Practical Data Hiding in TCP/IP

Kamran Ahsan
Government of Ontario
123 Edward Street
Toronto, Ontario Canada
+1 416 212-3100
kamran.ahsan@cbs.gov.on.ca

Deepa Kundur
University of Toronto
10 King's College Road
Toronto, Ontario Canada
+1 416 946-5181
deepa.kundur@utoronto.ca

ABSTRACT
This work relates the areas of steganography, network protocols and security for practical data hiding in communication networks employing TCP/IP. Two approaches are proposed based on packet header manipulation and packet ordering within the IPSec framework. For the former the Internet protocol IPv4 header is analyzed to identify covert channels by exploiting redundancy and multiple interpretations of protocol strategies; by passing supplementary information through IPv4 headers we demonstrate how security mechanisms can be enhanced in routers, firewalls, and for services such as authentication, audit and logging without considerable additions to software or hardware. For the latter approach, we show the use of packet sorting for steganographic embedding with IPSec can allow for enhanced network security.

Keywords
Data hiding, steganography, covert channels, TCP/IP, network security, toral automorphism, IPSec, packet sorting.

1. INTRODUCTION
Modern computer networks, such as the Internet, are designed for communication, connectedness and collaboration; their specification is "open" (i.e., publicly available) which presents difficulties with respect to security. As the Internet permeates our daily lives, there is an immense need to address issues of protection; flexible security for evolving network applications is required. This work attempts to integrate traditional network security with another emerging technology, *data hiding*. We first identify covert channels in TCP/IP, and then suggest application scenarios in which we make use of the supplementary bandwidth to enhance network security for current computer networks.

As described in [1], a covert channel is a communication link between two parties that allows one individual to transfer information to the other in a manner that violates the system's security policy. Covert channels are classified into *covert storage channels*, in which one transfers information to another by writing to a shared storage location, and *covert timing channels*, in which one signals information to another by modulating temporal system resources. We focus on covert channels in computer networks for which data hiding takes place by making use of the network packet streams as the cover object. Since these packets traverse different network topologies and are shared at various network nodes before reaching their intended destination, we take into account network behavior for data hiding design.

The authors wish to thank the support of Bell University Labs (BUL) and the Nortel Institute for Telecommunications (NIT) of the University of Toronto for providing funding for this work.

Workshop Multimedia and Security at ACM Multimedia'02, December 6, 2002, Juan-les-Pins on the French Riviera.

1.1 Framework
We assume that our communicating parties denoted *Alice* and *Bob*, transfer information *overtly* over a computer network and employ data hiding involving the TCP/IP protocol suite to *covertly* communicate supplementary information. Data hiding is employed through a *stego-algorithm* that takes as input the covert message C_k, a sequence of network packets $\{P_k\}$ known as the *cover-network packet sequence* and possibly a secret key to generate a *stego-network packet sequence* $\{S_k\}$ that contains the overt payload of $\{P_k\}$ while piggy-backing C_k. The stego-network packet sequence $\{S_k\}$ is sent to Bob over a computer network. The covert message C_k traverses a generally non-ideal channel characterized by the network behavior on the packet S_k. Keeping in view a model of the channel, Bob deciphers the covert message to produce C^*_k.

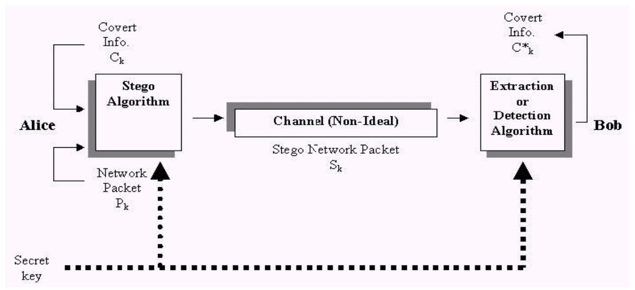


Figure 1. The general covert channel framework in TCP/IP

relevant points below:

- For reasons of security, Alice and Bob may employ the use of a symmetric secret key, so that only Bob can detect the information accurately. In addition, for the packet sorting approach, Alice and Bob are both assumed to implement IPSec.

