

Building Covert Channels over the Packet Reordering Phenomenon

Adel El-Atawy and Ehab Al-Shaer
 School of Computing
 DePaul University
 Chicago, IL, USA
 Email: {aelatawy, ehab}@cs.depaul.edu

Abstract—New modes of communication have shown themselves to be needed for more secure and private types of data. Steganography or data-hiding through covert channels can be highly motivated by today’s security requirements and various needs of applications. Moreover, the amount of information in the Internet traffic is not bounded by what is contained in packets payload; there is considerable hidden capacity within packets and flows characteristics to build robust and stealthy covert channels.

In this paper, we propose using the packet reordering phenomenon as the media to carry a hidden channel. As a naturally occurring behavior of packets traveling the Internet, it can as well be induced to send a signal to the receiving end. Specific permutations are selected to enhance the reliability of the channel, while their distribution was selected to imitate real traffic and increase stealthiness. The robustness of such channel is analyzed, and its bandwidth is calculated. A simple tool is implemented to communicate over the natural phenomenon of packet reordering. Reliability and capacity of the techniques are evaluated and promising results show the potential of the proposed approach.

I. INTRODUCTION

Data transmission over normal data channels in the Internet has been always characterized by being plain, in the open and available to anyone to collect and analyze. Even with virtual private networks, IPSEC, or any other form of data encapsulation and encryption, the channel itself is known to exist. In other words, everyone is aware of the communication taking place even if the content itself is not readily available. Therefore, new modes of communications have shown themselves to be needed for more secure and private types of data where being evasive to adversaries is badly needed.

Steganography can be incorporated into new modes of communication that can be highly motivated by today’s security requirements. Knowing that a communication exists between two parties is a valuable piece of information even if the content is not known. An example can be borrowed from the stock market, where knowing that an entity has an ongoing communication with a competitor might give an insight into current deals being negotiated. Also, with sensitive corporate information, leaking the information through hiding is far more effective. Another important and undesired application of covert channels is when deployed by hackers and zombie masters to control their slaves. Evading detection will give hackers an edge and pushes detection mechanisms to their limit. Covert channels over network traffic has been around for years. However, their design and implementation has been limited to a few types that are either easy to detect or have

a low capacity. Applications for having a side-channel along the main data channel (*i.e.*, payload of packets traversing our networks) can be tremendous, and vary from peaceful applications to a tool that can be used by hackers and all other types of cyber-criminals.

We propose a technique that can use the unconventional channel of packet order to be our covert communication medium. By manipulating the order of packets sent over the network at the sender-side, we emulate the packet reorder phenomenon which takes place naturally. Interceptors and intermediate nodes will not be able to order these packets due to the overwhelming computational cost of buffering and sorting packets at the network core. On the other hand, the receiver at the network layer, will be able to observe the packet order as the secret signal. Moreover, the reorder extent is designed not to affect the normal operation of the transport layer protocol. The transport protocol (*e.g.*, TCP) will perform its duty in reordering the packets for delivery to the application layer transparently of the underlying hidden operations.

A typical case at the receiver’s end is to receive ordered packets from $1, 2, \dots, n$ (*i.e.*, sending order). If there exist any deviation from this order, the transport layer will perform the reorder and forward the payloads to the application. However, this unordered sequence bares more information that we will utilize as our covert channel medium. In our case, we intentionally send out-of-order packets where different permutations hold different meanings (or codes). Therefore, we claim we have a channel that is independent from the packet’s payloads and not very sensitive to inter-packet jitter. Our proposed method go well beyond previous work that used packet payload to hide the secret messages. Also, we go a step further than techniques that used packet inter-arrival time as the main media for data communication. Our focus is on possible ways to send data hidden over the Internet without affecting the traffic headers or payload themselves. The proposed mechanism includes 1) applying coding theory for careful selection of permutation patterns to enhance robustness and error-resistance, 2) calculating distribution of codewords to evade detection by imitating natural Internet traffic, and 3) the decoder technique for retrieving the sent secret codes.

The goal of this work is to build a channel which is reliable and practical while having a reasonable bandwidth. In Steganographic systems, there are always three factors competing: bandwidth vs robustness vs stealthiness. By increasing the bandwidth, the channel used to hide information (host channel) becomes more susceptible to noise and being discovered

by adversaries. Lowering the hidden bandwidth (*i.e.*, data transmitted in the host channel) causes the transmission to be more stealthy and more resistant to change due to external noise. We will investigate the parameters that will affect each of these factors, and analyze the consequences of changes applied to each one.

In the following sections, we start with an overview of the phenomenon of out-of-order packets from studies of its abundance and persistence in Section II. Then, a brief description of related work in covert communication is in Section III. A simple channel is described in Section IV to introduce the idea to the reader, and to show the potential of such channels. The full-fledged channel is then described in Section V. The evaluation and final discussion are provided in Sections VII and VIII, respectively.

II. PACKET REORDERING PHENOMENON

Packet reordering in network traffic is receiving packets in a different order than that used to send them. Previous studies [14] showed that it is mainly due to one of two reasons: 1) local built-in parallelism and connection slicing, and 2) multi-path forwarding routes [18]. The first reason was shown to be more influential. As the need for faster processing speeds and the increasing ease of adding parallel processing capabilities, the packet reordering phenomenon is not going to disappear in the near future. Results [6] showed that around 90% of all sessions in the experiment had packets reordered, with the percentage of packets transmitted out-of-order ranged from 0.1% to 3%. However, more recent studies [15], [16], showed that the reasons behind packet reordering are extended to include: 1) load balancers, 2) layer 2 and TCP retransmissions, and 3) DiffServ handling of some flows that violate their quality limits. All these factors are and will stay persistent in modern day data networks. Therefore, we can claim we have a property that is abundant, and can be observed as a natural behavior. Moreover, introducing more out-of-order packets is not necessarily against the overall session and network performance [3].

It is important to study the applicability of relying on packet order as a reliable and capable channel. For a property to be useful as host for building covert channels, it needs to have a static pattern to a reasonable extent. This pattern needs to be rich enough to create a sort of background noise in which we can hide our covert channel. For example, in image-based steganography, the ideal host images are those of complex nature not images with pure plain areas. Packet ordering satisfies this condition as studies below show. Another factor to choose host properties, is that it has to be neither easy to monitor/record nor excessively studied. Again, packet ordering fits the profile perfectly. Despite being studied in the literature [3], [5], [14], [15] with respect to abundance and effect, it is almost always overlooked in practice by administrators due to irrelevance or because of the prohibitive cost of doing so.

