction to convey secret message and can be further divided into non-graph-theory-based steganography and graph-theory-based steganography. For non-graph-theory-based steganography, in order to prove whether social network behavioral data can provide instructions for botnets, Pantic and Husain [16] use the length of Twitter to realize the transmission of secret data. This method has certain concealment, but it is easy to be detected. Zhang [17] proposes a method to send secret message by marking "love." The behavioral steganography method of marking "love" attempts to send message in the WeChat Moments, which is widely used in China, but there are insufficient embedding capacity and detection resistance. In view of the problem that the relationship between friends is not considered in [17], Hu et al. [18] use the behavioral correlation between sender and friend to calculate the reasonable probability of marking "love" on a social network, which improves the security, but the embedding capacity needs to be improved. Regarding the "prisoner model," the security issue between sender and receiver is not considered; Yang et al. [7] give a constraint framework to ensure content security and behavioral security and provide a reference method for the behavioral security of social network steganography. Li et al. [19] propose a framework for reposting posts and other network activities to hide secret message. This framework hides secret message in interactive activities and provides a theoretical basis for social network steganography. For graph-theory-based steganography, methods of using a graph theory to describe steganography can be traced back to [20]. This paper describes the method of using a dynamic data structure in memory to store secret messages. Nechta [21] constructs an undirected graph of secret message on social networks to achieve covert communication. The method proposed by Wu et al. [22] conveys secret message through an undirected graph and enhances the security of communication through a directed graph; however, there are some redundancies in the nodes of this method. To address the problem that the secret data may be exposed due to the attacker's mastery of the reconstruction graph process, Wu et al. [23] improve the secret data security by remapping the correspondence between the vertices of the graph structure with a key, but the embedding capacity of the method still needs to be improved.

In response to the above problems, this paper proposes a high-capacity and robust behavioral steganography method for cross-platform social networks based on timestamp modulation. This method uses accounts in multiple social networks to construct a directed secret message graph on the sender side and hides the secret message in the timestamps of the interaction behaviors generated by the secret message graph. At the receiving end, the receiver uses the shared account and key to extract the secret message through the public social networks. The main work of this paper is as follows:

(1) This paper proposes a method to construct a directed secret message graph across multiple social networks and map the secret message into the timestamp generated by the edge set of the secret message graph. Compared with existing methods, this method has a very high embedding capacity, and the effective number of bits transferred and the embedding rate are improved.

(2) This paper proposes a distribution modulation algorithm that hides secret messages into multiple

social networks by timestamps and fits the frequency characteristics of ordinary user interactions. This algorithm increases the cost of attackers to analyze the covert communication, reduces the probability of anomalous behaviors, and improves the detection resistance.

(3) This paper proposes an algorithm to overcome the problem of information extraction failure due to interaction delay by increasing the redundancy time. The algorithm increases the number of mappings between participant account indexes and timestamps to avoid the possibility of mismatch between them and solve the robustness problem.

The rest of this paper is organized as follows. Section 2 briefly introduces the work related to SNBS. Section 3 introduces the method proposed in this paper. After that, Section 4 gives the experimental results and evaluation. Finally, this paper is concluded by Section 5.

2. Related Work

Steganography based on graph theory uses user's accounts and interactions on social networks to construct secret message graphs to realize covert communication. Compared with SNMS, steganography based on graph theory has high robustness and there is no statistical anomaly for multimedia steganography, and it is effective against multimedia steganography analysis techniques. Therefore, this section will briefly introduce the previous research based on graph theory steganography, so as to explain the feasibility of designing steganography methods based on graph theory.

Nechta [21] used social network behaviors to construct undirected graphs for covert communication, abbreviated as SSN. It constructs a secret message undirected graph $G(V, E, \varphi)$, where V denotes the set of vertices in the graph, E denotes the set of edges in the graph, and φ denotes the adjacency function. This function is shown in equation (1) and is used to define whether the edges between V_a and V_b exist in graph G. $|V|$ denotes the number of vertices in set V. Moreover, vertices (v_a, v_b) are exhaustively enumerated, where $a < b$. The number of bits that can be transmitted by this method is N_1 shown in equation (2).

$$\varphi = \begin{cases} 0, & E_{a,b} \notin E, \\ 1, & E_{a,b} \in E \end{cases} \tag{1}$$

$$N_1 = |V| * \frac{(|V| - 1)}{2}. \tag{2}$$

To facilitate the comparison of the ability of existing methods to deliver the size of secret messages, we take the product of the number of interactive behaviors h and the average number of bits carried by each behavior as the embedding capacity $c = h * l_s$. Let the average number of bits carried by a single behavior be l_s. In this paper, the numbers of bits sent in [21–23] are denoted as N_1, N_2, N_3, and the numbers of interaction behaviors are denoted as h_1, h_2, h_3, and their embedding capacities are denoted as c_1, c_2, c_3, respectively. The embedding capacity in [21] is shown by the following equation:

$$c_1 = \frac{h * N_1}{h_1} = h * |V| * \frac{(|V| - 1)}{2h_1}. \tag{3}$$

The paper in [21] proposed a novel method to enable covert communication on social network, which provides a good reference for steganographic methods to social networks.

Wu et al. [22] used undirected graphs to hide secret message and directed graphs to hide topology and enhanced the security of the proposed method, abbreviated as NGTASS. For a graph $G(V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{e_1, e_2, \ldots, e_n\}$, each bit of the secret message binary string is embedded as an edge of the undirected graph $G_0(V_0, E_0)$ and then the message graph G_0 is embedded into the directed graph $G_1(V_1, E_1)$. The edges in the directed graph G_1 correspond to the edges in the undirected graph G_0, and the direction is from the high index vertex to the low index vertex. If we ignore the direction pointing, G_0 is a subgraph of G_1 actually. The premise of this method is that the steganography and extracting processes share a set U containing all vertices in G_1. The interaction information of the social network is public, and the receiver can observe the interaction of the sender. When a receiver obtains G_0 and G_1, the secret message will be extracted. It is worth noting that the method uses $|V|$ accounts to achieve covert communication, and one of the vertices is used to hide the topological information; in fact, $|V| - 1$ vertices are used to encode the secret message. N_2 is shown in equation (4) and c_2 is shown in equation (5).

$$N_2 = \frac{(|V| - 1) * (|V| - 2)}{2}, \tag{4}$$

$$c_2 = \frac{h * N_2}{h_2} = \frac{h * (|V| - 1) * (|V| - 2)}{2h_2}. \tag{5}$$

This method uses undirected graphs to encode secret message and uses additional vertices to hide the topology, which reduces the embedding capacity.

Wu et al. [23] based their work on [22] to remap the correspondence between the vertices of the graph structure by a key, abbreviated as SGSIR. This method controls $n + 2$ vertices, which are indexed from v_0 to v_{n+1}, and the vertex set consisting of v_1 to v_n is denoted by V. The edges whose starting and ending points belong to the vertex set V are denoted as the edge set E. The edge set E has m edges, where m is an integer power of 2. For the embedding process, the method selects m edges in E based on a random seed R and assigns an index. Then, the secret message is converted into a binary sequence, and every $\log_2 m$ group is converted into a decimal sequence $D = \{d_1, d_2, \ldots, d_m\}$ and is allocated $m + 1$ build operations. Finally, four types of build operations are performed to send the secret message to the social network. For the extracting process, a receiver reconstructs the graph structure and extracts the secret message by using R, V, and public social network. The method controls $|V|$ vertices to achieve covert communication, but vertices v_n and v_{n+1} are used to denote 0 and 1, respectively, so the method uses $|V| -$

2 vertices to encode the secret message. N_3 is shown in equation (6) and c_3 is shown in equation (7).

$$N_3 = 2^{\lfloor \log_2 ((|V|-2)*(|V|-3)/2) \rfloor} * \lfloor \log_2 \frac{(|V|-2)*(|V|-3)}{2} \rfloor, \tag{6}$$

$$c_3 = \frac{h * N_3}{h^3} = \frac{h * 2^{\lfloor \log_2 ((|V|-2)*(|V|-3)/2) \rfloor} * \lfloor \log_2 ((|V|-2)(|V-3|)/2) \rfloor}{h^3}. \tag{7}$$

This method uses an undirected graph to construct a secret message graph and uses interactive remapping to improve security. Even if the attacker successfully reconstructs the graph structure, he cannot obtain the embedded data. At the same time, each behavior can carry more secret messages. However, it consumes a certain amount of vertices to hide information such as key and topology and uses undirected graphs to realize covert communication, which reduces the embedding capacity of the method.

