Computers and Mathematics with Applications 63 (2012) 437–447

Contents lists available at SciVerse ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

An asynchronous covert channel using spam

Aniello Castiglione [a,*], Alfredo De Santis [a], Ugo Fiore [b], Francesco Palmieri [c]

[a] Dipartimento di Informatica "R.M. Capocelli", Università degli Studi di Salerno, Via Ponte don Melillo, I-84084 Fisciano (Salerno), Italy

[b] C.S.I., Università degli Studi di Napoli, I-80126 Napoli, Italy

[c] Dipartimento di Ingegneria dell'Informazione, Seconda Università degli Studi di Napoli, I-81031 Aversa (Caserta), Italy

ARTICLE INFO

Keywords:
Stegosystem
Raptor codes
E-mail steganography
Spam steganography
Asynchronous covert channel
Secure communication

ABSTRACT

Current Internet e-mail facilities are built onto the foundation of standard rules and protocols, which usually allow a considerable amount of "freedom" to their designers. Each of these standards has been defined based on a number of vendor specific implementations, in order to provide common inter-working procedures for cross-vendor communication. Thus, a lot of optional and redundant information is being exchanged during e-mail sessions, which is available to implement versatile covert channel mechanisms.

This work exploits this possibility by presenting a simple but effective steganographic scheme that can be used to deploy robust secret communication through spam e-mails. This scheme can offer unidirectional asynchronous one-to-one or one-to-many covert channel facilities that are able to bypass the most sophisticated firewalls and traffic analyzers. Its implementation neither affects the involved transport protocols nor causes any perceivable performance degradation or data loss to the end-users. The proposed scheme allows one to manage possible filtering/loss of the e-mails being the vehicle of the secret information. A novel retransmission method based on the Raptor codes has been adopted. The use of Raptor codes is key to correctly and efficiently manage the difficulty or impossibility to retransmit e-mails in the case of a unidirectional secret communication starting from one sender and directed to many recipients. In order to evaluate the performance characteristics of the proposed scheme, an empirical estimation of the covert channel bandwidth has been performed.

1. Introduction

Several powerful and sophisticated security systems have been introduced to monitor communications over the network, inspect their content and react accordingly when inappropriate, dangerous, or simply suspicious content is detected. However, this power in the hands of some individuals within agencies or organizations raises the issue of possible misuse. Communications intended to be private might be illegally intercepted, transcribed or manipulated without consent, putting the right to free speech and expression at risk. As a consequence, covert channels for data communications crossing the borders of some countries or companies are receiving attention from the scientific community. A covert channel is defined [1] as "any communication channel that can be exploited by a process to transfer information in a manner that violates the system's security policy". The covertness of such a channel is related to the data exchange being hidden. This also implies that any third party should have no means to discover the existence of such a communication channel, let alone the content of the messages exchanged. That is, the involved parties apparently interact in full compliance with the security policy of both

* Corresponding author. Tel.: +39 089 969594; fax: +39 089 969600.
E-mail addresses: castiglione@ieee.org, castiglione@acm.org (A. Castiglione), ads@dia.unisa.it (A. De Santis), ugo.fiore@unina.it (U. Fiore), francesco.palmieri@unina.it (F. Palmieri).

0898-1221/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2011.07.068

the end systems as well as the transport infrastructure, while exploiting some hidden communication facilities piggybacking (ideally) undetectable data onto legitimate content. This has led to an emerging discipline, steganography, which is the Greek expression for *covered writing practice*. Steganography is therefore associated with concealing the existence of the message when secret information is hidden in innocently looking covering data.

In this scenario, unsolicited e-mails (commonly known as *spam*) offer an interesting opportunity for implementing effective and flexible asynchronous covert communication channels over the internet. First, spam messages are e-mail messages, and hence, they rely on the protocols that have carried for decades, and still carry, regular e-mail communications on a worldwide scale. The protocols involved in the whole e-mail lifecycle are built on ASCII control strings that offer plenty of room to encode secret data. Second, spam messages are routinely discarded and they do not receive any additional attention besides that, which is strictly necessary to classify them as spam. This suggests the possibility of a robust secret communication that is not likely to be perceivable even by the most sophisticated firewalls and traffic analyzers. The presented steganographic system does not affect the involved transport protocols and mechanisms nor changes the cover e-mail message contents. In addition, it does not cause any noticeable performance degradation or data loss to the end-users. The proposed scheme is able to deal with loss of spam e-mail messages which carry pieces of steganographic information by using a special algorithm that is able to reconstruct "some" missing information from a predetermined amount of received e-mails.

2. Related work

Many covert channel schemes are available in the literature and a significant number of them focus on covert storage channel implementations based on several common network or application layer internet protocols. In general, only a few of them do not use the IP protocol to survive and make use of different carrier such as Microsoft Office documents or PDF file [2,3]. However, the use of e-mail as an asynchronous covert channel has been proposed only indirectly: to cite only an example, [4] focuses on HTTP headers and only hints the possibility of using SMTP headers in the same way, without giving details. The main ideas behind the use of e-mail headers as a vehicle for data hiding appeared in [5]. This work extends the previous one, emphasizing the choice of spam as the preferred covert medium, completing the details of message sequencing, retransmission and framing techniques in the general context of data hiding. In addition, an original contribution of this paper is the introduction of a Forward Error Correction (FEC) code aimed at enhancing the covert channel resilience to loss or corruption of stego-blocks, possibly as a consequence of active countermeasures. To this end, a code belonging to the class of digital fountain codes has been adopted.

3. Motivation and rationale of the proposed solution

3.1. Context

Digital steganography can usually be implemented according to two distinct approaches: *injection* and *substitution*. Both use bit locations or character string markers to locate specific places into the cover "vector" object (sometimes referred to as the *support*) where the hidden message bits are to be inserted or replaced for the existing ones. In general, the wider and deeper the modifications to the original support, the easier it is for an external observer to detect the changes. Data compression techniques are therefore usually applied to the original message to be hidden, to save valuable space in the transporting vector. The overall channel robustness and security are increased through proper encoding (possibly providing error detection/correction features) and symmetric encryption techniques using a pre-shared key, preventing secret data from being tampered with and restricting their detection. Particular care should be devoted to preventing the involved end-processing or transport applications (programs, communication protocols, etc.) from revealing the presence of embedded data within the cover, e.g. raising application errors due to unrecognizable file format, or inadvertently modifying or removing the embedded data, as a result of transcoding or optimizations. A very common practice is to encapsulate information into a legitimate protocol, without altering its behavior and semantics, to easily bypass firewalls and content filters.

