Implementing a Covert Timing Channel Based on Mimic Function

Jing Wang[1,2,3], Le Guan[1,2,3], Limin Liu[1,2,⋆], and Daren Zha[3]

[1] Data Assurance and Communication Security Research Center, CAS, Beijing, China
[2] State Key Laboratory of Information Security, Institute of Information Engineering, CAS, Beijing, China
[3] University of Chinese Academy of Sciences, Beijing, China
{jwang,lguan,lmliu}@lois.cn,
zhadaren@ucas.ac.cn

Abstract. Covert timing channel is a mechanism that can be exploited by an attacker to conceal secrets in timing intervals of transmitted packets. With the development of detection techniques against such channel, it has become increasingly difficult to exploit a practical covert timing channel that is both detection-resistant and of high capacity. In this paper, we introduce a new type of covert timing channel. Our novel encoding technique uses mimic functions as the basis to accomplish the mimicry of legitimate traffic behaviors. We also design and implement a mimicry framework for automatically creating this new type of covert timing channel. In the end, we utilize the state-of-the-art detection tests to validate the effectiveness of our mimicry approach. The experimental results show that the created covert timing channel can successfully evade the detection tests while achieving a considerable channel capacity.

Keywords: network security, covert timing channel, mimic function, detection resistance.

1 Introduction

Covert channel was first introduced by Lampson [15], and originally, studied in the context of multi-level secure systems. Ever since Girling started the study of covert channel in the network scenario [10], the security threat posed by network covert channel has attracted increasing attention. Network covert channel involves in a wide range of attacks, e.g., data exfiltration [18], DDoS attacks [12], privacy enhancement [13], and packet traceback [20].

Traditionally, network covert channel is classified into storage channel and timing channel. The former exploits the redundancy of network protocols, i.e., random/unused bits in packet header, while the latter manipulates inter-packet delays (IPDs) of network traffic. Storage channels can be discovered by observing the anomalies in the patterns of packet header fields. The detection of timing

⋆ Corresponding Author.

X. Huang and J. Zhou (Eds.): ISPEC 2014, LNCS 8434, pp. 247–261, 2014.

channels is usually based on statistical analysis of shape and regularity in packet timing intervals. Compared with storage channels, it is more difficult to detect timing channels due to the high variation in timing intervals. To thwart such channels, researchers have proposed substantial detection techniques [2,16,8], the focus among which is on using entropy-based method [8]. This promising method has been proven an effective way for detecting various covert timing channels.

In recent years, in order to evade detection, several works propose to create covert timing channels by mimicking the statistical properties of legitimate traffic [9,19,14]. In particular, Walls et al. [19] first revealed the entropy-based detection method can be defeated. Their approach uses a half of packets to smooth out the anomalies caused by covert traffic. As a consequence, the detection resistance results in a significant reduction in channel capacity. By far, it remains a challenge work to implement a covert timing channel that is both detection-resistant and of high capacity.

In this paper, we propose a new type of detection-resistant covert timing channel with a considerable channel capacity. Leveraging a stenographic technique—mimic function in the encoding method, covert traffic generated has the similar statistical properties with legitimate traffic. The statistical properties include the first-order and high-order statistics, which actually correspond to the shape of distribution and inter-packet dependencies in network traffic, respectively.

To construct practical covert timing channels, we also develop and implement a mimicry framework, which is intended for automatically mimicking, encoding, and transmitting. More specifically, the framework includes five phases: filtering, symbolization, modeling, encoding, and transmission. Through these phases, covert traffic which is statistically approximate to legitimate traffic is generated, and finally is transmitted to the Internet. The covert timing channel constructed by the framework can also be adjusted by certain parameters in line with different requirements in undetectability, error rate, and capacity.

In the end, we conduct a series of experiments to validate the effectiveness of our mimicry approach. Experimental results show that the covert timing channel built from the mimicry framework can successfully evade the entropy-based tests while achieving almost 3-6 times the throughput of Liquid [19].

The rest of this paper is organized as follows. Section 2 introduces the related work in covert timing channel. Section 3 describes mimic functions and our mimicry framework. Section 4 shows the experiment results on the effectiveness of our mimicry approach. Finally, Section 5 concludes this paper.

2 Related Work

The timing of network packets can be utilized to leak secret information. In [2], Cabuk et al. presented IPCTC, the first IP covert timing channel. IPCTC employs a simple on/off encoding scheme. During a specific timing interval, the sender transmits a 1-bit by sending a packet and a 0-bit by not sending a packet. Gianvecchio et al. [9] explored a model-based covert timing channel, which is named

MBCTC in short. MBCTC models distribution functions for legitimate traffic, and mimics its first-order statistics. In [18], Shah et al. proposed a novel passive covert timing channel, which can leak typed passwords over the network without compromising the host or creating additional traffic. Sellke et al. [17] proposed the "L-bits to N-packets" encoding scheme for building a high-capacity covert timing channel, and quantified the data rate of that scheme. Walls et al. [19] presented a detection-resistant covert timing channel relying on the idea of Jitterbug. The main idea is to insert shaping IPDs into covert traffic, so as to smooth out shape distortion generated by Jitterbug.

