Two secure online steganography schemes based on HTTP request sequences

Leonel Moyou Metcheka, Stephane Gael Raymond Ekodeck, René Ndoundam

▶ **To cite this version:**

Leonel Moyou Metcheka, Stephane Gael Raymond Ekodeck, René Ndoundam. Two secure online steganography schemes based on HTTP request sequences. 2022. hal-03706829

HAL Id: hal-03706829
https://hal.science/hal-03706829

Preprint submitted on 23 Jul 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Two secure online steganography schemes based on HTTP request sequences

Leonel Moyou Metcheka[1,2,3], **Stephane Gael Raymond Ekodeck**[1,2,3], **René Ndoundam**[*1,2,3]

[1]Team GRIMCAPE
[2]Unversity, IRD, UMMISCO, F-93143, Bondy, France
[3]Department of Computer Science, University of Yaounde I, Yaounde, 812, Cameroon

*E-mail : ndoundam@yahoo.com

Abstract

The interest of steganography is to find ways to leak confidential information without being detected. Several methods proposed in the literature implement strategies for secret embedding in one or more covert media with increased security performance, by enhancing alteration resistance. Nevertheless, these embedding strategies are vulnerable to detection with blind universal steganalysis. This paper sheds light on two new secure covert channels, perfectly driven by online web operations that do not leave fingerprints. The design features of this undetectable scheme are based on lightweight requests for secret encoding coupled to the cover media independence. Experiments on this new approach have shown remarkable security results compared to existing ones. The results are significant and present promising research contributions.

Keywords

Undetectability ; Lightweight covert channel ; Tagged URLs ; HTTP request sequences

I INTRODUCTION

Steganography is the art and science of hiding information so that its presence cannot be detected. Its aim is to set up a covert channel for secret communications [14] so that no one except the recipient is aware of the existence of the secret message [12]. Classic steganography techniques conceal secret messages in digital content such as text, image, video and audio files [18, 31, 33, 35]. Hence, three properties are used to evaluate the performance of a steganographic scheme [29]. Firstly the capacity, which refers to the amount of secret bits that could be hidden inside a stego-covert. Secondly the robustness, which assesses the ability of the scheme to resist modification or destruction of the secret message. And finally the security, which assesses the inability of an intruder to detect the secret information exchanged. Ultimately, security is the most critical property desired to achieve undetectable communications [6].

Recently, distributed data hiding in multi-cloud storage environment [32] has been introduced based on the files integrity. This secret channel allows to bypass steg analysis since nothing is inserted or modified in the multimedia files handled. However, the content of the key and the exposed method are complex. Indeed, four parameters must be defined in the key while the process of inserting and extracting the secret requires managing multiple clouds with many multimedia files. In this paper, we propose two lightweight covert channels independent to

multimedia files that performs secret communications using HTTP request sequences. The two main parameters involved in the key are the tagged URLs and the IP addresses used in the communication. The method is based on the simple operations of opening web pages from the selected URLs. The secret is distributed via URLs opened by the sender in a methodical way, depending on the embedding method used.

The rest of the paper is organized as follows : section 2 describes existing steganographic schemes and their limits. The lightweight cover secure channels contribution are presented in section 3. Experimental results are done in section 4 and finally section 5 is devoted to the conclusion.

II COVERT CHANNELS DETECTION

This section discusses the steganographic schemes security that insert confidential information inside the covert media. Existing work is highlighted to show that an attacker is able to perform progressive scans of these covert media in order to detect or destroy the presence of the secret. The cover channels detection based on text, images, video and audio are presented.

Text steganography is classified into three categories [20]: format-based methods, linguistic methods as well as random and statistical generation methods. Format-based methods modify the format or the layout features of the cover text without altering the sentences or the words. Features consist of word spacing, line spacing, font style, text color [16, 18, 30]. Linguistic or natural language processing-based methods modify the syntax and semantics of text content using features such as synonyms, abbreviations, word similarity to hide secret bits [27, 37]. Random and statistical generation methods are used to generate automatically covert text with regard to the statistical properties of the secret message. To produce covert text in the natural language, data compression techniques or random covert can be use [17, 21]. Assuming an attacker who has access to the stego text, he can intentionally insert, modify, delete, reorder words or characters or even change the formatting of the text (color, font) and the character encoding (ASCII, UNICODE, UTF8). This inevitably leads to the destruction of the secret message. Furthermore, statistical analysis can be carried out to detect the covert channel [10, 28].