Efficient, yet concise, metrics for measuring the amount of packet reordering has been the center topic for several research work. In [15], a survey of reordering metrics is presented. Reorder Density (RD) and Reorder Buffer-occupancy Density (RBD) are the most applicable metrics to use in our application. They measure the reordered packets with respect to

how far the received packets are away from what is expected, and how much buffering is needed for reorder, respectively. In this paper we use a variant of RD throughout the analysis and discussion. However, RBD can be used as well with little or no change to the logic or implementation.

The study of the packet reordering phenomenon over the Internet has taken different turns. Researchers focused on how probable it is, the factors that affected it, and what are the effects on network stability and performance. In section III, we will quickly show some of the related work in the literature that address covert network channels in general. This includes the techniques used for creating covert channels, especially those built over intangible media as packet timing (which was almost the sole target of all recent papers proposing covert communication over network traffic).

III. RELATED WORK

Previous work on covert channels in Internet traffic have focused on hiding information in either payload (not network based at all), or inside IP header fields. Some work used IP options fields, or source port patterns to hide the secret information. Others used timing information of packet sequences. The closest work to what we are planning to do, and the most recent, is the work by Shah et al in [21], that used timing channels for leaking types passwords from victim machines and [1] that used packet sequences/order as a storage media.

Previous work on covert channels [19] in network traffic can be divided according to the storage media used: packet payload, packet header, or packet timing/behavior. Different researchers have focused on identifying the possible applications in public networks [24] and their theoretical capacity bounds [23].

a) Using Header Fields: These schemes are simple yet have a wide variety as shown in [1] where custom techniques for each common protocol are shown including secure protocols (*e.g.*, IPSec). A mixed technique that uses both the specifics of TCP time-stamping and flow timing of packets is presented in [11], in a joint technique between both categories; real packet field storage and timing channels. In [13], specifics of wireless operation has been used in the 802.11 standard to embed the required secret information.

b) Using Timing Channels: Timing channels were originally suggested to attack crypto-systems and to leak information between HI and LO processes (security-clearance wise) [12]. Many researchers worked on analysis of timing information over Internet traffic to obtain worst case bandwidth variation and end-to-end delay/jitter measurements. These are necessary in the analysis of channel capacity. In [21], a device (JitterBug) that plugs between the keyboard and the machine of the victim, encodes the typed characters into inter-character delay. Although the encoding used in their work is very simplistic, and there was no formal analysis of the capacity, the idea was very attractive and shows considerable innovation. Also in [8], information is embedded in timing channels created by the sender where on/off sending of packets encodes the intended signal; a simple technique and very intuitive but will have a strong effect on the overt communication bandwidth. The work in [1] is very close to our approach where packet order is manipulated to store information. However, their approach is limited by having a specific number of

patterns, and lack of generalization in the analysis. Also, code selection was not made to specifically to evade packet order metrics in the literature.

There has been extensive theoretical study of this type of side channels. In [22], it was shown that for real time systems to exist, security requirements have to be relaxed as the bandwidth of covert-timing channels associated with system events can increase indefinitely. The capacity of transmission channels having a queue as the source of non-determinism is given in [2]. The authors showed that it is possible to exceed the raw capacity of the channel by augmenting this channel with a side-timing-channel. The analysis was theoretical and not directly applicable to our network environments, yet it shows the high potential of such techniques.

c) Counter-Steganography Techniques: Techniques for discovering the existence of covert channels is far less mature than the hiding techniques themselves, and the theoretical work behind them. The majority are based on either; 1) utilizing the fact that allegedly contaminated data have some statistical properties that reveals alterations, and 2) depending on the host data itself being resistant to change. In [8], techniques are presented that manipulates packet timing in order to ruin the secret channels. A classic technique was suggested before in [12], where fuzzy timing is introduced to add non-determinism into the timing channels and significantly lower its bandwidth. Their focus was information leaking between OS processes, but the same concept still applies. In [7], a framework to detect data-hiding via tunneling in web traffic was shown. By monitoring statistical properties expected to be prevalent like packet size and inter request timing, alerts can be issued when suspected tunneling occurs. Information theory and entropy measurements were used in [10] to detect covert timing channels by observing changes in the information content in the packet timing. A very effective technique but its applicability is still limited to timing channels only.

IV. SIMPLE CHANNEL DESIGN

In a connection-oriented session, the order of packets received is irrelevant at the application level thanks to underlying layers that sort received packets to reach the correct stream structure. In such connections (*e.g.*, TCP flows), the original packet order is maintained by buffering out-of-order packets till intermediate (*i.e.*, late) ones arrive at the destination. Each packet is marked with some sequence identifier (*e.g.*, *sequence_number* in TCP or IPSec) whose nature and range depends on the end-to-end protocol used. As indicated before, many reasons contribute to causing this unordered reception of packets. It has always been considered a nuisance to the receiver and fixing it was one of the top priorities of such protocols. In our application domain, this phenomenon can be used or, more specifically, induced, to build a covert channel.

The normal/perfect received order of packets is $1, 2, \dots, n$. If packets were sent in a specific order, then there is a great chance they will be received as such. Therefore, this out-of-order was always considered as either rare or an annoying property of some channels. However, if we considered the received packet order as a source of information, then we have a channel that is independent from the packet's payloads and not very sensitive to inter-packet jitter. If packets were never received out-of-order as in circuit switched networks, then

this channel has zero capacity as its output is deterministic. However, this is not the case, and packet reordering is not that rare over the Internet (see [6]).

The question is: can we send information bits masqueraded in the form of packet order? If we assumed that packets in-order is one state of the channel (*i.e.*, conveying one specific symbol, say a 0 bit), then other packet orders that deviate from the perfect order are other symbols of our alphabet. The number of different orders we can impose over the packet sequence implies the capacity. Quantitatively, the logarithm of the number of distinguishable orders equals the channel's capacity in bits. Therefore, for n packets the maximum capacity will be $\log n! \sim n \log n$. Taking a file transfer as our example: a 700MB file, using 1.5Kbyte packets. This will have $8.7Mbit \sim 1.1MB$ (466K packets). Impressive as it may seem, this range is far from being practical as it literally destroys the host channel (no TCP stack can handle reordering this number of packets without failing). Moreover, this will jeopardize the stealthiness of the channel. Therefore, we will take a simple subset of this full-permutation space for demonstration: only adjacent packets can be reordered together. The coding space now is limited to 1 bit per packet pair (*i.e.*, 0:in-order, 1:swapped), reducing the overall capacity to $233Kbit \sim 30Kbyte$. In other words, the entropy of this channel is 1-bit per symbol, where a symbol is represented via a packet-pair. In terms of the overt channel, the side channel provides a capacity of 0.5 bits per overt packet.

