Journal of Universal Computer Science, vol. 25, no. 2 (2019), 74-97
submitted: 14/8/18, accepted: 27/2/19, appeared: 28/2/19 © J.UCS

Methods for Information Hiding in Open Social Networks

Jędrzej Bieniasz, Krzysztof Szczypiorski
(Warsaw University of Technology
Warsaw, Poland
J.Bieniasz@tele.pw.edu.pl, ksz@tele.pw.edu.pl)

Abstract: This paper summarizes research on methods for information hiding in Open Social Networks. The first contribution is the idea of StegHash, which is based on the use of hashtags in various open social networks to connect multimedia files (such as images, movies, songs) with embedded hidden data. The proof of concept was implemented and tested using a few social media services. The experiments confirmed the initial idea. Next, SocialStegDisc was designed as an application of the StegHash method by combining it with the theory of filesystems. SocialStegDisc provides the basic set of operations for files, such as creation, reading or deletion, by implementing the mechanism of a linked list. It establishes a new kind of mass-storage characterized by unlimited data space, but limited address space where the limitation is the number of the hashtags' unique permutations. The operations of the original StegHash method were optimized by trade-offs between the memory requirements and computation time. Features and limitations were identified and discussed. The proposed system broadens research on a completely new area of threats in social networks.

Keywords: information hiding, open social networks, hashtag, StegHash, SocialStegDisc, filesystem
Categories: C.2.0, C.2.4, D.4.3, D.4.6, K.6.5

1 Introduction

Identifying new multimedia [Szczypiorski, 2016] [Fridrich, 2009] and network [Mazurczyk, 2016] steganography methods and their countermeasures are the main research contributions to steganography in the last few years. Less attention has been paid to text steganography [Chapman, 2001], and we have revisited this attractive subject in combination with social networks to discover probably a new area of threats in internet services. We utilized the popularity of using specific labels called hashtags across social networks and other internet services. With almost no limits for the construction of hashtags, due to the thousands of languages worldwide with dozens (or even hundreds) of alphabets, the infinite set of indexes could be explored. In our work we abstract from the linguistic level and forget the exact meaning of the hashtags as understood by humans. The method of StegHash is the main idea proposed by Szczypiorski [Szczypiorski, 2016a]. It is based on the use of hashtags on various social networks to create an invisible chain of multimedia objects, like images, movies or songs, with embedded hidden messages. These objects are indexed by permutations of preliminarily chosen hashtags. For every set of hashtags containing *n* elements there is the factorial of *n* permutations, which could be used as individual indexes of each message.

One of the original ideas of applying the StegHash technique was to establish an index system like in existing classic filesystems, such as FAT (File Allocation Table) or NTFS (New Technology File System). It would result in the creation of a new type of steganographic filesystem beyond previous efforts in this area, where the main ideas were to deny the filesystem operations or to deny the existence of the stored data. A new type of technique called SocialStegDisc [Bieniasz, 2017] is the proof of concept of the application of the StegHash [Szczypiorski, 2016a] method for new steganographic filesystem. The original environment of StegHash was modified to introduce the basic concepts of classic filesystems, such as Create-Read-Update-Delete operations or a defragmentation process. Furthermore, time-memory trade-offs were proposed in the design. The concept was tested to obtain operational results and proof of correctness. The results and the design were analyzed to discover as many features of the method as possible.

As it was mentioned, this paper summarizes results from [Szczypiorski, 2016a] and [Bieniasz, 2017]. The contributions beyond them included in this paper are:

- Section 2 – considerations about state-of-the-art techniques of preserving hidden data in multimedia files after processing by OSNs;
- Section 6 – full section with implementation of SocialStegDisc *proof-of-concept* system with testing;
- Section 7 – full section with discussion about the method;
- Section 8 – full section with malware and digital media foresincs perspective on detection and analysis of such techniques like StegHash or SocialStegDisc;
- Section 9 – consideration of StegHash and SocialStegDisc application for cyberfog security approach;

This paper is structured as follows: Section 2 briefly presents the state of the art in social network steganography, including a background to text steganography. Section 3 contains a presentation of the idea of the StegHash method and a typical scenario for the preparation of the steganograms. Section 4 introduces the idea of SocialStegDisc as an application of StegHash for the steganographic filesystem. Section 5 presents the implementation of SocialStegDisc *proof-of-concept*. Section 6 reports experiments on SocialStegDisc. In Section 7 the results and evaluation of SocialStegDisc's operation are presented. Section 8 includes a discussion on the possibility of the detection of the proposed system. Section 9 finally concludes our efforts and suggests future work.

2 Related work

In this section, we present a review of the literature on the three main aspects that are relevant for a full comprehension of our work. The first aspect is applying OSNs to the operations of steganographic techniques. We investigated how OSNs could be used in hidden communication scenarios. Secondly, we focused on how OSNs could impact the preservation of hidden data embedded in multimedia files during their upload and storage. This knowledge is crucial to consider a particular OSN for use in the operation of StegHash. Finally, we look at steganography techniques used beyond filesystems. This establishes the context to determine how the design of SocialStegDisc broadens the research in this domain.

2.1 Use of OSNs for steganographic techniques

In [Beato, 2014], Beato et al. presented two models of communication: high-entropy and low-entropy. The high-entropy model utilizes multimedia objects, such as images, video and music, etc., to embed hidden messages. This is recognized as classic multimedia steganography. This type of communication is characterized by high steganographic throughput, but the channel is easily detectable. The second model uses a null cipher approach, where the text data (e.g. status updates) carry secret information. The mechanism to decode the steganogram location and the hidden message need a pre-shared code. This communication features lower steganographic throughput, so it is proposed that this method is applied to covert signaling channels, while the main steganogram can be uploaded to another online service.
The efforts of Castiglione et al. [Castiglione, 2011] could be identified as expanding low-entropy steganographic methods. The first method utilizes filenames to carry hidden messages, so it could be used in OSNs that preserve the original filenames. The authors utilized the default naming schemes of popular digital camera producers, where a photo sequence number was chosen as the carrier of the hidden data. This method has a low steganographic throughput but detection is hard, so the scheme is generally safe. The second method takes advantage of inserting tags in images. The proposed covert communication channel requires the uploading of multiple images with the tagging of multiple users. Based on a predefined image and user sequence, a binary matrix can be determined and used to decode hidden messages. This method also has a relatively low steganographic throughput, but it is hard to detect.

Chapman et al. [Chapman, 2001] and Wilson et al. [Wilson, 2014] presented linguistic approaches to hide information in Twitter. Steganograms are carried by a bitmap determined by a permutation of language. The channel is considered to be very secure, although it requires a human processing of the tweets. It has a very low steganographic throughput.

All of the state-of-the-art methods operate with a single OSN, except the signaling channels designed in [Beato, 2014]. They utilize either a classic image steganography approach, which can be detected easily, or more sophisticated methods, for which the steganographic throughput is relatively low, but higher undetectability is introduced. It could be summarized as the classical trade-off aspect of steganography, where the ease of the method is traded off for a higher undetectability rate. The other disadvantage in the proposed methods is the fact that a steganogram sender is linked to the various user accounts that are required for the method. Such behavior could be recognized as suspicions by OSN providers.

2.2 Preserving hidden data in OSNs

To investigate how to preserve hidden data in multimedia files during uploading to OSNs we analyzed the results on image watermarking techniques [Potdar, 2005]. Image watermarking differs from steganography in terms of hiding the author's mark instead of the secret message. Furthermore, watermarks can be explicitibly visible. The aim of this is to protect copyrights and to authenticate an originator [Naskar, 2014]. In [Chomphoosang, 2011], the researchers investigated watermarking techniques to be used in OSNs, resulting in attacks and possible solutions. In [Hiney, 2015], the authors verified various approaches to eliminate the compression effect and algorithms sanitizing steganography on the side of the Facebook servers. Research shows that Facebook algorithms are efficient for detecting steganography, in particular for hidden information of greater size. Applying wavelet decomposition for watermarking is presented in [Banos, 2015]. The authors showed the effectiveness of the method by simulating attacks on OSNs. The most far-reaching proposals included redesigning the uploading services of OSNs, like in [Zigomitros, 2012] where a dual watermarking scheme was constructed.

