Covert Channels in Personal Cloud Storage Services: the case of Dropbox

Luca Caviglione†, Maciej Podolski*, Wojciech Mazurczyk*, Massimo Ianigro†
*Warsaw University of Technology, Institute of Telecommunications, Warsaw, Poland
podolskimaciej91@gmail.com, wmazurczyk@tele.pw.edu.pl
† Institute for Intelligent Systems for Automation (ISSIA), Genova, Italy
luca.caviglione@ge.issia.cnr.it, ianigro@ba.issia.cnr.it

***Abstract*—Personal storage services are one of the most popular applications based on the cloud computing paradigm. Therefore, the analysis of possible privacy and security issues has been a relevant part of the research agenda. However, threats arising from the adoption of information hiding techniques have been mainly neglected. In this perspective, the paper investigates how personal cloud storage services can be used for building covert channels to stealthy exchange information through the Internet. To have a realistic use-case, we consider the Dropbox application and we present the performance evaluation of two different covert communication methods. To understand the stealthiness of our approach and propose countermeasures, we also investigate some behaviors of Dropbox in a production quality deployment.**

I. Introduction

Nowadays, cloud computing frameworks are widely used both from desktops and mobile appliances to access resources on-demand at reduced costs. Within this panorama, personal cloud storage is definitely one of the most popular applications [1]. Owing to this success, a core part of the research focused on security and privacy threats of cloud computing [2], [3], [4], together with new issues such as the handling of Big Data [5]. As regards personal cloud storage applications, [6] addresses hazards characterizing the entire life-cycle of the stored information. The works [7], [8] focus on solutions to guarantee security and service level agreement requirements. Reference [9] discusses how to undertake digital forensics investigations on personal data storage applications, while [10] proposes a general framework for a publicly auditable infrastructure.

However, emerging threats and attacks taking advantage of information hiding have been partially neglected. Even if such technique can be also used for licit purposes (e.g., to prevent censorship), it is primarily utilized to develop new malware able to covertly exfiltrate data while remain unnoticed for long times or to bypass sandbox-based policies of smartphones [11], [12]. Despite the growing attention on the adoption of covert channels to perform attacks, only few works consider cloud computing [13], [14]. The most relevant is [15], which demonstrates how to create a side channel in cloud storage services when the deduplication feature is enabled. The main idea is that, if a file is already present in the cloud, all clients would be informed to prevent unnecessary uploads of data. This knowledge could be used as a vector to transmit information or to perform a bruteforce attack against a stored secret. However, this method has a very limited feasibility, since service-wide deduplication is no longer used in commercial services and countermeasures have been developed [16]. Reference [17] partially deals with information hiding, since it addresses the data leakage among virtual machines running on the same server by means of covert channels using load or cache measurements.

As today, many personal cloud storage applications exist, for instance Apple iCloud, Microsoft OneDrive, Google Drive and Dropbox. However, Dropbox is the most popular one and outperforms the others in terms of users and produced traffic [1], [18]. In this perspective, the paper discusses the design and implementation of different covert channels exploiting the internals of the Dropbox platform, which is considered as a synecdoche of personal cloud storage services. Understanding this type of covert channels is of particular importance, since they mainly lie within huge traffic volumes and mixed sets of information. Thus, their detection or mitigation could lead to cyber security issues involving Big Data. To prevent that a too aggressive exchange of hidden information is easily recognized as an anomaly and to propose countermeasures, this work also partially investigates some network behaviors of Dropbox. However, providing a detailed traffic analysis is outside the scope of this work, and it has been already done in [1], [19].

At the best of our knowledge, this is the first work applying information hiding techniques to enable two endpoints covertly exchanging data through Dropbox. Prior investigations only focus on classic security and privacy attacks (see, e.g. [20]), but they are outside the scope of our work. Therefore, the main contributions are: *i*) the analysis of different covert channels targeting personal cloud storage services, *ii*) the implementation and the performance evaluation of two techniques, *iii*) a specific "behavioral" analysis of Dropbox users, and *iv*) the definition of countermeasures to prevent hidden channels in personal cloud storage services.

The rest of the paper is structured as follows. Section II briefly introduces the architecture of Dropbox. Section III presents the reference scenario and possible covert channels exploiting personal cloud storage applications. Section IV deals with the design of two techniques, Section V presents numerical results collected in a real-world testbed, and Section VI proposes countermeasures and mitigation techniques. Lastly, Section VII concludes the paper.

II. Dropbox: an Overview

Dropbox[1] is a cross platform application running on OSX, iOS, Android, Windows/WindowsPhone and Linux. In essence, it offers a storage service, which can be accessed from the browser or automatically synced with the local file system via an ad-hoc client interface running in background. It also enables collaboration by allowing users to exchange up-to-date contents located in a shared folder. The Web version of Dropbox offers a basic backup service, which can be used to revert to a prior version of a file, for instance to cope with accidental deletions or modifications.

To store and synchronize files, Dropbox relies upon two different cloud computing infrastructures [21]. The first is based on the Amazon S3 storage service, which is responsible for physically hosting files. While the organization of the file system of each user is forced by the guest OS, the Dropbox architecture internally adopts namespaces. Each user (or shared folder) has a unique root namespace and contents are located by using relative paths. Then, the client interface exchanges with the Amazon cloud the traffic needed to upload, update and retrieve a given file together with metadata describing the content. When a user locally creates or changes a file, it is synchronized with remote servers. To avoid excessive bandwidth requirements and to increase the Quality of Experience (QoE), files are split into blocks having a maximum size of 4 Mbytes, each one marked with an SHA-256 hash. The list of hashes is used to identify the file in a unique way in the framework including the file system journal [22]. Handling small portion of data in a separate manner allows to transmit to servers (named block servers) only blocks that have changed, rather than the whole file. This is used by the client interface to perform further optimizations. The first is delta encoding, which detects modifications and upload/download blocks selectively. The second improves the throughput by means of compression [19]. An additional enhancement pipelines the transmission of blocks to reduce the latency experienced by users [23]. According to the Dropbox site, the core functionalities for exchanging files are implemented via librsync [24]. The produced traffic represents the data plane of Dropbox and it is transported via TCP and encrypted with the Secure Socket Layer (SSL) [1].

Dropbox also enables devices located in the same LAN to sync. Recalling that files are stored in the Amazon datacenter, the normal approach requires that a device sends data to the cloud while the other retrieves the modified blocks from the Internet. From version 2.10, Dropbox implements a "stream sync" feature, i.e., the downloading client interface retrieves blocks even if the uploading peer has not completed the commit. This can still lead to performance bottlenecks, hence users would benefit if a direct communication among the two devices is possible. To this aim, Dropbox uses an additional protocol, named db-lsp, composed of two parts. The first is db-lsp-disc and queries the local network every 30 seconds to find devices by means of broadcast UDP messages sent on port 17500. If a device is found, the second part of the db-lsp directly starts an encrypted TCP conversation on the same port to negotiate parameters and directly exchange data. To avoid multiple incoherent snapshots in the system, the Amazon cloud has to be updated as well, but this can happen in background without degrading the perceived QoE. At the time of writing, the db-lsp is only supported by client interfaces running on desktops.

[1] http://www.dropbox.com

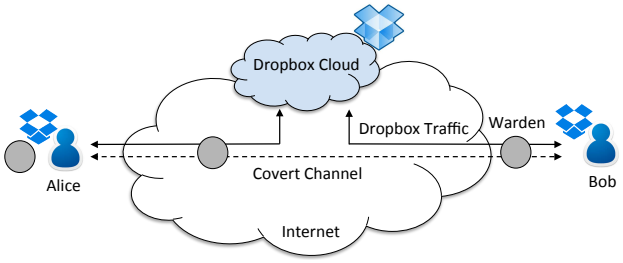


Fig. 1. Reference scenario for the development of a covert channel targeting the Dropbox personal cloud storage service.