Image steganography is classified into three types: spatial domain techniques, transform domain techniques and adaptive techniques [36]. The spatial domain based image steganography directly modify the image pixels to integrate the secret bits. Usually, the techniques employed are: Least Significant Bit (LSB), palette based techniques, Pixel Value Differencing (PVD), multiple bit-plane based techniques and histogram shifting [22]. Instead of directly manipulating the pixels, secret insertion is done by modifying the coefficients in the transform domain. Here, the techniques used include: Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT) [23]. Adaptive methods are designed to target specific regions or pixels of the cover image to insert secret data to improve the efficiency of steganographic techniques. Human Visual System (HVS) is used for embedding as well as a threshold value [25]. Likewise, the detection of secret messages in images requires some advanced statistical testing such as higher-order statistics, Markov random fields, linear analysis and wavelet statistics [34]. Other recent techniques use blind universal detection methods to identify any new or unknown embedding schemes used in an image to hide a secret [11, 26].

Video steganography employs techniques applied to images for the embedding of secrecy, particularly in the spatial and transform domain [36]. However, the LSB technique of the spatial

domain is the most used to insert the secret bits in the videos [24]. Methods employing machine learning are implemented to detect secret messages inserted in video media [19].

Audio steganography is subdivided into four main categories: low-bit encoding, phase encoding, spread spectrum coding and echo hiding [15]. Regarding the low-bit coding technique, the secret is inserted into the audio sample using the LSB technique [13]. Phase encoding uses the phase components of sound so that it is not perceptible by human listening [1]. The secret bits are integrated as phase shifts in the phase spectrum of the audio signal. Spread spectrum coding consists in randomly spreading the secret data bits across the frequency spectrum of the audio signal [7]. Finally, the echo hiding technique hides the secret by introducing an echo in the discrete audio signal [5]. Detection methods can be applied for a specific type of audio steganography [9]. Also, universal detection methods are able to signal the presence of secret messages in audio media [8].

The common point of steganography techniques applied to multimedia files is the modification properties of the cover media for secret embedding, therefore leaving traces which can be detected by steg analyzes. Likewise, if an algorithm can determine the presence of a secret message in a covert media, then the whole steganographic system used is considered broken[4]. Considering these limits, we want to propose a secure steganographic scheme independent of multimedia files because it can be detected by current steganlysis methods, while preserving the integrity of the medium used.

III CONTRIBUTION

The basic idea is to use everyday online operations to have a stealthy signature that is undetectable. These operations concern the consultation of websites which is in a way unsuspected and goes practically unnoticed in case of concealment of secret information. The interest of this new lightweight covert channels is to achieve perfectly undetectable communications without the usual exchange of stego multimedia files. The sender consults the web pages and the receiver controls the website containing these web pages. The URLs of web pages are tagged in order to conceal information when browsing. Therefore, the contribution is divided into the following three main points :

- The lightweight covert channel: the scheme needs simple operation;
- Undetectable by steg analysis : the technique is based on usual internet browsing;
- Covert media independent : the technique used does not require the exchange or modifications of multimedia files.

Two schemes based on a stealth signature are proposed using online web browsing operations. The first scheme uses HTTP requests based on dependency inside sequences with permutations to transfer the secret. While the second scheme uses HTTP requests based on dependency between sequences and applications to transfer the secret.

3.1 Notations and hypothesis

These notations are useful both for the secret insertion and extraction:

- S : a secret message in binary ;
- B : the base value ;
- π : a permutation of order n such that $\pi = (\pi_0, \pi_1, ..., \pi_{n-1})$;
- r : the rank of a permutation of n elements;
- U : a list of n URL such that $U = (U_0, U_1, ..., U_{n-1})$;
- P : a list of m IP addresses such that $P = (p_0, p_1, ..., p_{m-1})$;

- $U_0, U_1..., U_{n-1}$: a sequence of URL such that :
  - $\forall U_i \in U, 0 \leq i \leq n-1$;
  - $\forall i_1, i_2, 0 \leq i_1, i_2 \leq n-1$;
  - if $i_1 \neq i_2$ then $U_{i_1} \neq U_{i_2}$.
- U^i : the i^{th} block of n URL selected : $U_0^i, U_1^i, ..., U_{n-1}^i$;

3.2 HTTP requests based on dependency inside sequences

3.2.1 Overview

In this first scheme, the HTTP request sequences are carried out by the sender in a methodical way in order to transfer the secret data. Indeed, the requests within a sequence are ordered and therefore dependent on each other. The communication model proposed in Figure 1, shows the sender who transfers the secret data by making sequences of HTTP requests to the receiver, which has a set of web pages accessible online. The access URLs to some of these web pages are labeled and called as tagged URLs. These URLs are used by the sender to encode the secret while the receiver uses the source IP address of the sender as well as the tagged URLs to decode the secret. Note that the numbers present in these requests represent the identifiers of the accessible URLs. Throughout Figure 1, several types of requests are applied to URLs. In the set of URLs available in the web server, a short list of n URLs is identified to form the tagged URLs. HTTP requests can be made on tagged URLs as well as on untagged URLs. A distinction is made between HTTP requests containing tagged URLs, coming from the IP address of the sender and normal users. To encode and decode the hidden data bits, the ranking and unranking functions of Myrvold et al. are used [3]. These two functions are described in the next section.