Image watermarking techniques are classified into two types [Potdar, 2005]:

1. Spatial domain watermarking,
2. Transform domain watermarking.

Spatial domain watermarking is about modifying particular pixels of images to embed data. The simplest method of this type is the Least Significant Bit [Bamatraf, 2010], where the bit with no impact on the image's look is modified. The simplest and most popular spatial domain methods are generally vulnerable for image processing and transformations [Mishra, 2014], so more complicated schemes are under consideration. One of the improved LSB methods is Intermediate Significant Bit (ISB) [Zeki, 2009], where the watermarked section is found by the range of each bit-plane. Another advanced method is Patchwork [Bender, 1996], where the watermark is applied by changing the brightness of the pairs of randomly selected pixels.

Transform domain watermarking is about exploiting the coefficients of the transformed image to embed a mark. To transform the domain, four main techniques are considered:

1. Discrete Cosine Transform (DCT),
2. Discrete Wavelet Transform (DWT),
3. Discrete Fourier Transform (DFT),
4. Singular Value Decomposition (SWD).

In DCT algorithms, the image is divided into blocks and the selected set of coefficients are modified to embedding a watermark. DCT is a reversible [Cox, 1997]. DWT decomposes the image into three spatial directions and the watermark is embedded in the wavelet coefficients [Kundur, 1998]. DFT decomposes the image for phase and magnitude, and a watermark is embedded into the magnitude part. Fourier Transformation is robust against geometric attacks [O'Ruanaidh, 1997]. In SVD, the singular matrices of the frequency domain and spatial domain offer space for embedding a watermark. The loss of information in such a technique is very low. SVD could be combined with other techniques to exploit the widest range of properties, for example: SVD-DFT [Zhang, 2006], SVD-DWT [Lai, 2010] or DFT-DWT-SVD [Ansari, 2012]. In general, transform domain watermarking provides high robustness [Olanrewaju, 2011].

2.3 Steganographic filesystems

A breakthrough in the development of steganographic filesystems was the design by Anderson, Needham and Shamir [Anderson, 1998]. The authors proposed a method similar to that of invisible ink. A user with a password is able to find and decode hidden information. The StegFS system presented in [McDonald, 2000] is a practical

implementation of the concept presented in [Anderson, 1998] as an extension of a standard filesystem, Linux Ext2fs.
In recent years, distributed filesystems used in clusters of various kinds have been gaining popularity. For such systems, proposals for applying steganography to hide information in them also appear, for instance in [Venckauskas, 2013]. The main idea behind the method presented in [McDonald, 2000] consists of using multiple files to encode hidden information by means of their relative positions in clusters.

Among recent proposals, the work presented in [Neuner, 2016] is worth highlighting, in which Neuner et al. presented a way of hiding information in the timestamps of files saved in classic filesystems. They discovered that information may be effectively hidden in the 24 bits (3B) coding the nanosecond part of the timestamp in the NTFS filesystem. For each file, it is possible to use two types of timestamps: of creation and of last access. In this way, one file creates 6B of space for hidden data.

3 Idea of StegHash

The method of StegHash [Szczypiorski, 2016a] is based on the use of hashtags in the OSNs to connect multimedia files (like images, movies, songs) with embedded hidden messages. The initial set of hashtags is the base for constructing the indexes, which are unique pointers to these files. For a set of hashtags containing n elements there is the factorial of n permutations, and every single instance produces an individual and unique index for a given part of the data. The system consists of a secret initial set of hashtags (a password) and a secret transition generator. Using the transition generator starting from the initial set of hashtags, the link between indexes could be established and then hidden data could be explored as a chain from one to another (Fig. 1). The set of hashtags is logically independent from the OSN technology and could also be used on other Internet web services. The key issue is how to determine the placement of the next multimedia object with hidden data having generated a new index. A search engine dedicated to OSNs should be utilized, due to its capacity to search the hashtags as a primary way of marking messages in social media. The built in search option of the given OSN could also be used.

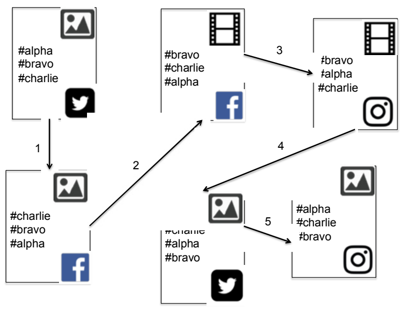


Figure 1: Example of StegHash method.

Let *l* be the length of an address in bits for creating the index for the group containing *n* hashtags:

$$l = \lceil log_2\, n!\, \rceil, \qquad n > 1\#(1)$$

The length of the address and percent of wasted addresses as a function of *n* is shown in Figure 2. For $n \in \{5, 10, 12, 22, 28, 29\}$ the number of wasted addresses is below 20%.

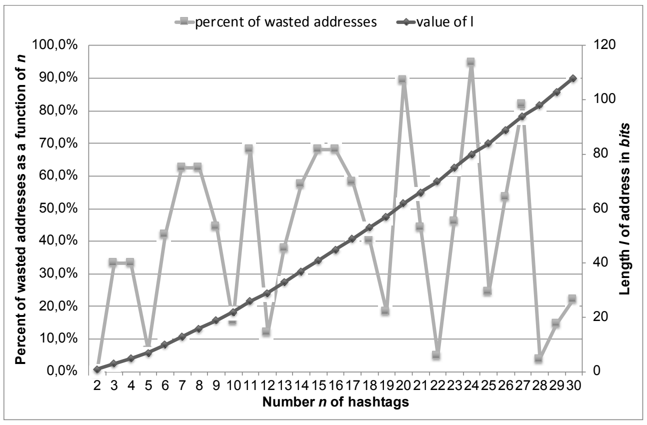


Figure 2: Length of address and percent of wasted addresses as a function of n.

0	1	2	#alpha	#bravo
1	2	1	#bravo	#alpha

0	**0**	**0**	1	2	3	#alpha	#bravo	#charlie
0	**0**	**1**	2	1	3	#bravo	#alpha	#charlie
0	**1**	**0**	1	3	2	#alpha	#charlie	#bravo
0	**1**	**1**	2	3	1	#bravo	#charlie	#alpha
1	**0**	**0**	3	1	2	#charlie	#alpha	#bravo
1	**0**	**1**	3	2	1	#charlie	#bravo	#alpha
1	**1**	**0**	x	x	x			
1	**1**	**1**	x	x	x			

Figure 3: Examples for n $\in$ {2,3}.

For example, for two hashtags there are two bits for addressing with no wasted space and two permutations (Figure 3); for three hashtags there are three bits (two addresses wasted) and six permutations (Figure 3). To start to use Steghash, there are four main aspects to consider:

1. An algorithm to create a dictionary – dependent only on *n*.
2. A set of hashtags to create a dictionary – secret initial set.
3. The mapping of the indexes into a dictionary.
4. A secret transition generator to create the link between the addresses (a chain).

