Towards Reversible Storage Network Covert Channels

Wojciech Mazurczyk
Przemysław Szary
Warsaw University of Technology
Warsaw, Poland
wmazurczyk@tele.pw.edu.pl
przemyslaw.szary.stud@pw.edu.pl

Steffen Wendzel
Worms University of Applied Science
Worms, Germany
wendzel@HS-Worms.de

Luca Caviglione
National Research Council of Italy
Genova, Italy
luca.caviglione@ge.imati.cnr.it

ABSTRACT

The use of network covert channels to improve privacy or support security threats has been widely discussed in the literature. As today, the totality of works mainly focuses on how to not disrupt the overt traffic flow and the performance of the covert channels in terms of undetectability and capacity. To not void the stealthiness of the channel, an important feature is the ability of restoring the carrier embedding the secret information into its original form. However, the development of such techniques mainly targets the domain of digital media steganography. Therefore, this paper applies the concept of reversible data hiding to storage network covert channels. To prove the effectiveness of our idea, a prototypical implementation of a channel exploiting IPv4 flows is presented along with its performance evaluation.

CCS CONCEPTS

• **Security and privacy → Network security**; *Distributed systems security*; *Information flow control*; Pseudonymity, anonymity and untraceability.

KEYWORDS

Network Steganography, Covert Channels, Information Hiding, Reversible Data Hiding.

ACM Reference Format:
Wojciech Mazurczyk, Przemysław Szary, Steffen Wendzel, and Luca Caviglione. 2019. Towards Reversible Storage Network Covert Channels. In *Proceedings of the 14th International Conference on Availability, Reliability and Security (ARES 2019) (ARES '19), August 26–29, 2019, Canterbury, United Kingdom.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3339252.3341493

1 INTRODUCTION

Information hiding and steganography embrace a variety of techniques to hide and transfer data. For instance, they can be used to store information for copyright management or to guarantee provenance, covertly transmit data to prevent censorship or hide sources in investigative journalism [14]. However, information hiding is becoming increasingly adopted to empower cyber attacks [10], [17]. Recent examples of threats show how it can be used to create covert channels to allow malware to secretly interact with a remote Command & Control (C&C) facility, exfiltrate sensitive data, implement side channels enabling two colluding processes or virtual machines to bypass sandboxes, or to exploit anti-forensics techniques [2].

In general, information hiding is used to embed secret data into a suitable carrier, which can be exchanged between two endpoints to bypass a third-party observer, often called the *warden*. For the case of cybersecurity, a very interesting scenario is when the carrier embedding information is a network flow or traffic artifact (e.g., TCP segments or the behavior of the throughput) and the two secret endpoints communicate through the Internet [5, 9]. Many of the methodologies proposed in the literature alter the carrier permanently when embedding data [4, 18]. This behavior can be used by the warden to spot the hidden data transfer, thus partially voiding the network covert channel or facilitating the development of detection tools [9, 18]. Therefore, being able to revert the carrier to its original form, i.e., before the injection of the secret by the sender or its extraction by the receiver, it is mandatory to have effective network steganographic mechanisms. At the same time, anticipating such an issue permits to fully evaluate network security and to correctly decide where the detection should be performed.

Reversible data hiding (RDH) techniques have been already addressed in the literature for the case of digital media steganography, i.e., where the carrier is a digital artifact like a picture, movie or audio track. For instance, in [13] authors mention reversible techniques such as lossless compression, difference expansion, histogram shifting, prediction-error expansion, integer-to-integer transformation and pixel-value ordering. Other two recent works addressing RDH are [15] and [8] presenting steganographic techniques to not permanently alter the exploited digital images.

