symmetry

MDPI

Article

A Novel High-Capacity Behavioral Steganographic Method Combining Timestamp Modulation and Carrier Selection Based on Social Networks

Mingliang Zhang [1,2,3], Zhenyu Li [1,2,3,*], Pei Zhang [1,2,3], Yi Zhang [1,2,3] and Xiangyang Luo [1,2,3]

[1] Zhengzhou Institute of Information Science and Technology, Zhengzhou 450001, China; zmlmail2021@163.com (M.Z.); peizhang0810@163.com (P.Z.); tzyy4001@sina.com (Y.Z.); luoxy_ieu@sina.com (X.L.)
[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
* Correspondence: zheenyuli@gmail.com

Abstract: Behavioral steganography is a method used to achieve covert communication based on the sender's behaviors. It has attracted a great deal of attention due to its robustness and wide application scenarios. Current behavioral steganographic methods are still difficult to apply in practice because of their limited embedding capacity. To this end, this paper proposes a novel high-capacity behavioral steganographic method combining timestamp modulation and carrier selection based on social networks. It is a steganographic method where the embedding process and the extraction process are symmetric. When sending a secret message, the method first maps the secret message to a set of high-frequency keywords and divides them into keyword subsets. Then, the posts containing the keyword subsets are retrieved on social networks. Next, the positions of the keywords in the posts are modulated as the timestamps. Finally, the stego behaviors applied to the retrieved posts are generated. This method does not modify the content of the carrier, which ensures the naturalness of the posts. Compared with typical behavioral steganographic methods, the embedding capacity of the proposed method is 29.23∼51.47 times higher than that of others. Compared to generative text steganography, the embedding capacity is improved by 16.26∼23.94%.

Keywords: data hiding; behavioral steganography; post selection; social networks; covert communication

Citation: Zhang, M.; Li, Z.; Zhang, P.; Zhang, Y.; Luo, X. A Novel High-Capacity Behavioral Steganographic Method Combining Timestamp Modulation and Carrier Selection Based on Social Networks. *Symmetry* **2022**, *14*, 111. https://doi.org/10.3390/sym14010111

Academic Editor: Fengyong Li

Received: 8 December 2021
Accepted: 5 January 2022
Published: 8 January 2022

1. Introduction

Steganography [1] is a technique for sending secret messages without being perceived by others. The embedding process is generally symmetric with the extracting process. The sender uses the key to hide the secret message in the carrier, then the receiver uses the key to obtain the secret message from the carrier. Social networks are ideal carriers for steganography because of the wide geographical distribution of users, rich usage scenarios, and large data volumes involved. It is worth noting that covert communication in social networks is also carried out in a symmetric way. This allows the sender and receiver to achieve covert communication without establishing a peer-to-peer channel, and the communication behaviors are difficult for a third party to notice in particular. This ensures the concealment of the communication and the security of both the sender and the receiver. The study of the use of steganographic methods in social networks has important theoretical and practical value. It has attracted widespread attention from scholars in this field.

The carriers of steganographic methods based on social networks include image, text, audio, video, behavior, etc. Steganography can be grouped into carrier selection, carrier modification, and carrier synthesis (generation) according to different embedding principles [2]. Steganography based on social network carrier selection includes image selection steganography [3,4], text selection steganography [5,6], and video selection steganography [7,8]. When sending secret message, this type of method is used for finding a carrier

that conforms to the secret message through the constructed carrier database [9]. It does not modify the carrier data and can effectively resist attacks of steganalysis, but its low embedding capacity is still a challenge. The steganographic methods used for carrier modification based on social networks consist of image modification steganography [10,11], text modification steganography [12], audio modification steganography [13], and video modification steganography. They make use of the covert features of human organs and the redundant features of digital carriers to embed a secret message into the carriers by slightly modifying the social network carriers [14]. These methods are characterized by a high embedding capacity, robustness, and anti-detection performance. With the development of machine learning, however, the steganographic methods used for carrier modification may face new threats [15–18]. Social network-based generative carrier steganography methods are grouped into generative image steganography [19], generative text steganography [20,21], generative audio steganography [22], etc. Early generative methods conformed to statistical features, but the limitations of algorithms and computational power lead to content that does not conform to common sense and can be easily recognized [23]. With the development of artificial neural networks (ANN) and the increase in computing power, the statistical features and contents of the generated stego are more natural and their quality has been significantly improved. However, Yang et al. recently pointed out that the better the quality of the stego generated, the lower the concealment may be [21]. This has caused some experts and scholars to worry.

In recent years, social networks have developed rapidly. Scholars realize that social networks not only contain huge multimedia data but also rich behaviors, such as likes, forwards, posts, comments, and shares, which can be used for covert communication. Zhang [24] and Hu et al. [25] used WeChat, a mainstream social software in China, to realize covert communication. Li et al. [26] sent secret messages by reposting posts. Yang et al. [27] embed secret message through statistical features of posts. Nechta [28] proposed a method for covert communication through the behavior of adding friends. Wu et al. [29,30] performed covert communication on social networks by constructing graph structures. This type of method does not modify the carrier content, resulting in a higher robustness and invisibility. However, its embedding capacity still needs to be improved.

To improve the embedding capacity, this paper proposes a carrier selection high-capacity behavioral steganographic method based on timestamp modulation. The main work is as follows:

- A method is proposed to indicate to the positions of mapping keywords in posts through timestamps. This method greatly improves the embedding capacity while keeping the carrier natural.
- An adaptive retrieval algorithm for posts with mapping keywords is given. When the target post cannot be retrieved on a given social network, this algorithm can automatically adjust the matching parameters. This ensures that secret messages are sent successfully.

The remainder of this paper is organized as follows: Section 2 briefly introduces related work on behavioral steganography. Section 3 introduces the method proposed in this paper. The performance of the proposed method is analyzed in Section 4. After that, Section 5 gives the experimental results. Finally, we summarize the full text and discuss the direction of further research in the future.

2. Related Work

The work in this paper focuses on behavioral steganography on social networks based on timestamp modulation. To this end, this section focuses on the typical behavioral steganographic methods and timestamp steganographic methods. The advantages and limitations of the introduced methods are also discussed.

2.1. Behavioral Steganography Based on Social Networks

Many behavioral steganographic methods use graph theory to hide information, so we first introduce basic symbols. A graph is denoted by G. V is the set of vertices in G

and the set of edges in G is denoted by E. $|V|$ denotes the number of vertices in V and n denotes the number of accounts actually controlled by the sender. Corresponding to social networks, the vertices represent social network accounts and the edges represent interactive behaviors between accounts.

Nechta [28] proposes an undirected graph construction using "request to add friend" as an interactive behavior to implement a covert communication method. This method uses the adjacency function to define whether an edge exists between v_a and v_b in $G(V, E, \varphi)$, which is shown in Equation (1).

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

It first constructs an undirected graph G and traverses the vertex (v_a, v_b) to generate the edge, where $a < b$. When sending a secret message, only the edge corresponding to binary 1 generates behavior and the secret message can be sent. The embedding principle is shown in Figure 1.

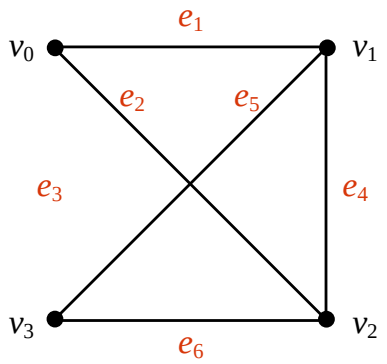

Figure 1. The embedding principle in [28].

The sequence of secret messages can be extracted by sequential splicing $S_M = \{e_1, e_2, \ldots, e_6\}$ = $\{110111\}$. This method is easy to implement and can successfully send a secret message to the receiver through a lossy channel.

Wu et al. [29] propose a covert communication method. This method uses undirected graphs to hide a secret message and directed graphs to hide the topology. Then, they enhance the security of the proposed method. This method uses additional vertices to hide the topology with strong security, but this also reduces the embedding capacity.

Wu et al. [30] propose a method to remap the correspondence between vertices of graph structure by key based on [29]. This method uses $n + 2$ vertices, which are indexed from v_0 to v_{n+1} and $V = \{v_1, v_2, \ldots, v_n\}$. The set of edges whose start and end points belong to the vertex set V is denoted as E, which is denoted by:

$$E = \{e_i = (v_a, v_b) | v_a, v_b \in V, 1 \leq i \leq m, 1 \leq a \neq b \leq n\}, \tag{2}$$

where m is the numbers of edges and m is an integer power of 2.