We draw the basic framework of this type of method, as shown in Figure 1. The main steps can be briefly described as follows:

(i) Encrypting secret message: encrypt the secret message to be transmitted

(ii) Converting secret message to binary: convert ciphertext information into binary which is easy to send

(iii) Building a secret message graph: the secret data is modulated into a secret message graph

(iv) Releasing the secret message graph to the social networks: the edges in the secret message graph will be interactively generated on the social networks in turn

(v) Extracting secret message: the extracting process of these methods is the reverse process of the embedding process

Based on the principles of existing methods, it is clear that behavioral steganography methods based on graph theory using undirected graphs to hide secret message have shortcomings. They convey less secret message and have lower embedding capacity. Moreover, they use a single social network to achieve steganographic communication, which requires frequent operations on the social networks. Timestamp is one of the public properties of social networks, and if the secret message is transformed into timestamps distributed over multiple social networks and used to construct directed graphs, it can effectively reduce the number of interactions generated in a single social network and improve the detection resistance and embedding capacity. Therefore, in this paper, we focus on a high-capacity, anomaly-detection-resistant behavioral steganography method using social networks timestamps.

The main notations in this paper are shown in Table 1. The notations without subscripts have a general description, and the subscripts s and r denote belonging to the sender and receiver, respectively.

3. Proposed Method

Timestamp is a time representation, also called Unix timestamp, which is defined as the total number of seconds from 00:00:00 GMT on January 01, 1970, to a certain point in time. The proposed method modulates the secret message as timestamps, constructs a directed secret message graph, and releases the edges of the directed secret message graph to social networks in the form of interactive timestamps. In this section, we describe our proposed method through a framework diagram of the method, several key steps, and an example.

The basic idea of this method is as follows: firstly, the sender-controlled accounts are used as the set of vertices of the graph, the possible edges between the vertices are traversed, and index numbers are assigned to construct the secret message graph. Then, the edges in the graph are used to construct the interaction index matrix, donated by **I**, whose elements are called interaction indexes, and the secret message is converted into interaction indexes, and the mapping relationship between interaction indexes and timestamps are constructed. Finally, the secret message is modulated into timestamps by using interaction indexes as an intermediate form, which enables the embedding process of covert communication. The framework of the proposed method is shown in Figure 2; Steps 1–5 belong to the embedding process, and Steps 6–9 belong to the extracting process.

Step 1. Data preprocessing: using the key, the plaintext is encrypted into ciphertext, and the ciphertext is converted into decimal data suitable for subsequent steps.

Step 2. Generating interaction index matrix: the accounts controlled by the sender are used as the set of vertices of graph, and the possible edges between vertices are traversed. Hash the key, take some of the values as random seed to assign index numbers to the edges in the graph, and build a matrix called interaction index matrix.

Step 3. Generating cross-platform interaction sequence and construct the cross-platform directed interaction graph: the preprocessed data is used to generate a sequence for interaction through an interaction index matrix, which can construct an interaction graph among multiple social networks.

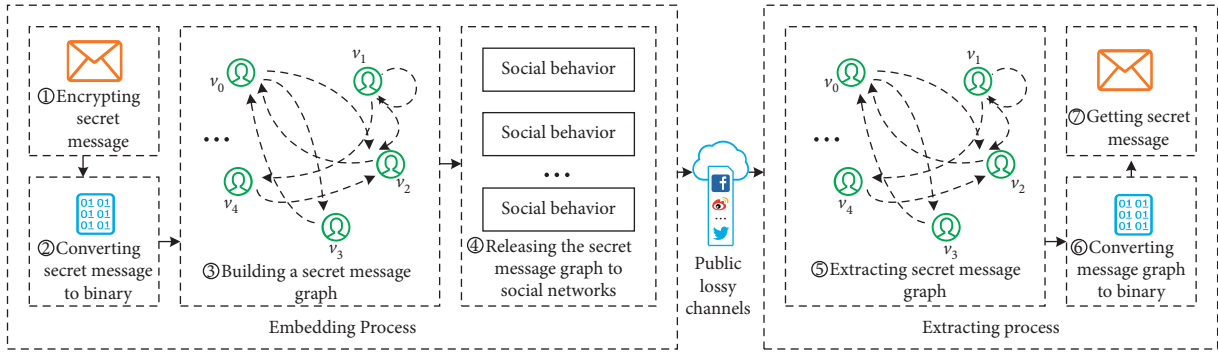

Figure 1: Basic framework of behavioral steganography based on graph theory.

Table 1: Notation table.

Notations	Explanation
A_s	ASCII code of the sender character
d	Maximum interaction delay
E	The set of edges in G
F	Time interval sequence of behaviors
G	Graph
$\mathbf{I}$	Interaction index matrix
l_b	Number of accounts actually controlled by the sender
l_p	The binary to be sent
l_r	Binary with secret message
l_s	Embedding capacity of a single behavior
m	Number of edges in G
p	Time interval
r_b	Effective bit rate
R	Random seed
S	Timestamp index sequence
S_R	Random sequence
t_c	Current timestamp
V	The set of vertices in G
B_s, B_r	Grouped binary for sender and receiver
D_s, D_r	Decimal sequence of sender and receiver
E_s, E_r	Interaction sequence of sender and receiver
n, $\lvert N\rvert$	Number of accounts actually controlled
T_s, T_r	Sender and receiver timestamp sequence
V_s, V_r	The set of vertices of the sender and receiver

Step 4. Generating cross-platform robust timestamp sequence: this step firstly obtains data about the user's behaviors on the corresponding social networks, which is used to analyze the interaction patterns. Secondly, the behavior generated by a user at one moment may be recorded at the next moment, which will cause a delay in the timestamp of the behavior, resulting in the receiver not being able to extract the message correctly. This problem can be solved by using a time redundancy control mechanism, for which it is necessary to obtain the maximum time delay over a while. Finally, the mapping relationship between vertices and timestamps is constructed by random seed, and the timestamp sequence of behaviors is generated by combining the interaction sequence, interaction delay, and interaction patterns.

Step 5. Releasing secret message graph: combine directed interaction graphs and timestamp sequence to construct secret message graph and release it to social networks.

Step 6. Regenerating the interaction index matrix: the receiver uses the account set and key to generate the interaction index matrix. The process of generation is the same as Step 2.

Step 7. Extracting timestamp sequence: the receiver uses the accounts to obtain user data from the corresponding social networks. The receiver parses the user interaction data in turn to obtain a sequence of timestamps, the elements of which are of the form <sender index, timestamp>.

Step 8. Reconstructing directed secret message graph: the receiver senses and obtains the maximum interaction delay of these networks over a while and tries to reconstruct the index of interaction participants by the maximum interaction delay and timestamp. Then, parse the <sender index, timestamp> element into a <sender index, interaction index> element, and construct a directed secret message graph.