- The stego-network packet, S_k, may pass through one or more intermediate network nodes in order to reach Bob. The covert channel, by definition, must be non-detectable by these nodes which can be ensured if and only if the intermediate node finds no critical difference between $\{P_k\}$ and $\{S_k\}$ while processing the sequence. The reader should note that we define "non-detectability" with respect to standard network node capability and not a human observer.
- At an intermediate node, a stego-network packet S_k may be dropped due to the non-availability of buffer capacity. However, this possibility is assumed to be nonexistent in our analysis of the proposed algorithms. We are focusing on that network traffic which is most unlikely to be dropped due to buffer unavailability. Such a condition is possible, if we consider QoS mechanisms through which network traffic can be prioritized as a preferred class. In addition, we assume there is remote possibility that the same stego network packet is corrupted during transmission and consequently be dropped by the data link layer mechanisms.

The next section summarizes our work in comparison to the previous contributions in the area of data hiding in computer networks to provide perspective for this paper.

1.2 A More Complete Picture

Our research can be considered, in part, a logical extension to [2] and [3] in which Handel & Sandford and Wolf propose the reserved or unused fields in packet headers for data hiding; we adopt more specific approaches. The proposed data hiding scenarios are more practical and robust since they are based on redundancies and multiple interpretations of process strategies of the Internet protocol. This makes the scenarios non-detectable to various automated security mechanisms and intermediate nodes.

This research also expands on the work presented in [4] as concrete steganographic encoding and decoding techniques are suggested for interoperability with network security mechanisms such as firewalls and routers. The header fields we select for data hiding are flexible for sending covert data without interfering with standard network processes while being accessible to routers and firewalls. In contrast, [4] suggests the use of the TCP sequence number and acknowledgement number fields, which reflect the number of bytes sent and acknowledged. These fields, we believe, are therefore inflexible for covert data transfer.

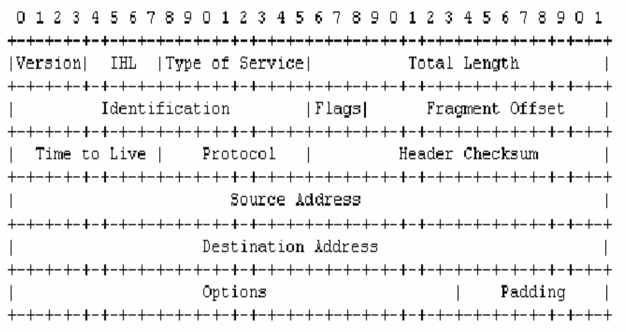
Overall, we attempt to provide improved data hiding approaches based on several interesting papers in the area. Moreover, we take into account, in part, the effect of the network on the stego-network packet S_k, and provide novel applications scenarios utilizing concepts proposed by Ackermann *et al.* [5]. Together, we hope to present a more complete picture of the data hiding in computer networks. The remainder of the paper details our proposed data hiding techniques.

2. PACKET HEADER MANIPULATION

In designing our data hiding approaches, we consider functional interfaces required in all IP implementations; a summary of these interfaces is found in [6]. In this way, we avoid identifying covert channels specific to particular TCP/IP software implementations, which are not available for general use.

The reader should note that the layered structure of networks requires the IP datagram to encapsulate information received from the transport layer. For example, IP headers encapsulate ICMP messages and IGMP report and query messages. Covert channels in the IPv4 header can, therefore also, be associated with those identified in the TCP, ICMP or IGMP headers.