When embedding the secret message, the method first selects m edges in E based on a random seed R and assigns indexes to them. Then, the secret message is converted into a binary sequence and each $\log_2{}^m$ group is converted into a decimal sequence $D = \{d_1, d_2, \ldots, d_m\}$. Finally, the secret message is sent to the social network by performing $m + 1$ operations. When extracting the secret message, the receiver rebuilds the graph structure with the shared parameters.

2.2. Timestamp-Based Steganography

A timestamp is a form of recording time, which is calculated from 1 January 1970 00:00:00. Taking Beijing time as an example, the timestamp of 1 October 1980 00:00:00 is 339177600, and the timestamp of 1 October 2020 00:00:11 is 1601481611. The behaviors generated by social

network users have time attributes, which record the generation time of the behaviors such as likes, comments, and reposts.

Recently, some scholars have carried out research on covert communication based on timestamps. Giffin et al. [31] sent a secret message by modulating the timestamp of data packets. This method changes the timestamp by modifying the Linux kernel code and is able to deliver the secret message accurately on low-latency channels. However, on high-latency channels, the receiver may not be able to extract the secret message correctly. Neuner et al. [32] used file creation timestamps and modified timestamps in an operating system to hide a secret message. The timestamp for hiding secret messages does not differ from the normal timestamp and has some resistance to detection. Bedia et al. [33] used the timestamp field in IPV4 to achieve covert communication. Experiments show that the method is able to deliver secret messages correctly, but the embedding capacity of the method is low.

From the above introduction, we can see that the existing behavioral steganography methods have a good performance in terms of security, but that their embedding capacity still needs to be improved. In addition, the steganography method based on timestamps still has great limitations in its performance of embedding capacity. To improve the embedding capacity, this paper uses a combination of timestamps and social network steganography methods to correlate the positions of keywords in posts using behavioral timestamps. Among the related works most relevant to our paper are [28–30], as these are all based on behavioral steganography in social networks. In Section 5, we will compare these methods in detail.

3. Proposed Method

Firstly, this section introduces each step of the proposed method. Next, three key steps are explained in detail. Finally, the process of sending secret message is explained by an example.

In order to achieve the high embedding capacity of the behavioral steganography on social networks and at the same time ensure the naturalness of content and behaviors, we propose a symmetric covert communication method that combines the time attribute of the behavior and the carrier selection. It converts secret message into high-frequency mapping keywords and adaptively retrieves eligible keyword posts on social networks. The behavioral attributes are dynamically used to point to the positions of keywords in posts, which in turn greatly improves the embedding capacity of behavioral steganography.

There are 9 steps in this method, as shown in Figure 2. Steps 1–5 belong to the embedding process, and steps 6–9 belong to the extracting process.

Step 1: Map secret message. This first combines the commonly used secret words with the public word frequency table to generate a table named a self-built word frequency table. Then, a mapping relationship table is constructed by combining the self-built word frequency table with the public word frequency table. The words in the secret message are called secret keywords. Finally, the secret keywords are converted into mapping keywords by the mapping relationship table that has been disordered, which can map one word to another one. The purpose of the self-built word frequency table is to ensure that all keywords in the secret message exist in the mapping relationship table shared by the sender and receiver. The purpose of the mapping relationship table is to map a keyword in the self-built word frequency table to another keyword so as to prevent the secret message from directly appearing in the post and ensure the security of the secret message. The out-of-order mapping table is used to fine-tune the order of the keywords in the mapping relationship table according to the key. If the key does not match, the secret message cannot be extracted by the receiver.

Step 2: Measure behavioral delays. The purpose of measuring behavioral delays is to address the impact they have on timestamps. Automated interactions on social networks, behavioral delays for a while are recorded and the maximum behavioral delay is obtained.

Step 3: Adaptively retrieve mapping posts. The purpose of this step is to find a set of posts that can contain all the secret keywords. We set an initial number of keywords and

group the mapping keywords into subsets according to the initial number of keywords. Each subset is called a mapping group. The post that contains one mapping group is dynamically retrieved on the social networks and the post is called a mapping post. If the mapping post that contains the mapping group is not found, the initial number of keywords is shortened and the retrieval continues. If it is retrieved, the information of the post is saved. This retrieval process does not end until all the mapping keywords are retrieved.

Step 4: Generate a stego timestamp sequence. The purpose of generating the stego timestamp sequence is to hide the positions of the mapping keywords in the mapping post into timestamps. The timestamp of the post that already exists on social networks is no longer affected by the behavioral delay, and this kind of post is noted as an ordinary post. To hide the positions of the mapping keywords, the accounts controlled by the sender interact with other posts. The behaviors generated in this process are called interactive behaviors. The positions of the mapping keywords in the mapping posts are specified by both the timestamp of the ordinary posts and the timestamp of the interactive behaviors, which are shown in Figure 3. In Figure 3, the colored font indicates the timestamps of the interactive behaviors, such as t_1, t_2, and so on, while the black font indicates the timestamp of the ordinary post, such as t_3, t_6. When hiding the positions of the mapping keywords with timestamps, the sender first extracts all the positions of mapping keywords from all the mapping groups to form a mapping position sequence and converts the mapping position sequence into a binary position string. Then, the timestamps of ordinary posts and interactive behaviors can carry a secret message of different lengths. The binary position string is divided according to their length. Finally, the split binary string is used to modulate the timestamp sequence, which is called the stego timestamp sequence.

Step 5: Generate interactive behaviors. The purpose of generating interactive behaviors is to release the stego timestamp sequence to social networks. The sender's account interacts with mapping posts and ordinary posts at the time corresponding to the stego timestamps, generating interactive behaviors. The secret message will eventually be hidden on the social network.

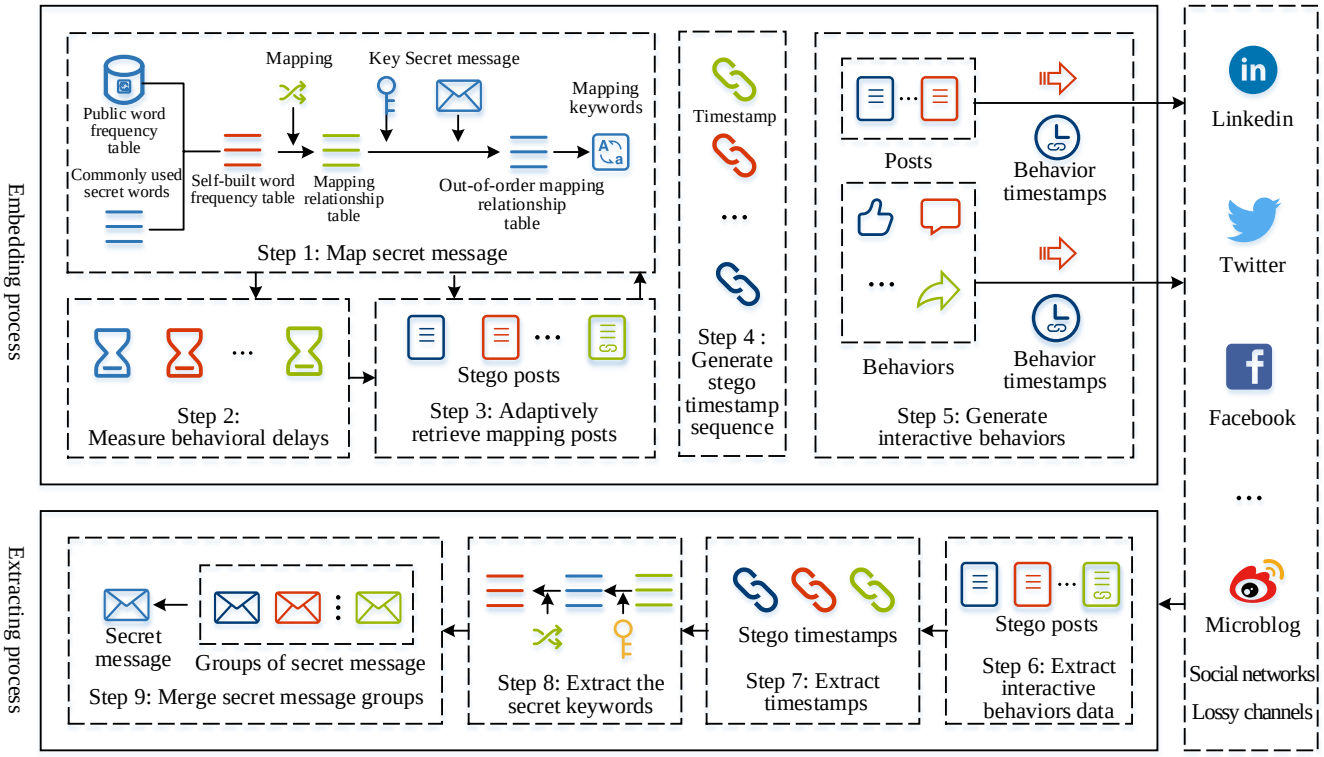

Figure 2. Steps of the proposed method.