Furthermore, a TCP stream with a high rate of out-of-order packets (*e.g.*, 50% of pairs on average being flipped) will look suspicious if monitored, and reducing this is a further step towards better stealthiness. Assuming a pair is swapped with a probability of 0.05, the entropy per packet-pair will be lowered to $H(0.05) = -0.05 \log 0.05 - 0.95 \log 0.95 = 0.2864bits$ and the overall capacity will be $66.8Kbit \sim 8.3Kbyte$.

In this channel, we omitted some basic features that are highly needed for successful communication. Namely, error detection and correction on the symbol level. If a packet pair is reordered due to naturally occurring network behavior, our induced packet order might be canceled out causing a 1-bit to be received as a 0-bit or vice versa. It is not possible to detect this kind of errors other than relying on multi-bit code words (*e.g.*, adding even/odd parity, or more sophisticated codes as Hamming code, Reed-Solomon, etc) or plain error detection via CRC-like mechanisms. In the following section, we will discuss how to build a real channel with enough details to support this feature in an intrinsic manner. This is achieved by extending the unit of transmission to be multiple packets for each transmitted symbol rather than a packet-pair. Moreover, better distribution and selection of packet-ordering patterns will be discussed in order to satisfy the stealthiness property as well as boosting the resilience for errors.

V. PACKET REORDER CHANNEL

In order to extend our simple prototype to a realistic and valuable channel, some important features are needed. These features include error resistance via error detection and correction on the basic transmission level that corresponds to the physical layer in traditional channels. Also, for our specific application, the stealthiness of the channel is an important goal by definition. Therefore, the overall reorder

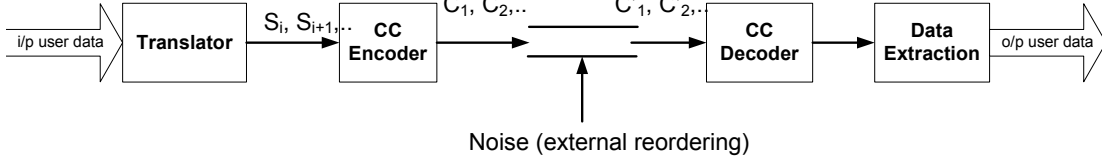


Fig. 1. Overall channel design. Starts with user data, translated to symbols that is decoded into permutation codewords to be sent to the receiver. The receiver decode the permutation into the original data symbols.

should be kept within normal levels. This is represented via two factors: *reorder depth*, and *reorder volume*. The former indicates the farthest packets that can be swapped, or the extent by which a packet can be moved, and the latter represents the percentage of packets that are out of order. Both metrics can be represented concisely in the ReOrder Density (RD) and the Reorder Buffer-occupancy Distance (RBD) metrics proposed by Piratla in [16], [17]. The above two main features, if carefully addressed, will result in a channel that is, both, robust and stealthy to a great extent. In this section, we address each of our goals by: 1) defining our channel parameters, 2) selecting the codewords (*i.e.*, permutation patterns), 3) enhancing error resilience by better codeword selection, 4) avoid detection by adjusting codeword probabilities.

A. Model and Notation

Let N be the total number of packets transmitted in the overt channel. Our covert channel (CC) will be built using every k packets from the overt channel (OC) to represent one codeword/symbol. A code word is a specific permutation of the k packets. Each codeword will be one of l different values selected from the set of code words L ($l = |L|$). Therefore, the upper bound on the information content of codeword is $\log_2(l)$ bits (when codewords are selected uniformly). Thus resulting in a total capacity of $N/k \times \log_2(l)$ bits. In our simple example from the previous section, we have $k = l = 2$ resulting in an overall data volume of $N/2$ bits. The two possible codewords used were “ordered” and “swapped” resulting in a plain single-bit binary coding. After introducing non-uniform use of codes for better stealthiness, the average information per codeword drops to $H(p_L)$, where p_L is the probability distribution of the L codewords the source uses to encode his hidden message.

In the case of having the transmission reorder of more than 2 packets exploited into a single codeword, say k packets; we will be having a maximum of $k!$ codewords. For the simple channel, this did not reflect any difference in our analysis. However, for $k > 2$, the capability of having error detection and correction is introduced, and some changes in our simple calculations are needed.

In general, an n -bit block code is denoted by (n, m) block code, if m bits of information were encoded into n -bit blocks. In that case, the coding rate R is equal to m/n . However, in our model, the difference in notation from bits per codeword to packets per codeword, and the use of permutation instead of bit values will change the resulting expressions. The block code will be denoted a $(k \log_2(k), \log_2(l))$ block code, and will have a code rate of $\log_2(l)/k \log_2(k)$. This will be needed when discussing the error correcting capabilities.

N	total packets in OC
k	packets to represent each codeword
N/k	number of transmitted symbols in CC
L	set of valid codewords
l	size of set L , $l = L $
p_L	probability distribution of codewords
$H(p_L)$	bits represented by each codeword
R	code rate, $R = \log_2(l)/k \log_2(k)$

TABLE I
NOTATIONS AND SYMBOLS USED

From now on, packets in the overt channel will be denoted by P_1, P_2, \dots, P_N . Inside code block i (*i.e.*, symbol S_i transmitted over CC), the constituent packets will be denoted by $C_{i,1}, C_{i,2}, \dots, C_{i,k}$. Then, the overall sequence will be $C_{1,1}, \dots, C_{1,k}, C_{2,1}, \dots, C_{2,k}, \dots$. If the symbol index is understood, or the discussion is not referring to a specific symbol then the first subscript will be dropped: $C_{i,j} \sim C_j$.

The communication between sender and receiver is assumed to be established after communicating through another form of secure communication to agree on system parameters (*e.g.*, k , l , etc) and the flows to be used. Although this information can be guessed with some extra effort by the receiver, but will not be assumed in this paper.