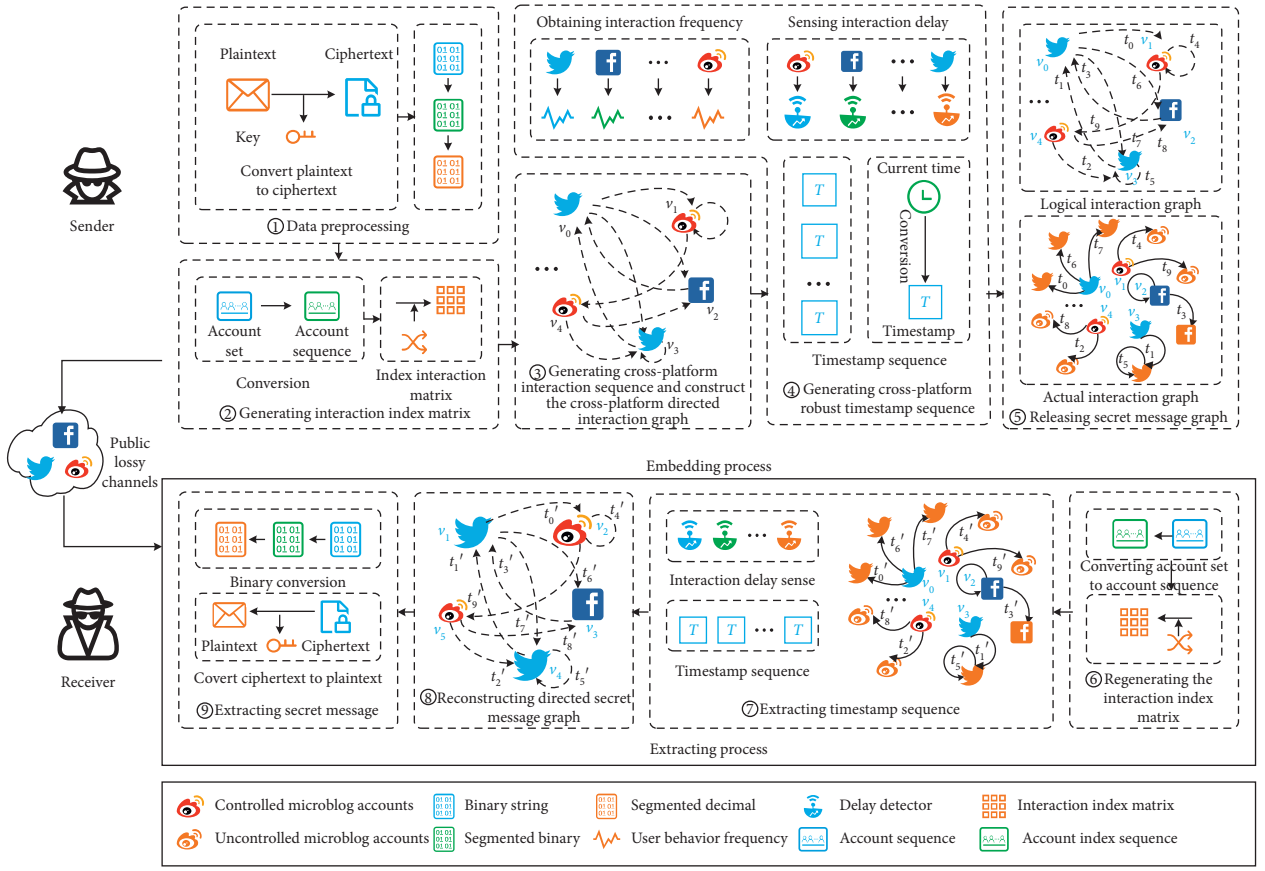

FIGURE 2: Framework of the proposed method.

Step 9. Extracting secret message: decimal secret data can be extracted by interaction indexes. Then, the secret message can be obtained through a share key.

In this paper, we use graph theory to describe the interaction behaviors between accounts. For a directed graph $G = V, E, G = <V, E>, V = \{v_0, v_1, \ldots, v_{n-1}\}$ and $E = \{e_0, e_1, \ldots, e_{m-1}\}$. $\forall e \in E, e = <u, v>$, which represents an interaction between accounts u and v. u is the account that generates the behaviors, and v is the account that interacts with u. For a directed graph G, with n vertices in V and m edges in E, n vertices can generate at most n^2 different interaction behaviors. We use l_b to denote the number of binary bits carried by each behavior. For coding purposes, m will be an integer power of 2, and $l_b = \log_2 m$ will be an integer. In fact, the following relationship will exist between l_b, m, and n in this method (note: "$\lfloor \cdot \rfloor$" means rounding down):

$$l_b = \lfloor \log_2 n^2 \rfloor. \tag{8}$$

From equation (8), when n is determined, l_b and m are obtained. For example, when $n = 6$, $l_b = 5$ and $m = 32$. The main steps of the method are described in detail in the following subsections.

3.1. Generating Interaction Index Matrix. An n-order matrix $\mathbf{I}$ is initialized, whose elements are all -1, and, after the hash key, take part of the value as the random seed, and generate m different pseudorandom indexes i in $\mathbf{I}$, where $i \in [0, m-1]$. At this time, the $n^2 - m$ elements of $\mathbf{I}$ are still -1. The pseudocode to generate matrix $\mathbf{I}$ is shown in Algorithm 1.

3.2. Generating Cross-Platform Interaction Sequence and Construct Cross-Platform Directed Interaction Graph. The binary data obtained by encrypting the plaintext is divided into groups of 8 bits; each group is an element of $A_s = \{a_0, a_1, \ldots\}$. Then, S_s is divided into l_b groups and converted to decimal, denoted as $D_s = \{d_0, d_1, \ldots\}$. If the number of binaries in the remaining S_s is less than l_b, it needs to append 0 until its length is l_b. $\forall i_{j,k} \in \mathbf{I}$, j, k are the row index and column index of matrix element i, respectively. By matching the positions of the elements in set D_s in matrix $\mathbf{I}$, the row and column indexes of the elements are obtained, denoted as $<v_j, v_k>$, and append them to E_s. The generation process of E_s is shown in Algorithm 2, which is used to implement the cross-platform interactive sequence assignment. A cross-platform directed interaction graph can be constructed based on E_s and V_s.

Accounts on different social networks cannot interact with each other directly; for example, an account on Twitter cannot comment on a Facebook post. To enable the relationship between accounts on social networks, we assign timestamp indexes to accounts, denoted as $S_s = \{s_0, s_1, \ldots, s_{n-1}\} = \{0, 1, \ldots, n-1\}$. A timestamp-

```
Input: V_s, R
Output: I
(1) Generate a random sequence S_R of length m from R
(2) n = len(V_s) //The len function represents the length of the acquired sequence
(3) Initialize an n -order matrix I, whose elements are all −1
(4) for i = 0 to m − 1 do
(5)     temp = S_R[i]
(6)     row = ⌊temp/n⌋
(7)     column = (temp − row ∗ n)%n
(8)     I[row, column] = i
(9) end for
(10) return I
```

Algorithm 1: Interaction index matrix generation algorithm.

```
Input: V_s, I, D_s
Output: E_s
(1) l = len(D_s)
(2) for i = 0 to l − 1 do
(3)     temp = D_s[i]
(4)     Get the row and column of temp in matrix I, and assign them to j and k respectively
(5)     E_s.append ([v_j , v_k])
(6) end for
(7) return E_s
```

Algorithm 2: Cross-platform interaction sequence allocation algorithm.

timestamp index mapping is constructed. For a given timestamp t, there is a unique timestamp index corresponding to it. For example, for accounts v_a, v_b, and v_c, v_a and v_b are two accounts on Facebook and v_c is an account on Twitter; v_a commented on a post of v_b at timestamp t. If the mapping value of t is c, it means that when timestamp is t, v_a established a relationship with v_b, which is called logical interaction. v_a interacted with v_b for real, which is called actual interaction. The logical and actual interactions are referred to as interactions. The graph formed by the logical interaction and its vertices is called the logical interaction graph. The graph formed by the actual interaction and its vertices is called the actual interaction graph, which is shown in Figure 2.

3.3. Generating Cross-Platform Robust Timestamp Sequence. User behavior data can be obtained from the corresponding social networks by accounts, and the time interval p of the interaction behaviors of the regular user is extracted. The frequency of the interaction interval within p is statistically distributed in minutes, and the time interval sequence F to be interacted is obtained by p. F is a two-dimensional array, the dimension represents the corresponding social networks, and the second dimension stores the timestamps of specific social network interaction behaviors. Before sending a secret message, the interaction delay of the currently used social network is constantly sensed and the maximum interaction delay d corresponds to the number of redundancies in the redundancy mapping over time.

Get the current time and convert it to the timestamp form t_c. The mapping between V_s and timestamp can be constructed by R to obtain the timestamp index. The timestamp of the interaction behavior is used to specify the time when the behavior was generated, and its sequence is denoted by T_s. T_s is generated by Algorithm 3, which has an initial value of t_c.

Algorithm 3 uses the method of adding redundant mapping to increase the robustness of this method. $\forall t \in T_s$, the algorithm executes the preset interactive behavior at t; then the secret timestamp can be posted to the social networks. Combined with the generated timestamp sequence and the posts crawled in Step 4, release the secret message graph to the social network at the corresponding timestamp.

3.4. Reconstruction of the Directed Secret Message Graph. Initialize an empty interaction sequence, and, from the timestamps that have been arranged in ascending order in Step 9, the timestamp index s can be calculated according to the following equation:

$$s = \left\lfloor \frac{t}{(d+1)} \right\rfloor \% n. \tag{9}$$

For any interaction element $<v_j, v_k>$, j is the index of the behavioral initiator in V_r, and k is the index of s in S. Get the interaction element $<v_j, v_k>$ from j, k and append it to E_r. Similarly, the elements in T_r are executed in sequence to obtain the interaction sequence E_r, which reconstructs the secret message graph. Combining the subscripts of the

```
Input: V_s, E_s, t_c, d, S, F, p
Output: T_s
(1) l = len(E_s)
(2) n = len(V_s)
(3) for j = 0 to p − 1 do
(4)   for i = 0 to l − 1 do
(5)     e = E_s[i]
(6)     receiver_index = e[v_j]] //Get the receiving account of the interaction sequence
(7)     tsi = S[receiver_index]
(8)     repeat
(9)       t_c = t_c + F[j][i]
(10)      if i == 1 and ⌊t_c/d⌋%n == tsi then
(11)        T_s.append(t_c)
(12)      if else { ⌊t_c/(d + 1)⌋ %n == tsi and T_s[i]] > T_s[i + 1] }
(13)        T_s append(t_c)
(14)      else
(15)        t_c = t_c + 1
(16)      end if
(17)  until i == l − 1
(18)  E_s append (< v_j, v_k >) //The append function means append the current element to the sequence E_s
(19)  end for
(20) end for
(21) return T_s
```