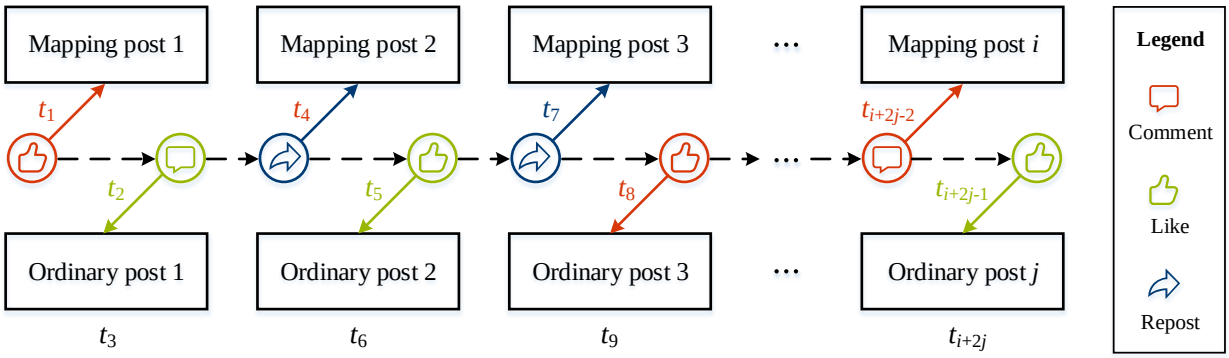

Figure 3. Schematic diagram of generating a sequence of stego timestamps.

The following are the steps used to extract the secret message.

Step 6: Extract interactive behavior data. The purpose of this step is to extract data from the accounts shared by the sender to extract secret keywords. According to the information such as the mapping relationship table and the number of accounts shared by the sender, the receiver extracts interactive behavior data over a period of time from the corresponding accounts of the social network. This data includes behaviors such as posted posts, reposted posts, comments, and likes.

Step 7: Extract timestamps. The purpose of extracting the timestamps is to obtain the positions of keywords. When extracting timestamp information from the interactive behavioral data, the stego timestamps are identified based on the secret key and identification fields.

Step 8: Extract the secret keywords. The positions of the keywords in the mapping posts are determined according to the mapping timestamps, and the mapping keywords are converted to secret keywords by the mapping relationship table. The embedding and extracting of a secret message is symmetric and the step is the reverse process of Step 4.

Step 9: Merge secret message groups. The secret message is extracted by merging the secret message groups.

There are three key steps in this method, which are: mapping a secret message, adaptively retrieving mapping posts, and generating a stego timestamp sequence. Next, the details of the key steps will be introduced in turn.

3.1. Map Secret Message

When sending a secret message, two problems will arise if secret keywords are carried directly by mapping posts. First, secret keywords may not be commonly used words. Even the public word frequency table may not contain certain out-of-the-way secret keywords. If they appear directly in the posts, this may cause anomalies. Second, keywords that are not frequently used have a low probability of appearing on social networks and may not be easily retrieved.

For this reason, we will take two measures to solve these two problems. On the one hand, a self-constructed word frequency table is constructed by combining the commonly used secret keywords with the public word frequency table. In other words, the self-built word frequency table contains both secret keywords and the public word frequency table. For example, when we send *the Declaration of Independence* as a secret message, there is a word "sufferable" that does not appear in the public word frequency table. For this reason, we have added "sufferable" to our selected public word frequency table. On the other hand, a mapping relationship is constructed between the self-built word frequency table and the public word frequency table. This is based on the principle that the commonly used secret keywords are mapped to the high-frequency words in the public word frequency table. In addition, the public word frequency table in the self-built word frequency table is mapped to the high-frequency words as much as possible. In this way, the secret keywords do not appear directly in the posts and the first problem is solved. The secret keywords are

mapped as high-frequency words that are easily retrieved on social networks, and thus the second problem is solved.

The disordered mapping relationship table is denoted as M_v, and the secret keywords table T_s to be sent by the sender is converted into the mapping keywords table T_m by M_v. This process can be formalized as:

$$T_m = M_v(T_s, k, M_r), \tag{3}$$

where k denotes the key and M_r denotes the mapping relationship table.

The word frequency table composed of frequently used secret keywords is denoted by T_f. The self-built word frequency table is composed of T_f and the public word frequency table T_w. The construction process of M_r is shown in Algorithm 1.

In the Algorithm 1, it can be found that a small number of low-frequency words in the self-built word frequency table are ignored.

Algorithm 1: Generation algorithm for the mapping relationship table.

Input: T_w, T_f
Output: M_r
$counter \leftarrow 0$
`/* The len function means the length of the sequence. */`
while ***len***$(T_f) - 1 >$ *counter* **do**
  $index \leftarrow T_w$.**getIndex(**T_f[*counter*]**)** `/* The getIndex function means obtaining the position of the element in the sequence. */`
  M_r.**append(**[T_f[*counter*]**,** T_w[*index*]]**)** `/* The append function means appending an element to the sequence. */`
  $counter \leftarrow counter$ **+ 1**
$counter \leftarrow 0$
while ***len***$(T_w) -$ ***len***$(T_f) - 1 >$ *counter* **do**
  $value \leftarrow T_w$[*counter*]
  if *value* ***not in*** M_r **then**
    M_r.**append**([*value*, T_w[*counter*]])
  $counter \leftarrow counter + 1$
return M_r

3.2. Adaptively Retrieve Mapping Posts

The initial number of keywords is denoted by l_i, which is used to specify the maximum number of mapping keywords contained in a mapping post. The number of mapping keywords contained in a retrieved mapping post is often not l_i. Its actual number of keywords is denoted by l_r. The algorithm for adaptively retrieving mapping posts is shown in Algorithm 2.

The Algorithm 2 first takes l_i mapping keywords from T_m. Next, the mapping posts containing l_i of the specified mapping keywords are retrieved on the social networks. If no keyword is found, the number of keywords searched in the previous round is subtracted by one and the retrieval continues. If a post is found, the number of keywords is set to l_i and the retrieval continues until all the mapping keywords are hidden in the found mapping posts. If it is still not found when $l_r = 0$, the retrieval fails, which rarely happens.

It is worth noting that l_i will affect the efficiency of retrieval. When l_i is too large, posts containing l_i mapping keywords may not be retrieved on social networks, which can lead to a decline in the number of keywords. For each reduction, the retrieval will be performed again, which will consume additional time.

Algorithm 2: Algorithm used for adaptively retrieving mapping posts.

Input: T_m, l_i
Output: S_r

$l_c \leftarrow l_i$
Initialize a result sequence S_r
$i_s \leftarrow 0$
while *true* **do**
  $M_s \leftarrow T_m[i_s : i_s + l_i]$
  if $M_s == Null$ **then**
    return S_r
  ret = **searchPosts(**M_s**)** `/* The searchPosts function means to search for posts on social networks */`
  if *ret* != *Null* **then**
    S_r**.append(***ret***)**
    $l_c \leftarrow l_i$
    $i_s \leftarrow i_s + l_c$
  else if $ret == Null$ & $l_c > 1$ **then**
    $l_c \leftarrow l_c - 1$
  else Retrieval failed ;

3.3. Generate Stego Timestamp Sequence

In this paper, timestamps of ordinary posts and interactive behaviors are used to hide the positions of mapping keywords. This step focuses on three issues regarding timestamps: first, the factors that influence the secret message in the timestamp cannot be extracted correctly; second, the amount of information that the timestamp can carry; third, the process by which the secret message is converted into timestamps.