We present two data hiding scenarios in this section. Each technique makes use of redundancy in the representation of information in the Internet Protocol for effective data hiding.



```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2. The IPv4 Header

2.1 Data Hiding Scenario 1

A close study of [6] reveals that there exists redundancy in the Internet Protocol's fragmentation strategy. Figure 2 displays the IPv4 header. The Flags field contains fragmentation information. The first bit is *reserved,* the second is denoted *DF* (to represent Do not Fragment), and the third is denoted *MF* (to represent More Fragment). An un-fragmented datagram has all zero fragmentation information (i.e. MF = 0 and 13-bit *Fragment Offset* = 0) which gives rise to a redundancy condition, i.e. DF (Do not Fragment) can carry either "0" or "1" subject to the knowledge of the maximum size of the datagram. This aspect is exploited in Data Hiding Scenario 1.

Consider two workstations on the same network with users Alice and Bob who have decided to have a covert communication employing the protocol suite of the network. They are aware that the network administrator is very security cautious and the TCP/IP software is configured properly as per the security policy of the organization. Alice and Bob have knowledge of the MTU (maximum transmission unit) of their network and are aware of the fragmentation strategy, which follows the standard design considerations of IP [6].

Based on the above explanation, it can be shown the datagrams in Tables 1 and 2 bear the same meaning to the overt network provided that Alice and Bob have the MTU information beforehand. Thus, this redundancy leads to the possibility of covert information through judicious selection of each representation. Datagrams 1 and 2 sent by Alice can therefore communicate "1" and "0" respectively to Bob. The constraint, however, is that both parties require prior knowledge of the MTU.

Table 1. Datagram 1 Covertly Communicating '1'

Datagram 1	16-bit Id. field	3-bit flag field	13-bit frag. offset	16-bit Total len.
1	XX…XX	0 **1** 0	00...00	472

Table 2. Datagram 2 Covertly Communicating '0'

Datagram 2	16-bit Id. field	3-bit flag field	13-bit frag. offset	16-bit Total len.
1	XX…XX	0 **0** 0	00...00	472

To demonstrate how both datagrams are similar from the perspective of a network, we note that Datagram 1 is of moderate length, but fragmentation is not allowed since the DF bit is set. Datagram 2 of the same length has the fragmentation bit unset, yet fragmentation is not possible since it is below the value of the MTU. Since Alice and Bob know the MTU of their network and have agreed to send a datagram of size smaller than MTU there will be no fragmentation. Details can be found in [8].

2.2 Data Hiding Scenario 2

Data Hiding Scenario 2 involves the 16-bit identification field of the IPv4 header shown in Figure 2, through *chaotic mixing* (*toral automorphism systems*). This identification field carries a value assigned by the sender to aid in assembling the fragments of a datagram at the receiver. The only limitation on the identification field by the fragmentation strategy is that it is unique for a specific source-destination pair as long as the datagram is alive on the Internet [6].

2.2.1 Toral Automorphism Systems

Toral Automorphisms are strongly chaotic (mixing) systems [7] that have found application in digital image watermarking. A two-dimensional toral automorphism is a spatial transformation applied to two-dimensional square planar regions. The main mathematical structure in this process is a 2×2 matrix **A**, consisting of constant elements representing the toral automorphism mapping, of the form:

$$\begin{bmatrix} 1 & 1 \\ k & k+1 \end{bmatrix}$$

and constrained to have elements that belong to the set of positive integers and a determinant of 1. The latter property ensures the existence of the inverse automorphism represented by $\mathbf{A}^{-1}$, the inverse matrix. The iterated application of **A** on a point *r* (belonging to integer lattice **L** and having coordinates as *x* and *y*), result in a dynamical system, that can be expressed as follows:

$$r_{n+1} = Ar_n \pmod 1),\ n = 0,1,2,.... \quad \mathbf{(2.1)}$$

The dynamical system pertaining to the complete lattice **L** can be obtained by having all the points on the lattice subjected to iterated actions of the toral automorphism matrix, **A**. Therefore, we can represent this as $\mathbf{A}_N(k)$: $\mathbf{L} \longrightarrow \mathbf{L}$ and

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ k & k+1 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix} (\bmod N) \quad \mathbf{(2.2)}$$

It can be shown that the transformed points obtained at each iteration are statistically uncorrelated with each other and depend on the parameters *k* and *N* where *N* is the dimension of the lattice [9]. Moreover these dynamical systems are *periodic* in nature and are therefore capable of returning to their initial state after a specified number of iterations. The reader should note that the transformation of all the points in the lattice takes place in such a way that the lattice representing a digital image, for example, appears as a completely deformed image having no visual relation with the original image. The transformation, thus, has a random nature if the parameters are unknown. This is useful for scrambling as we will see in our application to data hiding in the IPv4 header identification field.