Injection steganography works by adding the embedded payload inside the original (unaltered) cover vector, and thus increases its final size. On the other hand, *substitution* steganography replaces what is viewed as an insignificant part of the cover vector, but it should be suitably designed to survive when processed by any "native" application using or transmitting the vector itself. The substituted portion of a cover message could be a "free-format" or unused field into a protocol unit or a segment of executable code. The object resulting from the steganographic process within the cover vector is often referred to as *stego-text*, *stego-block*, or more generically, *stego-object*. Thus, covert channels based on substitution steganography may usually involve syntactic variations of the overt channel attributes that are however semantically acceptable, according to a *semantic overloading* strategy.

3.2. Motivation

Confidentiality and anonymity are usually achieved through cryptographic techniques. However, cryptographic methods are regulated by law, and they are illegal in some countries. In addition, the observation of an encrypted data transfer

can raise suspicion about the reason why the parties have chosen to communicate so privately: what do they have to hide? In other cases, the mere act of communicating with an unauthorized party can trigger a reaction. Data compression techniques are therefore usually applied to the original message to be hidden, to save valuable space in the transporting vector. The overall channel robustness and security are increased through proper encoding (possibly providing error detection/correction features) and symmetric encryption techniques using a pre-shared key, preventing secret data from being tampered with and restricting their detection. Although such delayed control may be useful to people who operate botnets, this limitation will not likely satisfy all their requirements. The nature of the chosen carrier, the spam, also has other consequences. In fact, the one-to-many covert communication pattern turns out to be the most naturally suited for the proposed scheme, with one spam sender and many receivers. In the simplest form of this pattern, communication is unidirectional. This forbids the use of techniques based on acknowledgments or Automated Retransmission Requests [6] to handle data loss and packet reordering (possibly as the result of countermeasures [7]). Such state of things calls for the introduction of an encoding scheme aimed at permitting communication over a lossy channel.

3.3. Rationale

The proposed solution is based on a substitution approach and uses protocol headers to carry the payload. Hidden payload can, in fact, use the protocol headers or be delivered as a portion of the payload. In an e-mail, this would consist in employing text-based steganography on the message body or preparing steganographic images and sending them as attachments. In every e-mail, there are some "accessory" message headers and fields that are not usually shown, but are sent together with the e-mail content. These "accessory" components are often neither used nor affected by all the intermediate agents. Some of them look like random ASCII-encoded strings, whose usage and significance is restricted to end-side e-mail clients, which could be easily and effectively used to embed and hide secret messages. An underlying assumption in covert channels is that the protocol carrying the hidden data must be routinely used on the involved system, to avoid suspicion for the sudden introduction of a previously unseen protocol. This is certainly the case for e-mail. Accordingly, one-to-one or one-to-many unidirectional steganographic communication channels can be implemented by using sequences of e-mail messages flowing between the parties interested in the secret information exchange. Due to the transmission unidirectionality, and since the transmission channel is unreliable, with messages possibly undergoing unanticipated filtering, or unwanted modification as well as being delivered to their destination in the wrong order, some sort of mechanism is necessary in order to carry out any sequencing, integrity check and information loss detection at the receiver side.

In addition, since the e-mails within the cover vector sequence can be intermixed with other totally unrelated ones, some reliable method is needed to distinguish the "interesting" messages (those belonging to the covert channel communication) from the others, and identify the headers containing "useful" information, provably coming from the known originating party.

4. Implementing a covert channel in e-mail headers

The popularity of e-mail implies that most firewalling and access control mechanisms/policies tend to leave protocols associated with e-mail almost unaffected. Accordingly, the e-mail facility, by itself, can be considered an ideal vehicle for covert asynchronous information exchanges. In addition, the use of spam messages as the transport mechanism has the additional advantages that the involved media are considered background noise and thus normally ignored. The sheer volume of spam (more than 89% of all exchanged messages worldwide [8]) also means that the analysis would request substantial resources. In addition, there is some degree of deniability: if accused of participating in a covert communication activity, a party can assert to only have received spam. Deniability is important in some contexts, especially when censorship is involved.

4.1. The cover carrier

(E)SMTP is a text-based protocol, in which a sender communicates with a receiver by issuing command strings and supplying the necessary data over a reliable ordered data stream channel, typically a TCP connection. An (E)SMTP session consists of commands originated by a client (the initiating agent, or transmitter) and the corresponding responses from a server (the listening agent, or receiver) so that the session is opened, and session parameters are exchanged. Such sessions will be the main transport facilities used by the proposed covert channel implementation. The content exchanged in a session is sent in the "DATA" protocol unit and can be divided into two parts: the *headers*, forming a collection of field/value pairs organized as in the specification [9] and the *body*, that, when structured, is defined according to MIME [10]. For a simple text e-mail message, the body consists of a sequence of lines containing ASCII (7 bits) characters and the header fields are a set of name–value pairs corresponding to the various attributes of the e-mail message (e.g. "Sender", "Recipient(s)", "Subject", "Date", "Return-Path", etc.).

The covert message can be conveniently encoded in a portion of the delivering Mail Transfer Agent (MTA) chain described by the headers. Regular messages from the same source to the same destination usually follow the same chain, apart from unusual circumstances. However, fake lines can be added within the message before actually giving it out to the first MTA in the chain. MTAs will not, in general, inspect those headers so the "added" lines in the message will pass on untouched.