To begin with, let us consider the first problem. It often happens in life that when we access a website, we may have to wait for a short period before we can see the content of the page. In fact, when posting a post on a social network, it may be necessary to wait a while for this post to be seen by other users, even if this time is very short. This situation is the behavioral delay, which may result in the secret message not being properly embedded in the timestamps. For example, a sender intends to deliver a decimal number 2 and start hiding the message at timestamp 1735150139. The sender reposts a post when the timestamp is 1735150141. However, the behavior is delayed due to a series of requests and is only recorded by the social network at timestamp 1735150142. When the secret message is extracted, the receiver subtracts 1735150142 from 1735150139 to obtain 3. At this point, the receiver extracts the wrong secret message. To solve this problem, the sender needs to measure the maximum behavioral delay d_{max} on the social networks for a while before sending the secret message. When sending a secret message, the secret message is converted to decimal and multiplied by $d_{max} + 1$ to prevent errors in the secret data. The detailed analysis and data can be found in Section 4.1.

Next, we give equations for the number of bits that can be carried by different kinds of timestamps. An interactive behavior timestamp uses l_t bits to encode the positions of keywords, which are calculated as follows ($\lfloor . \rfloor$ denotes rounding down):

$$l_v = (10^b - 1)/(d_{max} + 1), \tag{4}$$

where b denotes the last b digits of the timestamp used to encode the positions in mapping posts. The number of bits that can be carried by the timestamp of an interactive behavior is denoted by l_v, and $10^b - 1$ denotes the length that can be used for encoding. Considering the existence of behavioral delays in social networks, this decimal number, if used directly to encode the message, will cause the message hidden in the timestamp to change, resulting in the secret message not being extracted correctly by the receiver. Equation (4) gives some redundancy capability.

$$l_t = \lfloor \log_2 l_v \rfloor = \left\lfloor \log_2 (10^b - 1)/(d_{\max} + 1) \right\rfloor \tag{5}$$

When sending a secret message, the secret message needs to be converted to binary and then to decimal. For this purpose, taking the logarithm of l_v and rounding down will give the number of bits l_t of binary that l_v can represent.

In addition, the timestamp of an ordinary post is denoted by t_o. The number of bits it can carry is denoted by l_o, and is calculated as follows:

$$l_o = \lfloor \log_2 (t_c - t_s) \rfloor \tag{6}$$

The timestamp when the sender is about to send a secret message is denoted by t_c. The minimum timestamp of a certain social network is denoted by t_s. When t_s is the earliest timestamp of this social network, l_o takes the maximum value. t_c is the timestamp of a behavior that already exists on the social network and is no longer affected by the behavior delay, so there is no need to set redundant information for this timestamp. For Twitter, l_o can take the maximum value when t_s is the timestamp of the first post on the Twitter. The timestamp for the interaction between the account controlled by the sender and the mapping post is denoted by t_m, and the time for interaction with the ordinary post is denoted by t_b. The timestamp of an ordinary post is denoted as t_o. If corresponding to Figure 3, t_m, t_b, and t_o can be t_1, t_2, and t_3, respectively.

A sender is able to send a secret message by interacting with a mapping post and an ordinary post on a social network. The number of bits that these two behaviors can carry is denoted by l_s, and the equation is as follows:

$$l_s = l_t + l_o + l_t \tag{7}$$

$$= 2\log_2 \left\lfloor (10^b - 1)/(d_{\max} + 1) \right\rfloor + \log_2 (\lfloor t_c - t_s \rfloor) \tag{8}$$

Finally, an algorithm for the generation of the stego timestamps is given as Algorithm 3. Among the parameters, the sequence of mapping groups is denoted by S_p, the generated sequence of stego timestamps is denoted by S_t. The algorithm first calculates the number of bits that can be hidden by different types of timestamps and then generates a sequence of secret timestamps based on it.

Algorithm 3: Algorithm used for the generation of stego timestamps.

Input: S_p, t_c, b, d_{max}, t_s, n
Output: S_t

$l_g \leftarrow$ **len**(S_p)
$l_o \leftarrow \lfloor \log_2 (t_c - t_s) \rfloor$;
Get the position of each mapping keyword and convert it to binary to get S_b
$l_v \leftarrow (10^b - 1)$ // $(d_{max} + 1)$
$l_t \leftarrow \lfloor \log_2 l_v \rfloor$
$l_{tm} \leftarrow S_b[0{:}l_v * l_g]$
for $i = 0; i < l_g - 1, i$++ **do**
    S_t.**append**(t_c + **dec**(l_{tm})*b) /*dec function means binary to decimal*/
    $t_c \leftarrow t_c$ + **dec**(l_{tm})*b
$l_c \leftarrow l_g * l_v$
while *true* **do**
    $s_o \leftarrow S_p[l_c{:}l_c + l_o]$
    $l_c \leftarrow l_c + l_o$
    $s_b \leftarrow S_p[l_c{:}l_c + l_v]$
    $l_c \leftarrow l_c + l_v$
    $t_l \leftarrow$ get the last 1 element of S_t
    S_t.**append**(t_l−**dec**(s_o))
    $t_l \leftarrow$ get the penultimate element of S_t
    S_t.**append**(t_l + **dec**(s_b))
return S_t

3.4. Example

In this subsection, we briefly describe the process of embedding and extracting secret messages using an example. In this example, the sender sends a secret message to the receiver on a social network. Suppose we send a secret message as "This is a secret message." Their mapping keywords are "can", "a", "good", "not", "search", and "." by Algorithm 1 respectively, which form a mapping group. Suppose the minimum timestamp available for the social network carrying the secret message is 1633017600 and the timestamp for sending the secret message is 1577808000. The maximum behavioral delay of the current network is 2. The last 2 digits of the timestamp are used to convey secret messages. So, t_c = 1633017600, t_s = 1577808000, b = 3, d = 2. Calculated by Equations (4)–(7), l_o = 25, l_t = 8, l_s = 41. The mapping posts containing this mapping group is retrieved on social networks by Algorithm 2, and one of the results is shown in Figure 4. The higher the frequency of the mapping keyword in the public word frequency table, the more likely the post containing the keyword is to be retrieved. The positions of the mapping keywords in the post are 9, 19, 27, 13, 22, and 30.

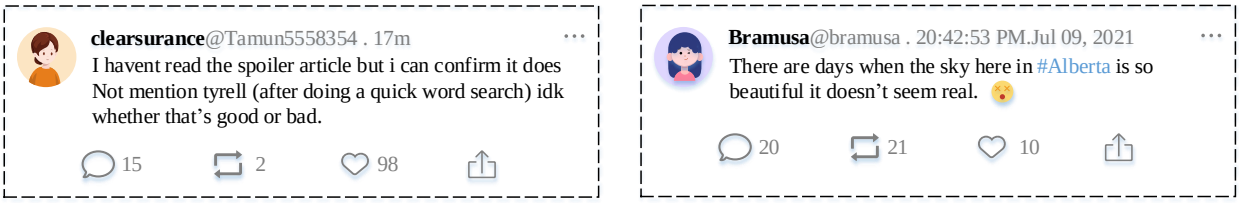

Figure 4. Posts retrieved based on the secret message sent.

The positions are converted to binary and are split into three groups. The number of each group is 8, 25, 8, and then each group is converted from binary to decimal. $t_m = t_c + 37 \times (d_{max} + 1) = 1633017711$, $t_o = t_m - 7183027 = 1625835084$, $t_b = t_m + 192 \times (d_{max} + 1) = 1633018687$. The calculation process and data are shown in Table 1. The mapping keywords are sent when the sender interacts with the post on the left side of Figure 4 at 1 October 2021 00:01:51. When the sender interacts with the post to the right of Figure 4 at 1 October 2021 00:09:36, the positions of mapping keywords are sent. Corresponding to the Figure 3, t_m is equivalent to t_1, t_b is equivalent to t_2, and t_o is equivalent to t_3. The extracting process is the inverse of the embedding process and will not be repeated here.

Table 1. The process when sending a secret message.