For the case of network carriers, the literature abounds of works on how information hiding techniques can be used to create network covert channels, see, e.g., references [9], [18], [6] and [12]. Two main paradigms can be used: *timing* in which the information is encoded by manipulating the evolution of a quantity over time (e.g., the shape of the throughput or the amount of data sent in a specific time window) or *storage* in which the information is injected in suitable parts of the traffic flow (e.g., in the bits used for padding a packet). Unfortunately, about the totality of techniques used to implement a network covert channel do not consider reversibility, even if some techniques could be easily endowed with RDH mechanisms. At the best of our knowledge, the only exception is presented in [11], where authors introduce a method further compressing the voice sampled in an IP telephony service to free space for embedding data. To avoid the disruption of the vocal flow and the detection due to a reduced audio quality, the transcoded voice samples are restored to the previous format, thus (implicitly) implementing an RDH technique.

Therefore, our work aims at filling this research gap and focuses on the definition of an RDH paradigm for the case of network covert storage channels. The main contributions of the paper are: the introduction of a conceptual framework to describe properties and issues of reversible information techniques when applied to network traffic; an assessment on the use of IPv4 as a carrier for RDH; a preliminary performance analysis to evaluate the effectiveness of the proposed reversibility concepts.

The rest of this paper is structured as follows. Section 2 introduces reversible data hiding, while Section 3 discusses the feasibility of building reversible storage covert channels in IPv4. Section 4 showcases the algorithms to endow the covert channel with reversibility and Section 5 deals with numerical results. Lastly, Section 6 concludes the paper and provides some future research directions.

2 PROBLEM STATEMENT

In general, developing a steganographic technique with reversibility properties should not be intended as a goal *per se*. In fact, pursuing RDH could require additional algorithms to restore the original flow, thus causing overheads in terms of software complexity or used resources. This may reduce the undetectability of the covert channel, for instance, due to increased CPU or power drains [3]. Despite this, RDH could be mandatory when in the presence of distributed wardens. In fact, if the traffic can be collected in different portions of the network and then compared, discrepancies in replicas of the carrier can reveal the presence of the hidden communication.

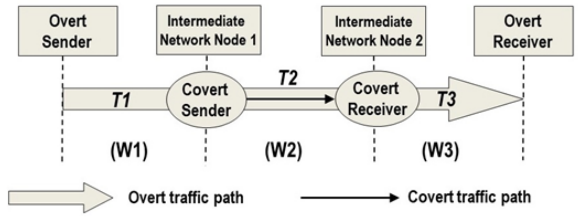

Figure 1: Reference scenario for the data exchange through a network covert channel considered in this work.

To better discuss this point, Figure 1 depicts the reference scenario where secret data is exchanged by two covert endpoints using a Man-in-the-Middle (MitM) scheme. In more detail, the covert sender injects secret information into the traffic flow generated by the overt sender and the covert receiver extracts the secret before the traffic reaches the overt receiver. The overt traffic (denoted as T1 in the Figure 1) is steganographically modified by the covert sender located in the intermediate network node 1 (INN1). Thus, due to the injection of data, it takes the form of T2. In the ideal case, the more T2 resembles T1, the better. If the covert channel is created by using a typical network steganography technique, the covert data is extracted at the intermediate node 2 (INN2), but the traffic is not restored to the previous form T1. Yet, if both the endpoints implement an RDH mechanism, when the data extraction process is finished, the traffic T3 passed to the overt receiver will be shaped to replicate T1 as close as possible.

As regards the detection, let us consider a warden able to inspect the overt traffic only in a single network location. The possible locations where a probe can be placed are reported in Figure 1 and denoted as W1, W2 and W3. Despite the location, the warden can only detect the covert communication by exploiting local data, for instance perturbation of some statistical indicators [9]. Instead, let us consider a *distributed* warden, that is a warden able to inspect the traffic in at least two portions of the network. For instance, the warden can collect steganographically modified traffic in a specific link (e.g., W2) and compare it with a flow gathered in other parts (e.g., W1 or W3). In this case, a comparison of the traffic gathered in two locations, e.g., (W1, W2) or (W1, W3) can reveal the presence of the covert communication (under the assumption that an algorithm able to detect alteration is available). Obviously, the more the measurements, the higher the chance to reveal discrepancies in the overt traffic and eventually to spot the channel. To counteract the presence of a distributed warden, let us assume to use an RDH-based method to create the covert channel. For instance, if the warden inspects traffic in W1 and W3, INN2 can fully "reverse" the flow by restoring the carrier to the form originally sent by the overt sender, i.e., this leads to T1=T3 making the covert data exchange undetectable.