2.2.2 Generation of Sequences

Consider a two-dimensional diagonal integer-lattice (i.e., the diagonal points are the only ones considered in the iterated application of **A**). For our use, the size of the defined lattice is represented by *K* to distinguish it as being a design variable (instead of *N* which was fixed as described above) and is defined to be the *main key*. This main key *K* determines the number of elements in a generated sequence. The parameter *k* as shown in (2.2) is defined as the *sub-key* in our scenario and affects the period of the iterated transformation. It can be shown that given the main key *K*, choosing a specific *k* generates a family of sequences that can be generated through the repeated iterative application of the toral automorphism [9]. The main and sub-keys, dictate the number of possible unique sequences that can be obtained (as the transformation is applied in succession) which we term *sorted sequences* of the original sequence. We can select a single *unique sequence* from the set of sorted sequences; this selection takes place with the use of a third key (which is to the number of times the toral automorphism is applied).

Therefore, once the main key, the sub-key and the third key are set, one can generate a specific sequence from an initial. To summarize, the main key determines the number of elements in a sequence, the sub-key the total number of possible unique sequences generated through iterated application of the toral automorphism, and the third key a specific sequence out of that total number of sorted sequences obtained. Thus, given all three keys one can map from an initial sequence to another sequence.

2.2.3 Scenario

The strength of a data hiding scheme depends on its non-detectability either by the administrator or by any automated network-monitoring scheme; its identification field appears to be perfectly "normal". Chaotic mixing provides structured scrambling and enables Alice and Bob to perform the following operations, in line with the framework detailed in Section 3. Alice and Bob both have prior knowledge all three secret keys discussed in the previous section. For the purpose of demonstrating our scheme, we let the data to be communicated covertly C_k be the capital letters of the English language; this requires that the choice of *K* and *k* be such that there are at least 26 elements in a sorted sequence.

2.2.3.1 Alice's End

Alice performs the following operations to encode a covert data symbol (i.e., letter of the alphabet):

1. Alice selects a specific sorted sequence from the keys: *K* and *k* and the third key (which may be dynamic for every covert symbol).
2. A look-up table is formed whereby each element of the sorted sequence from Step 2 is matched up with each one of the capital letters of the English language.
3. The resulting table is used to map each letter to an 8-bit binary equivalent.
4. By appending another independent randomly generated 8 bits to the result of the previous step, the 16-bit identification field of IPv4 header is formed such that it is unique and compatible for the IPv4 identification field.

2.2.3.2 Bob's End

To decode, Bob generates the look-up table with the information of all the keys. By looking at the identification field of the received packet, Bob can easily decipher the covert information being sent by Alice.

We present an example of the technique below. Suppose *K*=26 and *k*=1; the total number of sequences will be 42. Selecting *sequence number* as 8, the following table provides information regarding the identification field generation of IPv4 header:

Table 3. Generation of Identification Field

Alphabets	Seq. for 8th iter.	Binary Equ.	4-bit Encoding	Ident. field
A	1	0000 0001	01	0 1 X X
B	14	0000 1110	0 E	0 E X X
…	…	…	…	…
Y	6	0000 0110	0 6	0 6 X X
Z	19	0001 0011	1 3	1 3 X X

Table 3 shows an example of identification field generation based on the toral automorphism. The second column represents the 8th sorted sequence of the possible 42 sorted sequences based on *K* as 26 and *k* as 1. The third column represents the binary equivalent of the second column. Consequently, the fourth column expresses the binary equivalent in 4-bit form. The last column of Table 1 represents the identification field value of the IPv4 header.

Our method of using the identification field of the IPv4 header through the generation of uncorrelated sequences does not require that Alice and Bob to be on the same network; they could communicate across the Internet as well. The randomness in the identification field values makes this scheme non-detectable against the detection of secret data through packet filtering and stateful inspection type firewalls. The covert data is *scrambled* in an efficient way, so that statistical cryptanalysis and traffic analysis cannot, in general, be easily automated to decipher the information.