Position	9	19	27	13	22	30
Binary string	001001	010011	011011	001101	010110	011110
Split binary string	00100101	001101101100110101010110011				11000000
Converted Decimal	37	7183027				192
Timestamp calculation	$t_c + 37 \times (d_{max} + 1)$	$t_m - 7183027$				$t_m + 192 \times (d_{max} + 1)$
Timestamp	1633017711	1625835084				1633018687
Corresponding time	1 October 2021 00:01:51	9 July 2021 20:51:24				1 October 2021 00:18:07

4. Performance Analysis

Embedding capacity and robustness are important metrics for measuring the performance of steganographic methods. In this paper, embedding capacity refers to the number of bits carried by each behavior. Robustness generally refers to the property that the stego can be successfully communicated despite being attacked by an attacker or a channeled attack [11]. This section will analyze the performance of our method from these two aspects.

4.1. Robustness

In addition to using text to hide the mapping keywords, this paper also uses timestamps to hide the positions of the keywords. Generally, the text data can exist stably on social networks, and the text content except for blank characters is not modified. The factor that threatens the robustness of this method originates from the timestamps. This is because the expected behaviors must be executed at the same time as the time recorded by the social network. In practice, it is difficult to satisfy this condition.

Figure 5a shows the expected execution time and the actual execution time of behaviors for Weibo, Twitter, and Facebook for a certain period. The dotted line represents the execution time of interactive behavior, while the solid line represents the time when the interactive behavior is recorded by the social network. From Figure 5a, we can see that the time being recorded is not equal to the time being executed in most cases, which indicates that behavioral delays are present in most cases.

In Figure 5b, the value of the behavioral delays is obtained by calculating the difference between the actual time and the expected time, which indicates the maximum behavioral delay $d_{max} = 2$ during this time.

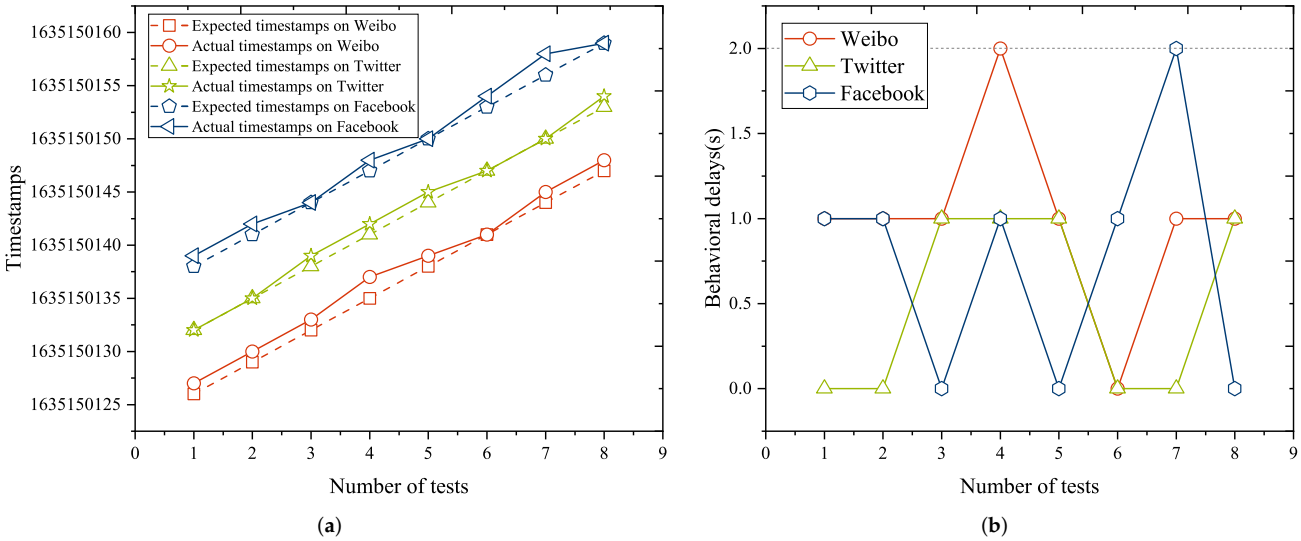

Figure 5. Behavioral delays on different social networks. (**a**) Differences in behavioral timestamps for a while; (**b**) Behavioral delays of multiple social networks for a while.

If there is a behavioral delay when sending a secret message, this will lead to an error in the positions of the keywords and subsequently cause the secret message not to be extracted correctly. To solve this problem, this paper uses a time-redundant control mechanism to measure the behavioral delays over some time and obtain the maximum value d_{max}. Then, the secret message that is converted to decimal is multiplied by $d_{max} + 1$ to resist the effect of behavior delays. For example, the secret data sent by the sender are 47, $d_{max} = 2$, and the current timestamp $t_c = 1635150139$. Next, an behavior can be executed until the timestamp is 1635150141. If the behavior is delayed by 2 s, the time of execution is recorded by the social network as 1635150143. The receiver divides this timestamp by 3 and takes the value downward, and the value obtained is still 47. Therefore, the robustness of this method on both text content and timestamp can be guaranteed.

4.2. Embedded Capacity

The size of the embedding capacity is influenced by various factors. It is related to variables such as the last b digits of the timestamp, the ordinary post timestamp l_o, and the maximum behavioral delay d_{max}. There are two questions that need to be addressed in this subsection. The first question is what is the appropriate value or range for each variable to take? For b, if $b \leq 2$ will lead to frequent operations and cause abnormal behavior. If b is too

large, it can encode more information but consumes too much time. In addition, we refer to many references to make l_o as maximum as possible and to make this method applicable to mainstream social networks. According to [34–36], mainstream social networks such as Facebook, Twitter, and Weibo already had a large amount of user and post data by 2011. Therefore, the starting time can be set to 1 January 2011 00:00:00, corresponding to a timestamp is t_s = 1293811200. Suppose the current time is 1 October 2021 00:00:00, then t_c = 1633017600, and l_o = 28 according to Equation (6). By Section 4.1, d_{max} should be greater than or equal to 2.

Table 2 gives the corresponding values of l_v, $\log_2 l_v$, $\lfloor \log_2 l_v \rfloor$ and l_s for different d_{max}. It shows that when $d_{max} = 1/2$ ($d_{max} = 1$ or $d_{max} = 2$), the value of l_s is the same. When $d_{max} = 3/4/5/6$, l_s is the same. When d_{max} are the same, the same l_s means the same number of bits sent.

Another question is how many bits can represent a position. To answer this question, we first crawl 10,731,668 posts from Twitter. We select 300,000 posts and divided them into 3 groups. Next, each post is divided into words, and the number of words in the posts is counted. Finally, the frequency of keywords appearing in each group of posts is counted. The corresponding experimental results are shown in Figure 6.

Table 2. Number of bits that can be carried by timestamp with different time delays.

d_{max}	l_v	$\log_2 l_v$	l_t	l_s	l_i
0	999.00	9.96	9	46	7
1	499.50	8.96	8	44	7
2	333.33	8.38	8	44	7
3	249.75	7.96	7	42	7
4	199.80	7.64	7	42	7
5	166.50	7.38	7	42	7
6	142.71	7.16	7	42	7
7	124.88	6.96	6	40	6

In this paper, the number of words in a post is denoted by X. Figure 6 shows $X \in [1, 60]$ for almost all the posts. According to $2^6 = 64$, we know that using 6 bits we can represent 64 positions, which can satisfy the need to index positions in the post. According to this, the last column of Table 2 gives the value of l_i. The remaining part is distributed in $[61, 67]$.

Let p_i denote the probability that the number of keywords is i. The expectation $E(X)$ of X is given by the following equation:

$$E(\mathrm{X}) = \sum_{i=1}^{67} i \cdot p_i \tag{9}$$

The results calculated by Equation (9) have been labeled in Figure 6. By calculating the average of 3 experiments, $E(X)$ = 22.67. This shows that the average number of words per post is 22.67, by which the average embedding capacity can be obtained. Assuming that each word contains an average of $\bar{l}$ letters, the average embedding capacity is $\bar{l} \cdot E(X) \cdot 8$. The actual embedding capacity depends on the secret message sent, and the experimental results can be seen in Section 5.2.