According to the previous reference scenario, we now introduce a definition for reversible network covert channels allowing to describe RDH applied to the case of network steganography.

Definition - *Reversible Network Covert Channels.* Reversible network covert channels are those where the covert receiver can restore the traffic to the form originally transmitted by the overt sender (full reversibility) or at least partially (quasi reversibility) before passing it to the overt receiver. Changes that would happen during network traversal, such as a checksum recalculation or alteration of the inter-packet timing due to buffering in routers, are not considered. Applying reversibility should not worsen the undetectability of the network covert channel □

We point out that the definition is quite general as the traffic could be fully restored only in an ideal case, whereas in realistic scenarios the carrier could be only partially reverted to its original form. In fact, traffic restoration is seldom an ON/OFF process and the flow produced by the covert receiver could have different degrees of accuracy, compared to the traffic produced by the overt sender. Therefore, we introduce three levels of reversibility to classify (reversible) network covert channels:

- *fully-reversible*: the secret receiver can completely revert the altered fields to the original state.
- *quasi-reversible*: it is not possible to precisely determine the original "state" of the carrier, hence the covertly communicating endpoints can only restore it in a statistical manner.
- *non-reversible*: the covert receiver has no access to the proper knowledge required to restore the protocol data units or network statistics to the original state.

For the case of fully-reversible channels, the secret sender can add information to reverse the process or the secret receiver can infer the original state by past observations. In general, a quasi-reversible network covert channel could enforce the secret receiver to "guess" the original state of the carrier in a probabilistic manner. For instance, this could be the case for mechanisms injecting information in the Time To Live (TTL) field of IPv4. In this case, the receiver has to observe previous TTL values sent from the overt sender and then re-match the original distribution. Typically, the longer the covert receiver is able to observe the statistical properties of a carrier, the more accurate the RDH technique would be. Moreover, a non-reversible channel could be the result of a warden performing some form sanitization (e.g., a firewall overwriting unused protocol fields) as well as some intermediate network nodes altering the state of the overt traffic (e.g., aggressive buffering in routers could reshape the inter-packet time or the throughput).

To complete our framework, we introduce the cases under which reversibility can be achieved. Specifically:

- *intrinsically*: the data hiding technique is constructed in such a way that the covert receiver is able to completely restore the overt traffic to its original form.
- *explicitly*: the covert sender also transmits the "state" of the overt traffic that is utilized as a hidden data carrier to allow the covert receiver to restore the carrier to its original state. This may require to sacrifice a part of the available steganographic bandwidth, thus, slowing down the covert communication exchange.
- *implicitly*: the covert receiver restores the overt traffic by removing the secret message and by "guessing" the potential original form of the overt traffic before the steganographic modifications were applied, for instance, via estimations or past observations.

As regards methods intrinsically equipped with some RDH technique, literature does not contain many of them. At the best of our knowledge, currently no network covert channels enjoy intrinsically RDH, with the only notable exception of [11], which under some circumstances (i.e., the specific encoding mechanism used to transcode the voice) has a built-in reversibility. We point out that this is a fortunate side-effect rather than a precise design choice. This also highlights the importance of investigating past implementation with the aim of understanding reversibility properties.