4.2. Spam as an ideal information exchange vehicle

As already highlighted in the previous section, in order to avoid any suspicion from a third party possibly observing the communication, these messages should deviate as little as possible from the "historical" distribution of regular messages previously exchanged between the involved parties. Spam is an ideal solution in this respect, since a useful analogy can be drawn between spam and background noise in a communication channel. In the unlikely case that the chosen mailboxes are immune from "regular" spam, a continuous flow of spam e-mail messages can easily be automatically generated by means of spam generation tools such those used in Spam Mimic [11]. The useful information can be embedded into randomly chosen messages within the flow. In addition, spammers tend to conceal the actual message source in order to protect it against intervention by network/system administrators who could block it. It can therefore be expected that the (E)SMTP headers in spam messages carry manipulated information. Thus, evidence of header manipulation alone is not enough to trigger suspicion of covert communication. The proposed scheme does not require that all the messages belonging to a specific covert channel flow are originated from the same source address, since other techniques are used to reliably associate the individual message to a channel. For the sake of brevity, details about these techniques are not given here.

Another advantage of spam is that these messages are usually inspected only by anti-spam systems and are seldom scrutinized by humans. Finally, it must be kept in mind that the majority of bulk senders avail themselves of zombie computers belonging to botnets. These machines run software that allows them to be remotely operated without the legitimate user being aware of it. As long as users are able to prove that their machine was under control of some other individuals, their liability for the actions performed with their machine usually decreases from misconduct to improper custody. A covert channel user wishing to limit the risk of being convicted on the basis of this message exchange may thus arrange fake evidence of infection or might even intentionally download a trojan.

4.3. The embedding facility

In general, many headers available in every e-mail envelope can be used as a vehicle for covert information exchanges. The `Received:` fields present in any e-mail message tell where the message has been originated and which route it took to get to its final destination. It contains a free-format ASCII string (the (E)SMTP `id`) uniquely identifying the involved MTA.

The `Message-ID:` field contains a unique identifier (the "local-part" address unit) which refers to the specific version of a particular e-mail message. In principle, each e-mail message should be assigned to a global-scale unique `Message-ID:`, so that this field can be used to trace and identify it. The uniqueness of the message identifier is only guaranteed by the host generating it. According to [12] this identifier is intended to be machine readable and not necessarily meaningful to humans, with it being reserved only for MTA or client internal use. The "local-part" in a `Message-ID:` can be anything, while the only restriction is that it should be unique (within the domain). Any attribute within the `Message-ID:` cannot be modified during the transmission process. Under the default setting, the e-mail softwares will not display them to users. Normal users have no interest in it except for special use such as mail debugging/backtracking analysis and steganography, as discussed in this work. Different software products will create `Message-ID:`'s in a different style, so that the version of the e-mail client software can in fact be guessed by just looking at the `Message-ID:` field. `Message-ID:` has no size-limit in theory, but in practice, a large `Message-ID:` may trigger suspicion.

For a MIME multi-part message (see [10,13,14]), the `boundary` parameter (for subtypes of "multi-part" media type), associated to a `Content-Type:` header, can also be used for steganography. The length of the `boundary` parameter can be up to 70 characters. It has been intended as the separator of the various sections in a multi-part message. The only restriction proposed in [13] is that "The boundary delimiter must not appear inside any of the encapsulated parts, on a line by itself or as the prefix of any line". For this reason it is very important that the composing agent can choose, and define, a specific boundary value which does not contain, as a prefix, the `boundary` parameter of the including `multipart` field.

Each of the previously presented headers could be used, in principle, to hide secret information in e-mail cover messages, starting, for example, from the `Received:` field. In fact, several fake `Received:` header lines, each one containing a valid (E)SMTP `id` string, can be inserted in an e-mail message, where only the bottommost one is really significant for the receiving side MTA. However, each of these lines should have been introduced by an intermediate relay step, and the presence of long e-mail paths is not too common in current settings, and hence may result suspicious. Furthermore, the generation of each of the above headers, with the associated (E)SMTP "*id*", is totally under control of the intermediate MTAs, so that by performing a cross-check between the "*id*" part in the fake `Received:` entries and the information that should be present into the log files of the involved transit relay servers, it would not be difficult to understand that some `Received:` lines have been forged, leading to the compromise of the entire stegosystem.

Starting from the above considerations, the headers to be used for embedding the secret messages and control information needed to implement the covert channel, are the `boundary` section of the `Content-Type: multipart`/field and the `Message-ID:` field. This choice is motivated by the fact that these headers are the only ones which are managed in an end-to-end fashion by the e-mail clients, and hence, have a minimum chance of being affected by consistency checks. Furthermore, they may be structured to contain data for which there is a certain degree of freedom, provided that all the basic specifications are met.

4.4. Providing end-to-end security, integrity and error tolerance

To provide end-to-end security on the covert channel, the communicating parties must share a secret key, which is necessary to ensure message confidentiality and authenticate covert messages by using stream encryption. In particular, an Advanced Encryption Standard (AES)-Rijndael ciphering scheme [15], operating in Cipher Block Chaining (CBC) mode [16], has been used for individual message encryption. It is an iterated symmetric key block cipher with a variable block length and a variable key length that can be independently specified to 128, 192 and 256 bits. In this encryption scheme, a plaintext message M is divided into b n-bit blocks c_i and the ciphertext e_i can be calculated from the cleartext c_i and vice-versa according to

$$e_i = AES_K(c_i \oplus e_{i-1}), \quad c_i = AES_K^{-1}(e_i) \oplus e_{i-1}, \ i = 1, 2, \ldots, b \tag{1}$$

each block c_i is separately encrypted and the value of the previous ciphertext block e_{i-1} is used to randomize the plaintext by combining (eXclusive-ORing and chaining) with plaintext blocks c_i to hide patterns and repetitions. A reliable integrity verification and replay prevention mechanism is also needed to verify that the "interesting" part of each hidden message has not been tampered with and to ensure that the associated message belongs to the covert channel flow. This can be implemented by using a keyed-Hash Message Authentication Code (HMAC) whose functionality and security are based on a cryptographically secure hash function [17] simultaneously verifying both the data integrity and the authenticity of a message. By using a sufficiently robust hash function and an enough long key, a successful HMAC check provides absolute guarantees to the recipient of the data that it came from the expected sender and has not been altered in transit.