The second datacenter composing the architecture is operated by Dropbox and it is in charge of authenticating users, handling the proper "signaling" traffic and keepalives to recognize if a host has disconnected. Specifically, it notifies the client interfaces about modifications of a shared content and it sends pointers to retrieve the needed blocks from the Amazon datacenter. In this case, proper servers (named metadata servers) "resolve" hashes and exchange information with the Amazon cloud to preserve the coherence of the information. Another important portion of the framework is given by the notification protocol, which dispatches all the changes performed on a file throughout the entire architecture. The resulting flow of traffic represents the control plane and its functionalities are implemented via an HTTP-based protocol exploiting long polling transmitted through TCP/SSL [1].

III. Reference Scenario and Analysis of Channels

In this section, we present the considered reference scenario and we showcase potential covert channels exploiting Dropbox. In the following, we use interchangeably the terms network covert channel, hidden communication, and steganographic channel, except when doubts arise [25], [26]. Besides, we consider network steganography as the technique to create covert channels for hidden communication. However, such covert channels do not exist in communication networks without steganography (only the "possibility" exists a priori) [26]. In general, covert channels are characterized by the following performance indexes [26], which will be used in the rest of the paper:

- *steganographic bandwidth*: defines the volume of secret data sent per time unit;
- *undetectability*: quantifies the inability of discovering the presence of secret information within the carrier;
- *robustness*: represents the resistance of a covert channel against impairments.

We assume that Alice and Bob want to establish a covert channel to communicate in a stealthy manner through the Internet. All the messages are inspected by a Warden, which can be placed everywhere in the path. This general use case is

a variant of the "prisoners' problem" formulated by Simmons [27] and depicted in Figure 1. For the sake of simplicity, the Amazon and Dropbox datacenters have been coalesced into a unique entity named "Dropbox Cloud". We point out that this scenario is general enough also to represent a malware using an information-hiding-capable communication layer to exfiltrate data from the host of a victim (Alice) towards a remote Command & Control facility (Bob) [28].

Different parts of the Dropbox architecture can be used as a carrier for storing secret information. As an example, Alice and Bob could share a folder via their client interfaces. In this case, each alteration of its content (e.g., the name of a file is modified) is notified to the cloud and then propagated to all the subscribers. Alice can encode secrets in well-defined patterns of operations, i.e., two consecutive changes of the same file denotes `1`, while deleting a file denotes `0`. On the receiving side, Bob can decode the information by knowing the shared secret, i.e., the encoding scheme used by Alice. Therefore, the covert channel lies within the sequence of legitimate actions propagated by the notification protocol of Dropbox.

A Warden can be arbitrarily placed to spot the secret flow of information. It can be deployed on the local host of Alice (or Bob), e.g., in the file system, to detect suspicious operations due to the ongoing data hiding process. It can be put in an intermediate network node to reveal the presence of the hidden channel from some features of the resulting legitimate traffic, e.g., the volume of notifications deviates from the average user behavior thus representing an anomaly. Consequently, the steganographic bandwidth of the covert channel between Alice and Bob and its detectability are tightly coupled [29]. For the considered example, the more secret bits are exchanged (i.e., the number of modifications performed to the shared folder), the higher is the traffic produced by Alice and Bob towards the Dropbox cloud. Thus, a proper trade-off between transmission rates and stealthiness has been searched for by means of traffic measurements.

A. Covert Channels Analysis

By considering the reference scenario depicted in Figure 1, we investigate potential covert channels exploiting personal cloud storage services.

First, we present methods using the network traffic of Dropbox as the carrier. Therefore, the secret sender is a client interface and the secret receiver decodes information in an intermediate network node, as depicted in Figure 2 - cases A and B. In more details:

1) volume-based: secrets are encoded by generating different amounts of traffic over the network. For instance, the secret can be produced by syncing a file multiple times (time-based encoding) or by varying its size (volume-based encoding). This method has two main drawbacks: *i*) a Warden can easily detect the covert channel by inspecting the overt network traffic or by monitoring the file system of the hosting machine [30], [31], [32], and [33]; *ii*) optimization algorithms prevent a fine-grained control on the traffic volumes, e.g., if a file/bock already exists in the cloud, only a notification is produced rather than a complete transmission [1], [19].

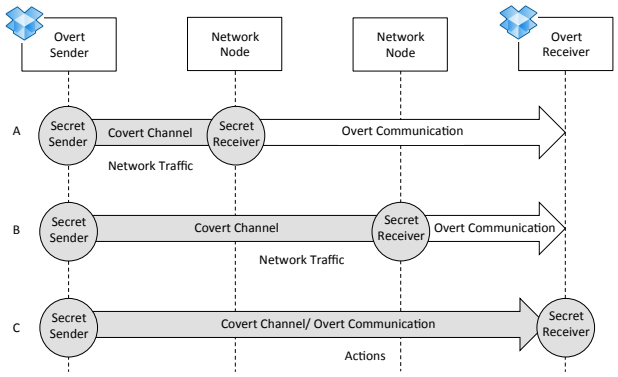


Fig. 2. Different covert/overt channels exploiting a personal cloud storage service.

2) throttle-based: secrets are encoded in well-defined values of the throughput. However, the presence of delta encoding may impede to have enough flexibility to obtain arbitrary throughput values to represent the information. The Warden can exploit the same techniques of 1) or simply void the channel via traffic normalization [30].
3) timing-based or storage-based: secrets are encoded by modifying certain features of the Dropbox traffic (e.g., delays between two packets) or by changing the header content. The Warden can utilize the same countermeasures presented in 1) and 2).
4) type-based: secrets are encoded by forcing the client interface to switch between the db-lsp and rsync protocols. For instance, the secret data `110` can be notified by producing network packets carrying a db-lsp, db-lsp and rsync sequence. Unfortunately, the usage of db-lsp limits the scope of the covert channel to the LAN.
5) time-stamp: each minute, the client interface sends a time-stamp to the Dropbox cloud within an HTTP request, which can be used as the carrier.

Second, we discuss methods considering Dropbox as a black box. In this case, protocols and internals are ignored, and data is injected by posing as a human user. In other words, the secret sender processes the content of a folder shared with the secret receiver, which infers information. The resulting covert channel lies within the flow of overt notifications/actions exchanged among client interfaces and the Dropbox cloud, as depicted in Figure 2 - case C. Specifically:

6) upload of new files: secret bits are hidden in the time at which the secret sender adds new files to the shared folder. Unfortunately, the intervals observed by the secret receiver heavily depend on the status of the network. This limits the "rate" of file uploads, reduces the achievable steganographic bandwidth, and makes the method very fragile to errors. The injection of a secret requires many operations on the file system, thus a Warden residing on the guest OS can easily spot the channel.
7) renaming of files: the hidden data is embedded in the name of files assigned by the secret sender. Since changing the filename only requires to update a metadata, this leads to a small amount of traffic. Even if a Warden placed in the network has limited chances to detect the

channel, one deployed on the host can spot massive renaming operations. Yet, reducing the throughput and using suitable name-encoding or encryption scheme for filenames can increase the stealthiness of the process.