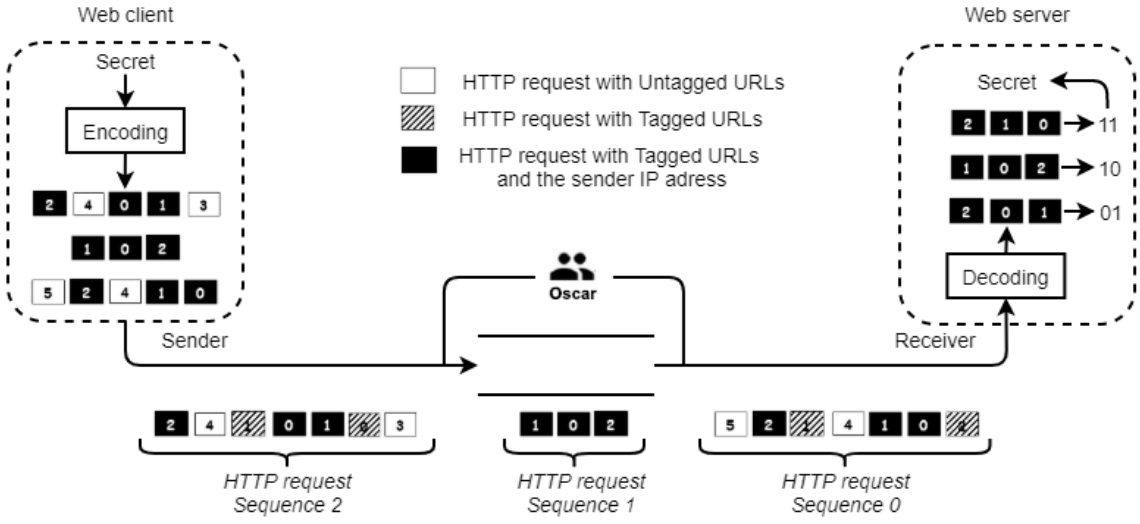

Figure 1: Overview of the proposed scheme.

3.2.2 Permutation generation functions

A permutation of order n is an arrangement of n elements. Thus, the number of permutations of n elements is $n(n-1)...2.1 = n!$. Consequently, at each permutation sequence $\pi_0, \pi_1..., \pi_{n-1}$ is associated a rank between 0 and $n! - 1$. To do such indexing, we use a ranking function and the reverse operation requires the unranking function. Indeed, for each of the $n!$ permutations of n symbols, the ranking function returns an integer in the interval $[0, n! - 1]$. Inversely, the unranking function takes as input an integer between 0 and $n! - 1$ and returns the permutation of n

symbols having this rank. Some related works [2] produce these permutations in lexicographic order. The works of Myrvold et al. [3] which are taken into account in this paper, generate these permutations but not in lexicographic order. Their respective pseudo-codes are described below.

Unranking function : to determine the permutation of n symbols of rank r, we must first initialize the array π to be the permutation identity $\pi[i] = i$, for $i = 0, 1, \cdots n-1$. Then call the unrank procedure below [3], with the following parameters: n, r and the initial permutation π. Once the procedure is completed, the expected permutation is available in the array π.

Procedure $unrank(n, r, \pi)$
begin
if $n > 0$ **then**
$swap(\pi[n-1], \pi[r \bmod n])$;
$unrank(n-1, \lfloor r/n \rfloor, \pi)$;
end;
end;

Ranking function : to determine the rank r associated with a permutation π of n symbols, we first initialize the array π^{-1} like this: $\pi^{-1}[\pi[i]] = i$, for $i = 0, 1, \cdots, n-1$. Then the rank function is called with the following parameters: n, π and π^{-1}. This returns the rank associated with the permutation π.

function $rank(n, \pi, \pi^{-1})$:integer
begin
if $n = 1$ **then** return(0) **end**;
$s := \pi[n-1]$;
$swap(\pi[n-1], \pi[\pi^{-1}[n-1]])$;
$swap(\pi^{-1}[s], \pi^{-1}[n-1])$;
return$(s + n.rank(n-1, \pi, \pi^{-1}))$;
end;