On the other hand, researchers have proposed various disruption and detection techniques to defend against those channels. Compared with disruption techniques, channel detection has two major advantages: 1) it has no effect on network performance; 2) it has an additional benefit that the hosts transmitting covert information can be located. Detection techniques are based on the fact that the creation of covert timing channels causes shape or regularity distortion in traffic's timing characteristics. The Kolmogorov-Smirnov test [16] is a nonparametric test that is used to determine whether a sample comes from a reference distribution or tell the difference between two samples. It has been experimentally proven that this test is able to sniff out abnormal traffic which distorts in shape. Cabuk et al. [2] investigated two detection tests, namely ϵ-similarity and regularity test. However, they are only effective to a minority of covert timing channels, because they are over-sensitive to the variation of traffic. Gianvecchio et al. [8] introduced a fruitful detection method using the combination of entropy and corrected conditional entropy, which is effective in finding out the anomalies in the shape and regularity, respectively. In the literature, this detection method has the best performance on detecting a wide variety of covert timing channels.

To evade the detection tests, the technology of mimicking legitimate traffic has been used in intelligent channel design [9,19,14]. In [9], the distribution of covert traffic is very close to that of legitimate traffic. However, owing to the lack of inter-packet dependencies, this kind of channel fails to evade the entropy-based detection method [8]. In [19], the goal of channel design is to defeat the entropy-based detection method. Its encoding method is based on that proposed by Shah et al. [18] but sacrifices a half of IPDs to evade detection tests, therefore, it has a low channel capacity, which is nearly 0.5 bit/packet. Kothari et al. [14] proposed an undetectable timing channel that uses a mechanism of Regularity Tree to mimic the irregularity of legitimate traffic. This channel maintains throughput of 1 bit/packet.

3 Our Scheme

In this section, we first introduce mimic functions, the basis of our encoding scheme. Then, we describe the mimicry framework that is designed to automatically create the new type of detection-resistant covert timing channel.

3.1 Mimic Functions

Mimic functions [21], which were introduced by Peter Wayner, are used to transmit hidden information as a subliminal technique. A mimic function changes input data so it assumes the statistical properties of another type of data, and consequently accomplishes the mimicry of identity. This technique has been applied in various scenarios, e.g., text steganography [1], digital watermarking [5], and code obfuscation [22]. Nevertheless, to the best of our knowledge, this technique has not yet been employed in the field of covert timing channel.

Regular Mimic Functions. Regular mimic functions use Huffman coding algorithm as the base. In Huffman coding, a table of occurrence frequency for each symbol in the input is required to build a variable-length code table, according to which a binary tree of nodes, namely a huffman tree, is generated. As a result, the symbols which occur frequently reside at the top, that is, they are given short representations, while the rare symbols are represented as long codes and located at the deep.

The inverse of Huffman coding can be used as mimic functions if the input is a random bit stream. The mimic process consists of two parts: compression phase and expansion phase. In the compression phase, the frequency table of each symbol in a data set A is estimated and the corresponding huffman tree is constructed. In the second phase, a data set B of random bits is expanded, specifically, variable length blocks are converted into fixed length blocks due to the huffman decoding operation, which is based on the huffman tree of A.

However, there is a problem that symbols occur in regular mimic functions output with different probabilities from the original ones. In fact, the regular model limits all symbols to have a probability which is a negative power of two, e.g., .5, .25, .125, and so on. The following technique can be used to solve this problem.

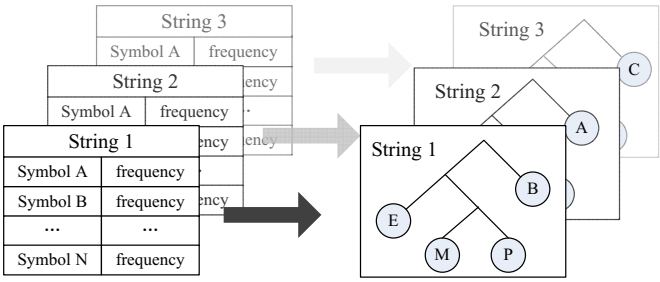

Fig. 1. The huffman forest

High-order Mimic Functions. Compared with regular mimic functions, high-order mimic functions capture more detailed statistical profile of data. In order