Algorithm 3: Cross-platform robust timestamp sequence generation algorithm.

sending and receiving account numbers of the elements of E_r, the decimal information carried by this sequence can be determined from **I** in Step 6. For any interaction element $<v_j, v_k>$, the decimal data value transmitted can be determined from the following equation:

$$\text{value} = \mathbf{I}[j, k]. \tag{10}$$

Using equation (10), the elements of sequence E_r are executed sequentially, and the obtained values are appended to the initially empty D_r sequence in turn. Then, D_r is converted into binary B_r and spliced into S_r. Finally, the secret message transmitted can be extracted.

$$I = \begin{bmatrix} 20 & 31 & 6 & 11 & 30 & 23 \\ -1 & 16 & 18 & 1 & 29 & -1 \\ 14 & -1 & 21 & -1 & 26 & 0 \\ 24 & 27 & 2 & 7 & 12 & 8 \\ 5 & 28 & 17 & 10 & 15 & 4 \\ 22 & 9 & 13 & 3 & 25 & 19 \end{bmatrix}. \tag{11}$$

3.5. Example. To make the proposed method easier to understand, this section gives an example to briefly describe the process of sending, using plaintext to deliver the secret message. The secret message sent is "hello"; $V_s = \{v_0, v_1, \cdots, v_5\}$; the key is "secret"; $d = 2$; $t_c = 1614704185$. Thus, we can get $n = |V_s| = 6$; then $l_b = 5$, $m = 32$, and $S = [s_0, s_1, \ldots, s_5] = [0, 1, \ldots, 5]$. When sending a secret message, the secret message is firstly converted to binary, divided into a group of every l_b bits, and then converted to decimal, and the conversion process is shown in Table 2. Then, generate a random sequence S_R of length m, assuming that at this time S_R = [17, 9, 20, 33, 29, 24, 2, 21, 23, 31, 27, 3, 22, 32, 12, 28, 7, 26, 8, 35, 0, 14, 30, 5, 18, 34, 16, 19, 25, 10, 4, 1], through the parameters of Algorithm 1, using Algorithm 1 to obtain **I**, as shown in equation (11). For example, the value of index 0, element 17 in S_R, calculated by Algorithm 1, is row = 2 and column = 5, so the element in row 2 and column 5 of **I** is 0.

Then, through equation (11) and Algorithm 2, the interaction sequence E_s is generated. For example, the element with index 0 in sequence D_s is 13, and its row = 5 and column = 2 in equation (11). Therefore, $<v_5, v_2>$ can be generated, which is shown in Table 3, and the constructed directed interaction diagram is shown in Figure 3(a).

Next, the timestamp sequence T_s is generated by Algorithm 3. Here, take $e_0 = <v_5, v_2>$ as an example, and briefly describe its generation process. $<v_5, v_2>$ means that v_5 needs to interact with v_2, and the subscript value 2 of v_2 is obtained by equation (9). The index of the timestamp corresponding to 1614704190 is 2. Therefore, v_5 performs a certain interaction behavior at timestamp 1614704190, which means that $<v_5, v_2>$ has a virtual interaction. The rest of the elements are calculated in the same way as shown in Table 3. The secret message graph is constructed based on E_s and V_s, as shown in Figure 3(b). Finally, by combining T_s and the post message, the secret message graph is released to multiple social networks at the corresponding timestamps.

The actual interaction graph generated by sending "hello" is shown in Figure 4. The red timestamp in the figure indicates that the interactive behavior is delayed, the green timestamp indicates normal sending, and the orange social network icon indicates that it is not under the control of the sender account.

Table 2: Process of converting plaintext to decimal.

Secret message	h		e	l	l		o
A_s	01101000		01100101	01101100	01101100		01101111
S_s	01101000011001010110110001101100011011111						
B_s	01101	00001	10010	10110	11000	11011	00011	01111
D_s	13	1	18	22	24	27	3	15

Table 3: Interaction diagram and timestamp sequence generation process.

Ds	13	1	18	22	24	27	3	15
Er	$<v_5, v_2>$	$<v_1, v_3>$	$<v_1, v_2>$	$<v_5, v_0>$	$<v_3, v_0>$	$<v_3, v_1>$	$<v_5, v_3>$	$<v_4, v_4>$
Ts	1614704190	1614704193	1614704208	1614704220	1614704238	1614704241	1614704247	1614704250
S_N	Facebook	Twitter	Twitter	Facebook	MicroBlog	MicroBlog	Facebook	Twitter

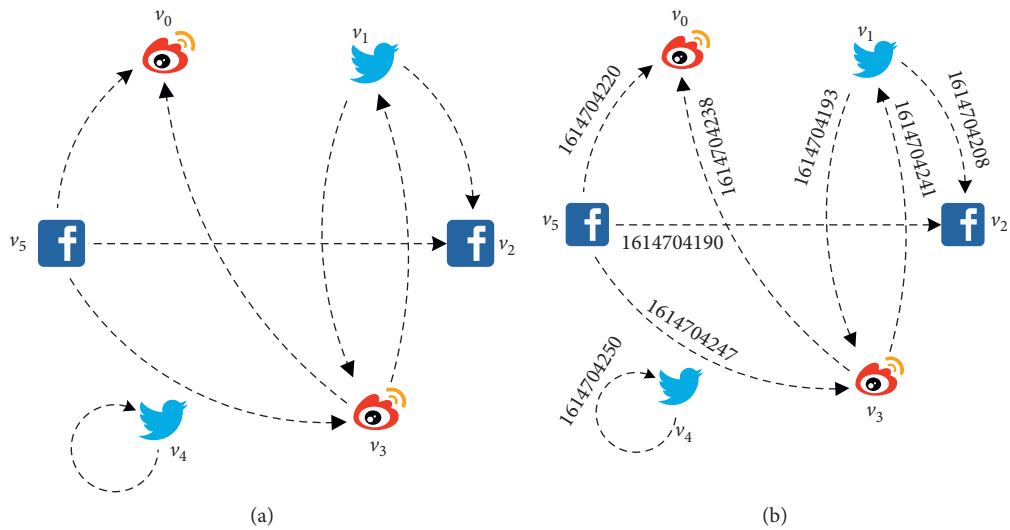

Figure 3: Logical interaction graph. (a) Logical interaction graph without timestamps. (b) Logical interaction graph with timestamps.

According to Figure 4, we can extract the timestamp and other messages. The extracting process is the inverse of the embedding process and will not be repeated here.

4. Experiments and Evaluation

To evaluate the performance of the proposed method, we implemented the methods in [21–23] and implemented the method proposed in this paper using Python.

4.1. Experimental Settings. The experiment uses *Gone with the Wind* as the secret message, "world" as the key, and 6 accounts on 3 platforms as the sending account. Therefore, $n = |V| = 6$, and then $l_b = 5$ and $m = 32$. To determine the maximum interaction delay, we measured the time delay of Facebook, Twitter, and Weibo within 2 hours before the experiment. The specific method is as follows: use an automated tool to automatically perform a certain behavior at the expected timestamp. Then the time data that have been extracted on social networks can be converted to a timestamp form. The corresponding results are shown in Figure 5. Through observation, we have conducted more than 200 interactions within 2 hours, and the maximum interaction delay is $d = 4$. In addition, Figure 6(a) shown by the interaction patterns of regular users on social networks can determine $p = [0, 16]$.

4.2. Comparative Experiments and Evaluation Related to Embedding Capacity. There are 4 groups of experiments in this subsection. The first group of experiments provided data support for the three following groups of experiments.