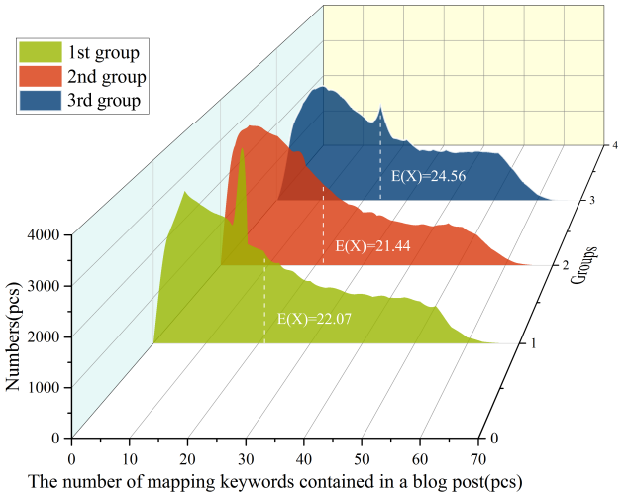

Figure 6. Number of words in each post.

5. Experiments

In this section, we design several groups of experiments to evaluate the performance of our method in terms of embedding capacity and number of behaviors. We use the third-party library Selenium [37] for python with the WebDriver [38] to control the browser, using them as automation tools which can simulate human behavior. It can be used to implement functions such as clicking the button on the webpages, inputting data, and obtaining data. In addition, we also conduct experiments and evaluations on the initial number of keywords l_i.

5.1. Experimental Settings

In our experiments, we use *the Declaration of Independence* as the secret message and the Kaggle [39] word frequency database as the public word frequency table. A total of 10,731,668 post data were crawled on Twitter using Twint [40]. They contain fields such as username, post, creation time, etc. Twint is a crawler tool on Github, which can accurately obtain posts, comments, and followers and other information through keywords within a stipulated time period. We use it as a tool for retrieving posts on social networks. Figure 5 shows that d_{max} should be greater than or equal to 2. For this reason, we conduct the following experiments under $d_{max} = 2$.

5.2. Comparative Experiments on Embedding Capacity

The maximum behavioral delay d_{max} and the initial number of keywords l_i carried by the mapping posts on social networks can have an impact on the embedding capacity. To this end, we first design a set of comparative experiments with a varying number of keywords to test the effect. The results are shown in Figure 7. Next, we select a set of parameters to compare with existing behavioral steganography work, whose data are shown in Table 3. Finally, our proposed method is compared with the generative text steganography method in terms of embedding capacity.

In the experiment corresponding to Figure 7, the secret message is divided into mapping groups by Algorithm 2. The number of bits that can be sent for a mapping post containing a secret message is shown in Figure 7. We can find the maximum, minimum, and average values carried by the groups.

When $l_i = 7$, the average value of the secret message that each group can carry is 164.95 bits, the highest is 328.00 bits and the lowest is 40.00 bits. When $l_i = 9$, both the maximum and average values increase to 352.00 bits and 167.23 bits, respectively. Figure 7 shows that when l_i is in a certain range, the amount of information carried by the group is gradually increased as l_i is raised.

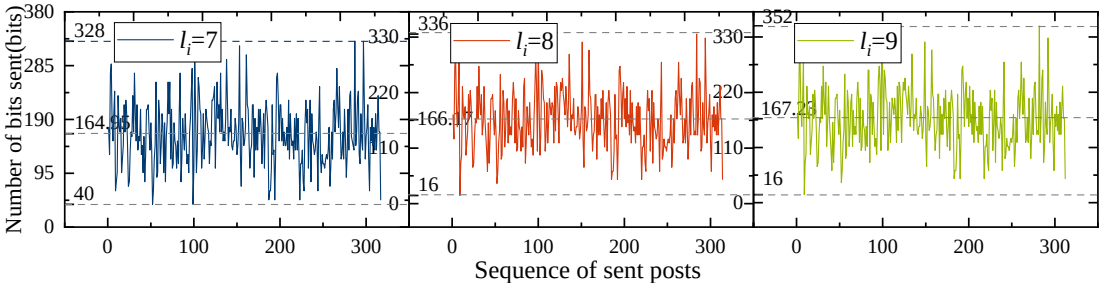

Figure 7. Number of bits carried by each group when sending *the Declaration of Independence*.

Table 3. Embedding capacity when d_{max} = 2 and l_i = 7 (bits).

	n = 7				n = 8				n = 9			
Group Number	**[28]**	**[29]**	**[30]**	**Our**	**[28]**	**[29]**	**[30]**	**Our**	**[28]**	**[29]**	**[30]**	**Our**
1–50	2.25	2.26	1.81	83.28	2.23	2.25	1.81	83.28	2.24	2.23	2.20	83.28
51–100	2.39	2.44	1.79	80.00	2.34	2.39	1.79	80.00	2.32	2.34	2.17	80.00
101–150	2.32	2.55	1.82	92.08	2.35	2.32	1.82	92.08	2.30	2.35	2.15	92.08
151–200	2.43	2.32	1.84	83.92	2.28	2.43	1.84	83.92	2.21	2.28	2.14	83.92
201–250	2.34	2.44	1.83	74.48	2.23	2.34	1.83	74.48	2.17	2.23	2.14	74.48
250–300	2.20	2.52	1.87	86.24	2.17	2.20	1.87	86.24	2.23	2.17	2.13	86.24
Mean value	2.32	2.42	1.83	83.33	2.27	2.32	1.83	83.33	2.24	2.27	2.16	83.33
Minimum	2.20	2.26	1.79	74.48	2.17	2.20	1.79	74.48	2.17	2.17	2.13	74.48
Maximum	2.43	2.55	1.87	92.08	2.35	2.43	1.87	92.08	2.32	2.35	2.20	92.08
Minimum multiple	**29.23**				37.91				31.64			
Maximum multiple	51.47				**51.47**				**43.15**			

To verify the performance of the embedding capacity, we implement the [28–30] and compare them with our method. When l_i = 7, *the Declaration of Independence* used as a secret message needs to be sent 317 times using our method. Every 50 times data is sent, the average of the bits carried by each behavior is calculated once. The experimental data are shown in Table 3. When n = 7, the 101st to 150th mapping posts are sent. The average embedding capacity of [28–30] and our method are 2.32, 2.55, 1.82, 92.08 bits, respectively. As n increases, there is a decreasing trend in the embedding capacity of the compared methods. This is caused by the increase in the number of behaviors, while the change in n has no effect on our proposed method. The maximum value of the embedding capacity of this method divided by the minimum value of the compared method yields the maximum multiplication of the embedding capacity increase. On the contrary, the minimum multiplication of the enhancement can be obtained. The Table 3 shows that our method has higher performance in embedding capacity than compared methods. It is 29.23∼51.47 times higher than the compared methods.

Our method is also compared with the generative text steganography method [20]. The embedding capacity of a generative text steganography method is the number of bits carried per word. According to Section 4.2, each post contains an average of 22.67 words. For this, we can get the corresponding embedding capacity under different parameters in [11]. The [19] point out that when [20] carries 4 bits per word, the probability of being recognized reaches 0.8. For this reason, we conduct a comparative experiment below 4 bpw. The experimental results are shown in the Table 4.

When each word carries 3 bits, each post can carry 68.01 bits in [20]. For our method, at d_{max} = 2 and l_i = 7, it can carry 83.60 bits per time. As l_i increases, the embedding capacity will increase, but the retrieval efficiency will decrease. When d_{max} increases, the embedding capacity decreases. The embedding capacity will exceed our method when carrying 4 bits per word in [20], but it has a high probability of 0.8 to be recognized, while our method uses

natural text without that risk. The [20] is safer when carrying 3 bits per word, and each post can carry 68.01 bits. Compared with this method, our method improves by 16.26∼23.94%. Thus, both groups of comparative experiments show that our method is superior in terms of embedding capacity.

Table 4. Comparison experiment with [20] on embedding capacity (bits).