2.3 Potential Applications

Associating supplementary information sent via covert mechanisms employing packet header manipulation algorithms find the following application scenarios:

1. Enhanced filtering criteria in packet filtering routers (firewalls). If the additional information pertains to a user or an application, a more reinforced filtering policy can be defined.
2. A client server architecture wherein several clients make a request to the FTP server, say of a library. A log file can be maintained, for audit purposes, based on the requests sent by various users. Moreover, serving the request by transferring a digital image to the user, say, can have the same user information or library information tied to the content packets. This scenario of tags tied to the content can allow for audit.
3. A logging process for the above application scenario based on the user or application specific-information completes the picture (i.e. logging of valid user), maintaining the record of user requests based on user information and ultimately serving the user requests by having either the user information or the server / source (library) information tied to the content packets to avoid unlawful use such as copyright violation.
4. Adding value to content delivery networks. A content delivery network is an overlay network to the public Internet or private networks, built specifically for the high performance delivery of content. Use of supplementary covert data adds intelligence to networking wherein the network makes path decisions based on more than simple labels such as IP address.

3. DATA HIDING BY PACKET SORTING

In this section, we describe, in general terms, the use of packet ordering for covert communication. There are *n*! ways in which to arrange a set of *n* objects. If the order of these objects is not of concern, then there is an opportunity through judicious selection of an arrangement of the *n* objects to covertly communicate a maximum of $\log_2(n!)$ bits. For a set of n=25 objects, for instance, 83.7 bits can be communicated. The capacity of data hiding increases dramatically for large n.

Applying this principle to our framework, we consider data hiding through network packet ordering. Modifying the order of packets requires no change of the packet content (i.e., the payload and the headers are not affected). Therefore, no major modifications are expected either in the protocol definition, design, or in the overall system. Based on these advantages, the feasibility of data hiding using packet sorting within the TCP/IP protocol suite is explored.

The packet sorting and resorting processes require a reference in order to relate packet numbers to their actual *semantic* order. The natural packet ordering of the cover-network packet sequence is needed so that the stego-network packet sequence ordering (sorting) can be undone (resorting) to extract out the covert and overt information within the packets, which we assume to be of use. The 32-bit *sequence number field* of the *authentication header (AH)* and *encapsulating security payload (ESP)* in the IPSec protocol provides information on natural ordering of the packet stream and is, therefore, utilized for data hiding. The primary objective of the identification field is to detect replay attacks and is, therefore, directly related to packet numbers and ordering.

Unlike the packet header manipulation approach, the cover object is a sequence of network packets $\{P_k\}$ to embed one covert message symbol rather than a single packet P_k. The stego-algorithm sorts the original cover-network packet sequence resulting in the generation of "sorted" packets having the sorted sequence numbers in their headers. The stego-network packet sequence $\{S_k\}$ may pass through one or more intermediate network nodes in order to reach Bob. Since the network cannot guarantee proper sequencing for packet delivery, the transmission process is modeled as a non-ideal channel characterized by the *position error(s)* imposed by the practical network behavior.

3.1 Sorting and Resorting Algorithm

A block diagram of the embedding and detection processes is shown in Figure 3 in which we see that Alice and Bob both implement IPSec. We, once again, borrow the concept of toral automorphism, to aid in our data hiding scenario. The sorting (i.e., embedding) and resorting (i.e., detection) algorithms are described below. Alice and Bob are both assumed to share a secret symmetric key prior to covert transmission consisting of the *main key* and *sub-key* as previously described. In contrast to the header manipulation scheme, the *third key* is the actual covert data sent and, therefore, only selected by Alice, and estimated by Bob.

3.1.1 Alice's End

1. Alice uses the three keys in order to generate the stego-network packet sequence (i.e., sorted sequence) from her end; the three keys uniquely determine the toral automorphism transformation as well as the number of times it is applied to the original order of the cover-network packet sequence to generate the sorted stego-network packet sequence. The main key K is from the set of positive integers and determines the number of elements in the sequence to be sorted. The sub-key k is a parameter of the toral automorphism matrix **A** detailed in [9].