The HMAC scheme works by hashing a shared key with the message itself, and then re-hashing a modified version of the above key with the resulting hash. That is, let K be a b-bytes long key, h a hash function such as MD5 or SHA-1, *opad* and *ipad* two bit patterns, consisting, respectively, of b repetitions of the byte 01011100 and 00110110. Furthermore, the symbol "$\|$" is used to represent concatenation, and $\oplus$ means Exclusive OR (XOR):

$$\mathrm{HMAC}_K(M) = h((K \oplus \mathrm{opad}) \| h((K \oplus \mathrm{ipad}) \| M)). \tag{2}$$

The inner and outer padding patterns (*ipad* and *opad* respectively) are needed to ensure that the inner and outer applications of the hashing function h work as if they were completely different functions. Such applications are specified in a way that greatly simplifies the processing of a relatively long text, by iteratively splitting it in parts. As long as the involved hash function is collision resistant, HMAC is secure. In fact, in this case also a weaker version of collision resistance is needed, so that even though MD5 (a commonly used hash function) is no longer considered completely secure, HMAC based on MD5 is still thought to be secure.

Furthermore, a certain degree of error tolerance is required to ensure that if some messages are lost or unrecoverably corrupted during transmission, a specific amount of received message blocks guarantees the correct retrieval of the original hidden message. To this end, a Raptor code has been chosen. This is an erasure-correcting code [18], i.e. a code intended for distribution of data over unreliable connections where retransmissions are impractical. The Raptor code is a FEC code. FEC codes can allow receivers to reconstruct transmitted objects without feedback from receivers to senders to request retransmission of lost parts. Raptor codes [19] are extensions of LT (*Luby Transform*) codes with linear-time encoding and decoding, and are widely used for the efficient data delivery over broadcast networks. These codes belong to the general category of *fountain codes*.

Fountain codes are record-breaking, sparse-graph codes for binary erasure channels, where messages are transmitted in multiple smaller chunks, each of which is either received without error or not received at all. Based on the analogy with fountains, these codes allow the reconstruction of the transmitted object by assembling digital droplets. The conventional data transmission protocols usually split a message up into k pieces/blocks and then repeatedly transmit each piece until it is successfully received. Consequently, a return channel is required for the transmitter to find out which blocks need to be retransmitted. In contrast, fountain codes generate blocks for transmission that are random functions of the whole message. The transmitter sprays such blocks at the receiver side without any knowledge of which of them are received. Once the receiver has obtained any m pieces – where m is just slightly greater than the original number of message pieces k – the whole message can be recovered. The computational costs of the best fountain codes are astonishingly small, scaling linearly with the message size.

The Raptor code can be viewed as a concatenation of codes, with the innermost code being an LT code, the intermediate a pre-code (usually a Low Density Parity Check or LDPC code [20]), and the outermost a linear code. The purpose of the outermost code is only to achieve a systematic code, i.e., a code that is capable of incorporating whichever source symbol which can be found in the encoding symbols that the scheme can produce. In a systematic code, the original symbols are transmitted unmodified from sender to receiver, with the addition of a number of repair symbols, generated on the basis of the source symbols. Given a frame of k bits of data, the encoding operation can be performed in two different stages. When considering the LDPC pre-coding stage, the k-bit frame is mapped to a new h-bit codeword, where the codeword bits are usually referred to as the intermediate symbols. In the second stage, the h-bit encoded frame is re-encoded by using an LT code operating at a dynamically determined code rate r, where a selected fraction of intermediate symbols are used to generate, through modulo-2 sum operations, $n = \frac{k}{r}$ output symbols. In detail, the i-th output symbol E_i is produced by XORing a number d_i (called the *degree* of E_i and randomly chosen according to a predefined probability distribution) of intermediate symbols, generated in turn by pre-coding. Intermediate symbols are not sent from sender to receiver and their

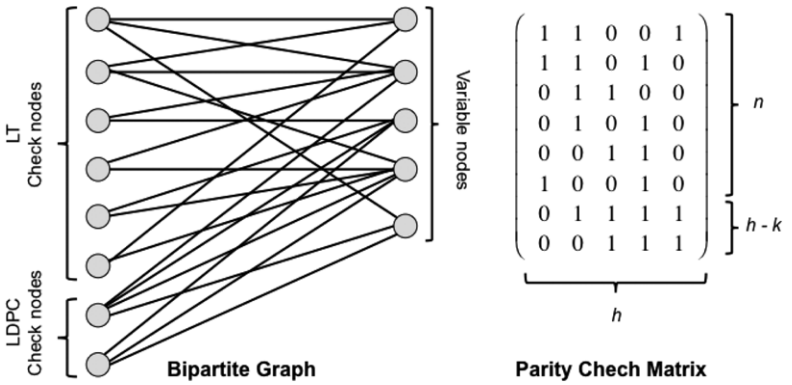


Fig. 1. Representations of a Raptor code.

only purpose is to generate the repair symbols. Certain pre-coding relationships must be satisfied within the intermediate symbols [18]. The d_i symbols are uniformly chosen between the intermediate symbols, giving the set T_i. Thus, the receiver needs to know the set of encoded symbols E_i and a way to determine the corresponding set T_i. This may be either explicitly transmitted or derived by the receiver through the same pseudo-random number generator used by the sender, provided that the two entities share enough information. To this end, an Encoded Symbol Identifier (ESI) is usually sent along with each intermediate symbol and the ESI is fed into a Pseudo-Random Number Generator (PRNG) whose parameters are shared by both parties.

The innermost LT code can be modeled through a bipartite graph [21] with h "variable nodes" on the first partition, representing the intermediate code bits, and n "check nodes" on the other one, representing the output code bits. An edge in the graph exists between a check node i and a variable node j if the bit j enters in input to the XOR operation whose output results to be the check node i. Whether the totality of nodes in either partitions of the graph have the same number of connected edges (i.e. the same degree), the code is said to be regular, otherwise it is irregular. The above graph may be represented by an $n \times h$ matrix H referred to as the *parity check matrix* where each element $H_{ij} = 1$ when there is an edge between the check node i and the variable node j, and 0 differently. The *Hamming weight* of a column or row vector in the matrix H can be determined by the number of entries in that vector with non-zero values. Whether the pre-code is an LDPC code, the Raptor code can be considered as a bipartite graph in which the check node partition is formed by LT and LDPC check nodes as shown in Fig. 1.