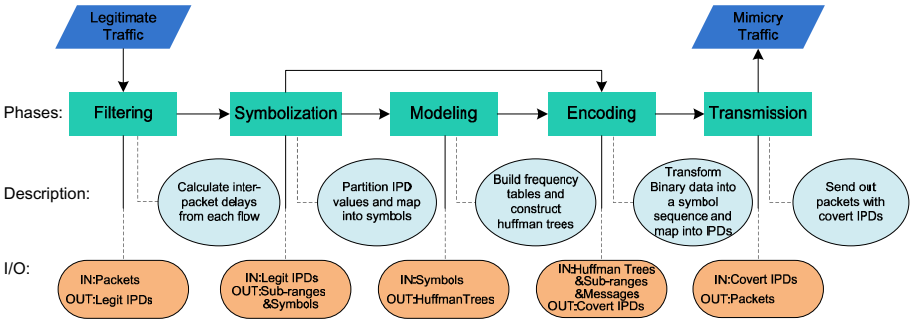

Fig. 2. The mimicry framework

to maintain regularity in the data, high-order mimic functions extract the inter-symbol dependencies by estimating the frequency of each symbol that follows a specific string of length $n-1$. High-order mimic functions build a huffman tree for each occurred string of length $n-1$, which results in a forest of huffman trees, as shown in Figure 1.

As a start, the encoding of high-order mimic functions requires one possible string as a seed. Given the seed, the encoding program locates the right huffman tree with the prefix of that selected string in the forest, and then uses the huffman decoding operation to determine the symbol that will follow the string. The resulting symbol and its preceding string of length $n-2$ form a new prefix of length $n-1$. The encoding program will take iterations in this order and output one by one. As the order n increases, the results become more and more approximate to the original data.

3.2 The Mimicry Framework

Given the advantage of high-order mimic functions, we decide to use their encoding technique as the basis for our scheme. To construct practical covert timing channels, we design a mimicry framework for mimicking, encoding, and transmitting. This framework includes five phases: filtering, symbolization, modeling, encoding, and transmission. Figure 2 gives an overview of our framework and a concise description of each phase. Details are expanded in the following paragraphs.

I. Filtering

In this phase, the packet sniffer captures packets from legitimate network connections. The packets are then classified into individual flows according to protocols, and source and destination IP addresses and ports. Generally, different types of traffic have different statistical properties. For example, HTTP and SSH protocols are both based on TCP/IP, but the difference between their traffic behaviors also exists and has been revealed by statistical tests [8]. Furthermore, the more specific traffic we filter out, the more precise statistical properties we can mimic.

For this reason, we choose a specific application protocol as a filtering condition. After trace classification, the packet analyzer calculates timing intervals between adjacent packets from each trace.

II. Symbolization

The input IPDs are mapped into corresponding symbols in this phase. In our application scenario, the objective is to mimic the statistical properties of legitimate traffic and thereby cover up the presence of covert timing channels. If IPD values are mapped to symbols one-to-one, the symbol set will be oversize due to the high variation in HTTP traffic, resulting in overload of the encoding program. To solve this problem, we sort all IPDs in ascending order, and partition IPD range into several sub-ranges. IPDs in the same sub-range are mapped to the same symbol.

The partitioning approach is of vital importance to the effectiveness of mimicking. Our approach is based on the observation that the IPD data is intensive in some ranges and rare in the other ones, and hence, the parameter of probability density provides a critical basis for partitioning. Specifically, the principle of our approach is that data has nearly uniform density within each sub-range. To achieve this, we firstly calculate the cumulative distribution function for IPD data. If data is uniformly distributed within a certain range, this part of the cumulative distribution curve exhibits a straight line. An inflection point is likely to exist between a high-density area and a low-density area on the cumulative distribution curve. Due to finite samples, the obtained cumulative distribution curve is very rough, and actually, composed of many line segments. In consideration of the objective and analysis described hereinbefore, we choose to use Douglas-Peucker algorithm [7] as the appropriate method of locating inflection points.

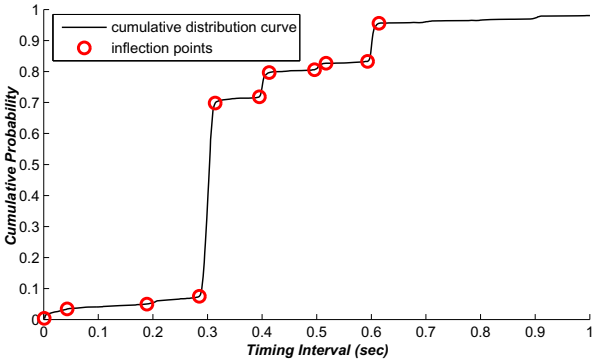

Fig. 3. Inflection points located by Douglas-Peucker algorithm ($\varepsilon = 0.01$)

The Douglas-Peucker algorithm is also known as the line simplification algorithm. The purpose of the algorithm is to produce a simplified polyline that approximates the original one within a specified tolerance ε. Figure 3 shows the

effect of Douglas-Peucker algorithm on locating inflection points in a cumulative distribution curve. According to these points, sub-ranges are determined and then legitimate IPDs are mapped to corresponding symbols.