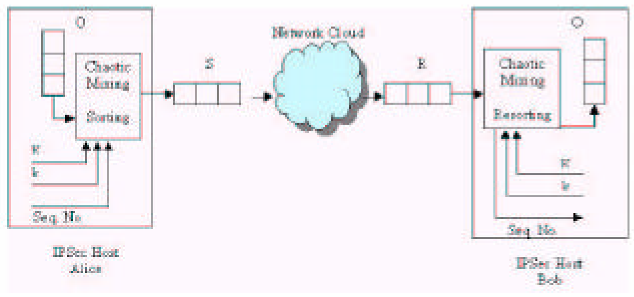


Figure 3. The block diagram of sorting and resorting process

a) Alice takes the covert data to be transmitted C_k and sets it as the third key which denotes the number of times **A** is applied to determine the particular stego-network packet sequence.

2. Alice physically assigns new-sorted sequence numbers to the data packets and transmits them.

3.1.2 Network Behavior

In an ideal network there is no physical change in order that packets are received; this is called a what-is-sent-is-what-is-received (WISIWIR) environment. However, a real network results in out of order delivery of packets, which is analyzed later in this paper.

3.1.3 Bob's End

1. Based on the knowledge of K and k knowledge, Bob generates all the possible valid sequences.
2. From the received sequence, Bob estimates the closest possible valid sequence transmitted (i.e., the third key in the toral automorphism process) to determine C_k*.

In the remainder of Section 3, we detail our embedding process under a WISIWIR environment. In Section 4, we relax this condition for more practical networks.

3.2 Algorithm Details

The embedding process takes a original packet sequence, O, as its input and is expressed as:

$$O = X_1 X_2 X_3 \ldots X_K \qquad \textbf{Equation 3.1.}$$

where K is the main key and X_i, for i = 1, 2, 3, 4,….,K, represents the ith packet of the packet sequence to produce the sorted packet sequence S. Referring to the periodic nature of toral automorphism systems [10], we know that after a specific number of iterations, N, all the points in integer lattice come back to their initial locations. The total number of unique sequences is represented as:

$$P = S(1), S(2), S(3), \ldots, S(N) \qquad \textbf{Equation 3.2}$$

where P is the set of possible sorted sequences generated from the chaotic sequence structure and $S(i)$ is the ith valid sequence. The reader should note that the chaotic mixing structure does not generate all possible permutations of the set of K packets and hence $N \neq K!$. To transmit a covert symbol, we effectively select the appropriate sequence from the set P.

The sorted packet sequence, as a result of chaotic mixing process, is therefore:

$$S(i) = X'_1 X'_2 X'_3 ..... X'_K \qquad \textbf{Equation 3.3}$$

where $S(i) \in P$ and i = 1, 2,….,N. Here X'_j , for j = 1, 2, 3, 4…K, represents the jth packet of the packet sequence. For packet sequences having more than K packets in their sequence, the sorting process follows [11]:

$$S(X'_{(M-1)K+x}) = (M-1)K + O(x);\, x \le K \qquad \textbf{Equation 3.4}$$

Here M represents the multiple of K, therefore if the packet lies within the first multiple of K, then M would be 1 and likewise 2 for second multiple and so on, and $O(x)$ represents the original sequence packet either equal to or less than the main key K. The received sorted sequence, R can be expressed as:

$$R = X''_1 X''_2 X''_3 .... X''_K \qquad \textbf{Equation 3.5}$$

At Bob's end, given K and k the received sequence, R passes through the resorting process to decipher the covert information, C_k. For WISIWIR, the received sequence can be perfectly estimated to be C_k. For the packet sequences having packets more than K, the resorting process follows [10]:

$$R(X''_{(M-1)K+x}) = (M-1)K + O(X''_{(M-1)K+x} - (M-1)K);\, x \le K \qquad \textbf{Equation 3.6}$$

4. Best Sequence Estimation in Resorting

In this section we relax our WISIWIR assumption. There is no guarantee that packets take specific routes. The Internet layer offers best effort connectionless delivery mechanism [6] and hence treats every packet individually. Packets thus experience varying levels of latency as they progress through the network.

Given this non-ideal nature of overt communication channel, we model the network behavior of introducing position errors (PEs) in a sequence of packets at the network layer. Internet packet dynamics are studied and analyzed in [11] and [12]. Paxson's work [11] involves a number of sites that ran special measurement daemons to estimate of various network parameters including out of order packet delivery. The analysis established the following characteristics of out of order receipt of packets:

- Out of order delivery is fairly prevalent in the Internet.
- Overall, 2% of all the first run data packets and 0.6% of the ACKs arrived out of order, whereas for the second run the percentages are 0.3% and 0.1% respectively.
- Reordering only rarely has significant impact on TCP performance since generally the scale of reordering is just a few packets. Reordering some times occurs in groups as large as dozens of packets, it usually involves only one or two packets.