Since the Raptor codes are based on LT codes, they inherit the main properties of digital fountain codes, among which the most remarkable are:

- the encoder is able to generate as many encoded symbols as it is necessary for the decoder to recover the source data;
- the decoder can recover the source from any set of received encoded symbols, provided that their number is greater (by a reasonable amount) than the number of source symbols. The decoding algorithm usually works according to a two stage scheme. The innermost LT decoder outputs a hard bit-reliability vector. Such a vector is then processed by the outer decoder (typically an LDPC one), by using a belief propagation algorithm.

Raptor codes, and fountain codes in general, are rateless, in the sense that the number of symbols generated and sent over the channel is not limited *a priori*. When the amount of symbols is instead capped to a chosen value q, a *fixed-rate* Raptor code is obtained. Raptor codes exhibit a performance very close to the ideal one [22], i.e., the code failure probability is so high that, also when only slightly more than k encoding symbols are received, the code can recover the source block. In fact, for $k > 200$ the limited inefficiency of the Raptor code can accurately be modeled as follows [23]:

$$p_f(m, k) = \begin{cases} 1 & \text{if } m < k, \\ 0.85 \times 0.567^{m-k} & \text{if } m \geq k. \end{cases} \tag{3}$$

Here $p_f(m, k)$ denotes the failure probability of the code with k source symbols if m symbols have been received. While an ideal fountain code would decode with zero failure probability when $m = k$, the failure for Raptor code is still about 85%. However, the failure probability decreases exponentially when the number of received encoding symbols increases. Combined with the ability of fountain codes to generate an unlimited quantity of encoding symbols, this allows one to "match" the loss characteristics of the transmission channel, using the expected erasure rate to adapt the percentage of redundant information that must be sent in order to achieve the desired failure probability.

5. The stegosystem

The stegosystem presented in this section is a proof-of-concept demonstrating the effectiveness of a covert channel based on the embedding mechanisms which use only some header fields in e-mail messages. It can be mainly structured in two parts: the one which is in charge of the hiding and the other responsible for the revealing.

5.1. The hiding process

For the hiding process, the stegosystem takes the secret message M to be embedded and performs several operations in order to prepare the bundle to hide in the appropriate headers (see Fig. 2(a)), as follows:

1. compress the secret message M and obtain $C^*(M)$ via the *deflate* algorithm using the ZLIB library [24];
2. let s be the size (in bytes) of $C^*(M)$;
3. split the compressed secret message $C^*(M)$ into $k = \lceil s/50 \rceil$ 50-bytes chunks $C_i(M)$ with $i \in [1, k]$;
4. apply the Raptor encoding to the set of all the chunks C_i, obtaining a number q of encoded symbols E_j, $j \in [1, q]$, and generate the fake e-mail messages needed to transport the above chunks; the total number of messages q is a parameter whose value will be determined starting from k on the basis of the expected rate of loss or corruption of the e-mails;
5. derive two 256 bit keys, K_e for the encryption and K_{hmac} for the HMAC authentication, from a passphrase by using the PBKDF2 [25], which is part of the PKCS#5 standard;
6. for each encoded symbol E_j compute $R = HMAC_{K_{hmac}}(E_j)$. The hash function for the HMAC computation has been chosen to be the MD5 for the sake of simplicity. The stegosystem can be further improved by choosing a different algorithm or by simply modifying it. A stronger hash function could have been used, for example the SHA-1 algorithm (which costs 20 bytes), but for the specific purposes of the stegosystem, where strong integrity is not a key aspect, it would have been a waste of space;
7. perform a concatenation creating, for each chunk/message, a control header $H = (ESI \| R \| TS)$. It consists of 22 bytes as follows:
   - a 2-bytes *ESI* which represents the *Encoded Symbol ID* needed for decoding;
   - the value R of the HMAC (16 bytes if using MD5) needed to perform the integrity check;
   - a 4-bytes timestamp *TS* useful for keeping track of the time of each operation as well as avoiding replay-attacks.
8. the header H is concatenated with the encoded symbol E_j and encrypted by using the AES algorithm in CBC mode to obtain a per-mail stego-block $W_i = AES - CBC_{K_e}(H \parallel E_j)$. It is important to note that the encryption key (K_e) is different from the one used for the HMAC authentication (K_{hmac}). In addition, a user has to remember a passphrase from which the two keys are derived;
9. the last step is to distribute the individual stego-block W_i in the corresponding e-mail prepared at step 4.

   A padding operation on the last message chunk is required if $s \bmod 50 \neq 0$. Each stego-block W_i will be dished out as follows:
   - the first 22 bytes of W_i are hex-encoded, before using them, in the `Message-ID:` field, resulting in 44 characters, which is below the limit fixed by the standard [12,9];
   - the remaining part of W_i is base 64-encoded and is embedded into the `boundary` sections of the `Content-Type: multipart`/field. Just one `boundary` section is used in each message in order to not raise suspicion.

It is worth noting that the small amount of information that can be embedded in a single e-mail message causes an increase in the overall coding overhead: since one message has only room for one symbol (plus the header), the ESI must be repeated for each symbol, and cannot be specified once for all the symbols sharing the same ESI. However, it should be kept in mind that this covert channel is not meant for the transmission of a large amount of data. Hence, a small extra overhead is tolerable, especially if one considers that the Raptor encoding achieves a very high probability of successful decoding at the price of only a slight amount of redundant data.

5.2. The revealing process

The revealing process is the reverse of the hiding process (see Fig. 2(b)). For each received e-mail i, the following tasks are performed:

1. derive two 256 bit keys, K_e for the decryption and K_{hmac} for the HMAC authentication, starting from a passphrase by using the PBKDF2, as in Section 5.1;
2. extract, from the appropriate e-mail headers, the stego-block W_i and decrypt it by computing $AES - CBC^{-1}_{K_e}(W_i)$ to obtain $(H \parallel E_i)$, where H is the header and E_i is the encoded symbol;
3. the header H is split into $(ESI \| R \| TS)$;
4. compute $HMAC_{K_{hmac}}(E_i)$ and compare it with the value R just extracted in the header H at the previous step;
5. if the comparison fails, the stegosystem discards the involved message as a compromised (it has been tampered with) or invalid one (it may be a background noise message and hence does not belong to the covert channel flow). Otherwise save the E_i symbol, along with the associated ESI and timestamps *TS*, to be used in the correct sequence, together with all the other $q - 1$ blocks to reconstruct the whole compressed secret message $C^*(M)$ via Raptor decoding.