3.2.3 Secret encoding and decoding illustration

The number of permutations of n elements is $n!$ and each permutation has a rank between 0 and $n! - 1$. Taking $n = 3$ for this illustration, the number of permutations is $3! = 6$. We enumerate all the arrangements of the elements $0, 1, 2$. Consequently, each permutation sequence of 3 elements will be associated with a rank between 0 and 5. Table 1 illustrates the different permutations and their respective ranks. Thus, the coding of binary secret 10 requires the use of the sequence: 1 0 2, because it corresponds to rank 2. Similarly, the decoding of the sequence 2 1 0, reveals the secret value 3 which is equal to 11 in binary.

Table 1: Ranks and permutation sequences for n = 3.

0	1 2 0	**2**	1 0 2	**4**	0 2 1	**1**	2 0 1	**3**	2 1 0	**5**	0 1 2

3.2.4 The stego key

Two parameters must be shared between the sender and the receiver :

- A list of tagged URLs: $U = (U_0, U_1, ..., U_{n-1})$, $n \geq 2$;
- A list of sender IP addresses : $P = (p_0, p_1, ..., p_{m-1})$, $m \geq 1$.

3.2.5 *Embedding scheme*

In this first scheme, the secret represented in binary is divided into blocks. Each binary block is encoded by selecting a sequence of URLs. This opening sequence of URLs is determined by applying the unrank function to the decimal value of each block. The sender makes HTTP requests to the web server in this predefined order. Therefore, the secret embedding algorithm is as follows :

Input

S : the secret message;

p_0 : the sender IP address ;

π : the permutation array ;

U : the tagged URLs list;

Output

U' : the URL selected after browsing;

Begin

1. The secret S is represented in binary as follows : $S = (z_{q-1}...z_1z_0)_2$, where $0 \leq z_i \leq 1$;
2. The number of tagged URLs is determined as follows : $n = |U|$;
3. The size of a secret block is evaluated as follows : $l = \lfloor log_2(n!) \rfloor$;
4. The number of blocks is evaluated as follows: $k = \lceil q/l \rceil$;
5. The secret is split into k blocks of l bits : $S[(i \times l) + j]$, $0 \leq i \leq k-1$ and $0 \leq j \leq l-1$;
6. For each secret block $i = 0, 1, ..., k-1$:
   - 6.1. Compute the decimal value of the block : $r = \sum_{j=0}^{l-1}(S[(i \times l) + j] \times 2^j)$;
   - 6.2. Initialize the array π to be the permutation identity $\pi[j] = j$, $j = 0, 1, \cdots n-1$.
   - 6.3. Determine the permutation sequence $\pi_0, \pi_1..., \pi_{n-1}$ by applying the unrank function with the following parameters : n, r and π;
   - 6.4. Determine the URLs sequence : $U_{\pi_0}, U_{\pi_1}, ..., U_{\pi_{n-1}}$
   - 6.5. For each URL U_{π_j}, $j = 0, 1, ..., n-1$:
     - 6.5.1. Send HTTP requests on the URL U_{π_j} with the sender IP address p_0 ;
     - 6.5.2. Get the HTTP response of the request.

End

3.2.6 *Extraction scheme*

On the server side, the receiver records the links requested by the client as well as the associated IP addresses. Then, it sorts through the incoming requests those coming from a known source IP address. The related URL links are extracted in order of opening. The rank function is applied to the different sequences of links in order to reconstitute the corresponding secret block. The dynamic programming of websites in python language uses these two instructions to determine respectively the URL of HTTP requests and the remote IP address of the clients : $request.META.get('SCRIPT\_URI')$ and $request.META.get('REMOTE\_ADDR')$. In addition the *time()* function is used to determine the time at which a URL is visited. Therefore, the secret extraction algorithm is as follows :

Input

p_0 : the sender IP address ;

U : the tagged URLs list;

π : the permutation array ;

π^{-1} : the inverse permutation array ;

Output

S : the secret message;