8) movement of files: secrets are encoded by moving or deleting files within the shared folder. Since each change on the file hierarchy of a Dropbox folder has to be propagated, bits are signaled via bursts of operations. Again, changes should be limited as to avoid triggering a Warden in the local file system.
9) size modulation: the hidden information is encoded by shrinking/inflating the size of files. If the number of changes/secrets is small, Dropbox optimizations can limit the traffic making the detection difficult. Indeed, aggressive adjustments lead to anomalous network behaviors resulting in to a channel with a poor stealthiness.
10) alteration of file: the hidden data is embedded in how the file is modified. Owing to optimizations, changes in a single block produce a minimal amount of traffic. This requires the Warden to identify short-living TCP bursts within the bulk of data, which is non-trivial especially if only one block has changed.
11) type of device: Dropbox can keep track of which device modified a file. Secrets are encoded by creating a device-to-file map, e.g., modifying a file on a smartphone is `1`, while on a desktop is `0`. Alas, such information is not propagated among client interfaces, but requires to log into the Dropbox website. The resulting implementation is byzantine and easy to spot.
12) modification of folder: this method exploits folders instead of files. From the traffic point of view, the resulting load is very modest, as blocks composing files are never transmitted. Yet, massive alterations of folders cause abnormal bursts of notifications that can can be detected by a Warden.

For the sake of completeness, we briefly present a technique directly injecting data in the Dropbox client interface. In this case, the cloud portion of the framework is bypassed via the db-lsp protocol, which is used to establish a direct TCP point-to-point channel over the LAN. Unfortunately, since August 2013, the Dropbox client interface has been updated three times per month (on the average), making the mechanism difficult to implement and maintain, especially for a real-world attacker targeting a mixed set of platforms. Moreover, the method could be beaten via normalization [30] or detected by using Deep Packet Inspection (DPI) [34].

Summarizing, methods 1) - 3) are not Dropbox-specific and have been previously used in the literature, see, e.g., [26], [35]. Additionally, the Warden can detect or block the covert channel by simply inspecting the traffic, thus reducing its effectiveness. Methods 4) and 5) have been reported only for the sake of completeness, since 4) can not be used on an Internet-wide scenarios and it is limited to desktops, while 5) is no more available as Dropbox now uses TLS preventing a secret receiver to access and decode the data embedded in the time-stamp. Methods 6), 8) and 12) should have poor bandwidth as to avoid that a Warden deployed on the host can easily spot the burden of operations performed on the file system. The method proposed in 9) has a poor stealthiness since the creation of new blocks produces non-negligible flows of data, which can be spotted. Lastly, 11) requires the Web version of Dropbox, thus making the implementation of the covert channel highly detectable. For such reasons, in the following we investigate techniques 7) and 10) since they are the most promising and rely upon different information hiding approaches with a mixed set of complexity.

IV. Design of the Covert Channels

In this section, we describe the design of two methods exploiting Dropbox to establish a covert channel.

As discussed in Section III, we assume that both the secret sender and the secret receiver (i.e., Alice and Bob) share a Dropbox folder populated with some innocent-looking files. We also assume that they known in advance the set of files to be used for the steganographic purposes. Other files can be present and act, for instance, as a decoy. Without loss of generality, methods can work in two modes: synchronous, i.e., both endpoints have to be active at the same time, or asynchronous, i.e., the receiving side does not need to continuously monitor changes.

A. Renaming of File Method

The renaming of file method, denoted in the following as REN, exploits the filename as the carrier, i.e., the secret information is encoded in a textual form. Accordingly, it solely relies upon text steganography. With L, we define the number of characters that can be used to hide information. Despite the file system of the guest OS, Dropbox limits the length of filenames to 255 characters, hence $L \leq 255$. To hide data, the secret sender implements a proper function, which maps the message to be transmitted M into a string of characters S containing the secret. If $S \leq L$, it can be injected into a single filename, while for $S > L$ multiple files are needed, or many rounds of processing have to be applied on the file. Figure 3(a) depicts a simple graphical representation of the method. More sophisticated schemes using only a fraction of L characters or using a cross-file approach to reduce the probability of being spotted by a Warden can be developed, but it is outside the scope of this paper. Possible examples for encoding secrets are: character existence, e.g, if a specific letter is present `1` is inferred and `0` otherwise, or a direct injection by using the UTF-8 encoding as a dictionary.

By considering a synchronous usage, the following steps are performed: 1) the original names of files in the shared folder are saved along with their MD5 signatures; 2) the secret sender modifies the name(s) of the chosen file(s) with characters encoding the secret data; 3) upon detecting changes, Dropbox synchronizes and propagates such information to the secret receiver; 4) upon detecting changes, the secret receiver decodes the secret data from the filename(s); 5) when the hidden data transfer is completed, the name of file(s), identified via their MD5 signatures, is restored.

When operating in synchronous mode, the secret sender is only aware that data has been synchronized with the

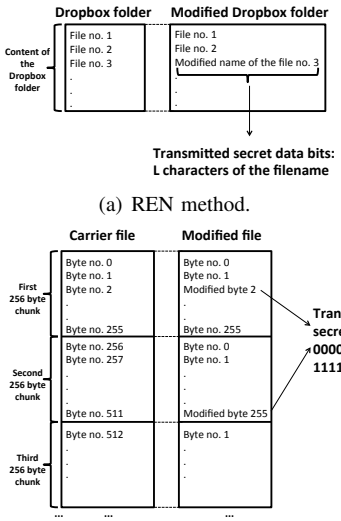
cloud, but it has no knowledge of the outcome. This could create incoherent statuses. For instance, the modification of a filename can occur twice before the receiving peer is notified. Hence, this requires a proper control protocol, e.g., to let the secret receiver acknowledge the secret sender that the content has been read. However, developing such form of signaling is outside the scope of this work. A possible workaround would be using an asynchronous mode. In this case, one can exploit the revision feature of Dropbox. The secret receiver can retrieve from the Web interface the list of all the changes made as to reconstruct the history of modifications to extract the secrets.

As regards a possible implementation, the REN mechanism can be realized by using the Dropbox client interface, via the web-based access or in a mixed form. For instance, the secret sender could change data by using proper scripts running on a shell, while the secret receiver can be a Rich Site Summary (RSS) daemon polling its web account and periodically inspecting filenames to extract the hidden information.

B. Alteration of File Method

The alteration of file method, denoted in the following as ALT, uses the contents of the file as the carrier, i.e., the secret data is embedded in "where" the file is modified. In essence, ALT takes advantage from the delta encoding algorithm, i.e., only parts of files differing from those already available in the personal cloud storage are uploaded. Let us assume a message to be covertly transmitted with a size of M bytes, denoted as $m_1, m_2, \ldots, m_M$. Then, with a little abuse of notation, the covert sender implements an encoding scheme to hide M into the S bytes of a file. To this aim, the carrier (i.e., the file) of size F is subdivided into K equal chunks, each one of C bytes, with $S < K = F/C$. Notice that chunks are only used for encoding the data and must not be confused with the blocks used by Dropbox to transfer the information. In the following, to avoid ambiguities, we refer to parts of the file altered for steganographic purposes solely as chunks. For the sake of simplicity, we assume $C = 256$ bytes, since it allows to transfer one byte of secret data per chunk. In Section V, we relax such hypothesis and we consider different values for C. Additionally, the smaller C, the higher K, then the resulting changes could be stealthier as only a small fraction of the original content is modified. We point out that $C = 256$ bytes allows to precisely add up to 4 Mbytes, which is the maximum block size supported by Dropbox.

Then, the encoding scheme used is as follows. First, the designated file is divided into K chunks. Then, the covert sender converts m_1 into its decimal representation, defined as d_1, which is used as a pointer to modify the carrier, i.e., the d_1-th byte of the first chunk is modified. Upon detecting changes, Dropbox synchronizes the content with the cloud and pushes data towards the secret receiver. The latter extracts the secret information by comparing the new and the old chunks contained in the updated blocks as to discover the position of the changed byte containing the secret, i.e., d_1. The process is then iterated for the remaining $M - 1$ bits of information and, finally, the file is restored. Figure 3(b) provides a graphical representation of the data hiding process.