B. Codeword Selection

From a covert channel owner point of view, the perfect environment is where: 1) no one suspects the presence of a CC (detectability constraint), and 2) no reordering is introduced from the network itself (noise in transmission). The first constraint limits the code size and codewords used. The longer the codeword (*i.e.*, higher k), the higher the probability a monitor will suspect the covert channel presence. The reason is that long reorders (*e.g.*, 10 packets in perfect reverse order) are not naturally occurring in data networks. The noise effect can be limited by introducing larger block codes and deeper swaps. Both constraints can be handled via code block size compromise and intelligent codeword selection.

Limiting the codeword size k to low values (*e.g.*, less than 5) will be enough for most cases, and it should be adjusted not to deviate significantly from the network’s naturally induced reorders. Codeword selection and their probability distribution p_L can affect both constraints. For example, we can put a constraint that no codeword should correspond to a complete reverse of more than 3 packets. Also, we can limit the distribution of packets out of order to a certain threshold to evade detection.

Each codeword is defined by the order of the constituent packets. We defined packets in a codeword to be $\langle C_1, C_2, \dots, C_k \rangle$. These values are assigned based on the original packets to be sent in that time order. Then, $\langle C_1, C_2, C_3, C_4 \rangle = \langle 1, 4, 3, 2 \rangle$ means we are defining one of the codewords to be a four packet-sequence sent in such an order that the second and fourth packets are swapped. At the receiving end, the transport layer will unknowingly resort them to be $\langle 1, 2, 3, 4 \rangle$ and forward the corrected sequence to the application layer. In the same time, a lower level module will be able to see the difference before resorting and extract the coded secret information.

Obviously, we have $k!$ ways to define codewords of length k . However, in order to be able to implement error-detection and correction, only a few of those will be used. The goal is to select a few, namely l codewords, out of the $k!$ possible permutations. This gives the previously calculated code rate that is sometimes also called efficiency $R = \eta = \log_2 l / \log_2 k!$. In our original prototype, $\eta = 1$ as expected, as there were no loss in efficiency to support error detection/correction. In a more sophisticated channel, take $l = 3$, $k = 4$, we have $\eta = 0.3456$. Normally, $\eta \leq 1$, where the equality is achieved if and only if all $k!$ codewords are to be used. This should be always avoided unless: 1) the value of k is very low (e.g., ≤ 5), and 2) the host channel is known not to introduce any further reordering of its own.

d) Detecting single errors:: To be able to detect (though, not correct) all single bit errors, all code words are selected to be even permutations. A permutation is even if it needs an even number of element-swaps to reach the desired order starting from the perfect order (i.e., all packets are sent in their intended order). Thus, our valid codewords must include the identity permutations (i.e., $\langle C \rangle = \langle 1, 2, \dots, k \rangle$). Any single transposition between packets will yield an error in transmission that can be detected, yet not corrected. Considering that a high portion of errors taking place in natural transmission is in the form of a single adjacent packet swap, such code can achieve good error detection rates. However, our target is to select codewords that can detect and correct multiple errors in transmissions. In other words, if one packet (or more) are shifted from their intended location as specified by the original codeword, the receiver should be able to detect and optionally correct those into the correct permutation.

e) General code selection: To be able to generate more powerful coding schemes, we formulate (and map) our permutation patterns into regular linear code blocks. Any of the l ($l \leq k!$) codewords $\langle C \rangle = \langle C_{1\dots k} \rangle$ can be represented efficiently using $k \log_2(k)$ bits. Each of the packets in a codeword is represented by a number that is essentially the RBD [17] before processing the packet. In other words, each packet in the codeword is represented by the number of packets of higher index received before it. $code(C_j) = \sum u(i-j)$, where $u(x) = 1$ (if $x > 0$) and $u(x) = 0$ otherwise. For example, receiving a codeword $\langle 1, 4, 3, 2 \rangle$ is translated to $0, 2, 1, 0$. The packet indexed by 1 was not preceded by any other packet with larger index, while packet with index 2 was preceded by another 2 packets with higher index, and so on. Obviously, the packet with the highest index will always be translated to 0 regardless of its order, so it will be omitted from

codeword	block coded	RD
$C_1 = \langle 1, 2, 3 \rangle$	00 00	$\langle 1, 0, 0 \rangle$
$C_2 = \langle 1, 3, 2 \rangle$	00 01	$\langle 2/3, 1/3, 0 \rangle$
$C_3 = \langle 2, 1, 3 \rangle$	01 00	$\langle 2/3, 1/3, 0 \rangle$
$C_4 = \langle 2, 3, 1 \rangle$	11 00	$\langle 1/3, 1/3, 1/3 \rangle$
$C_5 = \langle 3, 1, 2 \rangle$	01 01	$\langle 2/3, 0, 1/3 \rangle$
$C_6 = \langle 3, 2, 1 \rangle$	11 01	$\langle 1/3, 1/3, 1/3 \rangle$

TABLE II

EXAMPLE CODE USING $k = 3$ AND $l = k!$. THE BLOCK CODE IS WRITTEN IN BRGC. THE RD COLUMN SHOWS NORMALIZED HISTOGRAM ON THE METRIC AFTER THE EFFECT OF A SPECIFIC PERMUTATION

the mapping. Therefore, we represent each of the $k-1$ packets using $\log_2 k$ bits, resulting in an overall $(k-1)\lceil \log_2 k \rceil$ bits. This representation is asymptotically optimal, as the overall number of permutation patterns is $k!$, and it needs $\Theta(k \log_2 k)$ bits to be encoded.

Furthermore, in order to have a correct mapping regarding the Hamming distance induced by packet shift errors; we represent these numbers via Binary Reflected Gray Codes (BRGC) [20]. This representation is the easiest to compute from normal binary representation (i.e., $G = N \oplus (N/2)$), and we guarantee having each packet shift represented as a single extra Hamming distance.

In Table II, we demonstrate an example using $k = 3$, with L equals the whole possible set of permutations (i.e., $l = k! = 6$). Say, we need to detect single errors, then a distance of 2 is required. In such cases, the maximum number of codewords we can use (to increase the code rate) is 3. The selected code words L will be $\{C_1, C_4, C_5\}$, and the code rate will be $R = \log_2 3/3! = 0.264$. Using these selected block code, we will be able to detect any single step shift of packets in the transmission due to natural causes. As another example, for the enhanced requirement of correcting a single error we need a Hamming distance of at least 3. Therefore, our only solution for selecting codewords will be $\{C_1, C_6\}$. Using this block code we will correct single errors while detecting up to two errors. In general, for detecting e errors a minimum distance of $e + 1$ is needed, and for e -error correction the requirement increases to $2e + 1$.