Based on Paxson's findings [11], we focus on small scale reordering to model our non-ideal covert channel. We consider small scale reordering to be in the order of two, three or four position errors. Taking this into account, our covert data estimation process attempts to map the received sequence to an appropriate sent sequence. This process is inspired by [13] and detailed in the next section.

4.1 The Longest Subsequence (LSS) Method

To estimate the covert data the received sequence R''_x is compared to each possible candidate sent sequence $S(i)'_x$.

The *longest subsequence (LSS) method* consists of the following steps (the reader is referred to Ref. [8] for further details):

1. Compute the number of position errors (PEs) between the received and candidate sent sequence. We do a point-wise subtraction of each packet number between the received and candidate sent sequence. Through this subtraction, we identify locations where the position difference is zero (i.e., the packet locations match up for the two sequences). The number of PEs is obtained by subtracting number of zeros from the main key value, K.
2. If the PE is less than or equal to a threshold (depending on the assumed network behavior), then proceed. Otherwise term the candidate an *impossible sequence*.
3. Subject the received sequence to a right shift absolute subtraction (RSAS) [8]. Step 1 is repeated.
4. Truncate the last packet of the sent sequence and the first packet of the received sequence. Perform RSAS and identify and count the number of zeros.
5. Repeat Step 4 by continuing to shift until the first packet of the sent sequence undergoes RSAS with the last packet of the received sequence. For each one of these steps, identify and count the number of zeros (i.e., the number of places the positions of the received and candidate sequences match up).
6. Count the total number of zeros resulting from Steps 3 to 5.
7. Determine a *resultant subsequence* from the positions in the sequences where zeros are identified using the process identified in [8].
8. If the resultant subsequence conforms to both the received and candidate sequences then the subsequence is termed the *best estimate sequence*.
9. Otherwise the received sequence undergoes RSAS one more time. Subtract one from the total number of resulting zeros from this Step 6.
10. Repeat Steps 3 to 7 until the number of zeros in every step is counted, added to the previous and becomes equal to the resultant number of zero as worked out in Step 9.
11. Determine the corresponding resultant subsequence as in Step 7.

12. If the resultant subsequence does not conform in both the received and candidate sequences, then the received sequence is classified as an *error sequence.*
13. Otherwise it is termed a *longest subsequence* specific to the received candidate sequence pair.

The process is conducted for all candidates sent sequences. For each of the longest subsequences, the one having greatest number of zeros is termed the *longest subsequence* which provides an estimate of the transmitted sequence and hence the covert data. An *error condition* may arise when two longest subsequences have equal number of zeros. Once again, due to space limitations, the reader is referred to Ref. [8] for a more comprehensive description. Our goal in this paper is to provide a succinct description of the process.

4.2 Simulation and Testing

The process is simulated for different values of K ranging from 4 to 8 (inclusive), and k=1. The results are also valid for packet sequences that are multiples of these main key sizes. A practical communication network causing 3 to 6 (inclusive) position errors is simulated.

To provide an exhaustive and conservative measure of the success of our scheme in a practical network, we look at the decoding error rates of the covert data for all possible received packet sequences; thus, for a set of n packets, we identify the error rates for all n! possible permutations. This can be shown to provide an upper-bound on the error rates for standard network behavior.

For example, a 4-packet sequence has 24 permutations. Each one of the 24 permutations is treated as a received sequence. Each one of these received sequences then undergoes the RSAS process with each one of the 2 possible valid sequences sent (for K=4 and k=1, there will be only 2 sorted sequences and, hence, one bit of covert data can be sent). The best estimate process as detailed in Section 4 is then applied to decode the covert data.

From covert communication perspective, we would like to correctly decode the covert data. Naturally, *impossible sequences* are not considered and *errors* are not required. The dominance of these ignored sequences reduces the number of sequences mapped into one of the other three received sequence categories. The mapping of the received sequence to either an *evident sequence* or as *LSS based best estimate sequence* is therefore highly desirable for parties involved in covert communication. As per the framework detailed in Section 2, the non-ideal behavior of the network is considered in the form of introducing position errors (PEs) in the packet sequences. For each one of the position error scenarios considered (3 position error to 6 position error), the simulations show the availability of the desirable categories in percentages. Moreover simulations also point out which sent sequence is most likely to be mapped at the receiving end. Table 4 shows that sequence 3, $S(3)$, is most likely to be mapped.