(a) REN method.

(b) ALT method.

Fig. 3. Representation of methods used to encode secret data.

Same considerations of the REN method about using single or multiple files, the need of developing a proper signaling, as well as pitfalls regarding synchronous/asynchronous implementations hold. The ALT method requires to tightly interact with files, thus the covert channel should be implemented by using the client interface jointly with ad-hoc scripts to properly manipulate contents [36]. In fact, the need of reverting a file to its original form is more critical, as altering random bytes could lead to corrupted files, which can be easily spotted.

V. Performance Evaluation

To evaluate the performance of the covert channels, we implemented the secret sender and the secret receiver on Linux desktops by using the Python Dropbox APIs v. 2.2.0 and the CLI version of the client interface.

For each round of tests, two different cases have been considered. The first addresses an error-free scenario, i.e., the steganographic channel has been only affected by uncontrollable errors or delays introduced by the Internet. The second introduces additional impairments to quantify the robustness of channels. To this aim, delays have been added by using `netem` and we used `IPtables` to simulate losses and to avoid the need of a proxy. To collect traffic traces, we used the `Wireshark` traffic analyzer.

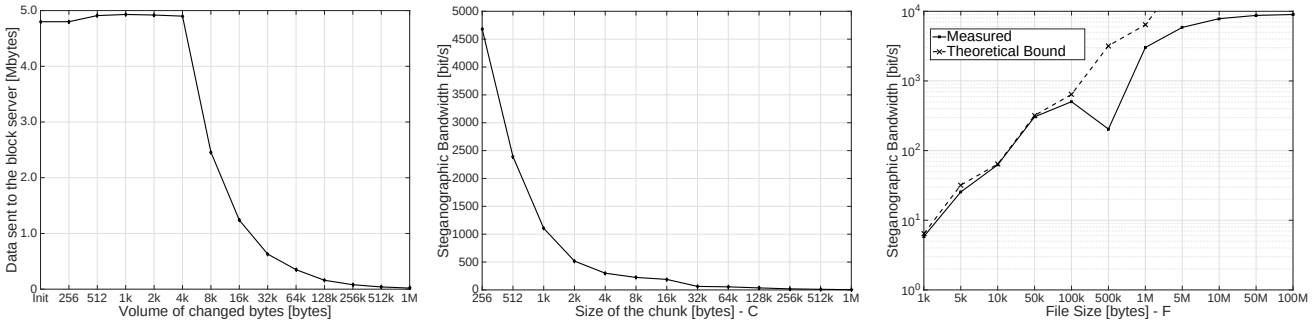
To have proper statistical relevance, each experiment has been repeated 10 times, and aimed at exchanging secrets of different lengths, which have been randomly generated to avoid correlations or biased distributions of the encoded information. To produce numerical results, we used `tshark` and ad-hoc Python and bash scripts.

As regards the traffic investigation for assessing the stealthiness of the proposed approaches, measurements have been conducted on the metropolitan area network of the National Research Council of Italy (CNR) in Bari, which daily serves more than $1,500$ unique users. Traffic has been captured for about one month and in two different periods of the 2015, i.e., from Feb. 11^{th} to 23^{rd}, and from March 8^{th} to 20^{th}. Data

TABLE I
DELAYS FOR THE ALT METHOD WITH DIFFERENT SIZES OF THE FILE AND DELAYS FOR THE REN METHOD WITH DIFFERENT FILENAME LENGTHS.

ALT											
	F [bytes]										
	1K	5K	10K	50K	100K	500K	1M	5M	10M	50M	100M
t_d [s]	5.48	6.29	5.12	5.24	6.35	7.85	10.83	27.89	41.78	187.67	364.21

REN					
	L [N. of characters]				
	1	64	128	192	255
t_d [s]	3.3	3.2	3.6	3.5	2.7
Bandwidth [bit/s]	2	162	286	442	753



(a) Impact of delta encoding on uploaded data. (b) Impact of the chunk size on the bandwidth. (c) Impact of the file size on the bandwidth.

Fig. 4. Behavior of the delta encoding used to upload data and analysis of the steganographic bandwidth when varying C and F in the REN method.

has been collected and processed on a Linux Workstation with two 2.40 GHz Intel Xeon E5-2609 CPUs. The machine has been directly connected via a gigabit link mirroring the traffic managed by the router towards the GARR network (i.e., the ISP of Italian Universities and Public Research Centers). The resulting dataset had a size of $1,400$ Gbyte and contained more than $1,742,279$ conversations seen between $289,392$ unique IP Source - IP Destination tuples. Collected traces have been anonymized by using `tcprewrite`. The traffic has been processed by using `tshark`, `tcpdump` and bash scripts.

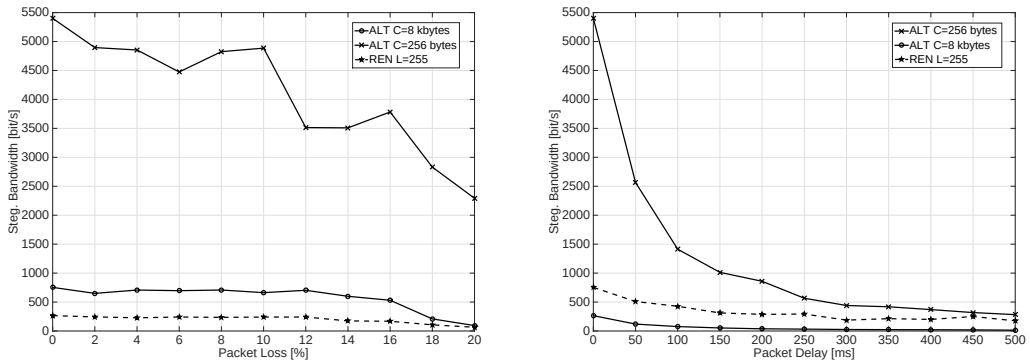
A. Steganographic Bandwidth

This round of tests evaluates the steganographic bandwidth of the covert channel, i.e., the rate at which the secret sender and the secret receiver exchange information. With t_d, we denote the delay in seconds between a request for synchronization from the client interface and the time when the Dropbox cloud completes the process and notifies users.

First, we investigate the performances achieved by the REN method. As discussed, the secret data is encoded within the filename, therefore longer names offer more capacious carriers. We considered different sizes of the carrier, i.e., $L = 1, 64, 128, 192,$ and 255 characters. We also assumed to have an UTF-8 encoding, hence each symbol can contain 8 bit of secret information. Table I presents the collected results. As shown, t_d is insensitive to the amount of secret data stored, i.e., L. This confirms that delays are only ruled by the network and the burden experienced by the Dropbox cloud and the client interface. The overhead in terms of metadata to be sent does not vary with L and, on the average, the parties synchronize by receiving the required notifications in $t_d \simeq 3$ seconds. The steganographic bandwidth, increases with L as expected and ranges from 2 up to 753 bit/s. However, delays impose an upper bound B given by $B = \frac{8 \cdot L}{t_d}$ bit/s.