The choice of algorithm has no impact on the security if the secret transition generator (point 4) is pseudorandom. For point 2, the balance between the popularity of some hashtags and the number of possible search results should be found. Typically one or two unpopular hashtags are enough to have a unique index and to have reasonable search outputs. Choosing only popular hashtags means that each multimedia object needs reviewing to find the hidden content. A secret transition generator initiated with a secret password, as used in StegHash, produces indexes in a chain to go step by step. The first address is the start, and if we used all the space it would be similar to a circular linked list for the data structure. A secret transition generator needs to be a function based on a pseudorandom code generator or a hash function.

18	#delta	#alpha	#bravo	**#charlie**
20	#delta	#alpha	#charlie	**#bravo**
19	#delta	#bravo	#alpha	**#charlie**
21	#delta	#bravo	#charlie	**#alpha**
22	#delta	#charlie	#alpha	**#bravo**
23	#delta	#charlie	#bravo	**#alpha**
6	#alpha	#delta	#bravo	**#charlie**
10	#alpha	#delta	#charlie	**#bravo**
7	#alpha	#bravo	#delta	**#charlie**
0	#alpha	#bravo	#charlie	**#delta**
11	#alpha	#charlie	#delta	**#bravo**
2	#alpha	#charlie	#bravo	**#delta**
8	#bravo	#delta	#alpha	**#charlie**
12	#bravo	#delta	#charlie	**#alpha**
9	#bravo	#alpha	#delta	**#charlie**
1	#bravo	#alpha	#charlie	**#delta**
13	#bravo	#charlie	#delta	**#alpha**
3	#bravo	#charlie	#alpha	**#delta**
14	#charlie	#delta	#alpha	**#bravo**
16	#charlie	#delta	#bravo	**#alpha**
15	#charlie	#alpha	#delta	**#bravo**
4	#charlie	#alpha	#bravo	**#delta**
17	#charlie	#bravo	#delta	**#alpha**
5	#charlie	#bravo	#alpha	**#delta**

for	**#delta**	go to	Facebook
	#alpha		Google Plus
	#bravo		Twitter
	#charlie		Instagram

Step	**From**		**To**	**Addr**
1	Facebook	➔	Instagram	**18**
2	Instagram	➔	Twitter	**20**
3	Twitter	➔	Instagram	**19**
4	Instagram	➔	Google Plus	**21**
5	Google Plus	➔	Twitter	**22**
6	Twitter	➔	Google Plus	**23**
7	Google Plus	➔	Instagram	**6**
8	Instagram	➔	Twitter	**10**
9	Twitter	➔	Instagram	**7**
10	Instagram	➔	Facebook	**0**
11	Facebook	➔	Twitter	**11**
12	Twitter	➔	Facebook	**2**
13	Facebook	➔	Instagram	**8**
14	Instagram	➔	Google Plus	**12**
15	Google Plus	➔	Instagram	**9**
16	Instagram	➔	Facebook	**1**
17	Facebook	➔	Google Plus	**13**
18	Google Plus	➔	Facebook	**3**
19	Facebook	➔	Twitter	**14**
20	Twitter	➔	Google Plus	**16**
21	Google Plus	➔	Twitter	**15**
22	Twitter	➔	Facebook	**4**
23	Facebook	➔	Google Plus	**17**
24	Google Plus	➔	Facebook	**5**

Figure 4: Example for n = 4 with addressing, pointers to social media networks and transition graph.

As stated previously, a search engine designed for the OSNs or the interior search mechanism of the given OSN should be used to find the next element in the chain. There are no effective search engines for some OSNs. To increase the performance of the system, one hashtag or more are utilized as the pointer to the next OSN. This may introduce a vulnerability into the StegHash method, because the prediction of this type of subaddressing could be linked with a given OSN. Figure 4 contains an example with four hashtags. The addressing scheme was generated and then a SHA-512 [NIST, 2015] based function was used to produce a chain. The last hashtag in the index represents the placement of the next message (for X go to Y). Figure 4 also shows a transition graph, which explains how a chain among the messages is built.

4 Idea of SocialStegDisc

There is a basic limitation to applying StegHash [Szczypiorski, 2016a] in SocialStegDisc [Bieniasz, 2017]: memory occupancy. Along with the increase in the number of *n* hashtags, the volume of the dictionary increases proportionately to *n*!. Therefore, keeping the entire dictionary in the memory places a heavy burden on the system. Generating the dictionary and restoring the set of used addresses will consume a large amount of time. The main goal behind keeping these dictionaries is ensuring uniqueness and reliability of addressing. Therefore, for a higher *n* it is proposed to introduce a dynamic verification of uniqueness of the generated addresses. Two space-time tradeoff-type modifications will be formulated on its basis along with an application of the linked list mechanism.

4.1 Dynamic verification of uniqueness of addresses

The first attempt to increase the efficiency of the StegHash method is to eliminate the storing of the results from the generation of the utilized address space when creating a chain and later restoring it. Such a set was stored because it was necessary to verify the uniqueness of the generated addresses. In this proposal, the uniqueness is verified by determining whether in the OSNs there is a file with a given address.

4.2 Space-time tradeoff modifications and linked list mechanism

From the perspective of a correct system operation, the system of addressing, which is indexing SocialStegDisc segments with hashtags, is of key importance. A characteristic feature of such addressing is non-linearity. Subsequent addresses are generated by means of a specific function on the basis of the initial address. The issue of addressing in SocialStegDisc is an analogy to the problem of memory space fragmentation in classic filesystems. In StegHash it was assumed that deleting files from such system is not a case, as the address space can be enormous with almost unlimited storage around the Internet. In case of SocialStegDisc the main consideration of design was to offer a full-fledged filesystem. *Delete* procedure is a required basic filesystem operation among *Read*, *Write* and *Update.*

Until now, a strong assumption was a prohibition of hiding the addresses for the system due to security reasons. This implies that it is not possible to apply mechanisms servicing memory fragmentation, for instance allocation tables or storing the index to another memory block together with the data. As a consequence, it is not possible to make available a function for deleting single files due to the resulting gap in the addressing, which if it is not known that the gap exists, would mean that the files located after the gap would be lost. Under this approach, once a file is added to SocialStegDisc, it will remain there. In this article, a solution is proposed that would allow the introduction of structures for the efficient management of resources (addresses), known from classic filesystems, to SocialStegDisc. To this end, the assumption of non-storing the addresses will be weakened by introducing such a mechanism that will simultaneously not decrease the security level.

Direct addresses to objects may be presented by means of an address to the object of reference (beginning of the SocialStegDisc instance) and a counter, whose value will serve to control the obtaining of the address of the subsequent object. In every multimedia object, a space for this counter is added to a hidden space. Due to this, the SocialStegDisc system assumes the structure of a linked list, which allows to supplement the system with file deletion. An example of how this operates is presented in Figure 5.

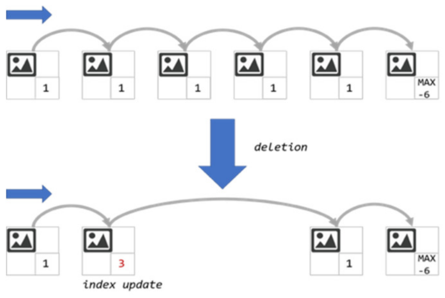


Figure 5: Mechanism of a linked list in SocialStegDisc. It allows to realize `delete` *procedure.*

Apart from hiding m bytes of proper data, it is necessary to place a minimum of p bits coding the numerical index in every multimedia object. The selection of the volume *p* determines the maximum length of the chain, for which the problem of information loss does not occur when the mechanism of file deletion is applied, and amounts to:

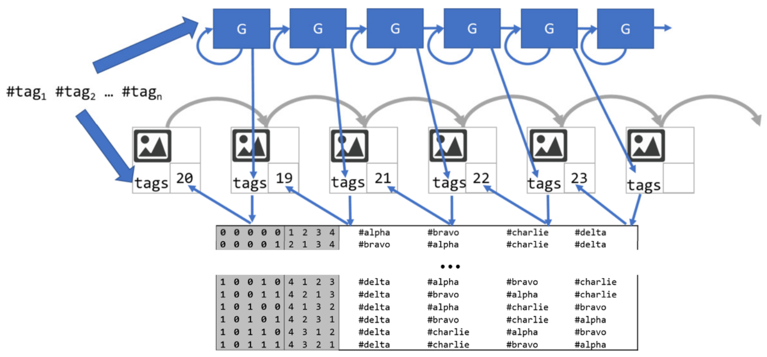
$$2p - 1\#(2)$$

The hidden counter will be used in the proposed modifications of the space-time tradeoff type, by using it to control the functions establishing the next address. This counter may store:

- The address code for the application of the StegHash method without the dictionary of the used addresses.
- The counter value that specifies how many times the procedure of generation of a subsequent address should be performed to actually obtain it – in the case of the elimination of both original dictionaries of the StegHash method.