	[20]				l_i		
d_{max}	**1 bpw**	**2 bpw**	**3 bpw**	**4 bpw**	**7**	**8**	**9**
2	22.67	45.34	**68.01**	90.68	83.60	84.16	**84.29**
3					81.27	81.76	81.92
7					**79.07**	79.38	79.53

5.3. Comparative Experiments on the Number of Behaviors

Frequent and large numbers of behaviors performed by the same user on social networks may cause behavioral anomalies. In addition, the methods compared in the paper achieve steganographic communication due to the use of graph theory, and they require a fixed length of information to be passed each time during the transmission of a secret message. If the sender sends a secret message that does not reach this length, a certain amount of redundant information is appended until this fixed length is satisfied. Therefore, the fewer the number of behaviors generated by sending a secret message, the better. For this purpose, we design a set of comparative experiments. When sending secret messages of the same length, we compare their performance in terms of the number of behaviors. The experimental results are shown in Figure 8a.

When sending 16-bit information, the number of behaviors required for [28–30] and our method are 10, 10, 13, and 2, respectively. It is worth noting that the number of behaviors for the compared methods fluctuates with the different messages sent. Specific experimental data are provided in Figure 8a.

In Figure 8b, we can observe the trend in the number of behaviors for each method as the number of bits sent increases. Figure 8b shows that as the number of bits passed increases, the number of behaviors for our method is lower than the compared methods.

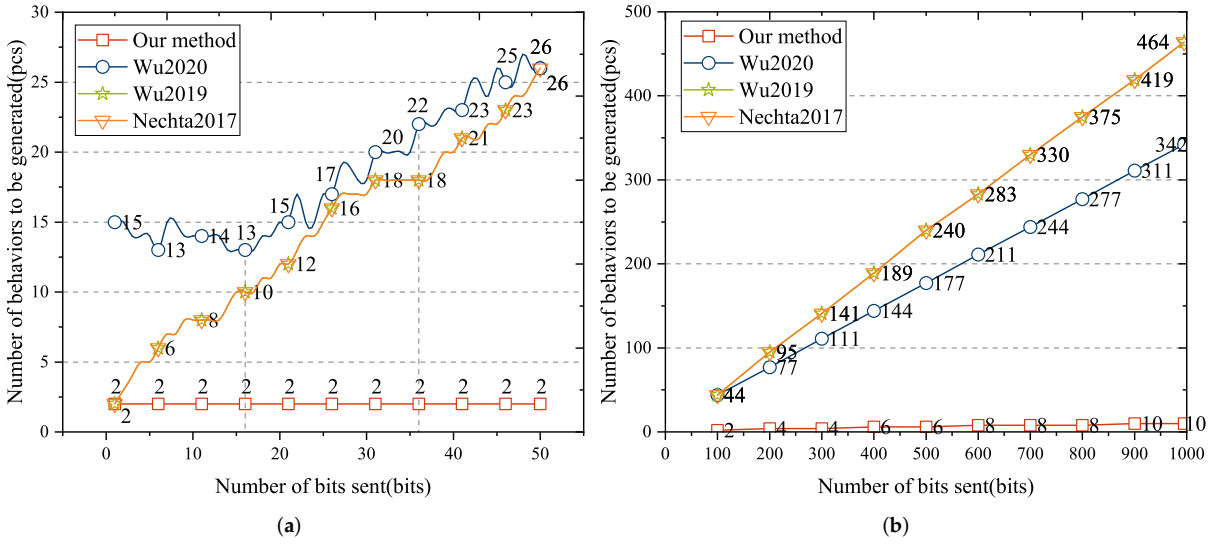

Figure 8. Number of behaviors generated by each method. (**a**) When sending 1–50 bits; (**b**) When sending 1–1000 bits. (Nechta2017, Wu2019 and Wu2020 correspond to references [28], [29], [30], respectively.)

5.4. Selection of Parameter l_i

When sending a secret message, the initial number of keywords in the mapping post is denoted as l_i. When a post containing l_i keywords is not retrieved, the mapping keyword sequence will be shortened, and then we will continue to retrieve appropriate posts. Considering that repeated retrievals affect the sending efficiency, we design a set of experiments to compare the sending success rate. This experiment can guide senders to set the appropriate l_i to achieve covert communication in an efficient way. The corresponding experimental results are shown in Figure 9. The actual number of keywords l_r in the mapping post with values less than 3% are not marked in Figure 9.

The different colors in Figure 9 indicate the different l_r. The percentage of each color indicates the probability of a mapping post being sent successfully when the actual number of mapping keywords is l_r. When $l_i = 7$, the success rate is 29.02%. When l_i is 7, the probability that a mapping post containing less than 6 keywords is sent successfully is 87%. When $l_r = 4$, mapping posts are more likely to be retrieved. Moreover, for different l_i, the success rates of their different l_r are all summed up equal to 100%, which indicates that the secret message can always be sent successfully.

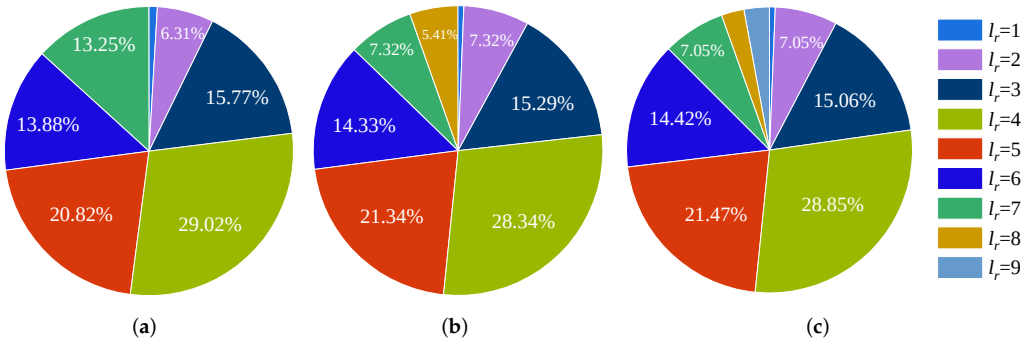

Figure 9. Success rate of sending with l_r mapping keywords for different l_i. (**a**) Success rate of each l_r when $l_i = 7$; (**b**) success rate of each l_r when $l_i = 8$; (**c**) success rate of each l_r when $l_i = 9$.

6. Conclusions

In this paper, we propose the use of a high-capacity behavioral steganography method on social networks based on carrier selection with timestamp modulation. This method uses natural post data to carry the converted secret message and utilizes timestamps of social network behaviors to indicate the positions of mapping keywords in the posts. Compared with typical behavioral steganographic methods, the embedding capacity of the proposed method is 29.23∼51.47 times higher than others because our proposed method can carry several words. Compared to generative text steganography, the embedding capacity is improved by 16.26∼23.94%. In future research, we will continue to work on increasing the embedding capacity of behavioral steganography.

Author Contributions: Methodology, M.Z.; validation, P.Z. and Y.Z.; formal analysis, M.Z., P.Z. and Y.Z.; investigation, P.Z. and Y.Z.; data curation, M.Z., P.Z. and Y.Z.; writing—original draft preparation, M.Z.; writing—review and editing, Z.L. and X.L.; visualization, M.Z.; supervision, X.L. and Z.L.; funding acquisition, X.L. and Z.L. All authors have read and agreed to the published version of the manuscript.

Funding: 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).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: Not applicable.