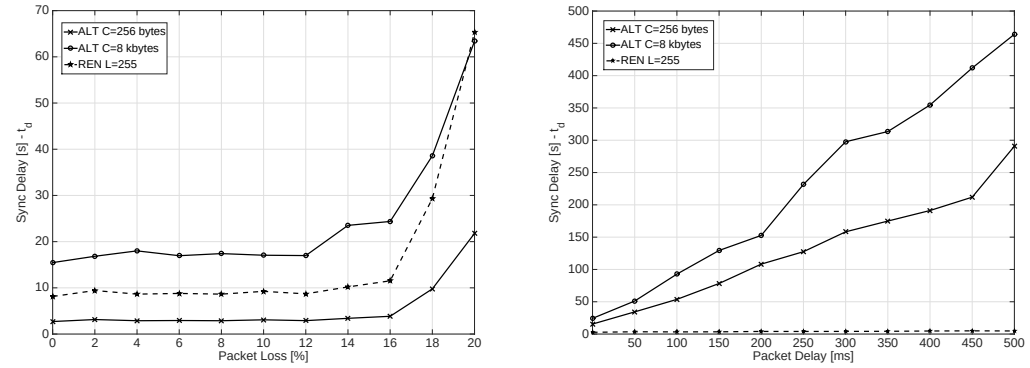
Second, we considered the ALT method. Compared to REN, it has a more complex interaction with the Dropbox cloud, thus we preliminary evaluated some behaviors to understand impacts over t_d and to elaborate on the undetectability of the covert channel. For this trial, a file having the size $F = 4$ Mbytes has been chosen. Figure 4(a) depicts the relation between the amount of changed data and the volume sent to block servers in the Amazon cloud. With 'init' we refer to the upload of a new file, i.e., block servers need to receive the content for the very first time. In this case, the amount of data to be transferred corresponds to the whole file, plus an overhead for the metadata, TLS encryption and handshaking. Surprisingly, the delta encoding is not always used, i.e., for changes needing to update less than 8 kbytes, the whole file is uploaded. Therefore, Dropbox applies optimizations by using an internal threshold, supposedly to perform a trade-off among CPU/disk utilizations and bandwidth savings.

The size of the carrier plays an important role also for the ALT method. Specifically, larger files have more data to be changed for embedding secrets. Since each modification in the local file requires to update the cloud, the t_d limits again the rates at which the secret sender can encode information. Table I reports the collected values with $C = 256$ bytes. As shown, t_d grows with F, but it remains bounded until



(a) Steganographic bandwidth with different packet losses.

(b) Steganographic bandwidth with different packet delays.



(c) Synchronization delay t_d with different packet losses.

(d) Synchronization delay t_d with different packet delays.

Fig. 5. Behaviors of the steganographic bandwidth and synchronization delay t_d for the REN and ALT methods with different impairements.

$F = 1$ Mbyte. Therefore, F is a critical parameter, as for undetectability reasons, it could be convenient having multiple files but with smaller sizes, rather than a unique big one. Recalling the mechanism used to inject the information, both F and C affect the achievable bandwidth. Figure 4(b) and 4(c) portrait their impact on the capacity of the covert channel. The best performances are obtained when the $C = 256$ bytes, as it allows to embed a greater amount of data and guarantees a covert channel with a bandwidth of about 4.5 kbit/s. Higher values of F account for more room in the carrier to inject secrets, but at the price of an increasing volume of traffic generated through the Internet. As regards the steganographic bandwidth, the theoretical bound is given by $B = \frac{F \cdot 8}{C \cdot t_d}$ bit/s, and the optimal upper bound is when $C = 256$ bytes. As depicted in 4(c), best performances are achieved for F greater than 5 Mbytes. Even if the capacity of the covert channel constantly grows with the unique exception of $F = 512$ kbytes, t_d prevents to reach the theoretical limit.

B. Robustness

To investigate how network impairments affect the performance of covert channels, we modeled hazards via packet losses and delays, e.g., to consider congestion in the Internet or intermittent connectivity of mobile devices. For both methods, the packet loss varied in the $0 - 20\%$ range, while delays varied in the $0 - 500$ ms range, as to consider different types of wireless mobility, i.e., from cellular to satellite [37]. For the REN method, we set $L = 255$ as it provides the best performances in terms of steganographic bandwidth. For the ALT method, we choose $F = 4$ Mbytes as it represents a good trade-off among the throughput of secret information and undetectability. In this case, we performed two different investigations to understand the roles of C and delta encoding. To this aim, we undertook experiments with chunks having the size of $C = 256$ bytes and $C = 8$ kbytes. Figure 5 depicts the collected results averaged over all the trials.

In more details, Figure 5(a) shows the impact of the packet loss on the steganographic bandwidth. In the case of REN, the covert channel compensates to losses smaller than 14%, whereas for more severe values its throughput reduces. A similar behavior is also experienced by the ALT method. Figure 5(b) depicts how packet delay influences the steganographic bandwidth. Similar for the packet loss, the delays reduce the throughput achieved by both methods. To understand such behavior, Figures 5(c) and 5(d) portrait the impact of packet losses and delays over the synchronization delay t_d.

For the case of REN the higher Round Trip Time (RTT) accounts for a performance degradation of the TCP/HTTP, hence limiting the amount of information that can be exchanged between the client interface and the Dropbox cloud. This is mainly due to the protocol hierarchy in charge of delivery the signaling and metadata. In fact, TCP/HTTP can absorb burst of errors, but they will stop properly working when in the presence of heavily impaired channels (e.g.,

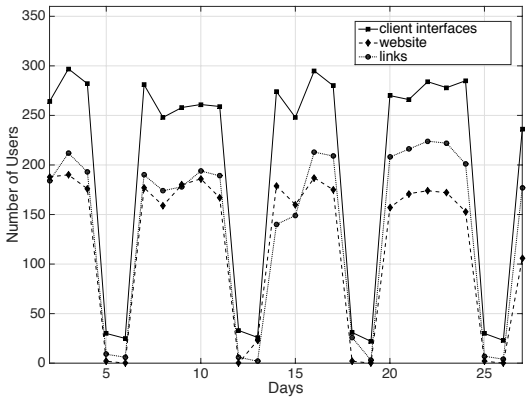


Fig. 6. Number of users accessing Dropbox subdivided for type of service.

due to retransmissions and timeouts). In this case, the client interface stops exchanging data with the Dropbox cloud, thus interrupting the flow of information between the secret receiver and the secret sender. Again, the high RTT reduces the number of operations that can be completed in a given timeframe by the ALT method. Similar to the error-free scenario, the covert channel achieves the best steganographic bandwidth for $C = 256$ bytes. Owing to the behavior of delta encoding (see Figure 4(a)), when $C = 8$ kbytes, less data is exchanged with Dropbox, including blocks sent to Amazon and information carried by the synchronization protocol. Oddly, this reduces performances as larger chunks encoding the same information require to alter different blocks. Instead, when $C = 256$ bytes the increased amount of traffic allows the TCP to better utilize the available bandwidth. This sort of "parallelization" compensates delays and saturates the bandwidth from the client interface towards Dropbox, leading to a higher flow of information between the secret sender and the secret receiver.

C. Undetectability

Undetectability is fundamental for characterizing a covert channel. To this aim, we discuss some behaviors of Dropbox to: *i*) tune the information hiding methods as to increase their stealthiness, and *ii*) enlighten the most critical features to develop an effective Warden. We recall that our scope is to investigate information hiding in personal cloud storage services (see, e.g., [1], [19] for comprehensive traffic characterizations).