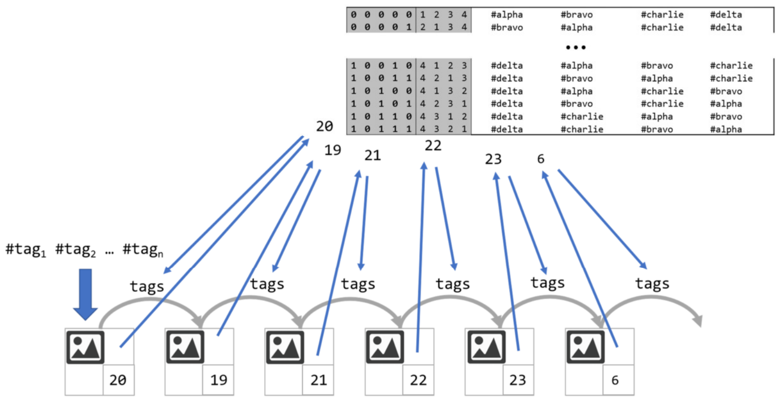
When StegHash with the modification eliminating the dictionary of used addresses is applied, and when a chain of objects is created, a code for the address to the next object is placed in the additional space, in line with the coding from the dictionary of permutations. The scheme of this procedure is presented in Figure 6.

During reading, the address code is retrieved from the hidden space and then converted into an address in the form of a set of hashtags. Thus, access to the next object is possible. To maintain the continuity of the address space, when deleting a file in the multimedia object preceding the beginning of the given file, the counter should be updated to the value retrieved from the last object storing the file to be deleted. The set of the used addresses should also be updated. The released addresses are not lost but they go back to the pool and may be generated in subsequent iterations of the pseudo-random generation function. The schemes for the read method is presented in Figure 7.



0 0 0 0 0	1 2 3 4	#alpha	#bravo	#charlie	#delta
0 0 0 0 1	2 1 3 4	#bravo	#alpha	#charlie	#delta
			•••		
1 0 0 1 0	4 1 2 3	#delta	#alpha	#bravo	#charlie
1 0 0 1 1	4 2 1 3	#delta	#bravo	#alpha	#charlie
1 0 1 0 0	4 1 3 2	#delta	#alpha	#charlie	#bravo
1 0 1 0 1	4 2 3 1	#delta	#bravo	#charlie	#alpha
1 0 1 1 0	4 3 1 2	#delta	#charlie	#alpha	#bravo
1 0 1 1 1	4 3 2 1	#delta	#charlie	#bravo	#alpha

Figure 6: Creation of a chain by means of combined StegHash method and linked list mechanism.



0 0 0 0 0	1 2 3 4	#alpha	#bravo	#charlie	#delta
0 0 0 0 1	2 1 3 4	#bravo	#alpha	#charlie	#delta
			•••		
1 0 0 1 0	4 1 2 3	#delta	#alpha	#bravo	#charlie
1 0 0 1 1	4 2 1 3	#delta	#bravo	#alpha	#charlie
1 0 1 0 0	4 1 3 2	#delta	#alpha	#charlie	#bravo
1 0 1 0 1	4 2 3 1	#delta	#bravo	#charlie	#alpha
1 0 1 1 0	4 3 1 2	#delta	#charlie	#alpha	#bravo
1 0 1 1 1	4 3 2 1	#delta	#charlie	#bravo	#alpha

Figure 7: Read of a chain by means of combined StegHash method and linked list mechanism.

Another proposed modification of the space-time tradeoff type is a total elimination of the original StegHash dictionaries and the use of the linked list mechanism to store relevant counters allowing the restoration of the chain. To this end, first of all, the algorithm of dynamic generation of addresses should be modified so that the function, apart from the new address, returns the current status of the sampling counter in the course of the generation of the subsequent addresses. In Figure 8, a scheme of a chain creation is presented. With the starting point known, the obtained counter allows the retrieval of the address returned in the course of the generation at the counter status. As these states of the counter are recorded in the hidden space defined for the linked list mechanism, they allow the chain objects to be traverse from a known initial address of the SocialStegDisc instance. In Figure 9, a scheme of a chain retrieval is shown.

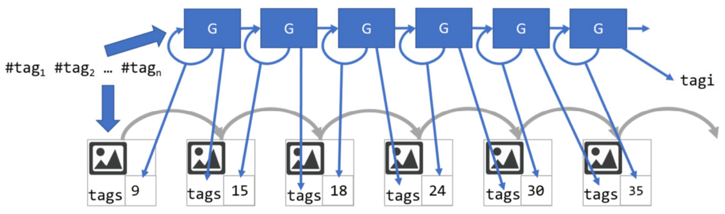


Figure 8: Chain creation with no dictionaries and a linked list mechanism.

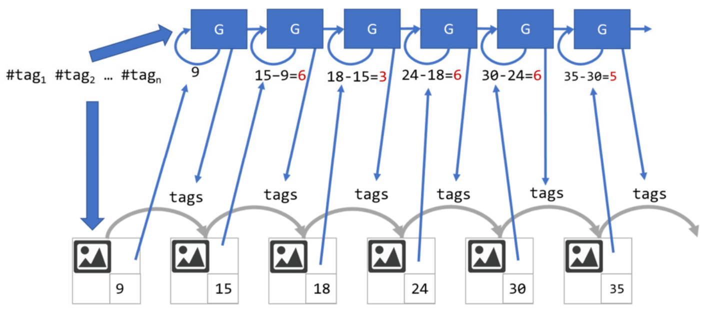


Figure 9: Chain retrieval with no dictionaries and a linked list mechanism.

In both cases, a file deletion that entails the creation of free address space, in this case updates the index in the object preceding the space to the value of the index from the last deleted file.

5 Implementation of SocialStegDisc

5.1 Environment

The system implementation proposal presented in this article is a proof-of-concept of the SocialStegDisc system. The main client application is the SocialStegDisc Shell terminal, which translates the client queries to appropriate operations of the file directory and operations on files, using the Controller module. In addition, two modules from StegHash were used:

- StegHash Engine – a module responsible for the creation of appropriate addressing of multimedia files with the use of hashtags.
- Database – contains a set of multimedia files used as a carrier of hidden data.