4.2.1. Comparative Experiments and Evaluation on the Average Number of Bits Carried by a Single Behavior. When n and the secret message are determined, the length of secret message required in [21–23] is determined. Starting from the first character of *Gone with the Wind*, we select the character length of the secret message required for the 4 methods. The number of bits carried by a single behavior obtained by delivering a secret message once has large randomness. To ensure that the average number of bits carried by a single behavior is representative, the average number of bits carried by a single behavior is calculated by sending the secret message 500 and 1000 times, respectively, and is denoted as l_s. According to the definition of embedding capacity in Section 2,

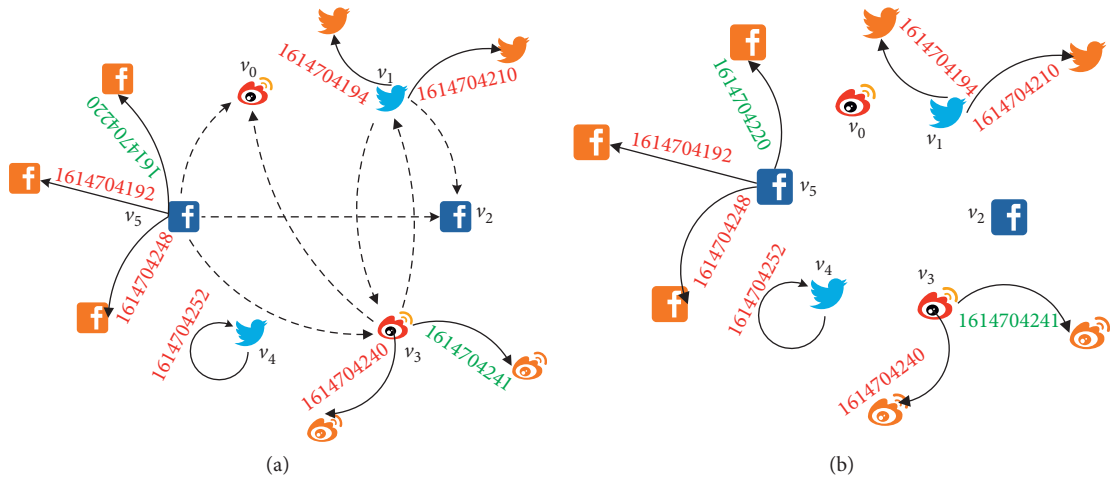

Figure 4: Actual interaction graph. (a) Mixed graph of logical and actual interactions. (b) Actual interactions graph.

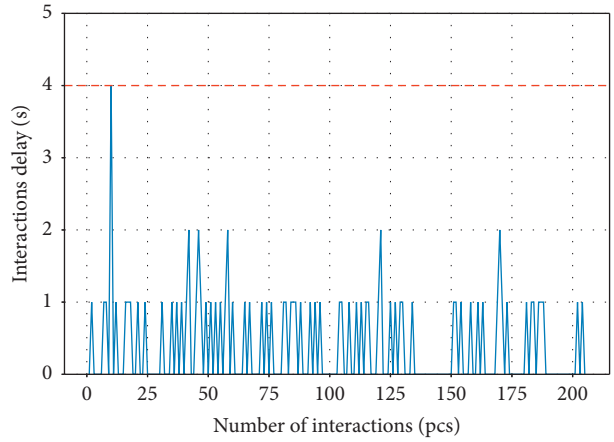

Figure 5: Interaction delay over a period of time.

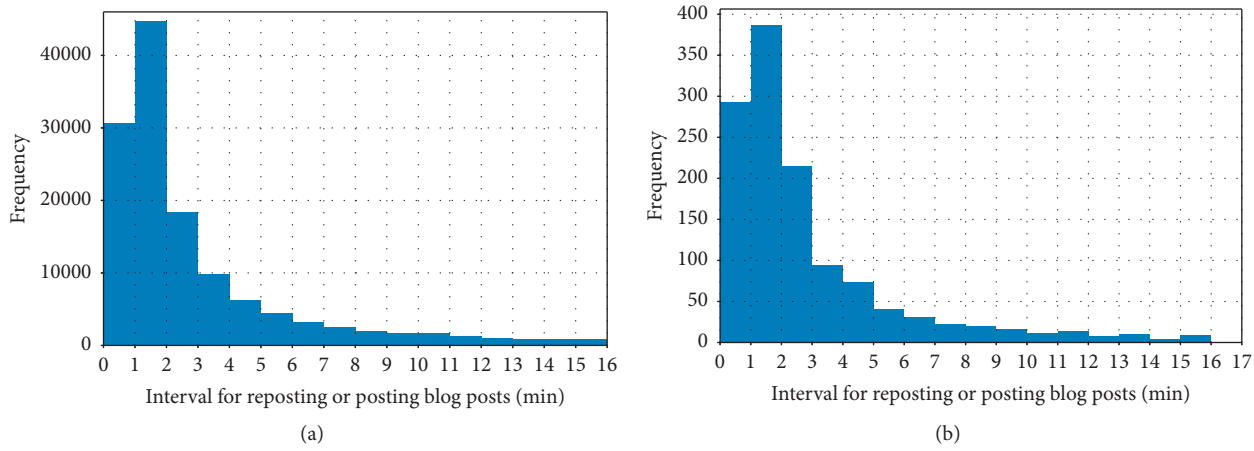

Figure 6: Interaction patterns. (a) Interaction patterns of ordinary users. (b) Interaction patterns of the proposed method.

l_s represents the embedding capacity of a single behavior. If the difference in the number of sendings is large and there is no significant difference in l_s, it means that l_s has stabilized and is representative.

Figures 7(a) and 7(b) show l_s calculated when $n \in [7, 16]$ and the secret message is sent 500 times. It shows that l_s of our method is higher in comparison to other methods. Besides, l_s is stable when sending 500 and 1000

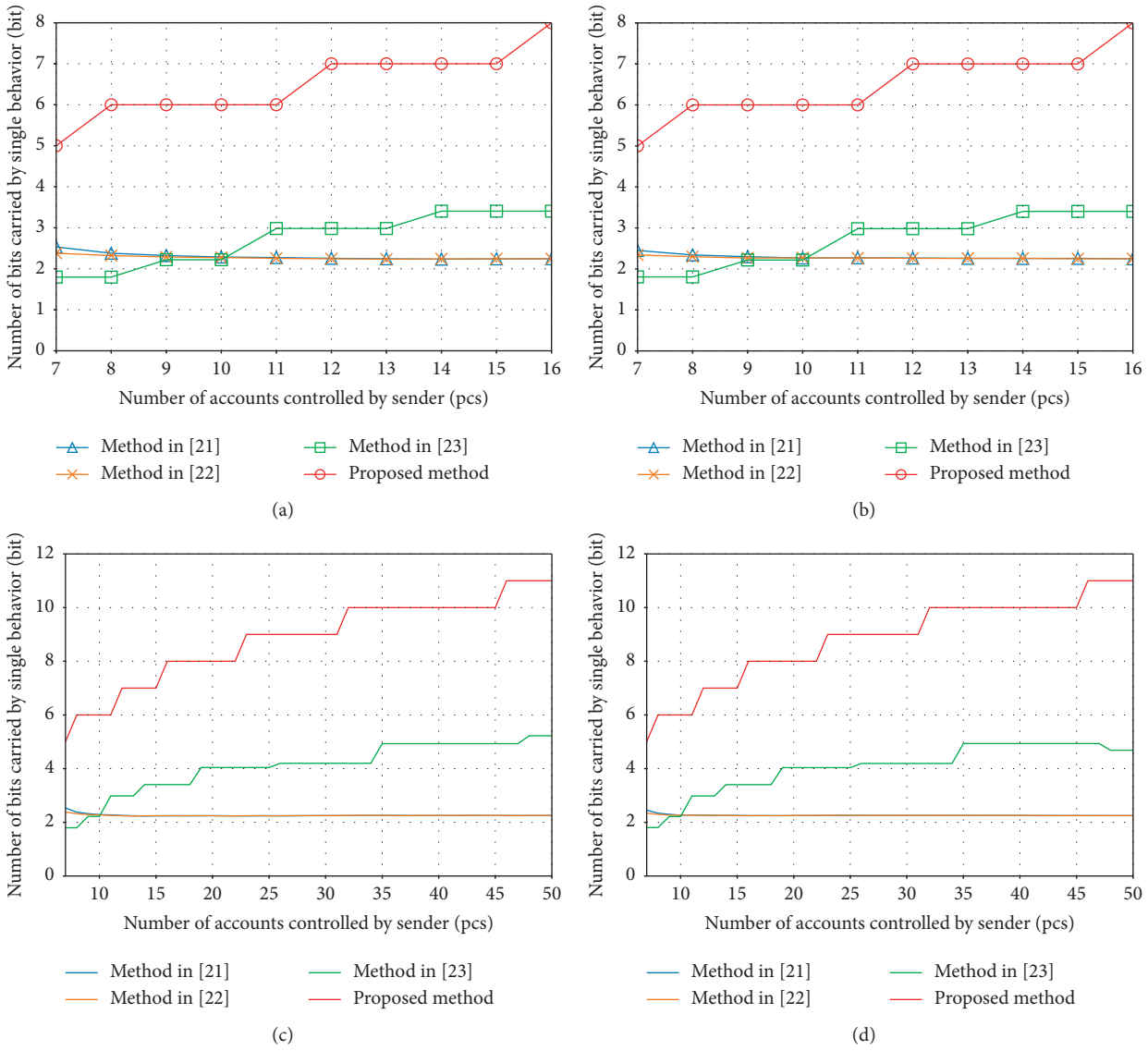

Figure 7: Comparison experiment of the average number of bits carried by a single behavior. (a) The average number of bits carried by a single behavior when $n \in [7, 16]$ and 500 messages are sent. (b) The average number of bits carried by a single behavior when $n \in [7, 16]$ and 1000 messages are sent. (c) The average number of bits carried by a single behavior when $n \in [7, 50]$ and 500 messages are sent. (d) The average number of bits carried by a single behavior when $n \in [7, 50]$ and 1000 messages are sent.

secret messages, respectively. Therefore, l_s is representative when sending 1000 secret messages. The following experiments will be based on this data.