III. Modeling

The modeling phase is a phase of extracting statistical properties of legitimate IPDs by constructing huffman trees. This phase takes the sequence of symbolized IPDs as the modeling object and mimicry target.

The first step is building frequency tables for all occurred strings of length $n-1$ in order to extract the n^{th} order statistics. The modeling function processes the input sequence in a sequential manner. When moving onto a string of length $n-1$, the function determines whether this pattern has already occurred. If so, the function gets the symbol that follows the string, and increments the corresponding frequency counter in the table appended to this pattern. If it is absent, then a new table is created and appended to it. These operations are repeatedly conducted until the search is completed. If the input sequence has a total of 2000 symbols, $2001-n$ strings of length $n-1$ can be collected, some of which may have the same pattern. The strings of length $n-1$, acting as the "prefixes" in mimic functions, are the indexes of their frequency tables. At the end of modeling phase, each frequency table is converted into a huffman tree. All the huffman trees with corresponding indexes are output to the next phase.

IV. Encoding

Based upon the modeling results, the encoding phase converts an arbitrary binary stream into a sequence of symbols, which has similar statistics with the mimicry target. The encoding function has three components: randomization, mimic encoding process, and inverse mapping.

Covert messages are mostly encoded into binary sequences with ASCII scheme [3], and each letter is represented as a code with length of 8 bits. Due to different occurrence frequency of letters [6], these binary sequences are non-random, whereas the input of mimic functions is required to be random. To solve this problem, each binary sequence is randomized by XORing with a pseudo-random bit stream. This stream is assumed to be known by the sender and receiver of the covert timing channel.

After randomization, the input of mimic encoding process is an arbitrary binary stream. To start the encoding of n^{th} order mimic functions, the first string of length $n-1$ is chosen as a seed. The right huffman tree of the specific string is located according to its index. Then the process performs the operation of walking the tree. From the root node, the process selects left or right branches according to the input bit, until it arrives at a leaf node, which represents a symbol. The most recently generated symbol and its preceding string of length $n-2$ forms a new string of length $n-1$. The encoding program will take iterations in this order and output one by one. In the end, a sequence of symbols is obtained.

Inverse mapping refers to an operation of inversely mapping symbols to IPD values. First, one symbol is mapped into the corresponding sub-range. Then, an IPD value is randomly selected from the sub-range.

V. Transmission

In this phase, the sender of the covert timing channel modulates the timing of packets corresponding to the sequence of IPDs, and then forwards the packets to the Internet.

3.3 Design Details

Prior Agreement: In this channel, there is a prerequisite that the sender and receiver should share the same mimicry target and have the same modeling results. One choice is to transmit the modeling results from the sender to the receiver. However, this solution requires a large amount of communication traffic before the covert channel is built up. The other choice is to collect the same legitimate traffic as the mimicry target by the two parties. They agree in advance on a particular period and a particular network trace. For instance, they single out the given trace during 8:00 AM and 8:05 AM. Even if several packets are lost or effected by network jitters, the modeling results will stay the same. This is because our partitioning method is based on the cumulative distribution function of IPDs, and thus slight changes during the transmission almost have no effect on the overall distribution. Moreover, slight changes have little influence on the generated huffman trees.

Decreasing Error Rate: In the symbolization phase, the distribution of IPDs in each sub-range is treated as the uniform distribution approximately. Accordingly, an IPD value is uniformly selected from the corresponding sub-range in the inverse mapping operation. When the selected IPD value is close to the cut-off point between the adjacent sub-ranges, a transmission error will probably arise due to network jitters. One solution is adding error correcting bits to covert data. In addition, the areas with a certain distance apart from cut-off points can be removed from the selection ranges, but this solution theoretically has a little influence on the mimicry effectiveness.

Parameter Selection: The tolerance ε in the Douglas-Peucker algorithm has dual influence on the practicality of our scheme. If ε is large, then the number of inflection points is small and sub-ranges are wide, thus the approximation to legitimate traffic is inaccurate. In contrast, if ε is small, then sub-ranges are narrow, causing a relatively high decoding error rate. Moreover, increasing the number of sub-ranges will increase the number of huffman trees. A large number of huffman trees will result in low efficiency of modeling and encoding. After comprehensive consideration, we choose $\varepsilon = 0.01$ in the following experiments.