Concerning covert channels explicitly developing reversibility, they can exchange the "state" of the original carrier. This is the opposite behavior characterizing implicitly channels, where the state can be only inferred. Such a meta-data can be in the form of a simple covert channel-internal control protocol. As discussed in [16], one needs to consider the available space per packet s_{pkt} for embedding such a control protocol. The information about the original state O of the overt traffic must then be embedded together with the covert data C, so that $|O| + |C| \leq |s_{pkt}|$. It follows, that when the storage of O within the available space s_{pkt} already consumes all the space, no additional covert data could be embedded, unless O could be compressed by the covert sender, i.e., using a compressor $\mathfrak{I}$, such as *zip*. Taking the considerations of [16] and the availability of a compressor into account, the maximum remaining space for covert data within one packet is: $max(|C|) = |s_{pkt}| - |\Im(O)|$. However, to unify the implementation of a channel for different potential original states $O_1 \ldots O_n$, the maximum amount of space that O can consume must be considered, which allows to calculate the always available space for C, i.e., $min(|C|) = s_{pkt} - max(|\Im(O_1)|, \ldots, |\Im(O_n)|)$.

Finally, to be able to express the "degree" of the RDH, we introduce the *reversibility factor*. We provide its definition below.

Definition - *Reversibility Factor.* Reversibility Factor is the fraction of the carrier's steganographic modifications that can be effectively reversed. □

3 FEASIBILITY ANALYSIS: THE CASE OF IPV4

In this section, we discuss the result of a feasibility analysis on the use of various fields in the header of IPv4 as carriers to implement a reversible storage network covert channel. We point out that the use of network protocols to create covert channels has been largely investigated in the literature. For instance, references [9] and [18] survey many techniques targeting the TCP/IP stack, whereas reference [7] is dedicated to evaluate different methodologies that can be used with IPv6. Moreover, reference [1] deals with techniques that can be used to covertly exfiltrate data through transitional mechanisms between IPv4/IPv6 networks. However, none of the aforementioned works provide an assessment of the adopted injection methodology in terms of reversibility. Therefore, we evaluated the reversibility of the most popular storage methods targeting fields of IPv4 and the outcome is resumed in Table 1.

As shown, not all fields can be successfully used to implement a covert channel. For instance, the alteration of the `Version` field could make routers to discard the datagram. A similar case is for `HL`: even if some encoding scheme can be implemented by modulating the size of optional fields, such values could be updated by intermediate routers, hence the original form of the header cannot be restored by the covert receiver.

For the case of `ToS`, its value can be estimated by inspecting past data, i.e., datagrams received by the host where the covert receiver operates. Instead, when using `Total Length`, `Identifier`, and the `Flags`, the steganographer can apply a different approach. For instance, when the overt transmission begins, the covert sender can forward a number of packets in an unmodified manner so that the cover receiver will be able to capture the original values of the fields in the IPV4 header. The knowledge inferred from such an initial trial of packets is then used to revert the flow of data to its original form with different level of reversibility. The number of unmodified packets must be agreed beforehand between the covert sender and the covert receiver to not interfere with the hidden data exchange. Moreover, using portions of the overt traffic to infer the "state" of the IPv4 header will lead to a reduced bandwidth for the covert channel as some data units cannot be used as carrier. A similar behavior characterizes covert channels using the `Protocol`, `Source IP Address`, and `Options` fields.

As regards the `Fragment Offset`, this field is usually set to zero, thus it can be easily restored by the covert sender. However, when packets are delivered through multiple routing paths, they might not all traverse the hop/node in charge of reversing the field to restore the carrier. As a consequence, the destination will not be able to restore the fragment offset, thus disrupting the flow or make the method irreversible. A similar consideration can be done for the `Padding`, except that its presence does not depend on the route.

The `TTL` can be estimated from past observations but this requires to know the routing path in its entirety, as well as some stability of the endpoints, e.g., mobility or multihoming may lead to changing values for the TTL [19]. Even if injecting the data in the `TTL` field could appear as a simple technique, covert channels using such scheme can be considered as protocol-independent. For instance, they can be easily ported to IPv6, which offers a similar field, the `Hop Count` [7] as well as to protocols using some form of counter to prevent loops or to limit their scope. Lastly, the `Destination IP Address` has to be considered non-reversible, as its alteration would lead to routing issues.

To the aim of evaluating the effectiveness of reversible storage network covert channels, in the following we evaluate the performances of three methods presented in Table 1, i.e., `DSCP`, `Total Length`, and `Src.Address`.