The remaining position error scenarios as resulted from simulations are analyzed in the same way.

4.2.1 *Mixed Network Behavior*

Assuming network as making mixed position errors randomly, Table 5 shows that the percentage of those received sequences which would be mapped to the same valid sequence is quite significant i.e. 42% for 4-packet sequences (2 and 3 PEs) and 35% for 5-packet sequences (3 and 4 PEs).

Overall, as detailed in Ref. [8] and summarized in this paper, the simulations suggest that the use of packet sorting is potentially viable for data hiding in computer networks.

4.3 Usage Scenarios

The data hiding process employing the packet sorting technique in the IP Sec environment finds following applications:

1. Preliminary (added) authentication in the IP Sec environment.
2. A mechanism to facilitate enhanced anti-traffic analysis by having packets with sorted sequence numbers.
3. Enhanced security mechanisms for IP Sec protocols; ESP operating in tunnel mode, which enables enriched security for virtual private networks (VPNs).

The details of these scenarios will be a topic of another paper.

5. Conclusions

This paper presents two practical data hiding approaches in the TCP/IP protocol suite. The techniques look at IPv4 header manipulation and packet ordering in an IP Sec environment to pass supplementary information through covert channels.

We demonstrate how IPv4 header manipulation can be used to pass supplementary information over the Internet. We present two practical data hiding techniques for TCP/IP based on fragmentation strategies and the identification field; the identification field value is independent of the fragmentation strategy in the Internet Protocol, making both takes jointly viable. The existence of these covert channels may be useful in enhancing several basic network functions such as router filtering, auditing and logging processes.

We have also presented a practical algorithm for packet sorting and resorting based on toral automorphisms. Our development is ad hoc in order to provide a proof-of-concept regarding the viability of using packet sorting in practical networks. Future work will look at formal development of a theory for packet sorting using coding strategies taking into account a position error model of the network behavior to guarantee optimal covert communication capacity.

6. REFERENCES

[1] "Department of defence trusted computer system evaluation criteria," Tech. Rep.DOD 5200.28-ST, Department of Defence, December 1985. Supersedes CSC-STD-001-83.

[2] T. Handel and M.Sandford., "Hiding data in the OSI network model," (Cambridge, U.K.), *First International Workshop on Information Hiding*, May -June 1996.

[3] M. Wolf, "Covert channels in LAN protocols," *Proceedings of the Workshop on Local Area Network Security (LANSEC'89)*, pp. 91 – 102, 1989.

[4] C. H. Rowland, "Covert channels in the TCP/IP protocol suite," *Tech. Rep. 5*, First Monday, Peer Reviewed Journal on the Internet, July 1997.

[5] R. Ackermann, U. Roedig, M. Zink, C. Griwodz, and R. Steinmetz, "Associating network flows with user and application information," *Eight ACM International Multimedia Conference*, Los Angeles, California, 2000.

[6] U. S. C. Information Sciences Institute, "Internet protocol, darpa internet program , protocol specification," September 1981. Specification prepared for Defense Advanced Research Projects Agency.

[7] D. Arrowsmith and C. M. Place, *An Introduction to Dynamical Systems.* Cambridge University Press, 1990.

[8] K. Ahsan, *Covert Channel Analysis and Data Hiding in TCP/IP*, M.A.Sc. thesis, Dept. of Electrical and Computer Engineering, University of Toronto, August 2002.

[9] I. Pitas and G. Voyatzis, "Chaotic mixing of digital images and applications to watermarking," in European Conference on Multimedia Applications Services and Techniques (EMAST' 96), vol. 2, pp. 687–695, 1996.

[10] T.Shiroshita, O. Takahashi, and M. Yamashita, "Integrating layered security into reliable multicast," in IEEE Third International Workshop on Protocols for Multimedia Systems, 1996.

[11] V. Paxson, "End-to-end internet packet dynamics," in IEEE/ACM Transactions on Networking, vol. 7(3), pp. 277–292, 1999.

[12] D. J.Mogul, "Observing TCP dynamics in real networks," in SIGCOMM '92, pp. 305–317, August 1992.

[13] D. I. Pullin, R. Manderville, and A. Corlett, "On the packet ordering problem,". Presented to the IPPM working group in the 51st IETF meeting in London, U.K. , August 2001.

Table 4. K= 6 (imprvd.); S(4); Network Behavior: 3 PE or less

Recvd. Sequence Category	**Seq. 1 S(1)**	**Seq. 2 S(2)**	**Seq. 3 S(3)**	**Seq.4 S(4)**	**Total**
Impossible	-	-	-	-	541
Error	-	-	-	-	15
Evident	33	36	40	34	143
Best Estimate	5	5	4	7	21
Total	38	41	44	41	720

Table 5. Mixed Behavior of Network

Main Key (packet sequences)	**Mixed Position Errors**	**Common Sequences**	**%age of Common Sequences**
4-packet	2 and 3 PEs	10	42%
5-packet	3 and 4 PEs	42	35%