Generally, the selection process is performed on the block code to achieve the desired characteristics. However, in our case this is not a straightforward task, as there will be some block codes that are impossible to convert back to a permutation pattern. The space defined by $(k-1)\lceil \log_2 k \rceil$ bits is an upper bound to $k!$, with $k = 5$ as the highest ratio between the two space sizes: $(k-1)\lceil \log_2 k \rceil / \lceil \log_2 k! \rceil = (5-1)\lceil \log_2 5 \rceil / \lceil \log_2 5! \rceil = 1.714$.

C. Codeword distribution

After selecting which codewords are valid to be used in order to obtain the desired error correction capabilities, comes the step to choose the input codeword probability distribution to our channel. Normally, the aim of choosing input probabilities is always to achieve the capacity of the channel being used given its error model. However, in our case, the goal is different where we are willing to sacrifice bandwidth in order to follow a behavior that will keep the stealthiness of the covert channel. For maximum transmission capacity, codewords should be used equally probable resulting

in a transmission capacity of $\log l$ bits per codeword, or $\log l/k$ bits per overt packet used. However, to satisfy the stealthiness requirement we should use some patterns more than others not to deviate significantly from the normal behavior and be susceptible to raising alerts and being detected. Using metrics as Reordering Density (RD) [4], [16] and Reordering Buffer-Occupied Density (RBD) [17], we will be able to know this target behavior.

For demonstration, let us consider the easiest case of $k = 2$. Although this will limit our code selection as there will not be any error detection/correction capabilities, we can still use it as a base for discussion. Assume a study of the host networks involved showed that a maximum unobserved RD histogram will have 90% of packets received as expected, and 10% are one step earlier. Let us denote that by the sequence $RD_t = \langle 0.9, 0.1 \rangle$, where RD_t is the target RD. As we have only two possible codes (*i.e.*, $\langle 1, 2 \rangle$ and $\langle 2, 1 \rangle$), then we just need to calculate the probability of using each to go below the detection radar.

Each code participate in a specific RD (RD_t) pattern RD_i which is the RD histogram caused by codeword i . As per our example, the first code $C_1 = \langle 1, 2 \rangle$ will contribute with a normalized RD equals to $RD_1 = \langle 1, 0 \rangle$ and the other code: $RD_2 = \langle 0.5, 0.5 \rangle$. Then,

$$\begin{pmatrix} 1 & 0.5 \\ 0 & 0.5 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} = \begin{pmatrix} 0.9 \\ 0.1 \end{pmatrix}$$

where p_i is the probability of code i . Solving for p_i we find that the identity permutation should be used 80% of the time, and the swapped pair should appear only 20% in order to satisfy the stealthiness requirements of the example network.

Generally, we can write $RD \times P = RD_t$ or in detail:

$$\begin{pmatrix} RD_{1,1} & \dots & RD_{l,1} \\ \vdots & \ddots & \vdots \\ RD_{1,k} & \dots & RD_{l,k} \end{pmatrix} \begin{pmatrix} p_1 \\ \dots \\ p_l \end{pmatrix} = RD_t, \quad (1)$$

where $[RD]$ is a k by l matrix where each column is an RD_i k -vector, $RD_{i,j}$ specifies the contribution of the i^{th} codeword on element j in the RD metric, and P is the l column vector of codeword probabilities to solve, and finally RD_t is the k vector with target overall RD effect of our covert channel.

A special case that can take place is having more than one codeword with the same metric vector. In such case, the calculations for calculating the distribution take place on the distinct vectors only. Formally, we solve the equation $RD \times P_{\bar{L}} = RD_t$, where \bar{L} is a subset of L where only codewords with distinct RD vectors are included. Thus, we have $\bar{l} = |\bar{L}| = rank(RD)$, assuming $l \geq k$. Once the equation is solved and probability of each distinct metric value is obtained, it will be evenly split over codewords sharing this value. The uniformity of the distribution is used to provide the maximum entropy of the distribution [9]. The same effect can be obtained by including all vectors and perturbing identical vectors with insignificant values to impose independence. However, this approach might force the solving algorithm into an unstable state due to the high error of dealing with tightly related vectors.

Code Rotation: Moreover, to further evade detection, we deploy another mechanism for code selection: Instead of just selecting l codewords and use them over our entire channel

k	2	3	4	5	6	7	8	9	10	11	12
E_C	0	1	2	3	4	5	7	8	10	12	14
E_D	0	2	4	6	8	11	14	17	21	24	28

TABLE III

THE EFFECT OF INCREASING THE CODEWORD SIZE (IN PACKETS) ON THE MAXIMUM DETECTABLE E_D AND CORRECTABLE ERRORS E_C .

lifetime, we select more than one group of l codewords, and rotate over after each sent block. The effect of this rotation is that over time all codewords will be observed by an adversary. Choosing the appropriate l now is more tricky, as it has to be chosen such that l divides $k!$ evenly for a complete coverage of all possible permutations. However, there is an advantage of having $k!$ as the total number of possible of codewords, as it is trivially factorized with a very wide factors span.

D. Performance and capacity calculations

Selecting the best set of codewords that satisfy the requirements of error detection and correction is a classically hard problem. The optimal solution is the largest set of codewords (*i.e.*, maximum code rate) that satisfies the capability requirements. A generally applicable algorithm is not yet reached. In traditional linear block codes, the size of such set (L) is bounded as follows:

$$l = |L| \leq \frac{2^n}{\sum_{i=0}^{d_{min}} \binom{n}{i}}, \quad (2)$$

where d_{min} is the minimum Hamming distance required for handling the specified error rate, and n is the number of bits in the block code (*i.e.*, $n \approx k \log k$). However, this formula will not be fully applicable in our case, as some of the 2^n codewords are not mapped to permutations we can use. Moreover, this difference need to be reflected in the numerator and denominator of the given formula complicating things further.