Begin

1. Determine the number of tagged URLs : $n = |U|$;
2. Determine the amount of bits hidden in a list of n URL : $l = \lfloor log_2(n!) \rfloor$;
3. Capture URLs in order of opening HTTP requests;
4. Extract the tagged one belonging to the list U;
5. Filter only URLs from the single known IP address p_0;
6. The URLs obtained is subdivided into k blocks of n URLs : $U^0, U^1, ..., U^{k-1}$
7. Initialize S with the empty string : $S = \varepsilon$;
8. For each block $U^i = U^i_{\pi_0}, U^i_{\pi_1}, ..., U^i_{\pi_{n-1}}$, $i = 0, 1, ..., k-1$:
   - 8.1. Initialize the array π as follows: $\pi = \{\pi_0, \pi_1, ..., \pi_{n-1}\}$;
   - 8.2. Initialize the array π^{-1} as follows: $\pi^{-1}[\pi[j]] = j$, for $j = 0, 1, \cdots, n-1$;
   - 8.3. Compute the rank r of the sequence: $\pi_0, \pi_1, ..., \pi_{n-1}$ with the following parameters: n, π and π^{-1};
   - 8.4. Convert the value r to binary on l bits : $S' = (z_{l-1}...z_1 z_0)_2$;
   - 8.5. Concatenate the binary string S' to S: $S = S'||S$, where the symbol $||$ denotes the concatenation operation;
9. Return the secret S.

End

3.3 HTTP requests based on dependency between sequences

3.3.1 Overview

In this second scheme, the sequences of HTTP requests carried out by the sender are dependent on each other methodically in order to transfer the secret. In contrast, the queries within a sequence are unordered and therefore independent of each other. The communication model illustrated in Figure 2, shows how the secret is distributed on several IP addresses belonging to the sender. Each participant hides its discrete value by making an HTTP request on the tagged URL with the assigned discrete value as index. The tagged URLs contained in these HTTP requests are indexed from 0 to $n-1$. These requests are therefore identified by the IP address of the sender and the selected tagged URL. During the process, each participant can select untagged URLs without hindering the secret embedding. Likewise, normal users can select both tagged and untagged URLs without any impact in the dissimulation process. On the server-side, the receiver filters HTTP requests by retaining only those containing the IP address of the sender and the selected tagged URL. Then, these requests are ordered according to the list of IP addresses of the sender: ip_0, ip_1, ..., ip_{m-1}. Finally, the secret is obtained by extracting, in the same order, the indexes of the tagged URLs contained in these HTTP requests.

The added value of this scheme compared to the first one is that no sequential order constraint is established during the selection of the URLs by different IP addresses. They perform HTTP requests independently of each other without disturbing the decoding process.

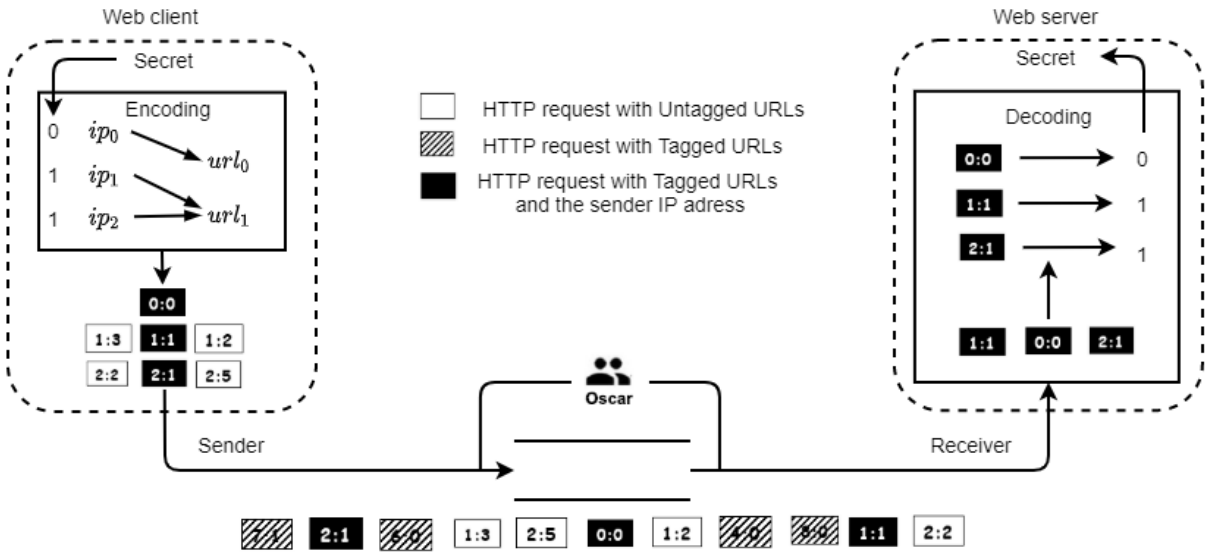

Figure 2: Overview of the proposed scheme.