4 IMPLEMENTING REVERSIBLE DATA HIDING

In order to implement a storage network covert channel, we developed an embedding scheme exploited by the covert sender and an homologous procedure implemented in the covert receiver. The algorithms used to inject and extract data are reported in Algorithm 1 and 2, respectively.

Specifically, for the case of the procedure used by the covert sender to embed data, the function `convert_to_binary()` converts the secret message into a binary form, e.g., from a human readable format to ASCII. We point out that this function could be complex, for instance also implementing some form of encryption or scrambling as to make the message impossible to read if spotted, or prevent its detection by using some form of entropy-based analysis [9]. Besides, the function `sniff_packet()` captures the traffic from the interface and the function `send_overtpacket()` is called a fixed number of times (5 in our example) and provides a "clean" representation of the carrier (this is a form of the *explicit* RDH mechanism as discussed in Section 2). In this way, the receiver can develop a template on which base the

Table 1: An analysis of the IPv4 header fields from the reversibility perspective (bold entries have been evaluated in Section 5)

Header field	Reversibility	Description
Version	-	Cannot be modified as it may lead to packet processing problems at intermediary devices
Header Length (HL)	Non-reversible	Cannot be directly modified as it depends on the number of IPv4 options and may lead to packet processing problems at intermediary devices
DSCP	Quasi/fully-reversible	Values in this field are predictable so they should be reversible
Total Length	Fully reversible	This field does not change during normal routing (except for fragmentation)
Identifier	Non/quasi-reversible	This field changes for every packet and thus it is difficult to be reversed. But some OSs have predictable sequence for the Identifier making it usable for reversibility purposes
Flags	Quasi-reversible	Flags are predictable and can be used for reversibility, especially the Don't Fragment flag for established flows (except the MTU path discovery case)
Fragment Offset	Fully reversible	These bits are often unused, so it is easy to reverse them completely
TTL	Quasi-reversible	In controlled environments it is possible to estimate the original value making the field reversible
Protocol	Fully reversible	This field is constant throughout a single connection, but intermediate hops might drop packets with values not present in the standard implementation
Header Checksum	Non-reversible	Incorrect checksums (due to injected secret information) would lead to packet drops by intermediate routers/noes
Src.Address	Fully reversible	Feasible but limited to the unidirectional use
Dst. Address	-	This field cannot be modified as it will lead to routing issues
Options	Quasi-reversible	This field is not always present, but it can be reversed
Padding	Fully reversible	This field is not always present, but it can be reversed

Algorithm 1 - Covert Sender Embedding

```
Input: secret_message, field
packet = sniff_packet(field);
for $i$ = 1 to 5 do
    send_overtpacket();
end for
while secret_message do
    $bindata \leftarrow$ convert_to_binary();
    for $data$ in $bindata$ do
        embed_data_with_steg(packet, field);
    end for
end while
stop_transmission()
```

restoration of the carrier. A special value to indicate the end of the transmission is sent to the covert receiver via `stop_transmission()`. The function `embed_data_with_steg()` depends on the specific steganographic method and targets the desired header element `field` and manipulates the given protocol data unit captured on the wire, i.e., `packet` is a pointer to a structure containing the packet. It must be noted that the function also transmits the packet.

At the receiving side, the extraction process (depicted in Algorithm 2) firstly checks how the real-life values from the selected field are changing. The function `select_most_common_value()` selects the most likely values that were measured. Such a pool of measurements constitutes a sort of snapshot of the carrier and it will be used later for restoring traffic. Afterwards, the covert receiver starts reading the secret message. This part of the algorithm also depends on the steganography method defined as `embed_data_with_steg()` in Algorithm 1. As an example, if we consider a message stored in ASCII format, for each 8 bits of information, a single element in the table is created. Then the modified value is reversed using the `known_value` and the packets will be forwarded to the overt receiver. This is achieved using the `forward_packet()` function. When the covert sender sends `stop_transmission()`, the receiver can consume the message or deliver it to the upper layers of the protocol stack.