For a k packet codeword, the resulting block code will be $O(k \log k)$ regardless of the mapping used. Therefore, the maximum number of errors we can correct E_C is bounded by $E_C \leq \lfloor (n-1)/2 \rfloor$, and the maximum number of errors detectable E_D is bounded by $E_D \leq n-1$, where n . These bounds are only achievable on the expense of having the minimum code rate R possible; $R = 1/n = 1/(k \log_2 k)$ by using only two out of the $k!$ possible permutations. Obviously, the two codewords that satisfy the maximum distance are the identity and the reversed permutations.

For a complete view of the channel's bandwidth, one can combine the information given into a single formula that is based on k , l , p_L , and desired error-handling capabilities.

$$Capacity = \frac{H(p_L)}{k} \text{ bits/packet}, \quad (3)$$

where $l = |L|$ is bounded as by Eqn. 2, and $H(\cdot)$ is the entropy of a given probability distribution [9]. If maximum capacity is required given a specific error-handling capability regardless of the stealthiness requirement, the $H(p_L)$ term will be replaced with its upper bound $\log l$. This in turn can be equated with

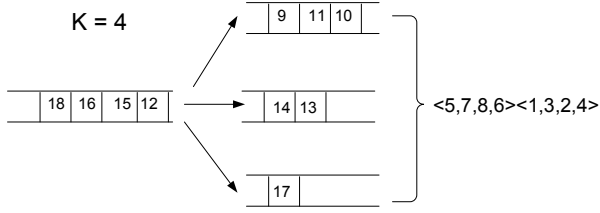


Fig. 2. The decoding process. Multiple partially received codewords are in temporary storage till completed, then forwarded to user.

its limit from Eqn. 2. The overall capacity is given by

$$Capacity \leq \frac{1}{k} \times \log \left[\frac{2^{\lceil \log k! \rceil}}{\sum_{i=0}^{d_{min}} \binom{\lceil \log k! \rceil}{i}} \right] \text{ bits/packet}, \quad (4)$$

where d_{min} is given as before based on the error detection/correction capabilities.

VI. IMPLEMENTATION DETAILS

The sender and receiver have to agree on an overt channel to host their covert communication. This includes defining the maximum allowable RD (or other similar metrics) to be used in setting the codeword usage distribution. Also, both parties have to agree on their choice of k , l , and L in order to successfully decode the sent information. In Figure 1, the overall operation is displayed in a simple diagram showing the data flow from the sender's to the receiver's side. The sender translates his data to l distinct symbols that are encoded to permutations and send over the network. The receiver receives every k packets and translates their order into symbols that are translated back to be read.

In our implementation, we send fabricated packets over IP using the *libnet* packet handling library. The sequence number of each packet is included in the payload explicitly. In a final implementation, these packets will be sorted by an application-transparent intermediate packet filter. This can take place as a hook to the protocol stack, or simply as an intermediate box that acts as a proxy for the sender's main machine. On the other side of the channel, the receiver is currently implemented as a sniffer process implemented over the *libpcap* packet capture library, and actual packets are received by a dummy application that just reads the stream of packets without any further processing. In the final implementation, the sender can receive the covert communication by essentially the same mechanism as the current implementation, while making use of the overt channel with any application level process (e.g., FTP client). It is important to mention that the current implementation shortcuts do not change the packet behavior in the network, nor the transmission capacity. Therefore, our implementation is a valid test-bed for the proposed covert channel.

The sender buffers every k packets and sends them in the order that encodes the covert channel input data. The process at the receiver (see Fig. 2) is more complicated due to the fact that some long span ordering might take place. This might cause codewords to interleave complicating the

Algorithm 1 Covert Channel Decode Packet

```

I ← Receive New Packet Index
curCW ← ⌊(I - indexStart)/k⌋
if curCW > codewordCount then
  codewordcount += (curCW - codewordCount)
end if
end if
CW[curCW] ← ((I - indexStart) mod k) + 1
while size(CW[⌊indexStart/k⌋]) = k do
  flush(CW[⌊indexStart/k⌋])
  indexStart ← indexStart + k
  codewordcount --
end while

```

Errors in codeword	0	1	2	3	4
prob. of errors	0.8732	0.0808	0.0376	0.0082	0.0002
CDF	0.8732	0.954	0.9916	0.9998	1.0

TABLE IV
ANALYSIS OF SER RATES IN A PACKET INDEX TRACE WITH 14%
REORDERING. CODEWORDS SENT USING $k = 4$.

decoding process. The receiver keeps track of the most recent incomplete codewords. Every received packet is added at the end of its corresponding codeword. These codewords are now represented only using their index, none of the packet's extra information is stored in this phase. Once a codeword has all of its packets received, it will be checked for errors (and corrected if needed and possible), and saved into the output queue. If the completed codeword is not the earliest needed, it is kept in the queue of incomplete codewords till all earlier codewords are completed, then flushed together into the output stream. These steps are demonstrated in Algorithm 1.

VII. EVALUATION

To evaluate the proposed covert channel, the main metrics to evaluate is the covert bandwidth per overt cost (e.g., packet, bit, or unit time), and the error rate of the covert transmission. The bandwidth can be calculated based on the channel parameters k , l , and the target reorder metric we have to stay below. This was addressed in Section V-D. The error rate on the other hand, need evaluation experiments to imitate the behavior over each codeword transmitted not the aggregate effect as given by previously mentioned metrics (e.g., RD , RBD , etc).

The channel's most important design parameter is k , the codeword size in packets. As in Table II, the effect of k on the maximum number of errors that can be detected is obvious. However, there shall be no need to go beyond the error correcting capabilities needed to successfully transmit through a specific channel, as that means a direct loss of bandwidth.

A single error in transmission is denoted by number of *Shift error Rate (SeR)* per codeword. *SeR* is equivalent to the single bit error rate used in ordinary channel analysis. It represents the probability by which a packet might shift one step. It can also be modeled as the probability of a single swap in position between two packets. An important property to verify is that multiple errors within the same codeword is a rare event. A trace with about 14% of out-of-order packets was used, and only the first 5000 packets were analyzed for this experiment. The results show a sharp decline in the number of errors within the same codeword (for this experiment, $k = 4$). In Table IV these results are shown. The RD of

code word size (k)	probability of undetected errors	probability of uncorrected errors
2	0.084654	0.084654
3	1.233E-03	0.019830
4	1.08 E-04	0.011152
5	3.50 E-05	0.007875
6	2.40 E-05	0.004656
7	7.00 E-06	0.000154

TABLE V
UNDETECTED ERRORS USING ONLY TWO CODEWORDS FOR TRANSMISSION ($l = 2$). THE TRACE USED HAS AN RD OF 0.877,0.114,0.0074,3.32E-4,1.64E-5.