3.3.2 The stego key

Two parameters must be shared between the sender and the receiver :

- A list of n tagged URLs: $U = (U_0, U_1, ..., U_{n-1})$, $n \geq 2$;
- A list of m sender IP addresses : $P = (p_0, p_1, ..., p_{m-1})$, $m \geq 1$;
- The base value B such that : $B \geq 2$ and $B = |U| = n$.

3.3.3 Embedding scheme

In this second scheme, the sender has control over multiple computers. These client machines are identified by their respective IP addresses. On the server-side, the receiver controls n web pages. The sender thus transcodes the secret to base B, then distributes the discrete values of the secret obtained between the client machines. Each client machine consults the URL that is mapped to the assigned discrete value. Therefore, the secret embedding algorithm is as follows:

Input
S : the secret message;

P : the IP addresses of the senders ;

U : the tagged URL list ;

B : the base value;
Output
U' : the URL selected after browsing;
Begin

1. The number of IP addresses of the senders is determined as follows : $m = |P|$;
2. The number of tagged URL is determined as follows : $n = |U|$;
3. The secret S is transcoded in base B as follows : $S = (z_{q-1}...z_1z_0)_B$,
   where $0 \leq z_i \leq B - 1$ and $m \geq q$;
4. The sender IP addresses $p_0, p_1, ..., p_{q-1}$ send HTTP requests respectively to URLs $U_{z_0}, U_{z_1}, ..., U_{z_{q-1}}$.
5. For $i = 0, 1, ..., q - 1$:
   5.1. Get HTTP response of each request.

End

3.3.4 Extraction scheme

The server side receiver reads the incoming requests. A filter is then applied to extract only the tagged URLs from known source IP addresses. Subsequently, each URL is associated with the corresponding position in the set of tagged URLs. The discrete values thus obtained are assembled to form the secret in base B, which will then be converted into base 2 to obtain the initial secret. Therefore, the secret extraction algorithm is as follows:

Input

P : the IP addresses of the senders ;

B : the base value;

U : the tagged URL list ;

Output

S : the secret message;

Begin

1. Capture URLs and source IP addresses in order of opening HTTP requests;
2. Extract only the tagged one belonging to the list U;
3. Filter only URLs from the known IP address list P;
4. Sort the URLs in the numbering order of the source IP addresses of the sender : $U_{z_0}, U_{z_1}, ..., U_{z_{q-1}}$;
5. Match each URL $U_{z_0}, U_{z_1}, ..., U_{z_{q-1}}$ with its respective position : $z_0, z_1, ..., z_{q-1}$;
6. Represent the discrete values obtained in base B : $S' = (z_{q-1}...z_1 z_0)_B$;
7. Recover the secret S by converting S' to binary.

End

IV EVALUATION

This section focuses on the evaluation of the two proposed schemes in terms of estimating the capacity of hidden bits and the number of HTTP request sequences necessary for a given secret embedding.

4.1 The sequence Capacity and number of HTTP request sequences

In the two proposed methods, the sender makes sequences of n HTTP requests using m distinct IP addresses. The secret S to transfer is represented in the base B and the number of discrete values obtained is noted: $|(S)_B|$. Table 2 evaluates the capacity of the secret bits of an HTTP request sequence as well as the number of HTTP request sequences generated for n tagged URLs.

Table 2: The Sequence capacity and number of HTTP request sequences of the two proposed schemes

	Sequence capacity	# HTTP request sequences
Scheme 1	$\lfloor log_2(n!) \rfloor$	$\left\lceil \frac{\lvert(S)_B\rvert}{log_2(n!)*m} \right\rceil$
Scheme 2	$\lfloor log_2(n^m) \rfloor$	$\left\lceil \frac{\lvert(S)_B\rvert}{log_2(n^m)} \right\rceil$

As shown in Table 3 the first scheme has a higher secret bit capacity compared to the second scheme for a secret of 1024 bits. This is because the information contained in the permutations increase rapidly with respect to the increase in symbol space n. Similarly, the number of HTTP request sequences generated for the 1st scheme is very low compared to the second.

Table 3: Comparison example of the two proposed schemes

	n	m	Sequence capacity	# HTTP request sequences
Scheme 1	64	8	296	1
Scheme 2	64	8	48	22

To generalize, the higher the values of n and m, the greater the capacity and the lower the number of request sequences. The latter is visible in Table 2, with fractions that only depend on n and m.