To observe the trend of l_s under larger n, we did a group of experiments at $n \in [7, 50]$, which are shown in Figures 7(c) and 7(d). Figures 7(c) and 7(d) show that our method is still higher than other compared methods in the larger n range. As n increases, the superiority of the proposed method becomes more prominent.

4.2.2. Comparative Experiments and Evaluation on Embedding Capacity. Figure 8(a) shows the embedding capacity under different n when the number of behaviors is 10, and Figure 8(b) shows the embedding capacity trend of n in a larger range. Although they have the same trend as Figure 7, they have different meanings. For example, when $n = 35$, the embedding capacity of our method is 100 bits.

Table 4 shows the embedding capacity of the four methods when the number of behaviors is 10, and n is 10, 15, 20, 25, 30, 35, 40, 45, and 50, respectively. Compared with other methods, our method improves the embedding capacity. Table 4 shows that when n is 10, 15, 20, 25, 30, 35, 40, 45, and 50, as n increases, the embedding capacity shows an upward trend. When n is 20, the rate of the embedding capacity between our method and the compared method has a minimum of 1.98. When n is 50, the rate of embedding capacity between this method and the compared method has a maximum of 4.89. When n is the else value, the rate of the embedding capacity between our method and the compared method is between 1.98 and 4.89. Therefore, the

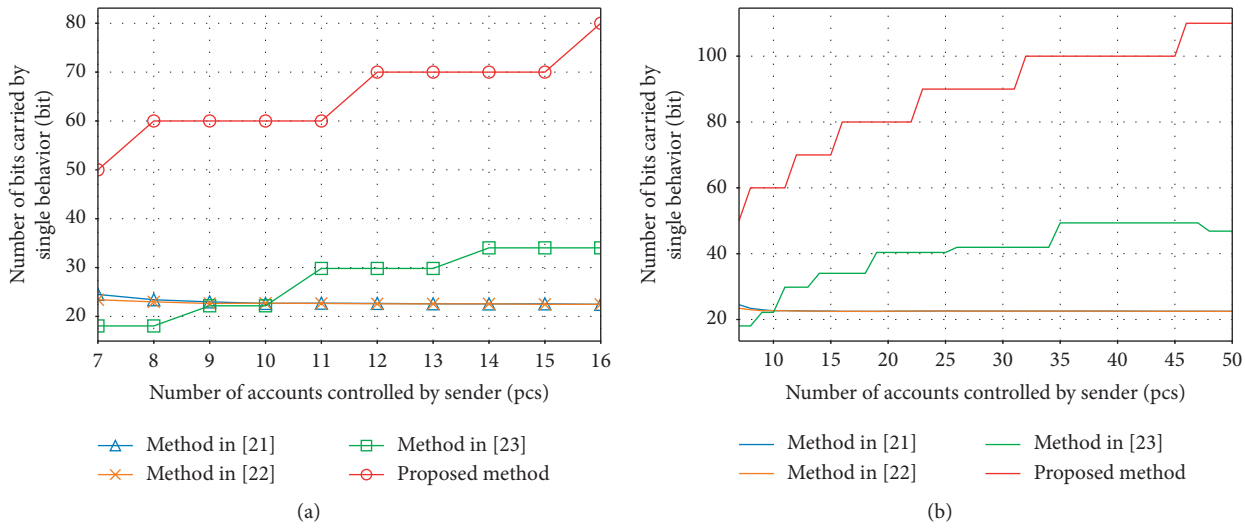

Figure 8: The embedded capacity in 10 interaction behaviors. (a) The embedding capacity of $n \in [7, 16]$. (b) The embedding capacity of $n \in [7, 50]$.

Table 4: Comparison data of embedding capacity (bit).

n		10	15	20	25	30	35	40	45	50
Method	Method in [21]	22.71	22.47	22.53	22.58	22.56	22.56	22.53	22.5	22.49
	Method in [22]	22.66	22.57	22.49	22.59	22.55	22.55	22.55	22.5	22.49
	Method in [23]	22.17	34.03	40.37	40.37	41.93	49.35	49.35	49.35	46.85
Proposed method		60	70	80	90	90	100	100	100	110
Minimum multiple of boost		2.64	2.06	1.98	2.23	2.14	2.02	2.02	2.03	2.34
Maximum multiple of improvement		2.7	3.12	3.56	3.99	3.99	4.43	4.44	4.44	4.89

experimental results show that the embedding capacity in our paper is significantly higher in comparison to the other methods. When $n \in [10, 50]$ and the step size is 5, it increases by 1.98 to 4.89 times.

4.2.3. Comparative Experiments and Evaluation on the Number of Behaviors. Suppose that the sender intends to send l_p binary bits, and the steganography method can send l_r binary bits each time. If $l_p < l_r$, the l_p binary sent is called the valid sent bits, and the remaining $l_r - l_p$ binary bits are invalid sent bits.

For the methods proposed in [21–23] and our method, when l_p bits are sent and l_p is not an integer multiple of l_b, $l_r = l_b - l_p \% l_b$ bits. The number of bits l_p sent by the sender may not be equal to l_r. In these four methods, 7 social accounts are controlled and 8-bit information needs to be sent. Our method [21–23] needs to be appended with 2, 7, 13, and 17 invalid bits, respectively, to convey the secret message. From this, the effective bit rate $r_b = l_p / l_r$ can be calculated as 0.8, 0.53, 0.38, and 0.33 for sending effective bits, respectively.

Figure 9 shows the number of behaviors generated by sending the same effective number of bits when n is 7. It shows that our method sends the same effective number of bits, and the number of behaviors generated is significantly lower compared to the other three methods. When using a suitable time interval to perform interactive behaviors, our method will consume less time. Figure 9(b) shows that as the number of effective bits sent increases, the number of behaviors of our method is still gradually increasing, which is smaller than the compared method. Compared with the other three methods, our method generates fewer behaviors when sending the same effective number of bits, and the effective bit rate is higher. At the same time, Figure 9 shows that our method is more flexible in sending message.

4.2.4. Comparative Experiments and Evaluation on Embedding Rate. The embedding rate in this paper refers to the number of bits that each account can carry; that is, embedding rate = c/n. The embedding capacity in this subsection is calculated when the number of behaviors is 10, and the embedding rate is shown in Figure 10. In Figure 10(a), the number of n ranges from 7 to 16. The embedding rate of our method is much higher compared to the other methods. In Figure 10(b), as n increases, the embedding rate generally shows a downward trend, but the embedding rate of our method is higher compared to the other methods.

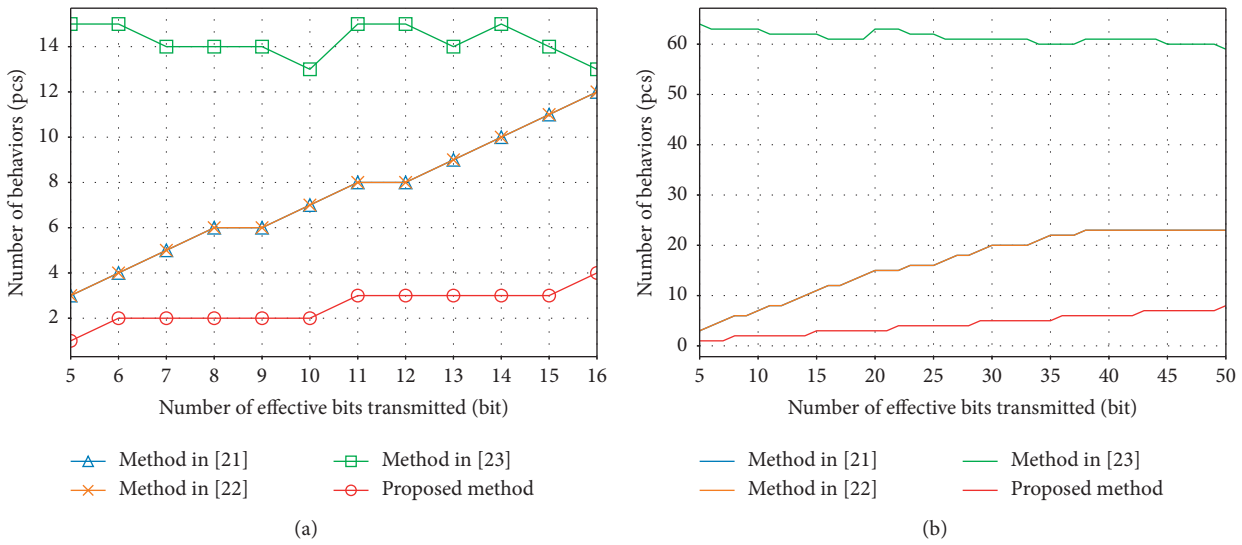

FIGURE 9: The number of behaviors generated under transmission of different effective bits. (a) The number of behaviors generated by the number of effective bits of [5, 16] when $n = 7$. (b) The number of behaviors generated by the number of effective bits of [5, 50] when $n = 7$.