5.2 Generation of addresses

The basic operation of the system is the generation of addresses. This index plays two roles:

1. Unambiguously identifies the object.
2. Identifies the social network in which it is embedded.

In StegHash [Szczypiorski, 2016a], addresses were generated by means of any type of permutation generation algorithm, and the uniqueness was ensured by rejecting the already drawn addresses. For SocialStegDisc, the generation of addresses in a chain must be characterized by restorability with the use of appropriate initial conditions. To this end, the approach of Rivest [Neuner, 2016] was used, who presented a reference code for a program generating a random sample of s size from set S. Assuming that s = S, the sample is a searched permutation of set S. In Figure 10, an exemplary code in Python of a chain generation method with no dictionaries stored, is presented.

```
def go_to_perm(seed, counter):
    input = ""
    for num in seed:
       input += str(num)
    i = 0
    output_list = []
    while(i < counter):
       if(len(output_list) == len(seed)):
             for num in output_list:
                    input = ""
                    input = str(num)
                    output_list = []
       i = i + 1
       hash_input = input + str(i)
       hash = hashlib.sha256(hash_input.encode('utf-8'))
       output = int(hash.hexdigest(),16)
       pick = int(output % len(seed))
       if not pick in output_list:
             output_list.append(pick)
    return output_list
```

Figure 10: Method of address generation based on reference address.

The main part of the method is used for the hash function for the pseudorandom generation of the next value that is a candidate to become an element of an output permutation. In the example of the code, the function SHA-256 was used [NIST, 2015], but it is possible to apply other hash functions, for instance SHA-512 [NIST, 2015]. Using a hash function with a greater output size reduces the number of sampling iterations when completing the output table. The output value from the hash function is converted to a hexadecimal number, based on which a decimal value is generated, which is a pseudorandom number generated in a given loop iteration. This number undergoes the operation of obtaining the remainder from division by the set size. Finally, it is verified whether the sampled value already exists in the output set from a given iteration of the permutation generation. If not, it should be added to the list. Generation ends when a table of numbers has been obtained for the size of the set undergoing permutation. The returned table of numbers is mapped to relevant hashtags.

5.3 Hosting in social media

When selecting a way of multimedia hosting, attention should be paid to multimedia files processing on the side of the social network services and the impact this processing has on non-breaching of the hidden data. This problem was investigated in Section 2.2. Therefore, it should be verified first whether the social networks selected for the needs of practical implementation of SocialStegDisc to ensure non-interference with the byte content of the files. Problems occurring with the sanitization of steganography on the side of social networks or with its deformation may be circumvented using an intermediary layer in the form of the file upload services. An advantage of such an approach is that such a service has no impact on the file content, and thus – there is a certainty of non-breach of the steganographic content. Then, in the social service using a hashtag mechanism only a link and a set of hashtags addressing this file are shared.

6 Practical evaluation of SocialStegDisc

6.1 Size of disk and files

The structure of the classical filesystem is an array of blocks, where a single block is the smallest unit of the disk operations. For SocialStegDisc, this unit is the amount of hidden data in a single multimedia file (images, movies, audiofiles). For example, StegHash uses 10 bytes per multimedia files.

In Table 1 the calculation of the required space for the index in bytes is presented as the function of the number of hashtags. This space is calculated by the formula:

$$p_B = \left\lceil \frac{log_2 n!}{8} \right\rceil, \qquad n > 1\#(2)$$

p [b]	p_B [B]	max. value	p [b]	p_B [B]	max. value	p [b]	p_B [B]	max. value
2	1	3	31	4	2147483647	60	8	1,15292E+18
3	1	7	32	4	4294967295	61	8	2,30584E+18
4	1	15	33	5	8589934591	62	8	4,61169E+18
5	1	31	34	5	17179869183	63	8	9,22337E+18
6	1	63	35	5	34359738367	64	8	1,84467E+19
7	1	127	36	5	68719476735	65	9	3,68935E+19
8	1	255	37	5	1,37439E+11	66	9	7,3787E+19
9	2	511	38	5	2,74878E+11	67	9	1,47574E+20
10	2	1023	39	5	5,49756E+11	68	9	2,95148E+20
11	2	2047	40	5	1,09951E+12	69	9	5,90296E+20
12	2	4095	41	6	2,19902E+12	70	9	1,18059E+21
13	2	8191	42	6	4,39805E+12	71	9	2,36118E+21
14	2	16383	43	6	8,79609E+12	72	9	4,72237E+21
15	2	32767	44	6	1,75922E+13	73	10	9,44473E+21
16	2	65535	45	6	3,51844E+13	74	10	1,88895E+22
17	3	131071	46	6	7,03687E+13	75	10	3,77789E+22
18	3	262143	47	6	1,40737E+14	76	10	7,55579E+22
19	3	524287	48	6	2,81475E+14	77	10	1,51116E+23
20	3	1048575	49	7	5,6295E+14	78	10	3,02231E+23
21	3	2097151	50	7	1,1259E+15	79	10	6,04463E+23
22	3	4194303	51	7	2,2518E+15	80	10	1,20893E+24
23	3	8388607	52	7	4,5036E+15	81	11	2,41785E+24
24	3	16777215	53	7	9,0072E+15	82	11	4,8357E+24
25	4	33554431	54	7	1,80144E+16	83	11	9,67141E+24
26	4	67108863	55	7	3,60288E+16	84	11	1,93428E+25
27	4	134217727	56	7	7,20576E+16	85	11	3,86856E+25
28	4	268435455	57	8	1,44115E+17	86	11	7,73713E+25
29	4	536870911	58	8	2,8823E+17	87	11	1,54743E+26
30	4	1073741823	59	8	5,76461E+17	88	11	3,09485E+26

Table 1: Maximal value of index taken as p bits and p_B bytes needed to encode them.

n	n!	p_B [B]
2	2	1
3	6	1
4	24	1
5	120	1
6	720	2
7	5040	2
8	40320	2
9	362880	3
10	3628800	3

Table 2: Number of possible addresses for n hashtags and needed bytes to encode their binary representation.

The maximum size of SocialStegDisc is calculated from:

$$S_D = n! \cdot (m + p_B)\#(3)$$

Table 3 summarizes how the size for files and total size (data and indexes) are related to the number of hashtags by assuming m = 5, 10, 15 as the number of bytes available for users' files per single multimedia file. These values are determined by formulas 4 and 5 analogically:

$$data\ [MB] = \frac{n! \cdot m}{1000}\#(4)$$

$$total\ disk\ size\ [MB] = \frac{n! \cdot (m + p_B)}{1000}\#(5)$$

n	*n!*	*p_B [B]*	*m = 5*		*m = 10*		*m = 15*	
			dane [MB]	*dane & Licznik [MB]*	*dane [MB]*	*dane & Licznik [MB]*	*dane [MB]*	*dane & Licznik [MB]*
2	2	1	0,01	0,012	0,02	0,022	0,03	0,032
3	6	1	0,03	0,036	0,06	0,066	0,09	0,096
4	24	1	0,12	0,144	0,24	0,264	0,36	0,384
5	120	1	0,6	0,72	1,2	1,32	1,8	1,92
6	720	2	3,6	5,04	7,2	8,64	10,8	12,24
7	5040	2	25,2	35,28	50,4	60,48	75,6	85,68
8	40320	2	201,6	282,24	403,2	483,84	604,8	685,44
9	362880	3	1814,4	2903,04	3628,8	4717,44	5443,2	6531,84
10	3628800	3	18144	29030,4	36288	47174,4	54432	65318,4

Table 3: Relation between size for files, total size (data and indexes) and number of hashtags.

6.2 Secrets

In StegHash, a user uses three secrets:

1. Initial set of hashtags for generation of dictionary.
2. Pointer for the first multimedia file with hidden data which is an initial set for the transition dictionary.
3. Generation function.

SocialStegDisc merges secrets 1. and 2. into one. The user generates pointers for the next files in the chain from the initial set and using the index value hidden in the file with the data together. Furthermore, as an empirical result, it is proposed to have knowledge of the last used address (set of hashtags) in the system as another secret. This does not violate the level of security of the system, but broadens the operations of SocialStegDisc with two functionalities:

1. Knowledge of iteration limit of the system.
2. From access to the last used object in the system, the size of the space left for the user is known.