4 Experimental Evaluation

We conducted a series of experiments to validate the effectiveness of our mimicry approach. The emphasis of our experiments is on determining if the covert timing channel exploited from the mimicry framework can evade the state-of-the-art detection tests. In addition, we examined the capacity of this new type of covert timing channel.

4.1 Experimental Setup

The detection tests are based on statistical analysis, therefore a large volume of network data is necessary in order to perform these tests. The selection of test data is detailed in the following paragraphs.

Legitimate Data Collection. In our experiments, we selected HTTP traffic as our mimicry target. The reasons include: 1) HTTP is the most widely used protocol on the Internet, thus the defensive perimeter of a network commonly allows HTTP packets to pass through. 2) The large volume of HTTP traffic makes it an ideal medium for covert communication. The HTTP traffic used in our experiments was extracted from publicly available data sets named NZIX-II [11]. The data sets contain the mixed traces of diverse network protocols. To filter out only HTTP traces, we used the destination port number 80 as the filtering condition. After that, HTTP streams were grouped into individual flows according to the source and destination IP addresses. IPDs calculated from each individual flow were jointed together to be our legitimate data set.

For different purposes, the legitimate data set is divided into two subsets: training set and test set. The training set, composed of 10,000,000 IPDs, is intended to initialize detection tests. The test set is used as the mimicry target for generating covert traffic, as well as the comparison object for the detection tests. This set contains 100 samples, each of which has 2000 IPDs.

Covert Data Generation. To automatically create the new type of covert timing channel, referred to as MFCTC hereinafter, we implemented each function of the framework on our testing machine, and integrated each into a complete pipeline for generating covert traffic. We then input legitimate data as the mimicry target. In the experiments, we set the tolerance ε in the Douglas-Peucker algorithm to be 0.01. The order of mimic functions was tuned to create different MFCTC data sets.

For the comparison purpose, we also implemented two existing covert timing channels: MBCTC [9] and IPCTC [2]. For MBCTC, each 100 packets of the test set are fitted to a model, which is used to generate covert traffic. For IPCTC, the timing interval is rotated among 0.04, 0.06, and 0.08 seconds after each 100 packets as suggested by Cabuk et al. [2].

4.2 Detection Resistance

The detection resistance is the objective of our mimicry approach. It can be estimated from two aspects: the *shape* and *regularity* of network traffic. The shape of traffic is described by first-order statistics, e.g., distribution. The regularity of traffic is described by high-order statistics, e.g., correction. In the experiments, we utilized the most advanced detection method—the entropy-based method [8], which uses the combination of entropy and corrected conditional entropy to examine the shape and regularity respectively. To our knowledge, the entropy-based detection method has the best performance on detecting various covert timing channels. In this section, we detail this detection method and show the detection results.

The Entropy-based Detection Method. This detection method is based on the observation that the creation of a covert timing channel changes the entropy of the original traffic in a certain extent. In information theory, entropy is used as a measure of the uncertainty in a random variable [4].

This detection method utilizes two metrics: entropy (EN) and corrected conditional entropy (CCE). The definition of entropy of a random variable X is given as:

$$EN(X) = -\sum_{X} P(x) log P(x)$$

The entropy describes the first-order statistics of traffic and can be used as a shape test. In addition, the corrected conditional entropy that is used to estimate the entropy rate can be used as a regularity test:

$$CCE(X_m|X_1, ..., X_{m-1}) = CE(X_m|X_1, ..., X_{m-1}) + perc(X_m) * EN(X_1)$$

Where $CE(X_m|X_1, ..., X_{m-1})$ is the conditional entropy of a random process $X = X_i$, $perc(X_m)$ is the percentage of unique sequence patterns of length m, and $EN(X_1)$ is the first-order entropy.

The detection method requires a large number of legitimate IPDs for training. For maximum effectiveness, this method divides the training data into Q equiprobable bins. When the Q bins have the same number of IPDs, the entropy reaches a maximum. Abnormal traffic, which has a different distribution, usually gets a low entropy score. Gianvecchio et al. chose $Q = 65536$ for EN test while $Q = 5$ for CCE test.

Dataset. In our experiments, we used seven data sets, including:

- ▷ HTTP training set: 10,000,000 HTTP IPDs
- ▷ HTTP test set: 200,000 HTTP IPDs
- ▷ IPCTC test set: 200,000 HTTP IPDs
- ▷ MBCTC test set: 200,000 HTTP IPDs
- ▷ 3^{rd}-MFCTC test set: 200,000 HTTP IPDs
- ▷ 4^{th}-MFCTC test set: 200,000 HTTP IPDs
- ▷ 5^{th}-MFCTC test set: 200,000 HTTP IPDs

To initialize the detection tests, we used the HTTP training set to determine the bin ranges for EN and CCE tests, respectively. MBCTC and MFCTC test sets are both generated by mimicking the HTTP test set. The three MFCTC test sets are based on third-order, forth-order, and fifth-order mimic functions, respectively.