Acknowledgments: We would like to thank Hao Li, Xiuting Wang, Guo Wei, Yanmei Liu and others for helping us check the details and providing us with valuable suggestions in this paper.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Neil, J.F.; Sushil, J. Exploring steganography: Seeing the unseen. *Computer* **1998**, *31*, 26–34.
2. Fridrich, J. *Steganography in Digital Media: Principles, Algorithms, and Applications*, 1st ed.; Cambridge University Press: New York, NY, USA, 2009; pp. 1–443.
3. Zhang, X.; Peng, F.; Long, M. Robust coverless image steganography based on DCT and lda topic classification. *IEEE Trans. Multimedia* **2018**, *20*, 3223–3238. [CrossRef]
4. Luo, Y.; Qin, J.; Xiang, X.; Tan, Y. Coverless image steganography based on multi-object recognition. *IEEE Trans. Circuits Syst. Video Technol.* **2021**, *7*, 2779–2791. [CrossRef]
5. Chen, X.; Sun, H.; Tobe, Y.; Zhou, Z.; Sun, X. Coverless information hiding method based on the chinese mathematical expression. In Proceedings of the 1st International Conference on Cloud Computing and Security (ICCCS), Nanjing, China, 13–15 August 2015; pp. 133–143.
6. Chen, X.; Chen, S. Text coverless information hiding based on compound and selection of words. *Soft Comput.* **2019**, *23*, 6323–6330. [CrossRef]
7. Tan, Y.; Qin, J.; Xiang, X.; Zhang, C.; Wang, Z. Coverless steganography based on motion analysis of video. *Secur. Commun. Netw.* **2021**, *2021*, 5554058. [CrossRef]
8. Pan, N.; Qin, J.; Tan, Y.; Xiang, X.; Hou, G. A video coverless information hiding algorithm based on semantic segmentation. *EURASIP J. Image Video Process.* **2020**, *2020*, 23. [CrossRef]
9. Li, Q.; Wang, X.; Wang, X.; Ma, B.; Wang, C.; Shi, Y. An encrypted coverless information hiding method based on generative models. *Inf. Sci.* **2021**, *553*, 19–30. [CrossRef]
10. Muhammad, K.; Sajjad, M.; Mehmood, I.; Rho, S.; Baik, S.W. Image steganography using uncorrelated color space and its application for security of visual contents in online social networks. *Future Gener. Comput. Syst.* **2018**, *86*, 951–960. [CrossRef]
11. Zhang, Y.; Luo, X.; Wang, J.; Guo, Y.; Liu, F. Image robust adaptive steganography adapted to lossy channels in open social networks. *Inf. Sci.* **2021**, *564*, 306–326. [CrossRef]
12. Peng, W.; Zhang, J.; Xue, Y.; Yang, Z. Real-time text steganalysis based on multi-stage transfer learning. *IEEE Signal Process. Lett.* **2021**, *28*, 1510–1514. [CrossRef]
13. Han, C.; Xue, R.; Zhang, R.; Wang, X. A new audio steganalysis method basedon linear prediction. *Multimed. Tools Appl.* **2018**, *77*, 15431–15455. [CrossRef]
14. Abd EL-Latif, A.A.; Abd-El-Atty, B.; Venegas-Andraca, S.E. A novel image steganography technique based on quantum substitution boxes. *Opt. Laser Technol.* **2019**, *116*, 92–102. [CrossRef]
15. Zhu, Z.; Li, S.; Qian, Z.; Zhang, X. Destroying robust steganography in online social networks. *Inf. Sci.* **2021**, *581*, 605–619. [CrossRef]
16. Wang, Z.; Chen, M.; Yang, Y.; Lei, M.; Dong, Z. Joint multi-domain feature learning for image steganalysis based on CNN. *EURASIP J. Image Video Process.* **2020**, *2020*, 28. [CrossRef]
17. Lin, Y.; Wang, R.; Yan, D.; Dong, L.; Zhang, X. Audio steganalysis with improved convolutional neural network. In Proceedings of the 7th ACM Workshop on Information Hiding and Multimedia Security (IH&MMSec), Paris, France, 3–5 July 2019; pp. 210–215.
18. Niu, Y.; Wen, J.; Zhong, P.; Xue, Y. A hybrid r-bilstm-c neural network based text steganalysis. *IEEE Signal Process. Lett.* **2019**, *26*, 1907–1911. [CrossRef]
19. Yin, Y.; Wu, H.; Zhang, X. Neural visual social comment on image-text content. *IETE Tech. Rev.* **2021**, *38*, 100–111. [CrossRef]
20. Yang, Z.; Guo, X.; Chen, Z.; Huang, Y.; Zhang, Y. RNN-Stega: Linguistic steganography based on recurrent neural networks. *IEEE Trans. Inf. Forensic Secur.* **2018**, *14*, 1280–1295. [CrossRef]
21. Yang, Z.; Zhang, S.; Hu Y.; Hu, Z.; Huang, Y. VEA-Stega: linguistic steganography based on variational auto-encoder. *IEEE Trans. Inf. Forensics Secur.* **2020**, *16*, 880–895. [CrossRef]
22. Shiu, H.; Lin, B.; Cheng, C.; Huang, C.; Lei, C. High-capacity data-hiding scheme on synthesized pitches using amplitude enhancement—A new vision of non-blind audio steganography. *Symmetry* **2017**, *9*, 92. [CrossRef]
23. Xiang, L.; Yang, S.; Liu, Y.; Li, Q.; Zhu, C. Novel linguistic steganography based on character-level text generation. *Mathematics* **2020**, *8*, 1558. [CrossRef]
24. Zhang, X. Behavior steganography in social network. In Proceedings of the 18th International Workshop on Digital Forensics and Watermarking (IWDW), Magdeburg, Germany, 23–25 August 2017; pp. 21–23.
25. Hu, Y.; Wang, Z.; Zhang, X. Steganography in social networks based on behavioral correlation. *IETE Tech. Rev.* **2021**, *38*, 93–99. [CrossRef]
26. Li, S.; Ho, A. T.; Wang, Z.; Zhang, X. Lost in the digital wild: Hiding information in digital activities. In Proceedings of the 2nd International Workshop on Multimedia Privacy and Security (MPS), Toronto, ON, Canada, 15–19 October 2018; pp. 27–37.

27. Yang, Z.; Hu, Y.; Huang, Y.; Zhang, Y. Behavioral security in covert communication systems. In Proceedings of the 18th International Workshop on Digital Forensics and Watermarking (IWDW), Melbourne, Australia, 25–27 November 2020; pp. 377–392.
28. Nechta, I. Steganography in social networks. In Proceedings of the 2017 Siberian Symposium on Data Science and Engineering (SSDSE), Novosibirsk, Russia, 12–13 April 2017; pp. 33–35.
29. Wu, H.; Wang, W.; Dong, J.; Wang, H. New graph-theoretic approach to social steganography. In Proceedings of the 2019 IS&T International Symposium on Electronic Imaging: Media Watermarking, Security, and Forensics, Burlingame, CA, USA, 13–17 January 2019; pp. 539-1–539-6.
30. Wu, H.; Zhou, L.; Li, J.; Zhang, X. Securing graph steganography over social networks via interaction remapping. In Proceedings of the 6th International Conference on Artificial Intelligence and Security (ICAIS), Hohhot, China, 17–20 July 2020; pp. 303–312.
31. Giffin, J.; Greenstadt, R.; Litwack, P.; Tibbetts, R. Covert messaging through tcp timestamps. In Proceedings of the 2nd International Conference on Privacy Enhancing Technologies (PET), San Francisco, CA, USA, 14–15 April 2002; pp. 194–208.
32. Neuner, S.; Voyiatzis, A.G.; Schmiedecker, M.; Brunthaler, S.; Katzenbeisser, S.; Weippl, E.R. Time is on my side: Steganography in filesystem metadata. *Digit. Investig.* **2016**, *18*, 76–86. [CrossRef]
33. Bedi, P.; Dua, A. Network steganography using the overflow field of timestamp option in an IPv4 packet. In Proceedings of the 3rd International Conference on Computing and Network Communications (CoCoNet), Trivandrum, India, 18–21 December 2019; pp. 1810–1818.
34. Zhuang, K.; Shen, H.; Zhang, H. User spread influence measurement in microblog. *Multimed. Tools Appl.* **2017**, *76*, 3169–3185. [CrossRef]
35. Speck, R.; Moussallem, D.; Ngomo, A.C.N. Twitter network mimicking for data storage benchmarking. In Proceedings of the 15th IEEE International Conference on Semantic Computing (ICSC), Laguna Hills, CA, USA, 27–29 January 2021; pp. 298–305.
36. Lombardo, G.; Fornacciari, P.; Mordonini, M.; Sani, L.; Tomaiuolo, M. A combined approach for the analysis of support groups on Facebook—The case of patients of hidradenitis suppurativa. *Multimed. Tools Appl.* **2019**, *78*, 3321–3339. [CrossRef]
37. Selenium. Available online: https://www.selenium.dev/ (accessed on 6 August 2021).
38. WebDriver | Selenium. Available online: https://www.selenium.dev/documentation/webdriver/ (accessed on 6 August 2021).
39. English Word Frequency | Kaggle. Available online: https://www.kaggle.com/rtatman/english-word-frequency/version/1 (accessed on 8 August 2021).
40. Twintproject/Twint: An Advanced Twitter Scraping & OSINT Tool Written in Python That Doesn't Use Twitter's API, Allowing You to Scrape a User's Followers, Following, Tweets and More While Evading Most API Limitations. Available online: https://github.com/twintproject/twint (accessed on 6 August 2021).