6.3 Testing social networks' operations

For this work, two social networks with hashtag mechanisms were used: Twitter and Flickr. For every social network two tests were conducted:

1. With original images with well-established steganography methods.
2. With images uploaded to social networks, downloaded from them and next used with well-established steganography methods.

The second case was determined by the assumption that it could help avoid the distortion of steganography as the image already processed by the social network's algorithms could be classified as benign.

The tests for Twitter were negative for both cases, whereas for Flickr all tests passed. It is proposed to avoid such a distortion by treating social networks as a layer for sharing URLs to a multimedia file stored on the service that do not violate the original structure. For testing this alternative design, Twitter was used to share Dropbox URLs to files. Tests were conducted successfully.

This approach forms the basis for the consideration of creating a generic design of such systems:

1. Pointers to objects with a hidden space: a secret coding of keys. In a generic design it could be any set of alphanumeric strings.

2. Storage for files: any Internet service where stored files are searchable by their description consists of pointers introduced at Point 1.
3. Transition generation function: as introduced for StegHash [Szczypiorski, 2016a], a secret transition generator based on a pseudorandom code generator or a hash function.

6.4 Experiments

We tested the system with different sizes of initial data to send: 50B, 100B, 1000B, 5000B, 10000B. The size of the space for the user's data was 10B, so 5, 10, 100, 500 and 1000 images were needed accordingly. One image had size of 9kB. For this setting we used minimal number of hashtags for each case: 3, 4, 5, 6 and 7 accordingly. The system was tested for two novel configurations introduced in Section 4. Experiments confirmed the initial results. For a higher demand for the number of permutations of hashtags the reliability decreases from 100% to 70%, but the mechanism to verify the upload could be introduced to trigger the retransmission. What is more noticeable is the time of operation. The transition generation function based on a hash function with the design from Figure 10 takes much more time to generate the next unique permutation of the hashtags as the number of them available decreases. Future research could place effort on developing transition functions to appease this aspect by needing less tries to find a new unique permutation of the hashtags. Obviously, the lower number of hashtags for the same number of objects to upload, the more it is observed in this kind of hang of the system.

Number of images	**Size of hidden data**	**Minimum number of hashtags**	**Time of simple uploading pictures [s]**	**Time of generating all permutations [s]**	**Number of needed iterations to generate all permutations**	**Time of generating needed permutation [s]**	**Number of needed iterations to generate needed permutations**
5	50	3	17.5	0.0005	133	0.0002	53
10	100	4	35	0.001	459	0,0004	96
100	1000	5	350	0.04	7334	0.015	2512
500	5000	6	1750	0.53	65962	0.077	12453
1000	10000	7	3500	18.39	888693	0.22	19887

Table 4: Performance of SocialStegDisc

Table 4 presents results of performance metrics from executing SocialStegDisc as the whole system and their main modules. This strategy of testing resulted from main assumptions:

- Time of uploading or checking existence of one picture is mainly a constant value, at 3.5 s; Reading of a single picture lasts 2.5 s;

- Time of generating permutations is non-linear function of needed number them, with strong rise with larger arguments;
- Extracting hidden data from a single steganograhic carrier lasts 1s;

		Create [s]		**Read [s]**	
Number of images	**Minimum number of hashtags**	**SocialSteg Disc ver. 1**	**SocialSteg Disc ver 2.**	**SocialSteg Disc ver. 1**	**SocialSteg Disc ver 2.**
5	3	17.5	35	12,5	25
10	4	35	70	25	50
100	5	350	700	250	500
500	6	1750	3500	1250	2500
1000	7	3500	7000	2500	5000

Table 5: Performance of Create and Read operartion for both versions of SocialStegDisc

Table 5 includes performance results of *Create* and *Read* operation for both versions of SocialStegDisc. For presented test cases, the system works predictably and with the accepted time. It should considered that this time might be altered, for example if the method needs to be less detectable or to cover in the background of normal traffic by pretending that operations of SocialStegDisc are not satistically different from it.

7 Discussion

7.1 Addresing scheme and hidden space

The issue of addressing in SocialStegDisc is an analogy to the problem of memory space fragmentation in classic filesystems, which has been highlighted when developing the concept. The SocialStegDisc concept introduces a mechanism of storing indexes to the next object in the system, due to which the drive assumes the structure of a linked list. Thanks to the application of a linked list mechanism, it is possible to efficiently use the space and propose a file deletion mechanism. The greatest problem is the calculation overhead and algorithms to verify the correctness of the operation scheme.

The system is characterized by non-linearity in filling in the byte space. This effect occurs when the last used object hides a smaller number of bytes than possible. On the basis of the number of bytes, it is possible to determine the length of the chain of multimedia files created in line with the StegHash technique essence. This size amounts to:

$$L = \left\lceil \frac{M}{m} \right\rceil \#(6)$$

where M is the number of bytes on which the operation is the performer, and m is the size of a single logical block. When M mod m $\not\equiv 0$ in the last multimedia file in the chain, fewer bytes are placed than the size of space allocated in the multimedia file (m).

7.2 Undetectability

The SocialStegDisc undetectability level depends on:

1. The level of undetectability in the open social networks of the applied methods of the multimedia steganography, serving to hide data.
2. The level of undetectability of the StegHash steganographic method, which is a system of indexing multimedia objects and building the SocialStegDisc drive space from single blocks.
3. The level of security of social networks with respect to the detection of anomalies in the network users' behavior. The SocialStegDisc system may communicate with the same service multiple times at short intervals. From the perspective of such a service, frequent and automated sending of queries will be treated as a threat, for instance a Denial-of-Service type attack.
4. The level of detectability of multimedia steganography. This problem was under research for Facebook, among others. In [Ning, 2014], the authors tested algorithms sanitizing steganography on the side of the Facebook servers. The research shows that Facebook algorithms are efficient for the detection of steganography known so far, in particular for hidden information of greater size.

In [Szczypiorski, 2015], a way of evaluation of network steganography techniques was proposed in three categories of undetectability: good, bad and ugly. The presented classification manner may be applied to other steganographic methods. StegHash may be considered a good steganographic method, which transitively means that the proposed SocialStegDisc technique is also characterized by this feature. The observer is unable to detect the ongoing communication anywhere in the network, even at the receiver of the hidden data.

7.3 Reliability

The reliability of the SocialStegDisc system should be assessed by the reliability of individual system operations, these operations are divided into dependent on steganography and independent of it.

Reliability due to multimedia steganography includes:

1. Degree of avoiding steganography cleaning algorithms on social networking sites. This problem was addressed in the assessment of undetectability in the Section 7.2 and it is closely related to the level of system reliability.
2. The influence of image processing algorithms, e.g. compression, on the bit content of a multimedia object. This issue was addressed in the study [Hiney, 2015] where a method was found to overcome this problem. In practical implementation, SocialStegDisc must be included.

It may be necessary to introduce a mechanism to confirm the placement of data in all upload operations. It means download of the loaded object and bit comparison with object before uploading. If the objects are identical, uploading a segment of the hidden data carried by this object is considered successful. When loading multiple objects, this mechanism becomes a large overhead in the operation of the entire system.

All interaction with websites is done using the TCP / IP stack protocols – IP, TCP and HTTP. The use of TCP in transport layer means the use of mechanisms of windows, confirmations, sequential numbers and retransmissions. The transport speed is adjusted to the conditions in the network, data can be read in the order of transmission and after detection of data loss, data transmission is repeated. Thanks to this, the transport of SocialStegDisc requests has a high level of reliability. On the other hand, if the execution of the request fails, the attempts can be made repeatedly until the positive result.