Detection Results. Our *first* set of experiments is to investigate the effect of the order of mimic functions on approximation. Theoretically, mimic functions with higher order capture more detailed statistical profile of data. Therefore, the statistics of the corresponding MFCTC traffic are more similar to those of original traffic. However, it is impractical to employ a mimic function with a very high order. Increasing n will increase the number of huffman trees exponentially. This means the order has the direct effect on the performance of data processing. In our experiments, we chose $n = 3, 4, 5$ for generating covert traffic, respectively.

In order to investigate the effect in reality, we ran EN and CCE test 100 times against HTTP test set and three MFCTC test sets, respectively. A sample

of 2000 IPDs was used in each time. To compare the results, we calculate the difference between the test score for each sample of covert data and that for the corresponding sample of legitimate data. The mean of the test scores and the comparative scores are shown in Table 1. The test scores for the three MFCTC test sets are all higher on average than those of legitimate test set. Whereas, with the increasing of the order, the comparative scores are decreasing gradually. This indicates that the higher order results in the more accurate approximation.

Table 1. The mean of the test scores and the comparative scores

test	LEGIT	3^{rd}-MFCTC		4^{th}-MFCTC		5^{th}-MFCTC	
	Mean	Mean	Diff	Mean	Diff	Mean	Diff
EN	16.214	19.566	3.894	19.416	3.666	19.340	3.565
CCE	1.949	2.016	0.117	1.996	0.106	1.987	0.098

Our *second* set of experiments is to determine if MFCTC traffic created from our mimicry framework can evade the entropy-based detection method. We ran each detection test 100 times against legitimate, 4^{th}-MFCTC, IPCTC, and MBCTC traffic, respectively. A sample of 2000 IPDs was used in each time. The mean of EN and CCE test scores are shown in Figure 4 and Figure 5, respectively. In theory, a low EN test score indicates the first-order probability of the test traffic is distinct from that of training data, and hence suggests this traffic is abnormal. In turn, the higher EN test score the traffic gets, the more similar to legitimate data it is. Analyzing in the same way, when the CCE test score is very high, the traffic lacks regularity. When the CCE test score is very low, the traffic is too regular. Our test results show that the mean EN test score of MFCTC is much higher than that of legitimate traffic, while the mean CCE test score is very close to that of legitimate traffic. The EN and CCE test scores of IPCTC are both too low. For MBCTC, the EN test score is higher than that of legitimate traffic, however, the CCE test score is too much higher.

In order to estimate the detection rates, we introduce a criterion called *false positive rate*, which was also used by Gianvecchio et al [8]. The false positive rate refers to the rate of legitimate samples that are incorrectly classified as covert. We calculate the *cutoff* scores for both tests to achieve a false positive rate of 1%. Any sample with a test score beyond the normal range, partitioned by the cutoff, would be identified as covert. The detection rates for MFCTC, IPCTC, and MBCTC samples are shown in Table 2.

The EN and CCE tests are both able to detect the presence of IPCTC. Although the EN detection rate for MBCTC is 0%, the CCE detection rate reaches up to 90%. This reveals that MBCTC only exploits the first-order statistics of legitimate traffic, but ignores the regularity. The EN detection rate for MFCTC is 0%, while the CCE detection rate is very low, only 6%.

We also investigate the distributions of the CCE test scores for legitimate and MFCTC samples, which are illustrated in Figure 6. Most scores for both fall between 1.8 and 2.2. There is a heavy overlap between the distributions. Moreover, with the increasing of the order of mimic functions, the overlap becomes