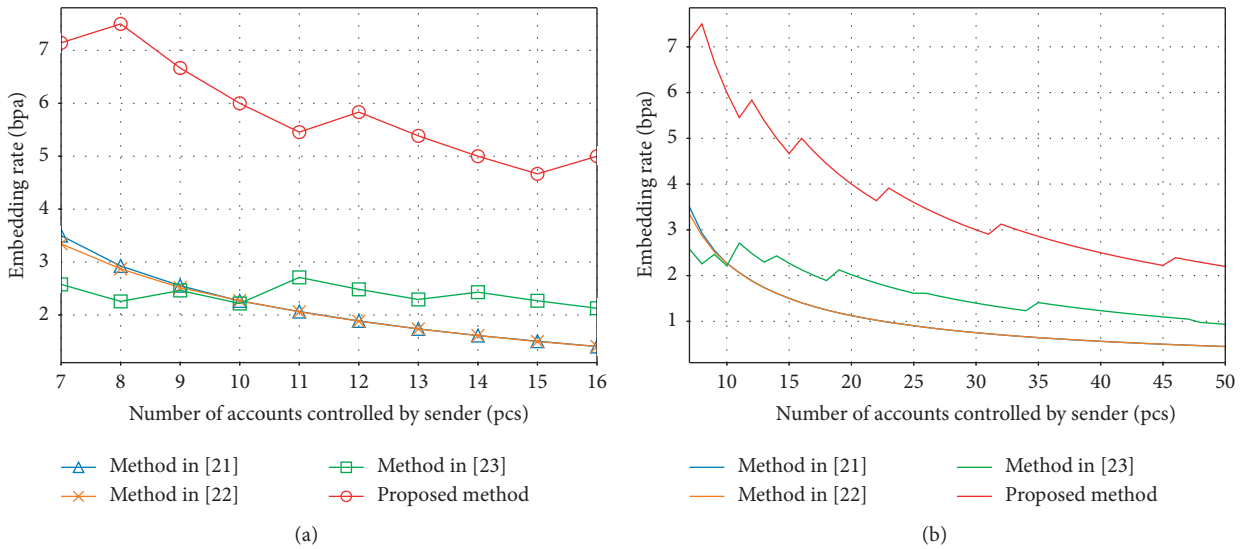

FIGURE 10: Embedding rate when the number of behaviors is 10. (a) Embedding rate when the number of behaviors is 10, $n \in [7, 16]$. (b) Embedding rate when the number of behaviors is 10, $n \in [7, 50]$.

This subsection conducts a comparative experiment of the average number of bits carried by a single behavior and conducts an experiment of embedding capacity on the basis of the average number of bits carried by a single behavior. The experimental results show that our method carries out an average number of bits and a large number of bits carried by a single behavior. The embedding capacity of each behavior has better performance. In addition, this subsection also conducts a comparative experiment on the number of behaviors and the embedding rate generated by the transmission of the effective number of bits. The experimental results show that when sending a certain length of effective bit, this method will generate fewer behaviors. Meanwhile, this method has a higher embedding rate for each account that sends secret message; that is, each account can carry more information.

4.3. Experiments and Evaluation on Detection Resistance. References [7, 16, 18] collect and fit the user's behavior frequency to avoid anomalies caused by the sender's behaviors. For this reason, we take the same method to ensure resistance to detection. Firstly, we crawled more than 160,000 posts from 94 bloggers in the "entertainment" section of public social networks and calculated the time interval between two adjacent posts or retweeted by the same blogger. Then, we merged the time interval data of all bloggers and sorted them and removed the top 10% and bottom 10% data to plot Figure 6(a). The time interval between retweets or posts on social networks is consistent with the frequency of interactions, and anomalous behavior can be avoided if the time interval between interactions on social networks is consistent with this pattern.

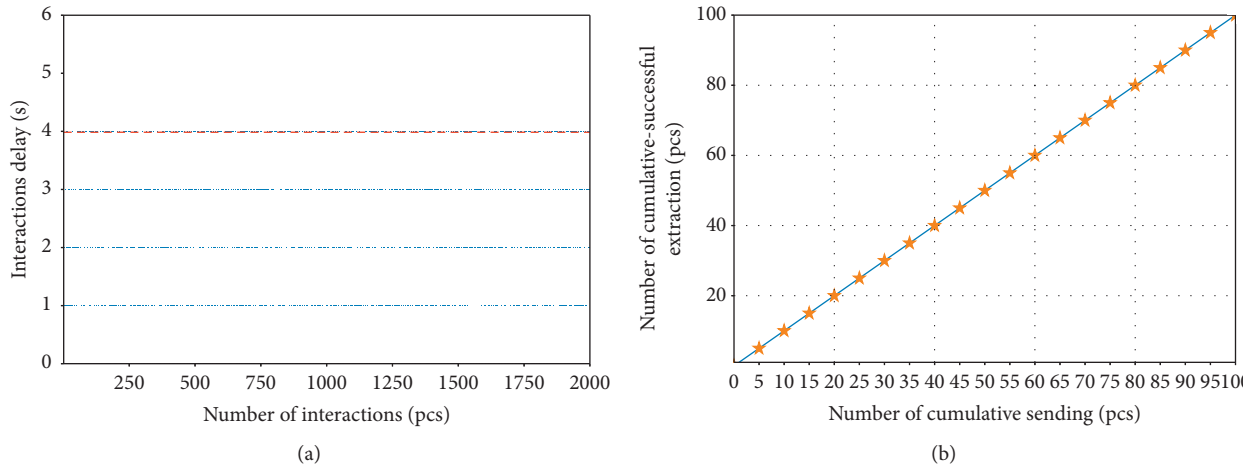

Figure 11: Robustness performance of the proposed method. (a) Interaction delay of the proposed method. (b) Extraction effect of the proposed method.

Table 5: Average time consumed to send 100 bits of data with different behavioral delay(s).

	$n = 7$			$n = 11$			$n = 15$		
Delay	1st group	2nd group	3rd group	1st group	2nd group	3rd group	1st group	2nd group	3rd group
$d = 2$	547.08	561.07	584.10	323.15	341.53	359.12	242.97	262.33	286.41
$d = 3$	554.20	569.94	587.16	331.38	348.69	364.47	247.34	265.33	284.78
$d = 4$	562.41	578.54	597.31	340.98	360.38	377.56	255.19	274.77	290.72

This experiment selects the first paragraph of text from *Gone with the Wind* as a secret message to send. Figure 6(a) shows the patterns of the interaction behaviors of ordinary users on social networks, and Figure 6(b) shows the patterns of the interactive behavior generated by our method. Based on Figure 5 of Section 4.1, $d = 4$ is the interaction time delay. As shown in Figure 6(b), the difference of the timestamps of the interaction behaviors during the secret message transmission is plotted as a frequency, and the time interval for generating the interaction behaviors is [0, 16]. The trend in Figure 6(b) is generally consistent with Figure 6(a) and the interval is also consistent, which indicates that the interaction behaviors generated by our method are consistent with the interaction behaviors of social network users. Therefore, our method can resist the analysis and detection of attackers in terms of behavioral anomalies.

4.4. Experiments and Evaluation on Robustness. Robustness is a measure of the stability of a method. When the carrier data passes through a lossy channel, the channel or the attacker may destroy the carrier data. The execution time of the behavior may be influenced by various factors, for which we need to verify the robustness of the method. In this paper, we ensure the robustness of the proposed method by sensing the interaction delay and using redundancy control mechanism. This method allows the receiver to extract the secret message correctly even if there is an interaction delay when the sender delivers the secret message. To verify the robustness of our method, we have done the following experiments.

From *Gone with the Wind*, part of the data is selected and sent 100 times as secret messages, and the maximum interaction delay $d = 4$ can be seen from Figure 5 in Section 4.1. A total of 24,800 behaviors are generated during the sending process, and the interaction delays of the first 2,000 behaviors are selected and plotted in Figure 11(a). Figure 11(b) shows the relationship between the cumulative number of secret messages sent and the cumulative number of successful extractions.