Algorithm 2 - Covert Receiver Extraction and Restore

```
1:  Input: field
2:  for $i = 1$ to 5 do
3:      table[] ← sniff_packet(field)
4:  end for
5:  known_value ← select_most_common_value(table)
6:
7:  stop_sniffing = false
8:  while not stop_sniffing do
9:      in_packet = sniff_packet()
10:     $bin$ ← extract_steg(in_packet, field)
11:     data.append(bin)
12:     if len(data) == 8 then
13:         letters.append(data)
14:         clear_data()
15:         if letters.last() == end_of_transmission then
16:             stop_sniffing = true
17:         end if
18:     end if
19:     field = known_value
20:     forward(in_packet)
21: end while
22:
23: for $x$ in letters do
24:     $x$ ← convert_to_string()
25:     secret_message.append(x)
26: end for
27: print(secret_message)
```

5 EXPERIMENTAL RESULTS

To assess the performances of reversible covert channels, we created an experimental setup by using different nodes communicating through a controlled network environment. To this aim, we used VirtualBox to create virtual machines hosting the covert sender and the covert receiver both implemented with Kali Linux. Instead, the endpoints generating the overt traffic are Ubuntu-based nodes. To implement the MitM attack used to gather and manipulate the traffic we used the `arpspoof` tool. The covert channels considered in this performance evaluation are those injecting data in the `Src.Address`, `ToS`, and `Total Length` header fields. To implement the embedding and extracting techniques, we used the `Scapy` library and ad-hoc Python modules. To capture and process the traffic, we used `thsark`. Then, by means of ad-hoc scripts we performed per-packet comparisons as well as we computed the performance indicators used in the following. As regards the warden, it is implemented in a separate virtual machine running Wireshark.

The experiments have been conducted in three different scenarios. In the first two scenarios, we considered an overt traffic flow composed of 50, 000 and 15, 000 ICMP packets, respectively. In the following, we will denote them as Sc. 1 and Sc. 2, respectively. Instead, in the third scenario, denoted in the following as Sc. 3, we used an overt traffic flow made of an FTP transfer of a file of 30 MBytes, which accounted for about 10, 000 IP datagrams. In all the cases, the considered network topology is composed of a controlled LAN environment containing one router and five hosts, i.e., the secret endpoints, the covert endpoints, and the warden.

Numerical Results

We now discuss the numerical results collected in our trials.

First, we measured the impact in terms of additional delays introduced by the use of a reversible mechanism to create the covert channel. Results of the delay experienced by each packet (averaged over the entire flow composed of 15, 000 data units) are reported in Table 2. Specifically, when no steganographic attempts are present, the delays are only limited to those introduced by the network, e.g., the time needed to traverse the different protocol stacks and buffers. Instead, when the secret sender starts injecting data, delays increase. This could be ascribed to the overhead introduced by the sniffing, embedding and re-transmission operations. Instead, when the RDH-enabled mechanism is introduced, the delays increase up to 10 times, compared with the direct, clean transmission of data. On one hand, this is consequence of the overhead introduced by the presence of additional intermediate nodes processing packets. On the other hand, the impact of a non-optimized implementation using Python could exacerbate delays. Therefore, as a part of our ongoing research, we aim at investigating a network covert channel with a reduced detectability, e.g., by preventing too many context switches between the kernel and user space accounting for overheads [14]. Moreover, we point out that experienced delays appear to be independent of the IPv4 field used as the carrier. In fact, for all cases and tests, results were similar.

Another important measure deals with the Reversibility Index (RI), which is a specialized implementation of the *Reversibility Factor* defined in Section 2. The RI quantifies the level of reversibility and it has been defined as the ratio between successfully reversed packets and all the packets composing the stream. We point out that, in this work, the RI is the same for all the considered covert channels, thus it does not depend upon the field used for embedding data.