4.2 Discussion

By comparing the two proposed schemes, the 1st having a dependency between HTTP requests within a sequence is slower in execution time because it follows an order of execution of requests to transfer the secret. However, it has a good capacity for secret bits and a reduced number of HTTP request sequences. On the contrary, the second scheme lifts this time limit with the absence of dependency between HTTP requests inside a sequence. But as a disadvantage the capacity of the secret bits is smaller and the number of HTTP request sequences higher. Finally, the two proposed schemes are complementary. Nevertheless, the 1st scheme is better than the second in terms of capacity of secret bits to transfer and the number of HTTP requests generated.

V CONCLUSION

In this paper, we have proposed two new secure covert channels, perfectly driven by online web operations that do not leave fingerprints. The design features of this undetectable scheme are based on lightweight requests for the secret encoding coupled to the cover media independence. The proposed scheme is simple and robust regardless of the size of the secret to be transmitted. Our scheme differs perfectly from others in that it does not modify or use files to conceal information. This considerably improves the scheme security. The comparative analysis showed that the 1st scheme is more efficient than the second in terms of capacity of secret bits to transfer and the number of HTTP requests generated.

REFERENCES

Publications

[1] W. Bender, D. Gruhl, N. Morimoto, and A. Lu. “Techniques for data hiding”. In: *IBM systems journal* 35.3.4 (1996), pages 313–336.

[2] C. T. Djamegni and M. Tchuenté. “A cost-optimal pipeline algorithm for permutation generation in lexicographic order”. In: *Journal of Parallel and Distributed Computing* 44.2 (1997), pages 153–159.

[3] W. Myrvold and F. Ruskey. “Ranking and unranking permutations in linear time”. In: *Information Processing Letters* 79.6 (2001), pages 281–284.

[4] J. Fridrich and M. Goljan. “Practical steganalysis of digital images: state of the art”. In: *Security and Watermarking of Multimedia Contents IV*. Volume 4675. International Society for Optics and Photonics. 2002, pages 1–13.

[5] D.-Y. Huang and T. Y. Yeo. “Robust and inaudible multi-echo audio watermarking”. In: *Pacific-Rim Conference on Multimedia*. Springer. 2002, pages 615–622.

[6] I. S. Moskowitz, L. Chang, and R. E. Newman. “Capacity is the wrong paradigm”. In: *Proceedings of the 2002 workshop on New security paradigms*. 2002, pages 114–126.

[7] D. Kirovski and H. S. Malvar. “Spread-spectrum watermarking of audio signals”. In: *IEEE transactions on signal processing* 51.4 (2003), pages 1020–1033.

[8] M. K. Johnson, S. Lyu, and H. Farid. “Steganalysis of recorded speech”. In: *Security, Steganography, and Watermarking of Multimedia Contents VII*. Volume 5681. International Society for Optics and Photonics. 2005, pages 664–672.

[9] W. Zeng, H. Ai, and R. Hu. “A novel steganalysis algorithm of phase coding in audio signal”. In: *Sixth International Conference on Advanced Language Processing and Web Information Technology (ALPIT 2007)*. IEEE. 2007, pages 261–264.

[10] Z. Chen, L. Huang, Z. Yu, W. Yang, L. Li, X. Zheng, and X. Zhao. “Linguistic steganography detection using statistical characteristics of correlations between words”. In: *International Workshop on Information Hiding*. Springer. 2008, pages 224–235.

[11] X.-Y. Luo, D.-S. Wang, P. Wang, and F.-L. Liu. “A review on blind detection for image steganography”. In: *Signal Processing* 88.9 (2008), pages 2138–2157.

[12] L. Y. Por and B. Delina. “Information hiding: A new approach in text steganography”. In: *WSEAS international conference. Proceedings. Mathematics and computers in science and engineering*. Volume 7. World Scientific, Engineering Academy, and Society. 2008.

[13] R. Sridevi, A. Damodaram, and S. Narasimham. “EFFICIENT METHOD OF AUDIO STEGANOGRAPHY BY MODIFIED LSB ALGORITHM AND STRONG ENCRYPTION KEY WITH ENHANCED SECURITY.” In: *Journal of Theoretical & Applied Information Technology* 5.6 (2009).

[14] C.-C. Chang and T. D. Kieu. “A reversible data hiding scheme using complementary embedding strategy”. In: *Information Sciences* 180.16 (2010), pages 3045–3058.

[15] P. Jayaram, H. Ranganatha, and H. Anupama. “Information hiding using audio steganography–a survey”. In: *The International Journal of Multimedia & Its Applications (IJMA) Vol* 3 (2011), pages 86–96.

[16] K. F. Rafat and M. Sher. “Secure digital steganography for ASCII text documents”. In: *Arabian Journal for Science and Engineering* 38.8 (2013), pages 2079–2094.