As the second key element of the system's operation, independent of steganography, the operation of the social network service server should be indicated. The level of reliability, in particular of the largest websites, is very high. These services are scalable and well-protected, for example, by resource overload attacks. However, the user SocialStegDisc may encounter, when interaction with the server is blocked due to:

1. Classifying the interaction SocialStegDisc with the service as a kind of dangerous anomaly, e.g. symptom of the attack Denial-of-Service;
2. Applying the limitation mechanism of access tokens - when exhausted, requests are discarded until the pool is renewed.
3. Changing the interface, blocking searching methods and other modifications introduced by operators of OSNs;

These aspects could limit the use of the method, especially regarding the duration of such communication or completely stop it. Regarding to [Fridrich, 1999], every steganography method is designed basing on *the magic triangle* which means trade-off between steganographic bandwidth, undetectability level and robustness of the method. These features of SocialStegDisc can be flexibly adjusted to the goals to be achieved. Obviously, real cyber threat actors could tolerate long time of working (low steganographic bandwidth) wether it could imply reliable cover for their actions. Higher steganographic bandwidth would mean that ther network traffic could be easier distinguished as abnormal activity and thus easily blocked. In that case, we do not much care if OSN could alter their operations. We would like the OSN operators to pay attention to the way their systems operate and this paper may constitute a collection of knowledge for them regarding new vectors of abuse of their systems.

8 Impact on digital forensics methods

The StegHash [Szczypiorski, 2016a] and SocialStegDisc [Bieniasz, 2017] methods have been established as a proof of concept for use during digital crimes, such as information leaks or communication between bad actors mainly observed as benign activities of users of social networks and other Internet services. It can be broaden to research on misusing public Internet services by a combination of steganographic methods to hide data and operations in the logic of using such services. It goes far beyond classical steganalysis methods to find the existence of that activity, for which methods and tools of digital forensics are needed. Unfortunately, they are currently not sufficient and this generates a need for a new approach to deal with such activities. First of all, finding the way to collect pieces of evidence is a must. Initially, it could be like looking for a needle in a haystack, because there are three main questions to answer:

- What to observe?
- How to observe?
- How to establish verifiable and formal evidence of a digital crime from observations and methods for observing?

All of them are related together, so an approach to one aspect impacts another and vice versa. The natural answer for establishing a class of problem is the big data approach for behavioral analysis. Collecting users' activity, for example:

- Upload and download multimedia files from open social networks;
- Existence of correlation between using an open social network and the next upload or download of a multimedia file from an Internet service;
- Timing of activity;
- Distribution of activity;

appears to be promising as a source for big data analysis to find indicators of the use of StegHash and SocialStegDisc. The next aspect is to choose the observation methods and architectures. We see a need for a comprehensive survey on such methods, architectures and designs, which is what we plan to do in the future. The main categories of algorithms to investigate at first can be identified easily: community detection in graphs, distinguishing automatic from human activity and finding patterns of behavior in large datasets. These algorithms could be used for data collected by observing the central network with aggregated data or in more distributed designs where the particular actors could be detected more accurately. The interesting idea is to combine the proposed algorithms with the design of a moving observer [Szczypiorski, 2015]. The combination could show the strength in increasing confidence and efficiency of detection of the bad actor by analyzing datasets generated at the following points and getting closer to the source of malicious activity. Another consideration is to find methods and algorithms that can be introduced on the open social network side, where simple quotas or blocking anomalous activity are not enough nowadays. The contribution of the authors in this paper is a foundation for their further research on defining a set of indicators for using methods such as StegHash or SocialStegDisc, finding algorithms and the best architectures for the execution of them to establish verifiable and formal evidence of digital crime in an optimal way. Furthermore, the authors see an opportunity to invent some methods that can be implemented in the restricted view of an open social network operator.

9 Conclusions and future work

In the initial experiment we proved that the concept of the StegHash method, as a new approach for combining text steganography with other carriers, like pictures, movies and songs, was correct. Every multimedia object is indexed with a unique permutation of a set of n hashtags. Between the objects there is a logical structure created by a set hashing function. On the basis of an input index (permutation of a set of hashtags), this function generates an index of the next object (permutation of a set of hashtags).

Please note that for *n* hashtags, *m* byte messages and 100% accuracy, we have the receiving capacity of $n!\cdot m$ bytes for storage, i.e., for $n = 12$ and $m = 10$ bytes, this would be 4.46 TBytes. Furthermore, a new steganographic filesystem called SocialStegDisc was formulated. Thus, subsequent blocks of hidden space may be read, recorded or modified in sequence, which is an analogy to servicing classic filesystems. The authors proposed the idea of SocialStegDisc as an improvement on the initial scheme of StegHash via a trade-off between memory requirements and computation time. During tests it was validated that SocialStegDisc works analogically to StegHash, but the effort of operations is more widely distributed. Furthermore, SocialStegDisc introduces a classical set of CRUD (Create, Read, Update, Delete) operations on files, which was not possible in the original StegHash method.

We recognized that StegHash and SocialStegDisc could support the development of a “cyberfog” security approach, where solutions to eliminate centralizing crucial, sensitive and critical information are constantly sought. Following [Kott, 2016], a “cyberfog” security approach assumes splitting the data into numerous fragments and dispersing them across multiple end-user devices. Data dispersion presents adversaries with uncertainty as to where to find relevant information and how to reconstruct it from captured parts. We see the possibility for the application of the StegHash indexing scheme and SocialStegDisc filesystems operations directly into such a “cyberfog”. We addressed the construction of such a system in [Bieniasz, 2018]. The design seems to offer a real and functional application. Moreover, we are planning to use big data analytics to find the context in systems that are similar to StegHash and to introduce new methods for the detection of such systems. Finally, in the future we will also pay attention to the theoretical aspect of building relations among hashtags by analyzing other mathematical functions and algorithms that output permutations.

Research into such issues as the SocialStegDisc technique focuses on their usefulness in real scenarios, especially with respect to information leak. It seems that the development of such concepts may serve only terrorists or other criminals in their attempts to hide communication supporting their actions. However, the purpose of the research into this method was to demonstrate its application options and to determine on this basis the vulnerability of the systems used in it. Such vulnerability may be taken into account during the following iterations of the system development and be entirely eliminated.

Acknowledgement

This work has been supported by the National Centre for Research and Development (agreement No. CYBERSECIDENT/369234/I/NCBR/2017) under the CyberSecIdent Programme.

References

[Anderson, 1998] Anderson, R., Needham, R., Shamir, A. 1998. “The Steganographic File System”. Proceedings of the Second International Workshop on Information Hiding, 73–82.

[Ansari, 2012] Ansari, R., Devanalamath, M. M., Manikantan, K. and Ramachandran, S. “Robust digital image watermarking algorithm in DWT-DFT-SVD domain for color images,” in Communication, Information & Computing Technology (ICCICT), 2012 International Conference on. IEEE, 2012, pp. 1–6.

[Bamatraf, 2010] Bamatraf, A., Ibrahim, R., and Salleh, M. “Digital watermarking algo- rithm using lsb,” in Computer Applications and Industrial Electronics (ICCAIE), 2010 International Conference, 2010, pp. 155–159.

[Banos, 2015] Banos, O., Lee, S., Yoon, Y., Le-Tien, T. et al., “A novel watermarking scheme for image authentication in social networks,” in Proceedings of the 9th International Conference on Ubiquitous Information Manage- ment and Communication. ACM, 2015, p. 48.

[Beato, 2014] Beato, F., De Cristofaro, E., and Rasmussen, K. B. "Undetectable communication: The Online Social Networks case," Privacy, Security and Trust (PST), 2014 Twelfth Annual International Conference on, Toronto, ON, 2014, pp. 19-26.

[Bender, 1996] Bender, W., Gruhl, D., Morimoto, N., and Lu, A. “Techniques for data hiding,” IBM systems journal, vol. 35, no. 3.4, pp. 313–336, 1996.