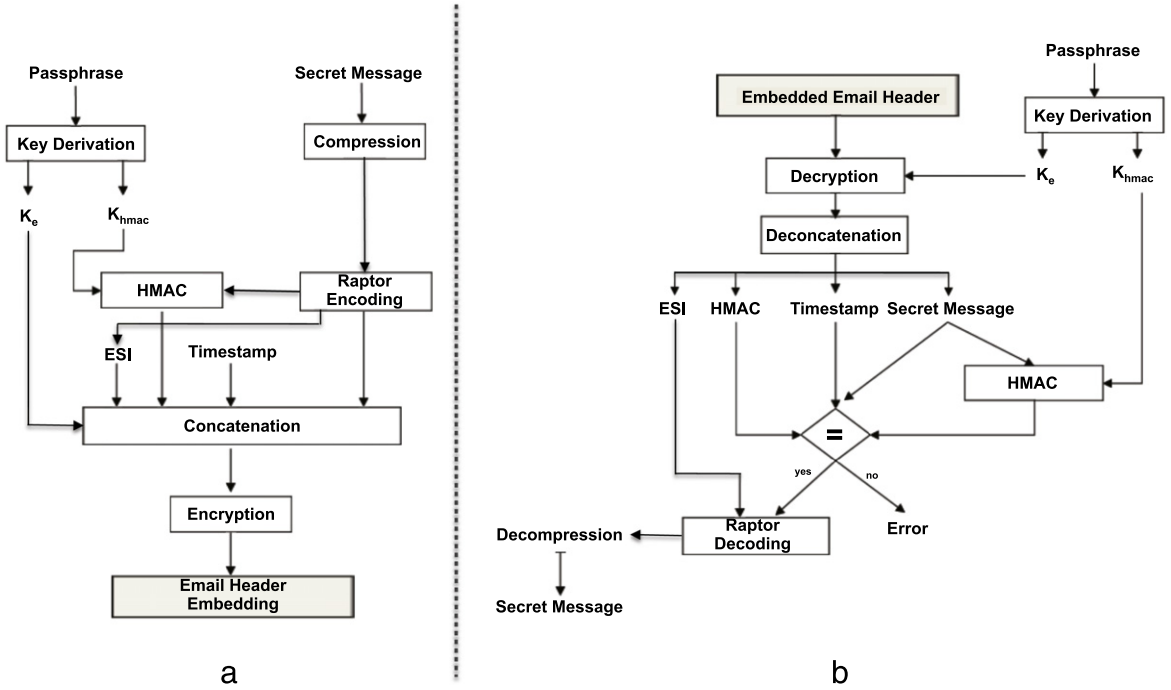


Fig. 2. The hiding part is shown in part (a). The corresponding revealing part is shown in part (b).

6. Performance considerations

In order to analyze the performance of the proposed mechanism, an empirical evaluation of its bandwidth has been proposed. The term "bandwidth" can be intended in two different meanings. Both the *hiding bandwidth* (i.e. the ratio of hidden data to cover data) and the *channel bandwidth* (i.e. the hidden bits transmitted per second) [26] can be quantified. To measure the hiding bandwidth $B_H = \frac{m}{Z}$, where m is the size of the hidden message and Z is the cover data volume, expressed in the same units as m, four components contributing to the ratio should be considered:

- the efficiency of the compression process $\chi = \frac{m}{m_C}$, i.e. the ratio of m to the compressed message size m_C;
- the overhead due to the Raptor encoding, $\rho = \frac{k}{q}$, i.e. the ratio of redundant symbols added, empirically determined starting from the expected failure rate over the covert media; recall that k is the number of input symbols and q the number of encoded symbols;
- the protocol overhead, $\pi = \frac{P}{P+H}$, i.e. the ratio of the payload P to the sum $P + H$ of the header size H and the payload P;
- the embedding overhead, $\eta = \frac{P+H}{s}$, i.e. the ratio of $P + H$ to the average carrier message size s.

The protocol overhead contributes equally for each symbol, since there is no grouping of symbols. The same applies for the embedding overhead (each message carries only one protocol unit). All the ratios can then be multiplied to get the hiding bandwidth:

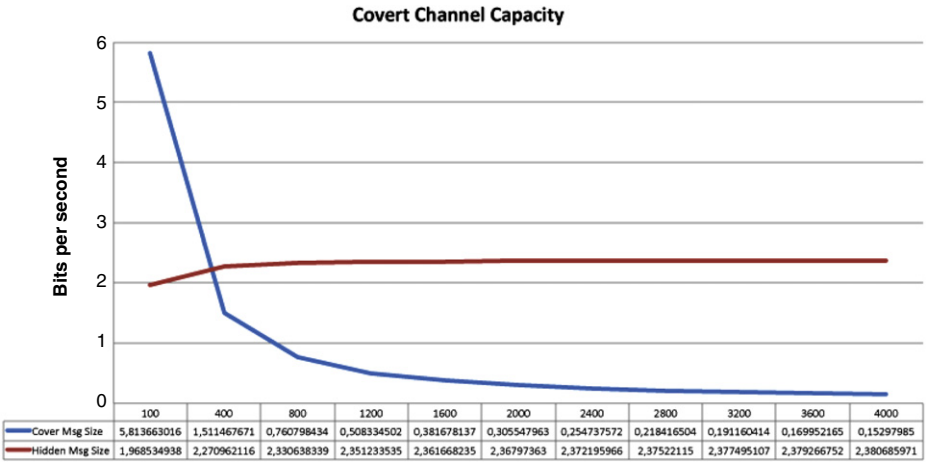
$$B_H = \chi \times \rho \times \pi \times \eta. \tag{4}$$

The channel bandwidth has been evaluated by using the *block sending model* presented in [27]. Accordingly, when transmitting a hidden message with a size of m bits, the resulting channel capacity $B_C = \frac{m}{T}$ bps can be calculated, where T is the time, in seconds, required to transmit the entire message, defined as

$$T = S + \frac{8 \cdot (B + N)}{V}. \tag{5}$$

Here, the first component S is the software overhead, measured in seconds and independent from the message size, B is the total size, in bytes, of the transmitted cover message(s), N is the network protocol overhead, in bytes, and V is the transport network speed in bit per seconds.