As a first step, we evaluate if implementing a covert channel within a client interface can be considered as an anomaly. To identify how an endpoint accesses Dropbox, the ideal method would require to "force" the TLS encryption and directly inspect the `host_int` and `host_id` fields used in the notification protocol. This is not a viable solution, thus we considered the endpoints of TCP conversations, as each Dropbox service has a specific range of IP addresses. As regards identification of hosts, the DHCP is only used for few guest devices, hence the noise introduced by dynamic addressing is very limited. Figure 6 depicts the number of hosts accessing the different services delivered by Dropbox. As shown, the majority uses the storage service via the client interface, but also the web-based platform has a relevant

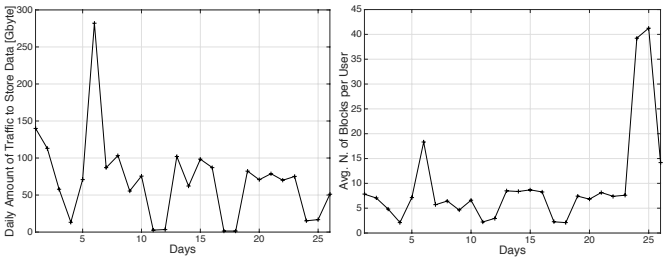


Fig. 7. Average daily behaviors of Dropbox users.

utilization. Since we expected a higher gap, we searched for links, i.e., the creation through the Dropbox website of URIs to publicly share files. We discovered that the high utilization of the website has to be primarily ascribed to the creation of a link, rather than for manually syncing contents via browser. Therefore, a covert channel carried by the flow of information of a client interface would not appear as an anomaly.

Concerning the distribution of users in the observed timeframe, drops represent the weekends. In this case, spotting the hidden flow of information via network analyzers could be simpler, as the bulk of data reduces. Consequently, one might imagine to use low-attention raising mechanisms, for instance by spawning the covert channel when the network traffic is assumed more consistent [38], [39]. A similar technique could be also adopted to hide from a Warden on the file system.

Recalling that REN only relies on the alteration of an entry in the metadata and does not require to update contents in the cloud, its network footprint is hard to be spotted by a Warden. In fact, on the average, it is required to inspect daily up to 60 Gbytes of data, with peaks of more than 250 Gbytes, as depicted in Figure 7(a). Instead, a Warden on the host could reveal the data hiding process, as too many changes on the file system could be suspicious. Recalling the values of t_d measured and reported in Table I, without additional impairments imposed by the Internet, an average synchronization requires about 3 seconds. Therefore, the software implementing the covert channel should restore the original name within t_d as to avoid that a Warden on the file system spots the change. Unfortunately, this increases the burden on the mass storage, e.g., due to continuous parse of the file system log, thus a proper trade-off is needed.

As regard the undetectability of ALT, we recall that the amount of Dropbox traffic depends on the actions performed by the user. Therefore, we measured both the volumes of data and the number of blocks stored by each user daily, as reported in Figure 7. On the average, a single user stores 9 new blocks per day, which is equal to ~ 36 Mbytes of data (recalling that a standard block is 4 Mbytes). Further investigations reveal that such a value is characterized by a very high variance, making hard to fix an effective threshold to decide whether a volume of blocks represents an anomaly. It appears clear that users prefer to do frequent and small changes to a file instead of producing brand new contents. This can only partially rule the throughput of data, as the optimized block transfer architecture of Dropbox plays a major role.

Surely, steganographic bandwidth can be traded for de-

tectability, since larger values of F and smaller sizes C imply more traffic observable over the network, and then, a more aggressive user activity that can be spotted. However, the more important trade-off is between the volume of secret information sent daily and undetectability. In other words, limiting the amount of secret data M to 100 bits would make the resulting flow of chunks between the secret sender and the secret receiver as indistinguishable from licit traffic. Obviously, more bandwidth or a greater amount of information can be exchanged by using multiple files, but this would inflate the traffic per user, and hence make the channel easier to spot.

D. Comparison, Implementation Issues and Summary

As shown, ALT allows to trade network stealthiness in terms of throughput by varying C, but this will have more impact on the file system. Conversely, the amount of traffic produced by REN is insensitive from L and the methods used to encode the secret message M. In our trials, we successfully used REN with character occurrence, alteration of the number of spaces, character existence, word-to-symbol map, direct UTF-8 encoding, and the number of character used (see, e.g., [40] for a thorough discussion on text steganography). We point out that such methods are general and can be mixed to mimic the normal behavior of a user during his/her daily working routine. Nevertheless, REN is robust against errors and losses but, its bandwidth should be limited to avoid the detection by observing the hosting file system.

As regards perspective implementation issues, the ALT method is the more challenging, as it can lead to a temporary corruption of the file used as the carrier. A simple workaround to minimize the risk of opening such a file is to use a low-attention raising technique, for instance enabling the covert channel when the user is idle (see, e.g., [38] for an information-hiding-capable malware using such approach on Android devices). Another approach relies on using contents that can not be corrupted *tout court*, such as multimedia ones. As an example, chunks used to embed secrets could be only limited to metadata or ID-tags of MPEG files. A more complex, yet preferred, way is to add to REN and ALT another layer of steganography. For instance, if the ALT is applied to an image, one might further encode the bytes of M within a pixel and considering also features of the neighborhood area, as to avoid the detection due to artifacts or corruptions [42]. Nevertheless, encrypting M could make the covert channel more robust, as it prevents to reconstruct the secret by recognizing the bytes changed in the file.

The computational overheads of both methods are quite modest. Indeed, ALT and REN require a proper design to not have too much impact on the hosting device, e.g., to avoid detection due to performance degradations or excessive battery drains. A conflicting design goal concerns the secret sender, which should be implemented to be as fast as possible. Unfortunately, this would require to have a spin-lock style behavior leading to a high-detectable process. Therefore, overheads could be reduced by considering the t_d, which prevents to reach theoretical bounds. In this case, the sender/receiver could be put in a sort of sleep state during "idle" periods due to delays. Concerning computational/storage overheads, the REN only requires to perform simple operations on the file-system and computing an hash, thus they can be considered as negligible. A similar consideration could be made for the ALT, even if it produces slightly increased costs in terms of duplication and restoration of files.

To summarize, results clearly indicate that the proposed approaches can covertly transfer data through the Internet and can be used to implement a communication layer allowing malware to hide its existence for long periods. If the amount of daily exfiltrated data is small, their detection could be very hard, thus they can be used for "long and slow" leakages [11].

VI. Wardens and Mitigation Techniques

One of the primary goals of investigating covert channels targeting Dropbox is the development of specific and effective Wardens or mitigation techniques to prevent the misuse of personal cloud storage services for information hiding attacks. Preliminary, we mention that the development of general countermeasures for such threats is still an open problem, as each method has its own and poorly generalizable characteristics [41]. This also reflects in many works proposing method-agnostic Wardens addressing the network traffic of embedding applications, rather than concentrating on the internals of the steganographic exploit [12], [30], [31], [32], [33].

The REN and ALT methods offer a wide variety of possible features to be used, as they range from the alteration of metadata producing a negligible amount of traffic to file changes leading to a more aggressive network behavior. Considering a Warden deployed on the host, it has to pursue two concurrent goals: decide whether a change in the file indicates a secret injection and detect the threat with a reasonable footprint in terms of computational and I/O burden. In general, spotting REN-like attacks within the host of the victim could be feasible, while for the case of ALT the need for a continuous parse of the file system log and check for differences among multiple versions of the file would make preferable shifting the detection in the network. We point out that only few works in the literature try to directly spot a covert channel in the device, with the exception of the colluding applications attack or digital media steganography only targeting the content and not a wider architecture (see, e.g., [12] and references therein).

Therefore, developing a Warden acting on the network could be a more efficient solution. As discussed, all the exchanges between the secret sender and the secret receiver are embedded within flows of the metadata protocol. Since Dropbox uses TLS-encrypted end-to-end conversations to transfer notifications, the use of DPI techniques is ineffective. As presented in Section V-C, if the Warden has a "good" knowledge of the block/user usage, this can be exploited to reveal the ALT covert channel or to force the secret sender to slow as to avoid to be detected as an anomaly. To this aim, an effective Warden could take advantage of a specific behavior of the Dropbox protocol. Especially, trials revealed that the number of uploaded blocks is equal to the TLS Application Data packets (i.e., `Type 23`, `0x17`) sent from the Dropbox cloud to the client interface with the push flag of the TCP set to true (i.e., `PSH = 1`).