[Bieniasz, 2017] Bieniasz, J., Szczypiorski, K.: “SocialStegDisc: Application of steganography in social networks to create a file system”, In Proc. of 3rd International Conference on Frontiers of Signal Processing (ICFSP 2017), Paris, France, 6-8 September 2017

[Bieniasz, 2018] Bieniasz, J., Szczypiorski, K.: „Towards Empowering Cyber Attack Resiliency Using Steganography”, In Proc. of 4th International Conference Frontiers of Signal Processing (ICFSP 2018), Poitiers, France, 24-26 September 2018

[Castiglione, 2011] Castiglione, A., D'Alessio, B., and De Santis, A. "Steganography and Secure Communication on Online Social Networks and Online Photo Sharing," Broadband and Wireless Computing, Communication and Applications (BWCCA), 2011 International Conference on, Barcelona, 2011, pp. 363-368.

[Chapman, 2001] Chapman, M., Davida, G., and Rennhard, M., “A Practical and Effective Approach to Large-Scale Automated Linguistic Steganography”, Proceedings of the Information Security Conference, October 2001, pp. 156-165.

[Chomphoosang, 2011] Chomphoosang, P., Zhang, P., Durresi, A., and Barolli, L., “Surveyoftrust based communications in social networks,” in 2011 14th International Conference on Network-Based Information Systems, 2011.

[Cox, 1997] Cox, I.J., Kilian, J., Leighton, F. T. and Shamoon, T., “Secure spread spectrum watermarking for multimedia,” IEEE Transactions on Image Processing, vol. 6, no. 12, pp. 1673–1687, 1997.

[Fridrich, 1999] Fridrich, J. Applications of data hiding in digital images. In ISSPA’99 Proceedings of the Fifth International Symposium on Signal Processing and its Applications (IEEE Cat. No. 99EX359), volume 1, page 9.

[Fridrich, 2009] Fridrich, J. “Steganography in Digital Media: Principles, Algorithms, and Applications”, Cambridge University Press; 1 edition, December 2009.

[Ning, 2014] Ning, J., Singh, I., Madhyastha, H., Krishnamurthy, S., Cao. G., Mohapatra, P. “Secret message sharing using online social media”, Proceedings of the Communications and Network Security (CNS), 2014 IEEE Conference on, 319-327.

[Hiney, 2015] Hiney, J., Dakve, T., Szczypiorski, K., and Gaj, K. "Using Facebook for Image Steganography", Proceedings of the Availability, Reliability and Security (ARES), 2015 10th International Conference on, Toulouse, 2015, pp. 442-447.

[Kott, 2016] Kott, A., Swami, A. and West B. J., "The Fog of War in Cyberspace," in Computer, vol. 49, no. 11, pp. 84-87, Nov. 2016.

[Kundur, 1998] Kundur, D. and Hatzinakos, D. “Digital watermarking using multires- olution wavelet decomposition,” in Acoustics, Speech and Signal Pro- cessing, 1998. Proceedings of the 1998 IEEE International Conference on, vol. 5. IEEE, 1998, pp. 2969–2972.

[Lai, 2010] Lai, C.-C. and Tsai, C.-C. “Digital image watermarking using discrete wavelet transform and singular value decomposition,” IEEE Transactions on instrumentation and measurement, vol. 59, no. 11, pp. 3060–3063, 2010.

[Mazurczyk, 2016] Mazurczyk, W., Wendzel, S., Zander, S., Houmansadr, A., Szczypiorski, K. “Information Hiding in Communication Networks: Fundamentals, Mechanisms, Applications, and Countermeasures”, Wiley-IEEE Press; 1 edition, February 2016.

[McDonald, 2000] McDonald, A., Kuhn, M. 2000. “StegFS: A Steganographic File System for Linux”. Proceedings of the Third International Workshop on Information Hiding, 463–477.

[Mishra, 2014] Mishra, A., Agarwal, C., Sharma, A. and Bedi, P. “Optimized gray- scale image watermarking using DWT-SVD and Firefly algorithm,” Expert Systems with Applications, vol. 41, no. 17, pp. 7858–7867, 2014.

[Naskar, 2014] Naskar, R. and Chakraborty, R. S. “Reversible digital watermarking: Theory and practices,” Synthesis Lectures on Information Security, Privacy, & Trust, vol. 5, no. 1, pp. 1–130, 2014.

[Neuner, 2016] Neuner, S., Voyiatzis, A., Schmiedecker, M., Brunthaler, S., Katzenbeisser, S., Weippl, E. “Time is on my side: Steganography in filesystem metadata”. 2016. Digital Investigation 18 : 76–86.

[NIST, 2015] FIPS 180-4: Secure Hash Standard (SHS). 2015. URL: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf [access: 25.02.2018]

[O’Ruanaidh, 1997] J. J. O’Ruanaidh, J. J., and Pun, T. “Rotation, scale and translation invariant digital image watermarking,” 1997.

[Olanrewaju, 2011] Olanrewaju, R. “Development of intelligent digital watermarking via safe region,” PHD, Electrical and Computer Engineering, International Islamic University Malaysia, Kulliyyah of Engineering, 2011.

[Potdar, 2005] Potdar, V. M., Han, S. and Chang, E. “A survey of digital image watermarking techniques,” in INDIN’05. 2005 3rd IEEE International Conference on Industrial Informatics, 2005. IEEE, 2005, pp. 709–716.

[Rivest, 2011] Rivest, R. “Reference implementation code for pseudo-random sampler for election audits or other purposes”. 2011. URL: https://people.csail.mit.edu/rivest/sampler.py [access: 25.02.2018]

[Szczypiorski, 2015] Szczypiorski, K., Janicki, A., Wendzel, S.: “The Good, The Bad And The Ugly: Evaluation of Wi-Fi Steganography”, Journal of Communications (JCM), Vol. 10(10), pp. 747-752, 2015.

[Szczypiorski, 2016] Szczypiorski, K. “StegIbiza: a New Method for Information Hiding in Club Music”, Proceedings of the 2nd International Conference on Frontiers of Signal Processing (ICFSP 2016), Warsaw, Poland, 15-17 October 2016, pp. 20-24.

[Szczypiorski, 2016a] Szczypiorski, K. 2016. “StegHash: New Method for Information Hiding in Open Social Networks”. IJET International Journal of Electronics and Telecommunication, 62 (4) : 347–352.

[Venckauskas, 2013] Venckauskas, A., Morkevicius, N., Petraitis, G., Ceponis, J. 2013. “Covert Channel for Cluster-based File Systems Using Multiple Cover Files”. ITC 42 (3) : 260–267.

[Wilson, 2014] Wilson, A., Blunsom, P., Ker, A.: “Linguistic Steganography on Twitter: Hierarchical Language Modelling with Manual Interaction”, Proc. SPIE 9028, Media Watermarking, Security, and Forensics 2014, 902803 (February 19, 2014).

[Zeki, 2009] Zeki, A. M. and Abdul Manaf, A. “A novel digital watermarking technique based on isb (intermediate significant bit),” World Academy of Science, Engineering and Technology, vol. 50, pp. 989–996, 2009.

[Zhang, 2006] Zhang, X. and Yang, Y. “A geometric distortion resilient image wa- termark algorithm based on dft-svd.” Jisuanji Gongcheng/ Computer Engineering, vol. 32, no. 18, pp. 120–121, 2006.

[Zigomitros, 2012] Zigomitros, A., Papageorgiou, A., and Patsakis, C. “Social network content management through watermarking,” in 2012 IEEE 11th Inter- national Conference on Trust, Security and Privacy in Computing and Communications. IEEE, 2012, pp. 1381–1386.