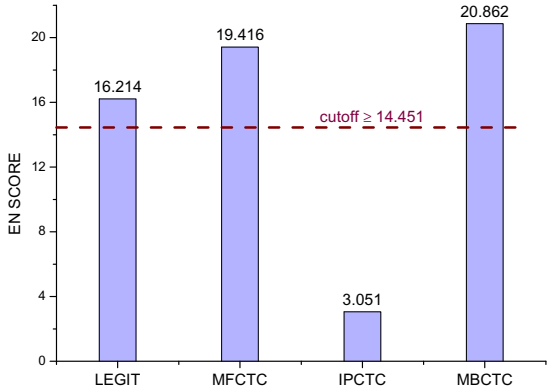

Fig. 4. EN test scores for legitimate, MFCTC, IPCTC, and MBCTC IPDs

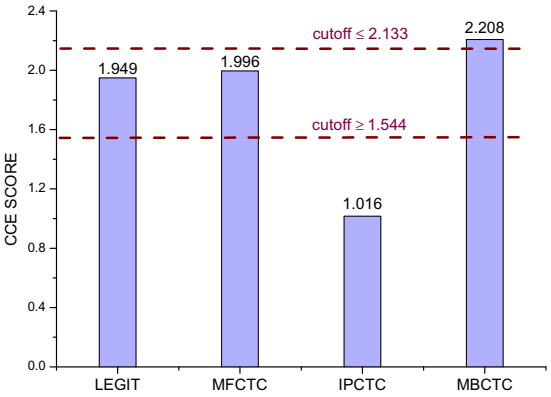

Fig. 5. CCE test scores for legitimate, MFCTC, IPCTC, and MBCTC IPDs

Table 2. The detection rates for MFCTC, IPCTC, and MBCTC samples

Test	LEGIT	MFCTC	IPCTC	MBCTC
	False Positive	Detection Rate	Detection Rate	Detection Rate
$EN \leq 14.451$	1%	0%	100%	0%
$CCE \leq 1.544$	1%	0%	100%	0%
$CCE \geq 2.133$	1%	6%	0%	90%

heavier. This implies that the detection rate can be reduced by increasing the order. In conclusion, the detection results indicate that our new type of covert timing channel is undetectable by the entropy-based detection tests.

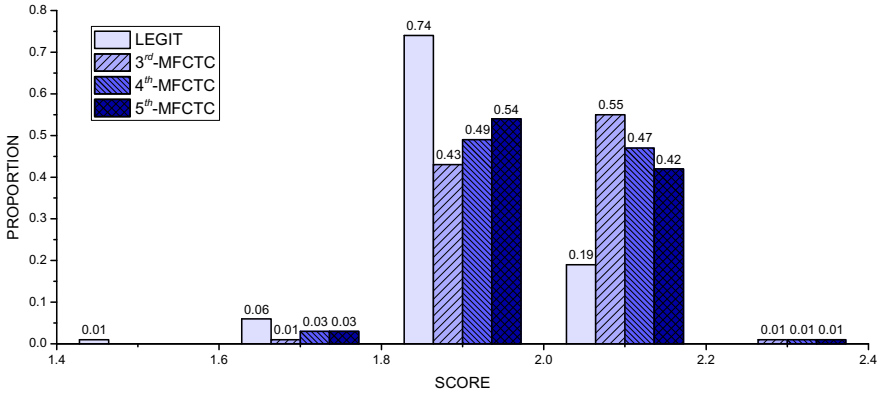

Fig. 6. The proportion of CCE test scores

Other Detection Tests. To be more convincing, we also ran Kolmogorov-Smirnov test [16] and regularity test [2] against MFCTC. The Kolmogorov-Smirnov test quantifies the maximum distance between two empirical distribution functions, so it can be used to examine the shape of traffic. If the test score is small, it implies that the sample is close to the legitimate behavior. For the regularity test, network traffic is separated to several windows with the same size, and the standard deviation is computed for each window. In general, the regularity score of legitimate traffic is high, because legitimate traffic changes over time. This test can be used to examine the regularity of traffic. Our results showed that the two detection tests both get 0% detection rate for MFCTC when the false positive rate is 1%.

4.3 Capacity

The channel capacity of MFCTC depends on the sizes and heights of huffman trees. Due to different mimicry targets and modeling results, the capacity of MFCTC is indefinite. To estimate the channel capacity, we count up the number of sub-ranges for 100 samples. When the tolerance $\varepsilon = 0.01$, these samples have 1044 sub-ranges in total, that is, there is nearly 10 sub-ranges on average. Consequently, for the first-order mimic function, each huffman tree has approximately 10 leaf nodes, thus the corresponding channel capacity is between 2 and 3.25 bits/packet in theory. Whereas, owing to the limited size of the mimicry target, only 2000 IPDs in each sample, huffman trees of high-order mimic functions have much fewer leaf nodes. For these 100 samples of 4^{th}-MFCTC, the mean of bit transmission rate is 1.5 bits/packet. When we enlarge the size to be 20000 and 200000 IPDs in each sample, the transmission rate increases to be 1.7 and 1.9 bits/packet, respectively. On the whole, the capacity of our encoding scheme is much higher than that of Liquid [19] and Mimic [14], which delivery 0.5 and 1 bit per packet respectively.

5 Conclusion

In this paper, we utilized mimic function to construct a new type of detection-resistant covert timing channel, which is undetectable by current detection tests while maintaining a relatively high channel capacity. We implemented a mimicry framework to automatically generate covert traffic. The framework includes five phases: filtering, symbolization, modeling, encoding, and transmission. The traffic generation process is as follows. Firstly, IPD values are filtered from legitimate traffic. Secondly, they are mapped into corresponding symbols. Thirdly, the statistical properties are extracted through building huffman trees. Next, based upon the modeling results, the encoding phase converts an arbitrary binary stream into a sequence of symbols, which has similar statistics with the mimicry target, and then inversely maps the symbols into IPD values. Finally, network packets with the mimicry IPDs are forwarded to the Internet.