If s is the average size, in bytes, of the cover carrier message, in order to calculate the total size of the transmitted message B all the spam messages of size s needed to transport m bits on the covert channel must be taken into account. First of all, it can be considered that according to [28] the systematic Raptor code defined in [18], and used in the proposed scheme, is characterized by a relative reception overhead (the extra amount of encoding data required to recover the original



	100	400	800	1200	1600	2000	2400	2800	3200	3600	4000
Cover Msg Size	5,813663016	1,511467671	0,760798434	0,508334502	0,381678137	0,305547963	0,254737572	0,218416504	0,191160414	0,169952165	0,15297985
Hidden Msg Size	1,968534938	2,270962116	2,330638339	2,351233535	2,361668235	2,36797363	2,372195966	2,37522115	2,377495107	2,379266752	2,380685971

Fig. 3. Covert channel capacity varying with message sizes.

unencoded data) of less than 2% with probability 99.9999%. Thus the required number S of spam messages is given by $S = \frac{m}{50} \cdot 1.02$ and hence $B = S \cdot s$.

On a state-of-the-art transport infrastructure, the network-layer protocol overhead can be estimated as $N = 58$ bytes (obtained by 20 bytes of TCP header + 20 bytes of IP header + 18 bytes of Ethernet header), and the average transmission speed $V = 100$ Mbps. Furthermore, the software overhead time has been estimated as $S = 4.38$ msec, according to [27]. Hence, the resulting capacity B_C can be determined parametrically from m and s as

$$B_C = \frac{m}{\left(4.38 + \frac{8\cdot\left(\frac{1.02\cdot m\cdot s}{50}+58\right)}{100}\right)} = \frac{625m}{1.02m \cdot s + 5637.5}. \tag{6}$$

Accordingly, when transmitting a hidden message with a size of m bits, from the chart in Fig. 3 can be observed how the channel capacity varies with the cover carrier message size and the transmitted message size. In particular, it can be easily appreciated that the capacity is almost unaffected from variation to the hidden message size whereas it can be significantly conditioned from the e-mail vector message size that, while very useful for the sake of covering and transport, can be considered as pure overhead.

7. Security analysis and possible countermeasures

Many existing covert channel schemes follow the security through obscurity approach, and in principle their detection or suppression is straightforward. An analyst wishing to reveal the presence of a covert channel such as the one presented above should identify, in the flow of spam, the content-carrying messages. She/he could isolate these messages by differential analysis, comparing some characteristics of the messages with the values commonly observed in the whole flow and finding deviations. Aspects that could be analyzed include:

- the distribution of header values, looking for unusual patterns or repetitions;
- the distribution of sender/recipient pairs and of the server chain followed, looking for differences with respect to similar messages;
- the actual content of spam messages, looking for signs of forgery: the absence of a URL, or a malformed one, while it is important for the message purpose (e.g. for an apparently phishing e-mail) it may result suspicious, because spammers can be expected to control their payload very accurately.

Among the possible active countermeasures, an attack with a reasonable impact and expected success ratio could involve choosing some messages at random and corrupting them or blocking them downright. Essentially, the effectiveness of such a "black-holing" strategy would be proportional to the percentage of messages blocked. Obviously, if all messages marked as spam were filtered, the covert communication would be fully stopped. On the other hand, since it would involve spam filtering in the middle of the transport chain and not at the endpoints, this scheme would be easy to detect by the endpoint themselves, that would suddenly see a dramatic drop in the amount of spam received. At the same time, even if spam were blocked, the proposed scheme could be also used with "regular" e-mail messages.

Since the proposed covert communication scheme is entirely deployed on the higher application layers and does not rely on any lower level network protocol feature or parameter, it can satisfactorily resists to other active attacks exploiting lower layer characteristics. It is also enough resistant to unexpected message tampering due to its robust error detection and data loss/corruption tolerance features, which greatly weaken the effect of content manipulation in transit. Being the

communication scheme totally asynchronous, it is not affected by extemporary adverse (i.e., network latency or packet loss) transport network conditions and sender–receiver party problems (i.e., delayed message processing in the presence of heavy loads). Also, no regularities exploitable for detection can be observed in the sender–receiver communication pattern because, due to the independent one-way communication model, no message acknowledgments or response mechanisms are provided.

The used transport facility ensures that all the messages conveying hidden data have size and frequency distributions similar to those characterizing the "normal" e-mail or spam traffic. This is an ideal feature for mixing/hiding the covert channel into normal network communication. Also, an active monitoring party which is aware of the presence of covert communication does not have significant advantages because, in order to detect such a channel, it necessitates to establish which k out of n e-mail/spam messages passing between two hosts are conveying hidden data. In particular, the random message traffic associated to the remaining $(n - k)$ messages will hide the presence of the covert channel, and render not simple to determine if a pattern would precisely identify the covert communication to an observer who eventually decides to inspect the traffic. Furthermore, covert communication is totally independent from message inter-arrival rates. Consequently, no known state-of-the-art detection methods based on the statistical analysis of the observed traffic (e.g., those proposed in [29]) can be effectively used to detect the presence of covert communication. As it is based on the clever use of almost hidden SMTP protocol features, without altering their semantics and format and leaving the message payload totally unaffected, the covert channel is also undetectable by using the detection techniques proposed in [30,31].