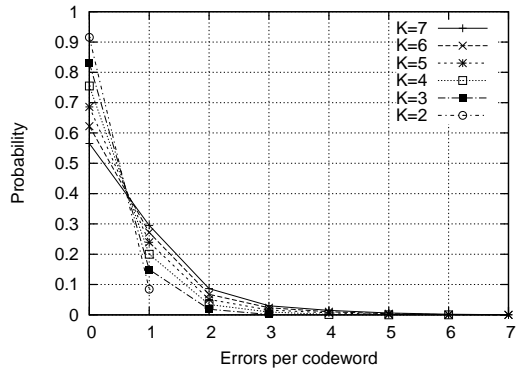


Fig. 3. Probability of different errors (SeR) for various values of K .

the trace used was $\langle 0.8545, 0.1343, 0.0105, 0.00054, \dots \rangle$. This specific trace was chosen for its heavy reordering where its out-of-order measures given even exceeds those provided in [15]. This ensures that our conclusions will be conservative estimates about the success of the channel in resisting external noise. The results show that if the code used is able to correct a single error (which is feasible for all $k \geq 3$) then 95.4% of codewords will be received successfully.

Another important dimension is how accurate can our transmission be if we sacrificed the bandwidth for maximum reliability. In other words, if we used only two permutations from the $k!$ possible ones, what will the percentage of undetected errors be. For the first few values of k ¹, Table V shows the percentage of errors goes undetected when applied for the same trace. Although the probabilities of undetected errors is considerably low, only detecting the error will require retransmission which is not possible in all cases. Obviously, the probability of errors that will go uncorrected (using the same trace and coding) will be higher. However, the values shown in Table V still show very high reliability despite the strict requirements of correcting errors rather than stopping at detection.

Figure 3 shows the effect of increasing k on the distribution of errors (SeR). Lower values of k give the highest probability of error-free codewords, but suffers low decay rate for the SeR distribution. Therefore, if moderate error-correction capability

¹From previous studies on extent of reordering, and its effect on TCP protocol buffers; values of $k > 10$ are not encouraged as it will severely degrade the stealthiness as well as the performance of the host application.

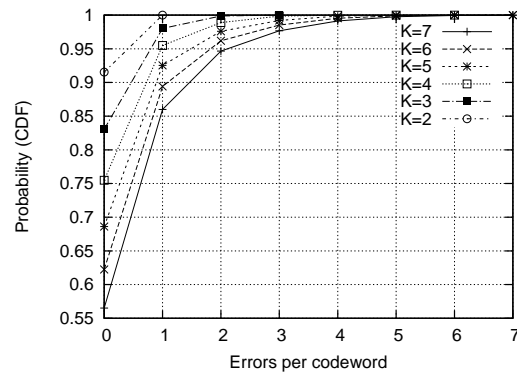


Fig. 4. Cumulative probability of different errors (SeR) for various values of K . $k = 2$ will not able to correct/detect any errors but is included here for completeness.

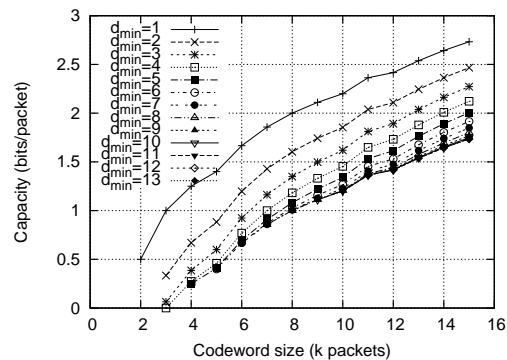


Fig. 5. Effect of increasing codeword size (k) while fixing minimum distance between codewords (d_{min}). Error detection: $d = E_D + 1$ or Error correction: $d = 2E_C + 1$

is provided, higher k 's will prove useful. The cumulative error distribution is shown in Figure 4, shows this effect. For example, If a code with $d_{min} = 3$ is used (feasible for $k \geq 3$), the reliability of the transmission increases to above 90% for $k = 3, 4, 5$ and increases to higher than 97% if 3 or less errors are corrected.

The effect of increasing the codeword size while fixing the error handling capabilities is presented in Figure 5. We can see that the capacity per packet can increase from almost no information transferred (~ 0 bits/packets) to as high as 2.5 bits/packet by increasing the value of k . Each curve represents a specific error capability represented in the minimum distance between codewords (d_{min}). On the other hand, the effect on increasing d_{min} on the capacity is shown in Figure 6. The drop is clear but gradual enough to give a wide range of valid design settings. For example, for $k = 6$ we can have capacity ranging from 1.7 down to 0.7 bits/packet in order to increase the error correction capability from 0 to 3 errors, or the detection capability up to 6 errors/codeword.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we propose a design for a covert channel that utilizes the packet order phenomenon as a mean for transmitting hidden signals. Packet ordering is a naturally occurring behavior in the Internet and it is to hard to be closely monitored and unlikely to raise suspicions. Moreover,

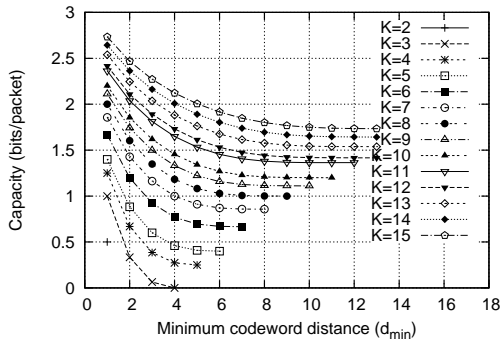


Fig. 6. Effect of increasing minimum codeword distance on the channel capacity per packet.

efforts to eliminate it will not be easily implemented due to the extremely high cost and the low incentive. Also, the main causes behind this phenomenon are not likely to vanish nor decrease in the near future.