Figure 11(a) shows that the interaction delay during the sending process does not exceed 4 seconds. From Figure 11(a), it can be observed that when $d = 4$, the secret message sent is always extracted correctly. This implies that the interaction delay is less than 4 seconds during the period when the secret message is sent. Otherwise, it is difficult for all secret messages to be extracted correctly. It also indicates that the redundancy number of redundant mappings can resist the effect of interaction delay on the correct extraction of secret messages. Figures 11(a) and 11(b) show that our method can resist the effect of interaction delay, which shows our method is robust.

4.5. Experiments and Evaluation on Time Consumption. To avoid behavioral anomalies, the behavioral frequency of some users needs to be fitted when sending secret messages, which requires a certain time sacrifice. In addition, the interaction delay can also have an impact on the consumed time. To examine the impact of different interaction delays on time consumption when sending secret messages of a certain length, we designed a set of experiments. The experiments consider the effect of interaction delay and the

number of accounts on the time consumed. Firstly, 100 sentences from *Gone with the Wind* were selected as secret messages and the time consumed was counted. Then, it was divided into 3 groups and the average time was calculated. Finally, the consumed time is converted uniformly to the time consumed when sending 100 bits, and the statistics are shown in Table 5.

When $d = 2$ and $n = 7$, it consumed 547.08 seconds on average after the first group of messages. When $d = 2$ and $n = 11$, it consumed 323.15 seconds on average after the first group of messages. When $d = 3$ and $n = 7$, it consumed 554.20 seconds on average after the first group of messages. Table 5 shows that the time consumed increases as the delay increases and decreases as the number of accounts increases. The reason for the above situation is that as the delay increases, the number of redundant mappings goes up and takes up more time. When the number of accounts increases, the number of simultaneously generated behaviors also increases, which will consume less time.

5. Conclusion

To enhance the communication security of edge IoT devices, this paper proposes a behavioral steganography method based on timestamp modulation. It utilizes timestamps under social networks to achieve cross-platform covert communication and uses directed graphs to encode secret messages to improve the embedding capacity. At the same time, the method reduces the number of behaviors generated on a single platform by spreading the secret message across multiple social networks, which reduces the probability of an attacker discovering the covert communication. The experimental results show that the proposed method is highly robust and resistant to detection, and the embedding capacity, the number of effective bits transmitted, and the embedding rate are better than those of the compared methods. However, there is still a gap in embedding capacity between social network-based behavioral steganography and traditional methods. In future research, we will continue to focus on and follow up the privacy protection in SIoT and further improve the embedding capacity of behavioral steganography.

Data Availability

Some or all data, models, or codes that support the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

Mingliang Zhang mainly proposed the research ideas, completed the coding of the experiments, visualized the results of the experiments, and completed the original manuscript. Xiangyang Luo mainly developed the proposed method, directed the experiments, reviewed and corrected the manuscript, and funded this work. Pei Zhang mainly directed the color matching of the graphs, collected the experimental data, and reviewed and corrected the manuscript. Hao Li mainly provided the data needed for the experiments, investigated the progress of research in related works, and reviewed and corrected the manuscript. Yi Zhang mainly verified the experimental results, investigated the research progress of related directions, and reviewed and corrected the manuscript. Lingling Li mainly directed the experiments, corrected the constant form in the algorithm to the variable form to ensure the correctness of the algorithm, and reviewed and corrected the manuscript.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant nos. U1804263, 62172435 and 62002387) and the Zhongyuan Science and Technology Innovation Leading Talent Project of China (Grant no. 214200510019).

References

[1] L. Atzori, A. Iera, G. Morabito, and M. Nitti, "The Social Internet of Things (SIoT)–when social networks meet the internet of things: concept, architecture and network characterization," *Computer Networks*, vol. 56, no. 16, pp. 3594–3608, 2012.

[2] A. A. Abd El-Latif, B. Abd-El-Atty, and S. E. Venegas-Andraca, "A novel image steganography technique based on quantum substitution boxes," *Optics & Laser Technology*, vol. 116, pp. 92–102, 2019.

[3] S. K. Ghosal, J. K. Mandal, and R. Sarkar, "High payload image steganography based on laplacian of Gaussian (LoG) edge detector," *Multimedia Tools and Applications*, vol. 77, no. 23, Article ID 30403, 2018.

[4] J. Ye, J. Ni, and Y. Yi, "Deep learning hierarchical representations for image steganalysis," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2545–2557, 2017.

[5] J. Fridrich, M. Goljan, and R. Du, "Reliable detection of LSB steganography in color and grayscale images," in *Proceedings of the 2001 workshop on Multimedia and Security: New Challenges*, pp. 27–30, Ottawa, Canada, October 2001.

[6] B. Guo, Y. Ding, L. Yao, Y. Liang, and Z. Yu, "The future of false information detection on social media," *ACM Computing Surveys*, vol. 53, no. 4, pp. 1–36, 2020.

[7] Z. Yang, Y. Hu, Y. Huang, and Y. Zhang, "Behavioral security in covert communication systems," in *Proceedings of the 18th International Workshop on Digital Watermarking*, pp. 377–392, Chengdu, China, November 2019.

[8] A. Wilson, P. Blunsom, and A. Ker, "Linguistic steganography on twitter: hierarchical language modeling with manual interaction," in *Proceedings of the IS&T Electronic Imaging on Media Watermarking, Security, and Forensics*, San Francisco, CA, USA, February 2014.

[9] F. Bertini, S. G. Rizzo, and D. Montesi, "Can information hiding in social media posts represent a threat?" *Computer*, vol. 52, no. 10, pp. 52–60, 2019.

[10] S. Mahato, D. K. Yadav, and D. A. Khan, "A novel information hiding scheme based on social networking site viewers' public comments," *Journal of Information Security and Applications*, vol. 47, pp. 275–283, 2019.

[11] H. Kang, H. Wu, and X. Zhang, "Generative text steganography based on LSTM network and attention mechanism with keywords, Electronic Imaging," in *Proceedings of the IS&T Electronic Imaging on Media Watermarking, Security, and Forensics*, no. 4, Burlingame, CA, USA, January 2020.

[12] S. Nagaraja, A. Houmansadr, P. Piyawongwisal, V. Singh, P. Agarwal, and N. Borisov, "Stegobot: a covert social network botnet, Information Hiding," in *Proceedings of the 13th International Workshop on Information Hiding*, pp. 299–313, Prague, Czech Republic, May 2011.

[13] K. Muhammad, J. Ahmad, S. Rho, and S. W. Baik, "Image steganography for authenticity of visual contents in social networks," *Multimedia Tools and Applications*, vol. 76, no. 18, pp. 18985–19004, 2017.

[14] K. Muhammad, M. Sajjad, I. Mehmood, S. Rho, and S. W. Baik, "Image steganography using uncorrelated color space and its application for security of visual contents in online social networks," *Future Generation Computer Systems*, vol. 86, pp. 951–960, 2018.

[15] J. Tao, S. Li, X. Zhang, and Z. Wang, "Towards robust image steganography," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 2, pp. 594–600, 2018.

[16] N. Pantic and M. Husain, "Covert botnet command and control using twitter," in *Proceedings of the 31st Annual Computer Security Applications Conference*, pp. 171–180, Los Angeles, CA, USA, December 2015.

[17] X. Zhang, "Behavior steganography in social network," in *Proceedings of the 12th International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp. 21–23, Kaohsiung, China, November 2016.

[18] Y. Hu, Z. Wang, and X. Zhang, "Steganography in social networks based on behavioral correlation," *IETE Technical Review*, vol. 38, no. 1, pp. 1–7, 2021.

[19] S. Li, A. T. S. Ho, Z. Wang, and X. Zhang, "Lost in the digital wild: hiding information in digital activities," in *Proceedings of the 2nd International Workshop on Multimedia Privacy and Security*, pp. 27–37, Toronto, Canada, October 2018.

[20] C. S. Collberg and C. Thomborson, "Watermarking, tamper-proofing, and obfuscation - tools for software protection," *IEEE Transactions on Software Engineering*, vol. 28, no. 8, pp. 735–746, 2002.

[21] I. Nechta, "Steganography in social networks," in *Proceedings of the 2017 Siberian Symposium on Data Science and Engineering*, pp. 33–35, Novosibirsk, Russia, April 2017.

[22] H. Wu, W. Wang, J. Dong, and H. Wang, "New graph-theoretic approach to social steganography," in *Proceedings of the IS&T Electronic Imaging on Media Watermarking, Security, and Forensics*, Burlingame, CA, USA, January 2019.

[23] H. Wu, L. Zhou, J. Li, and X. Zhang, "Securing graph steganography over social networks via interaction remapping, Communications in Computer and Information Science," in *Proceedings of the 6th International Conference on Artificial Intelligence and Security*, pp. 303–312, Hohhot, China, July 2020.