If not completely void, the bandwidth of channels targeting personal cloud storage services can be mitigated via traffic normalization. In fact, as presented in Section V-B, both methods are highly sensitive to delays. Therefore, a possible countermeasure introduces additional delays on the related traffic flows as to limit the achieved steganographic bandwidth. However, this is a delicate trade-off since it can degrade the QoE perceived by the entire user population. Besides, as discussed in Section IV-A, a malware wanting to attack Dropbox can use the RSS flow to implement a secret-receiver-side daemon. A simple approach could exploit the volume of traffic generated by Web syndication. Since it can be assumed as smaller if compared to the bulk of data generated by the control and data planes of Dropbox, a Warden can use syndication patterns as indicators to spot anomalies. Lastly, an effective Warden should also consider the presence of low-attention raising mechanisms.

Developing new mitigation techniques for threats using information hiding is part of our ongoing research. Specifically, for the case of personal cloud storage applications, the most promising approaches to be investigated are:

- energy-based: a recent field of research proposes the use of energy consumption as a more general indicator to spot anomalies [43]. This appears as particularly suited for covert channels, since their detection is tightly coupled with the specific information hiding method and poorly generalizable. Therefore, a Warden in the host can be used to reveal the malicious exploitation of personal cloud storage services by monitoring its consumption [44].
- by design: owing to the popularity of personal cloud storage applications, vendors like Dropbox, Google, Microsoft and Apple should consider to address information hiding hazards from very early implementation stages, as well as to deploy cloud-side countermeasures. As an example, metadata servers could analyze the usage pattern of an user and, if it deviates from a reference value, the steganographic bandwidth can be reduced by inflating t_d. This prevents to deploy a Warden on the device consuming CPU and battery resources [35].

VII. Conclusions

In this paper, we investigated covert channels targeting personal cloud storage services. We implemented two methods and tested their performances in terms of bandwidth and robustness in realistic scenarios. Results indicate that the proposed covert channels can be used to stealthy transfer information through the Internet. Specifically, personal cloud storage can be used as a "communication service" for the next-generation malware wanting to cloak its existence or covertly exfiltrate data from the hosts of attacked users. For this reason, the variety of data, together with the huge amount of traffic due to the vast user population, rise security issues that can lead to a Big Data problem. Therefore, understanding menaces arising from information hiding could prevent the need of performing resource-consuming traffic analysis or developing non-scalable countermeasures.

Future research aims at implementing and testing a novel set of countermeasures. Nevertheless, part of the ongoing research is devoted to extend the investigation of attacks based on information hiding to other cloud services.

References

[1] I. Drago, M. Mellia, M. M. Munafo, A. Sperotto, R. Sadre, A. Pras, "Inside Dropbox: Understanding Personal Cloud Storage Services", in Proc. of the Internet Measurement Conf., ACM, New York, NY, USA, pp. 481 - 494, 2012.

[2] S. Subashini, V. Kavitha, "A Survey on Security Issues in Service Delivery Models of Cloud Computing", Journal of Network and Computer Applications, Vol. 34, No. 1, pp. 1 - 11, Jan. 2011.

[3] K. Ren, W. Cong, W. Qian, "Security Challenges for the Public Cloud", IEEE Internet Computing, no.1, vol. 16, pp. 69-73, Jan./Feb. 2012.

[4] Z. Xiao, Y. Xiao, "Security and Privacy in Cloud Computing", IEEE Communications Surveys & Tutorials, vol. 15, no. 2, pp. 843 - 859, Second Quarter 2013.

[5] I. A. T. Hashem, I. Yaqoob, N. Badrul Anuar, S. Mokhtar, A. Gani, S. U. Khan, "The Rise of Big Data on Cloud Computing: Review and Open Research Issues", Information Systems, vol. 47, pp. 98 - 115, Jan. 2015.

[6] D. Chen, H. Zhao, "Data Security and Privacy Protection Issues in Cloud Computing", in Proc of the Int. Conf. on Computer Science and Electronics Engineering, Hangzhou, China, pp. 647-651, 2012.

[7] D. Zissis, D. Lekkas, "Addressing Cloud Computing Security Issues", Future Generation Computer Systems, Vol. 28, No. 3, pp. 583-592, March 2012.

[8] R. A. Popa, J. R. Lorch, D. Molnar, H. J. Wang, L. Zhuang, "Enabling Security in Cloud Storage SLAs with CloudProof", In Proc. of the USENIX Annual Technical Conf., vol. 242, June 2011.

[9] H. Chung, J. Park, S. Lee, C. Kang, "Digital Forensic Investigation of Cloud Storage Services", Digital Investigation, Vol. 9, No. 2, pp. 81 - 95, Nov. 2012.

[10] C. Wang, K. Ren, W. Lou, J. Li, "Toward Publicly Auditable Secure Cloud Data Storage Services", IEEE Network, vol. 24, no. 4, pp. 19-24, 2010.

[11] W. Mazurczyk, L. Caviglione, "Information Hiding as a Challenge for Malware Detection", IEEE Security & Privacy Magazine, Issue 2, March/April 2015, pp. 89-93.

[12] W. Mazurczyk, L. Caviglione, "Steganography in Modern Smartphones and Mitigation Techniques", IEEE Communications Surveys & Tutorials, vol. 17, no. 1, pp. 334 - 357, First Quarter 2015.

[13] E. Zielinska, W. Mazurczyk, K. Szczypiorski, "Trends in Steganography", Comms. of the ACM, vol. 57, no. 3, pp. 86 - 95, Mar. 2014.

[14] S. Katzenbeisser, F. Peticolas, "Information Hiding", Artech House Publishers, Norwood, United States, 2nd Edition, Dec. 2015.

[15] D. Harnik, B. Pinkas, A. Shulman-Peleg, "Side Channels in Cloud Services: Deduplication in Cloud Storage", IEEE Security & Privacy, vol. 8, no. 6, pp. 40 - 47, Nov. - Dec. 2010.

[16] S. Halevi, D. Harnik, B. Pinkas, A. Shulman-Peleg, "Proofs of Ownership in Remote Storage Systems", in Proc. of the 18th ACM Conf. on Computer and Communications Security, pp. 491 - 500, 2011.

[17] T. Ristenpart, E. Tromer, H. Shacham, S. Savage, "Hey, you, get off of my cloud: Exploring Information Leakage in third-party Compute Clouds", in Proc. of the 16th ACM Conf. in Computing and Communication Security, Chicago, IL, USA, 2009, pp. 199 - 212.

[18] W. Hu, Y. Tao, J. N. Matthews, "The Good, the bad and the ugly of Consumer Cloud Storage", ACM SIGOPS Operating Systems Review, vol. 44, no. 3, pp. 110 - 115, 2010.

[19] I. Drago, E. Bocchi, M. Mellia, H. Slatman, A. Pras, "Benchmarking Personal Cloud Storage", in Proc. of the 2013 Conf. on Internet Measurement, pp. 205 - 212, Oct. 2013, ACM.

[20] M. Mulazzani, S. Schrittwieser, M. Leithner, M. Huber, E. Weippl, "Dark Clouds on the Horizon: Using Cloud Storage as Attack Vector and Online Slack Space", in Proc. of the 20th USENIX Conf. on Security, 2011.

[21] Z. Li, C. Wilson, Z. Jiang, Y. Liu, B. Y. Zhao, C. Jin, Y. Dai, "Efficient Batched Synchronization in Dropbox-like Cloud Storage Services", in D. Eyers, S. Karsten (Eds.), Middleware 2013: ACM/IFIP/USENIX 14th Int. Middleware Conf. , pp. 307-327, 2013, Springer Berlin Heidelberg.