Our proposed channel is designed based on the idea of assigning different symbols to the different permutations a number of consecutive packets can have. By using only a subset of the permutations, we were able to add error detection and correction capabilities. Subsets used are rotated to evade detection. The codewords themselves are selected based on the traffic characteristics to follow closely the innate reordering characteristics of the host channel. The outcome was almost identical to the target as given by the reordering metric (RD in our evaluation). Also, by varying the codeword input distribution to the channel, as well as the subset of permutations to use, our channel have the ability to be extremely robust and resilient to external reordering effects. The channel's built-in error detection/correction capabilities increased the correctly received codewords to more than 98% in typical operating parameters. The evaluation using packet traces showed error rates as low as 0.1% if the appropriate coding scheme is selected. Moreover, the bandwidth and coding efficiency are theoretically calculated and their bounds discussed. Capacity was shown to reach higher than 2 bits per packet while maintaining the capability of correcting packet shift errors.

Future Work

The coding of the channel can be redesigned to allow cyclic codes to be used for better error handling. Also, dynamic adjustment of channel parameters to accommodate network changes is an important addition for a resilient channel. Another aspect that is needed for full applicability is to add learning capabilities to the initialization process in order to know the properties of the host connection. Also, having more than one technique so that the sender can select the most optimal one for the target channel will enhance the overall bandwidth as well as stealthiness of the covert communication. Using hybrid metrics of packet reordering to better evade detectors will be a plus. Moreover, investigating the effect of such techniques on the application layer performance is needed to make sure the channel can go unnoticed as desired. Another factor to be evaluated is the source of the reordering and the effect it will have on the channel success.

REFERENCES

- [1] Kamran Ahsan. Covert channel analysis and data hiding in tcp/ip. Master's thesis, School of Electrical and Computer Engineering, University of Toronto, 2002.
- [2] Venkat Anantharam and Sergio Verdú. Bits through queues. *IEEE Transactions on Information Theory*, 42, 1996.
- [3] Colin M. Arthur, Andrew Lehane, and David Harle. Keeping order: Determining the effect of tcp packet reordering. In *ICNS '07: Proceedings of the Third International Conference on Networking and Services*, page 116, Washington, DC, USA, 2007. IEEE Computer Society.
- [4] Tarun Banka, Abhijit A. Bare, and Anura P. Jayasumana. Metrics for degree of reordering in packet sequences. In *LCN '02: Proceedings of the 27th Annual IEEE Conference on Local Computer Networks*, pages 333–342, Washington, DC, USA, 2002. IEEE Computer Society.
- [5] J. Bellardo and S. Savage. Measuring packet reordering. In *ACM SIGCOMM Internet Measurement Workshop*, 2002.
- [6] Jon C. R. Bennett, Craig Partridge, and Nicholas Shectman. Packet reordering is not pathological network behavior. *IEEE/ACM Trans. Netw.*, 7(6):789–798, 1999.
- [7] Kevin Borders and Atul Prakash. Web tap: detecting covert web traffic. In *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security*, pages 110–120, New York, NY, USA, 2004. ACM Press.
- [8] Serdar Cabuk, Carla E. Brodley, and Clay Shields. Ip covert timing channels: design and detection. In *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security*, pages 178–187, New York, NY, USA, 2004. ACM Press.
- [9] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & sons, 1991.
- [10] Steven Gianvecchio and Haining Wang. Detecting covert timing channels: an entropy-based approach. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 307–316, New York, NY, USA, 2007. ACM.
- [11] John Giffin, Rachel Greenstadt, Peter Litwack, and Richard Tibbetts. Covert messaging through tcp timestamps. In *Privacy Enhancing Technologies*, pages 194–208, 2002.
- [12] Wei-Ming Hu. Reducing timing charmers with fuzzy time. *sp*, 00:8, 1991.
- [13] Christian Krätzer, Jana Dittmann, Andreas Lang, and Tobias Kühne. Wlan steganography: a first practical review. In *MM&Sec '06: Proceeding of the 8th workshop on Multimedia and security*, pages 17–22, New York, NY, USA, 2006. ACM Press.
- [14] Vern Paxson. End-to-end routing behavior in the Internet. *IEEE/ACM Transactions on Networking*, 5(5):601–615, 1997.
- [15] Nischal M. Piratla and Anura P. Jayasumana. Metrics for packet reordering - a comparative analysis. *International Journal of Communication Systems*, 21(1):99–113, March 2007.
- [16] Nischal M. Piratla, Anura P. Jayasumana, and Abhijit A. Bare. Reorder density (rd): A formal, comprehensive metric for packet reordering. In Raoof Boutaba, Kevin C. Almeroth, Ramo'n Puigjaner, Sherman X. Shen, and James P. Black, editors, *NETWORKING*, volume 3462 of *Lecture Notes in Computer Science*, pages 78–89. Springer, 2005.
- [17] Nischal M. Piratla, Anura P. Jayasumana, Abhijit A. Bare, and Tarun Banka. Reorder buffer-occupancy density and its application for measurement and evaluation of packet reordering. *Comput. Commun.*, 30(9):1980–1993, 2007.
- [18] N.M. Piratla and A.P. Jayasumana. Reordering of packets due to multipath forwarding - an analysis. In *ICC'06: IEEE International Conference on Communications*, volume 2, pages 829 – 834, 2006.
- [19] Niels Provos and Peter Honeyman. Hide and seek: An introduction to steganography. *IEEE Security and Privacy*, 1(3):32–44, 2003.
- [20] Carla Savage. A survey of combinatorial gray codes. *SIAM Rev.*, 39(4):605–629, 1997.
- [21] Gaurav Shah, Andres Molina, and Matt Blaze. Keyboards and covert channels. In *USENIX-SS'06: Proceedings of the 15th conference on USENIX Security Symposium*, pages 5–5, Berkeley, CA, USA, 2006. USENIX Association.
- [22] Sang H. Son, Ravi Mulkamala, and Rasikan David. Integrating security and real-time requirements using covert channel capacity. *IEEE Transactions on Knowledge and Data Engineering*, 12(6):865–879, 2000.
- [23] S. Voloshynovskiy, F. Deguillaume, O. Koval, and Thierry Pun. Information-theoretic data-hiding: Recent achievements and open problems. *International Journal of Image and Graphics*, 5(1):1–31, 2005.
- [24] Sviatoslav Voloshynovskiy, Frédéric Deguillaume, O. Koval, and Thierry Pun. Information-theoretic data-hiding for public network security, services control and secure communications. In *6th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Services (TELSIKS) 2003*, Nis, Yugoslavia, October 1-4 2003.