In order to validate the effectiveness of the mimicry approach, we performed the state-of-the-art detection tests against our new type of channel (MFCTC) and known channels (IPCTC and MBCTC). The results show that only MFCTC can successfully evade the detection tests. Moreover, MFCTC is able to maintain a considerable capacity, which is much higher than existing undetectable channels, i.e., Liquid and Mimic.

Acknowledgement. This work was supported by the National Basic Research Program (973 Program) of China (No. 2013CB338001) and the Strategy Pilot Project of Chinese Academy of Sciences Sub-Project XDA06010702.

References

1. Atallah, M.J., Raskin, V., Hempelmann, C.F., Karahan, M., Sion, R., Topkara, U., Triezenberg, K.E.: Natural language watermarking and tamperproofing. In: Petitcolas, F.A.P. (ed.) IH 2002. LNCS, vol. 2578, pp. 196–212. Springer, Heidelberg (2003)
2. Cabuk, S., Brodley, C., Shields, C.: IP covert timing channels: Design and detection. In: Proceedings of the 11th ACM Conference on Computer and Communications Security, pp. 178–187 (2004)
3. Cabuk, S., Brodley, C., Shields, C.: IP covert channel detection. ACM Transactions on Information and System Security (TISSEC) 12(4), 22 (2009)
4. Cover, T., Thomas, J.: Elements of information theory. Wiley-interscience (2006)
5. Cox, I., Miller, M., Bloom, J., Fridrich, J., Kalker, T.: Digital watermarking and steganography. Morgan Kaufmann (2007)
6. Dewey, G.: Relative frequency of English spellings. Teachers College Press, New York (1970)
7. Douglas, D.H., Peucker, T.K.: Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartographica: The International Journal for Geographic Information and Geovisualization 10(2), 112–122 (1973)
8. Gianvecchio, S., Wang, H.: Detecting covert timing channels: An entropy-based approach. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, pp. 307–316 (2007)

9. Gianvecchio, S., Wang, H., Wijesekera, D., Jajodia, S.: Model-based covert timing channels: Automated modeling and evasion. In: Proceedings of the 11th International Symposium on Recent Advances in Intrusion Detection, pp. 211–230 (2008)
10. Girling, C.: Covert channels in LAN's. IEEE Transactions on Software Engineering, 292–296 (1987)
11. WAND Research Group. Waikato internet traffic storage, `http://wand.net.nz/wits/nzix/2/`
12. Henry, P.A.: Covert channels provided hackers the opportunity and the means for the current distributed denial of service attacks. CyberGuard Corporation (2000)
13. Houmansadr, A., Nguyen, G.T., Caesar, M., Borisov, N.: Cirripede: Circumvention infrastructure using router redirection with plausible deniability. In: Proceedings of the 18th ACM Conference on Computer and Communications Security, pp. 187–200 (2011)
14. Kothari, K., Wright, M.: Mimic: An active covert channel that evades regularity-based detection. Computer Networks (2012)
15. Lampson, B.: A note on the confinement problem. Communications of the ACM 16(10), 613–615 (1973)
16. Peng, P., Ning, P., Reeves, D.: On the secrecy of timing-based active watermarking trace-back techniques. In: IEEE Symposium on Security and Privacy, pp. 334–349 (2006)
17. Sellke, S., Wang, C., Bagchi, S., Shroff, N.: TCP/IP timing channels: Theory to implementation. In: INFOCOM, pp. 2204–2212 (2009)
18. Shah, G., Molina, A., Blaze, M.: Keyboards and covert channels. In: Proceedings of the 15th Conference on USENIX Security Symposium, vol. 15 (2006)
19. Walls, R., Kothari, K., Wright, M.: Liquid: A detection-resistant covert timing channel based on IPD shaping. Computer Networks 55(6), 1217–1228 (2011)
20. Wang, X., Reeves, D.S.: Robust correlation of encrypted attack traffic through stepping stones by manipulation of interpacket delays. In: Proceedings of the 10th ACM Conference on Computer and Communications Security, pp. 20–29 (2003)
21. Wayner, P.: Mimic functions. Cryptologia 16(3), 193–214 (1992)
22. Wu, Z., Gianvecchio, S., Xie, M., Wang, H.: Mimimorphism: A new approach to binary code obfuscation. In: Proceedings of the 17th ACM Conference on Computer and Communications Security, pp. 536–546 (2010)