Table 3 presents the obtained results. According to the table, the covert channels exploiting the `Src.Address` and `DSCP` fields for the Sc. 2 are fully reversible. However, for the Sc. 3 the RI is close to 100% and therefore the method can be treated only as quasi-reversible. In other words, in this case almost every packet has been successfully reversed and only few of them were not correctly restored. This can be ascribed to the number of ICMP messages used for the

Table 2: Delay results for the reversible network covert channels scenario. Values are in ms.

	Direct	MitM	RDH
Avg.	0.5037	0.7517	4.8835
Std. Dev.	0.0345	0.128	0.685

Table 3: Reversibility Index (RI) [%] obtained in our trials for the three considered scenarios.

CC type	RI (Sc. 1)	RI (Sc. 2)	RI (Sc. 3)
Src.Address	98.94	100	95.59
DSCP	93.57	100	98.49
Total Length	51.1	55.61	6.65

experiments: 50, 000 and 15, 000 packets in Sc. 1 and Sc. 2, respectively. When sending the high number of ICMP traffic in Sc. 1 some of the messages were lost thus they could not be reversed. However, the increased amount of lost packet could be used as an indicator of the secret exchange of data. Therefore, the impact of reversibility on the various features characterizing the traffic should be carefully evaluated. At the same time, this underlines the importance of understanding all the aspects of the injection of data within an overt network flow, mainly in the perspective of developing effective countermeasures or mitigation techniques. A notable exception is the covert channel using the `Total Length` field in Sc. 3, where we failed to reverse the carrier. This is caused by the existing inter-dependency of the `Total Length` field of the IPv4 header with the `Length` field of the Ethernet frame. Specifically, by modifying the values in the header by embedding the secret data, the frame length adjusts itself automatically. Then, when we restore the value in the IPv4 header back, the corresponding frame length remains unchanged. This behavior has been also detected by Wireshark and increases the detectability of this method. In this case, it would be better to trade RDH properties for stealthiness of the covert channel.

To recap, this preliminary performance evaluation showcases the feasibility of implementing a reversible network steganography channel. However, a more thorough investigation is needed, for instance to produce reversed carriers that do not have statistical signatures, i.e., the RDH technique should match the distribution of the real values for the chosen field.

6 CONCLUSIONS

In this paper, we have introduced the idea of reversible data hiding already known from digital media steganography to network covert channels. Also, we have reviewed how fields in the header of IPv4 can be used to implement reversible storage covert channels. For three methods, we have shown a performance evaluation. Results indicate the feasibility of implementing reversible network covert channels. However, this requires to transfer some covert channel-free traffic in advance, thus penalizing the bandwidth that can be used to transmit the secret.

Future works aim at performing more detailed measurements and considering other paradigms besides the storage one. Nevertheless, we are working towards the development of proper countermeasures against malware endowed with reversible data hiding features, which could evade from many detection techniques.

ACKNOWLEDGMENTS

This work has been partially supported by the EU Project SIMARGL, Grant Agreement No 833042, and by the Polish National Agency for Academic Exchange, Grant Agreement No PPN/BEK/2018/00153.

REFERENCES

[1] Bernhards Blumbergs, Mauno Pihelgas, Markus Kont, Olaf Maennel, and Risto Vaarandi. 2016. Creating and detecting IPv6 transition mechanism-based information exfiltration covert channels. In *Nordic Conference on Secure IT Systems*. Springer, 85–100.

[2] Krzysztof Cabaj, Luca Caviglione, Wojciech Mazurczyk, Steffen Wendzel, Alan Woodward, and Sebastian Zander. 2018. The New Threats of Information Hiding: The Road Ahead. *IT Professional* 20, 3 (May 2018), 31–39.

[3] Luca Caviglione, Mauro Gaggero, Enrico Cambiaso, and Maurizio Aiello. 2017. Measuring the energy consumption of cyber security. *IEEE Communications Magazine* 55, 7 (2017), 58–63.