[17] Z.-H. Wang, H.-R. Yang, T.-F. Cheng, and C.-C. Chang. “A high-performance reversible data-hiding scheme for LZW codes”. In: *Journal of Systems and Software* 86.11 (2013), pages 2771–2778.

[18] S. G. R. Ekodeck and R. Ndoundam. “PDF steganography based on Chinese Remainder Theorem”. In: *Journal of Information Security and Applications* 29 (2016), pages 1–15.

[19] M. Fan, P. Liu, H. Wang, and X. Sun. “Cross correlation feature mining for steganalysis of hash based least significant bit substitution video steganography”. In: *Telecommunication Systems* 63.4 (2016), pages 523–529.

[20] S. Sharma, A. Gupta, M. C. Trivedi, and V. K. Yadav. “Analysis of different text steganography techniques: a survey”. In: *2016 Second International Conference on Computational Intelligence & Communication Technology (CICT)*. IEEE. 2016, pages 130–133.

[21] A. Malik, G. Sikka, and H. K. Verma. “A high capacity text steganography scheme based on LZW compression and color coding”. In: *Engineering Science and Technology, an International Journal* 20.1 (2017), pages 72–79.

[22] K. Muhammad, J. Ahmad, N. U. Rehman, Z. Jan, and M. Sajjad. "CISSKA-LSB: color image steganography using stego key-directed adaptive LSB substitution method". In: *Multimedia Tools and Applications* 76.6 (2017), pages 8597–8626.

[23] M. Saidi, H. Hermassi, R. Rhouma, and S. Belghith. "A new adaptive image steganography scheme based on DCT and chaotic map". In: *Multimedia Tools and Applications* 76.11 (2017), pages 13493–13510.

[24] S. Chen and Z. Qu. "Novel quantum video steganography and authentication protocol with large payload". In: *International Journal of Theoretical Physics* 57.12 (2018), pages 3689–3701.

[25] T. Luo, G. Jiang, M. Yu, H. Xu, and W. Gao. "Sparse recovery based reversible data hiding method using the human visual system". In: *Multimedia Tools and Applications* 77.15 (2018), pages 19027–19050.

[26] S. Wu, S. Zhong, and Y. Liu. "Deep residual learning for image steganalysis". In: *Multimedia tools and applications* 77.9 (2018), pages 10437–10453.

[27] L. Xiang, W. Wu, X. Li, and C. Yang. "A linguistic steganography based on word indexing compression and candidate selection". In: *Multimedia Tools and Applications* 77.21 (2018), pages 28969–28989.

[28] X. Zuo, H. Hu, W. Zhang, and N. Yu. "Text semantic steganalysis based on word embedding". In: *International Conference on Cloud Computing and Security*. Springer. 2018, pages 485–495.

[29] A. Gutub and K. Alaseri. "Hiding shares of counting-based secret sharing via Arabic text steganography for personal usage". In: *Arabian Journal for Science and Engineering* (2019), pages 1–26.

[30] B. Khosravi, B. Khosravi, B. Khosravi, and K. Nazarkardeh. "A new method for pdf steganography in justified texts". In: *Journal of information security and applications* 45 (2019), pages 61–70.

[31] S. Jiang, D. Ye, J. Huang, Y. Shang, and Z. Zheng. "SmartSteganogaphy: Light-weight generative audio steganography model for smart embedding application". In: *Journal of Network and Computer Applications* (2020), page 102689.

[32] L. Moyou and R. Ndoundam. "Distributed data hiding in multi-cloud storage environment". In: *Journal of Cloud Computing: Advances, Systems and Applications* (2020).

[33] U. Pilania and P. Gupta. *Analysis and Implementation of IWT-SVD Scheme for Video Steganography*. 2020.

[34] A. K. Sahu and M. Sahu. "Digital image steganography and steganalysis: A journey of the past three decades". In: *Open Computer Science* 10.1 (2020), pages 296–342.

[35] A. K. Sahu and G. Swain. "Reversible image steganography using dual-layer LSB matching". In: *Sensing and Imaging* 21.1 (2020), page 1.

[36] M. Dalal and M. Juneja. "Steganography and Steganalysis (in digital forensics): a Cybersecurity guide". In: *Multimedia Tools and Applications* 80.4 (2021), pages 5723–5771.

[37] G. Deepthi, N. V. SriLakshmi, P. Mounika, U. Govardhani, P. L. Prassanna, S. Kavitha, and A. D. Kumar. *Linguistic Steganography Based on Automatically Generated Paraphrases Using Recurrent Neural Networks*. 2022.