8. Conclusions

It can be claimed that the possibilities for establishing covert communications through the internet are almost endless. This work exploits the ability to hide secret information into asynchronous application-layer channels, such as an e-mail session, by embedding it into appropriately chosen header fields. The presented data embedding and extraction technique requires minimal engineering efforts for their implementation while resulting in a robust steganographic framework that can survive to the inspection of firewalls and traffic analyzers, by resisting to all the state-of-the-art active detection techniques based on revealing protocol as well as data format/content anomalies or specific statistic properties in network traffic distribution. The use of a FEC encoding increases the resilience of the proposed scheme to cover media loss or corruption. The used header fields have been chosen by considering only those directly generated by MUAs and not "touched" by MTAs.

A further improvement of this work could be the use of the body of the e-mails (either spam or not) as the covering material. Besides that, the authors' decision has been to not perform such "extension" in order to be completely independent from the e-mail content (which usually become scrutinized).

The ideal vehicle to transport the steganographic means is a particular kind of unsolicited e-mail, the spam. Hated by the overwhelming majority of people, it could quite easy become one of the most undetectable stegosystems on the internet.

References

[1] US Department of Defense, Trusted computer system evaluation criteria, Tech. Rep. DOD 5200.28-ST, December 1985.
[2] A. Castiglione, A. De Santis, C. Soriente, Taking advantages of a disadvantage: digital forensics and steganography using document metadata, Journal of Systems and Software 80 (5) (2007) 750–764.
[3] A. Castiglione, A. De Santis, C. Soriente, Security and privacy issues in the Portable Document Format, Journal of Systems and Software 83 (10) (2010) 1813–1822.
[4] Z. Kwecka, Application layer covert channel analysis and detection, Napier University Edinburgh Technical Report, 2006.
[5] A. Castiglione, A. De Santis, U. Fiore, F. Palmieri, E-mail-based covert channels for asynchronous message steganography, in: 2011 Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IEEE Computer Society, Seoul, South Korea, 2011, pp. 559–564.
[6] S. Zander, Performance of selected noisy covert channels and their countermeasures in IP networks, Ph.D. Thesis, Swinburne University of Technology Melbourne, 2010.
[7] D. Watson, M. Smart, G.R. Malan, F. Jahanian, Protocol scrubbing: network security through transparent flow modification, IEEE/ACM Transactions on Networking (TON) 12 (2) (2004) 261–273.
[8] M. Intelligence, Annual security report, Symantec Corp., 2010.
[9] P. Resnick, Internet message format, RFC 2822, April 2001.
[10] N. Freed, N. Borenstein, Multipurpose Internet Mail Extensions (MIME) part one: format of internet message bodies, RFC 2045, November 1996.
[11] P. Wayner, Spam mimic, March 2001. http://www.spammimic.com/.
[12] D.H. Crocker, Standard for the format of ARPA internet text messages, RFC 822, August 1982.
[13] N. Freed, N. Borenstein, Multipurpose Internet Mail Extensions (MIME) part two: media types, RFC 2046, November 1996.
[14] N. Freed, N. Borenstein, Multipurpose Internet Mail Extensions (MIME) part five: conformance criteria and examples, RFC 2049, November 1996.
[15] J. Daemen, V. Rijmen, The block cipher Rijndael, in: Proceedings of the The International Conference on Smart Card Research and Applications, Springer-Verlag, London, UK, 2000, pp. 277–284.
[16] W. Stallings, Cryptography and Network Security: Principles and Practice, 3rd ed., Pearson Education, 2002.
[17] M. Bellare, New proofs for NMAC and HMAC: security without collision-resistance, in: Dwork C. (Ed.), CRYPTO, in: Lecture Notes in Computer Science, vol. 4117, Springer, 2006, pp. 602–619.
[18] M. Luby, A. Shokrollahi, M. Watson, T. Stockhammer, Raptor forward error correction scheme for object delivery, RFC 5053, October 2007.
[19] A. Shokrollahi, Raptor codes, IEEE Transactions on Information Theory 52 (6) (2006) 2551–2567.
[20] R.G. Gallager, Low-Density Parity-Check Codes, MIT Press, Cambridge, MA, 1963.
[21] R.M. Tanner, A recursive approach to low complexity codes, IEEE Transactions on Information Theory 27 (5) (1981) 533–547.
[22] M. Luby, M. Watson, T. Gasiba, T. Stockhammer, W. Xu, Raptor codes for reliable download delivery in wireless broadcast systems, in: Consumer Communications and Networking Conference, 2006, CCNC 2006, 3rd IEEE, vol. 1, January 2006, pp. 192–197.

[23] M. Luby, T. Gasiba, T. Stockhammer, M. Watson, Reliable multimedia download delivery in cellular broadcast networks, IEEE Transactions on Broadcasting 53 (1) (2007).
[24] J. Gailly, M. Adler, A massively spiffy yet delicately unobtrusive compression library, zlib 1.2.5, April 2010.
[25] B. Kaliski, PKCS #5: password-based cryptography specification version 2.0, RFC 2822, September 2000.
[26] T. Handel, M.T. Sandford, Hiding data in the OSI network model, in: Information Hiding, Springer, 1996, pp. 23–38.
[27] C.G. Girling, Covert channels in LAN's, IEEE Transactions on Software Engineering 13 (February) (1987) 292–296.
[28] T. Stockhammer, A. Shokrollahi, M.W.M. Luby, T. Gasiba, Application layer forward error correction for mobile multimedia broadcasting, in: B. Furhet, S. Ahson (Eds.), Handbook of Mobile Broadcasting: DVB-H, DMB, ISDB-T and Media FLO, CRC Press, 2008, pp. 239–280.
[29] S. Cabuk, C.E. Brodley, C. Shields, IP covert timing channels: design and detection, in: Proceedings of the 11th ACM Conference on Computer and Communications Security, CCS'04, ACM, New York, NY, USA, 2004, pp. 178–187.
[30] G. Fisk, M. Fisk, C. Papadopoulos, J. Neil, Eliminating steganography in internet traffic with active wardens, in: Revised Papers from the 5th International Workshop on Information Hiding, IH'02, Springer-Verlag, London, UK, 2003, pp. 18–35.
[31] M. Handley, V. Paxson, C. Kreibich, Network intrusion detection: evasion, traffic normalization, and end-to-end protocol semantics, in: Proceedings of the 10th Conference on USENIX Security Symposium, vol. 10, USENIX Association, Berkeley, CA, USA, 2001.