[4] L. Caviglione, M. Podolski, W. Mazurczyk, and M. Ianigro. 2017. Covert Channels in Personal Cloud Storage Services: The Case of Dropbox. *IEEE Transactions on Industrial Informatics* 13, 4 (Aug 2017), 1921–1931. https://doi.org/10.1109/TII.2016.2627503

[5] Luca Caviglione, Steffen Wendzel, and Wojciech Mazurczyk. 2017. The future of digital forensics: Challenges and the road ahead. *IEEE Security & Privacy* 15, 6 (2017), 12–17.

[6] Adel El-Atawy, Qi Duan, and Ehab Al-Shaer. 2015. A novel class of robust covert channels using out-of-order packets. *IEEE Transactions on Dependable and Secure Computing* 14, 2 (2015), 116–129.

[7] Norka B Lucena, Grzegorz Lewandowski, and Steve J Chapin. 2005. Covert channels in IPv6. In *International Workshop on Privacy Enhancing Technologies*. Springer, 147–166.

[8] Bin Ma, Xiaoyu Wang, Bing Li, and Yun-Qing Shi. 2018. A Multiple Linear Regression Based High-Accuracy Error Prediction Algorithm for Reversible Data Hiding. In *International Workshop on Digital Watermarking*. Springer, 195–205.

[9] Wojciech Mazurczyk and Luca Caviglione. 2014. Steganography in modern smartphones and mitigation techniques. *IEEE Communications Surveys & Tutorials* 17, 1 (2014), 334–357.

[10] Wojciech Mazurczyk and Luca Caviglione. 2015. Information Hiding as a Challenge for Malware Detection. *IEEE Security Privacy* 13, 2 (Mar 2015), 89–93.

[11] Wojciech Mazurczyk, Pawel Szaga, and Krzysztof Szczypiorski. 2014. Using Transcoding for Hidden Communication in IP Telephony. *Multimedia Tools Appl.* 70, 3 (June 2014), 2139–2165.

[12] Wojciech Mazurczyk, Steffen Wendzel, Sebastian Zander, Amir Houmansadr, and Krzysztof Szczypiorski. 2016. *Information hiding in communication networks: fundamentals, mechanisms, applications, and countermeasures*. Vol. 7. John Wiley & Sons.

[13] Bo Ou, Xiaolong Li, Wei Li, and Yun-Qing Shi. 2018. Pixel-Value-Ordering Based Reversible Data Hiding with Adaptive Texture Classification and Modification. In *International Workshop on Digital Watermarking*. Springer, 169–179.

[14] Sabine Schmidt, Wojciech Mazurczyk, Radoslaw Kulesza, Jörg Keller, and Luca Caviglione. 2018. Exploiting IP telephony with silence suppression for hidden data transfers. *Computers & Security* 79 (2018), 17–32.

[15] Chang Song, Yifeng Zhang, and Guojun Lu. 2018. Reversible Data Hiding in Encrypted Images Based on Image Partition and Spatial Correlation. In *International Workshop on Digital Watermarking*. Springer, 180–194.

[16] Steffen Wendzel and Jörg Keller. 2011. Low-attention forwarding for mobile network covert channels. In *Proc. Communications and Multimedia Security (CMS 2011) (LNCS)*, Vol. 7025. Springer, 122–133.

[17] Steffen Wendzel, Wojciech Mazurczyk, Luca Caviglione, and Michael Meier. 2014. Hidden and Uncontrolled – On the Emergence of Network Steganographic Threats. In *ISSE 2014 Securing Electronic Business Processes*, Helmut Reimer, Norbert Pohlmann, and Wolfgang Schneider (Eds.). Springer Fachmedien Wiesbaden, Wiesbaden, 123–133.

[18] Steffen Wendzel, Sebastian Zander, Bernhard Fechner, and Christian Herdin. 2015. Pattern-based Survey and Categorization of Network Covert Channel Techniques. *Computing Surveys* 47, 3 (2015).

[19] Sebastian Zander, Grenville Armitage, and Philip Branch. 2007. An empirical evaluation of IP Time To Live covert channels. In *2007 15th IEEE International Conference on Networks*. IEEE, 42–47.