[22] Z. Li, C. Jin, T. Xu, C. Wilson, Y. Liu, L. Cheng, Y. Liu, Y. Dai, Z.-L. Zhang, "Towards Network-level Efficiency for Cloud Storage Services", in Proc. of the 2014 Internet Measurement Conf., pp. 115 - 128, 2014.

[23] P. Shlane, M. Huang, G. Wallace, W. Hsu, "WAN-Optimized Replication of Backup Datasets Using Stream-Informed Delta Compression", ACM Transactions on Storage, vol. 8, no. 4, pp. 1 - 13, 2012.

[24] The librsync homepage, http://librsync.sourceforge.net, last accessed: March 2016.

[25] F. Petitcolas, R. Anderson, M. Kuhn, "Information Hiding: A survey", Proc. of the IEEE, vol. 87, no. 7, pp. 1062 - 1078, Jul. 1999.

[26] W. Mazurczyk, S. Wendzel, S. Zander, A. Houmansadr, K. Szczypiorski, "Information Hiding in Communication Networks: Fundamentals, Mechanisms, Applications, and Countermeasures", Wiley-IEEE Press, April 2016.

[27] G. J. Simmons, "The Prisoner's Problem and the Subliminal Channel", in Proc. of CRYPTO, 1983, pp. 51 - 67.

[28] S. Wendzel, W. Mazurczyk, L. Caviglione, M. Meier, "Hidden and Uncontrolled - on the Emergence of Network Steganographic Threats", in Proc. of ISSE 2014 Securing Electronic Business Processes, pp. 123 - 133, Springer, Fachmedien Wiesbaden.

[29] J. Fridrich, "Applications of Data Hiding in Digital Images", in Proc. of 5th Int. Symp. on Signal Processing and Applications, pp. 1 - 9, 1999.

[30] M. H. Kang, I. S. Moskowitz, "A Pump for Rapid, Reliable, Secure Communication", in Proc. of the 1st ACM Conf. on Computer and Communication Security, Fairfax, USA, pp. 119 - 129, 1993.

[31] S. Cabuk, C. E. Brodley, C. Shields, "IP Covert Channel Detection", ACM Transactions on Information and System Security, No. 12, Vol. 4, pp. 22 - 29, April 2009.

[32] V. Berk, A. Giani, G. Cybenko, "Detection of Covert Channel Encoding in Network Packet Delays", Technical Report, Dartmouth College, 200

[33] S. Gianvecchio, H. Wang, "Detecting Covert Timing Channels: An Entropy-Based Approach", in Proc. of 14th ACM Conf. on Computer and Communication Security, pp. 307- 316, Alexandria, USA, 2007.

[34] D. E. Taylor, "Survey and Taxonomy of Packet Classification Techniques", ACM Computing Surveys, vol. 37, no. 3, pp. 238-275, 2005.

[35] L. Caviglione, W. Mazurczyk, "Understanding Information Hiding in iOS", IEEE Computer, vol. 48, no. 1, pp. 62 - 65, Jan. 2015.

[36] D. Kholia, P. Wegrzyn, "Looking Inside the (Drop) Box", in Proc. of the 7th USENIX Conf. on Offensive Technologies, pp. 1 - 9, Berkeley, CA, USA, 2013.

[37] L. Caviglione, "Can Satellites face Trends? The case of Web 2.0", Int. Workshop on Satellite and Space Communications, pp. 446 - 450, 2009.

[38] J.-F. Lalande, S. Wendzel, "Hiding Privacy Leaks in Android Applications Using low-Attention Raising Covert Channels", in Proc. of the Int. Conf. on Availability, Reliability and Security, pp. 701-710, 2013.

[39] R. Andriatsimandefitra, V. V. T. Tong, "Detection and Identification of Android Malware Based on Information Flow Monitoring", in Proc. of the 2nd Int. Conf. on Cyber Security in Cloud Computing, pp. 1 - 4, 2015.

[40] K. Bennett, "Linguistic Steganography: Survey, Analysis, Robustness Concerns for Hiding Information in Text", Purdue University, CERIAS Tech. Rep. 13, May 2004.

[41] J. J. Harmsen, W. A. Pearlman,"Capacity of Steganographic Channels", in Proc. of the 7th Workshop Multimedia Security, New York, NY, USA, pp. 11 - 24, Aug. 2005.

[42] W.-J. Chen, C.-C. Chang, T. H. N. Le, "High Payload Steganography Mechanism Using Hybrid Edge Detector", Expert System Applications, vol. 37, no. 4, pp. 3292 - 3301, Apr. 2010.

[43] A. Merlo, M. Migliardi, L. Caviglione, "A Survey on Energy-Aware Security Mechanisms", Pervasive and Mobile Computing, vol. 24, pp. 77 - 90, Dec. 2015.

[44] L. Caviglione, M. Gaggero, J. F. Lalande, W. Mazurczyk, M. Urbański, "Seeing the Unseen: Revealing Mobile Malware Hidden Communications via Energy Consumption and Artificial Intelligence", IEEE Transactions on Information Forensics and Security, vol. 11, no. 4, pp. 799 - 810, April 2016.

Luca Caviglione received the Ph.D. degree in electronic and computer engineering from the University of Genoa, Italy. He is a Researcher at the Institute of Intelligent Systems for Automation of the National Research Council of Italy. His research interests include p2p systems, wireless communications, cloud architectures, and network security. He is author or co-author of more than 100 academic publications, and several patents. He has been involved in research projects funded by ESA, EU and MIUR. He is a Work Group Leader of the Italian IPv6 Task Force, a Contract Professor in the field of p2p networking, and a Professional Engineer. He is involved in the technical program committee of many international conferences, and regularly serves as a reviewer for the major international journals. From 2011, he is an Associate Editor for the Transactions on Emerging Telecommunications Technologies, Wiley.

Maciej Podolski holds B.Sc. (2003) in Electronic and Computer Engineering with major in Telecommunications from Warsaw University of Technology (WUT) in Poland. He has been passionate about cyber security since early university years. Currently he works in security field. He is a member of Cisco TAC Security team, where he helps securing customers networks every day and resolves complex platform issues..

Wojciech Mazurczyk (SM'13) received the B.Sc., M.Sc., Ph.D. (Hons.), and D.Sc. (Habilitation) degrees in telecommunications from the Warsaw University of Technology (WUT), Warsaw, Poland, in 2003, 2004, 2009, and 2014, respectively. He is currently an Associate Professor with the Institute of Telecommunications, WUT, where he is the Head of the Bio-Inspired Security Research Group (bsrg.tele.pw.edu.pl). His research interests include bioinspired cybersecurity and networking, information hiding, and network security. He is involved in the technical program committee of many international conferences, including the IEEE INFOCOM, the IEEE GLOBECOM, the IEEE ICC, and ACSAC. He also serves as a reviewer for major international magazines and journals. Since 2013, he has been an Associate Technical Editor of the IEEE Communications Magazine (IEEE Comsoc).

Massimo Ianigro received the M.Sc. degree in Computer Science in 1992, from the University of Bari. Before joining the National Research Council of Italy, he has been awarded of research contracts from SGS Thomson Microelectronics and Digital Equipment Corp. on various themes (for instance, optical flow reconstruction, robotics, high performance computing and networking). During 1995 he has worked at Manchester University in the Computer Graphics Unit. He has been teaching in several master and PhD courses and he is responsible for the networking infrastructure and telematic services of the CNR Bari area. Also, he is a member of the GARR-CERT (Computer Emergency Response Team of the nation-wide network of academic institutions) and he is an expert appointed by Italian law enforcement agencies in many computer forensic cases. His current research interests include information and communication technologies, robotics, computer and network security, large scale infrastructures, computer forensics.