CoCo: coding-based covert timing channels for network flows

Amir Houmansadr and Nikita Borisov

Department of Electrical and Computer Engineering,
University of Illinois at Urbana-Champaign
{ahouman2,nikita}@illinois.edu

Abstract. In this paper, we propose CoCo, a novel framework for establishing covert timing channels. The CoCo covert channel modulates the covert message in the inter-packet delays of the network flows, while a coding algorithm is used to ensure the robustness of the covert message to different perturbations. The CoCo covert channel is adjustable: by adjusting certain parameters one can trade off different features of the covert channel, i.e., robustness, rate, and undetectability. By simulating the CoCo covert channel using different coding algorithms we show that CoCo improves the covert robustness as compared to the previous research, while being practically undetectable.

1 Introduction

A covert channel intends to conceal the very existence of a hidden message, *covert message*, by communicating it through a legitimate communication channel, i.e., the overt channel. Considering the computer networks, covert channels lie within two major categories: *covert storage channels*, and *covert timing channels* [5]. Covert storage channels work by modifying some unused/random bits in the packet header of a network flow [19, 12, 9]. Alternatively, a covert timing channel modulates the covert message into the timing pattern of network flow packets [3, 21]. A covert channel needs to be *undetectable*, meaning that the covert traffic should not be distinguishable from a legitimate traffic. Murdoch et al. show that many of the storage covert channels can be detected easily since the covert message modifies the benign pattern of the utilized header fields [17]. Also, different statistical tests have been utilized to detect covert timing channels [2, 7]. In addition to undetectability, a covert message needs to be *robust* to the perturbations caused by the communicating network and/or an adversary. Under given undetectability and robustness requirements a covert channel aims in maximizing its *rate*, i.e., the number of covert message bits sent per covert flow packets.

In this paper, we design CoCo as a reliable and adjustable framework for establishing covert timing channels. CoCo modulates the covert message in the inter-packet delays (IPD) of a network flow, while a coding algorithm is used to ensure the covert message robustness. A main contribution of CoCo over the previous work is that CoCo is adjustable: a user of the covert channel can tradeoff different features of the covert channel, e.g., undetectability, rate, and robustness, considering the application and network conditions. Recent research tends towards developing covert channels with *provable* undetectability [15], however, for many applications of covert channels a *practical* undetectability is sufficient. Instead of providing a provable undetectability for all applications CoCo adjusts its level of undetectability to what is needed for each specific application, resulting in significant improvements in robustness and/or the rate of the covert message. The adjustments are performed based on the channel models for noise and adversarial perturbations, and also the performance priorities of the covert channel users.

The CoCo covert channel is fast and easy to implement. To communicate a covert message, m, a *sender* S generates a network flow f for a given traffic model using a keyed IPD generator. The sender, then, *slightly* manipulates f according to c which is an encoded version of the covert message m, resulting in the covert flow, f^c. Finally, f^c is sent to a *receiver* R through a noisy channel, e.g., the Internet. The receiver extracts the covert message from a noisy version of f^c using a secret key which is only shared between the sender and the receiver. The channel noise is composed of two components: the natural noise of the overt network, e.g., network delays, and the perturbations performed by an adversary in order to degrade/destroy covert channels. CoCo is robust to both natural and adversarial perturbations of the communication channel by taking advantage of efficient coding algorithms. By adjusting the encoding algorithm and the *gain* of the covert message, as defined later, CoCo is able to balance between rate, undetectability, and robustness of the covert message. Our experiments show that the choice of the coding algorithm and also the encoding rate impacts the robustness performance of the covert message. We show through experiments that under reasonable requirements for practical undetectability CoCo improves the robustness performance significantly compared to similar previous research.

The rest of this paper is organized as follows; we mention some related work in Section 2. In Section 3, we discuss the system model and features of the covert timing channel. In Section 4, we describe a detailed description of the CoCo covert timing channel. In Section 5, we evaluate the detection performance of our covert channel using different coding algorithms. The undetectability of the CoCo covert channel is evaluated in Section 6, and the paper is concluded in Section 7.

2 Related Work

The very first covert timing channels are based on having a sender either send or remain silent in specific time intervals in order to communicate covert message bits 1 and 0, respectively [18, 11]. These schemes not only are limited by the need for accurate synchronization between the sender and the receiver, but also are easily detectable using statistical tests [3]. Later research on covert timing channels leans toward inserting the covert message directly in the inter-packet delays (IPD). In [4] a covert timing channel is proposed that tries to mimic the empirical distribution of the legitimate traffic by splitting the empirical IPDs distribution into two equally-sized *small-delay* and *large-delay* groups and sending a 0-bit (1-bit) covert message by randomly replaying an IPD from the small-delay (large-delay) set. Berk et al. also suggest to encode the messages directly into the inter-packet delays in order to maximize the covert capacity [2]. Shah et al. propose the keyboard JitterBug, a low capacity channel for leaking the typed information over an existing interactive network connection [21].

To defend against covert timing channels two different approaches have been taken in the literature. The first approach is to actively disrupt the network communication in order to destroy possibly existing covert timing channels. An example for this is the use of network *jammers* that apply random delays over the network packets [10]. Even though such disruptions can seriously devote the capacity of covert timing channels they also disrupt the expected performance of the legitimate traffic. This is more of a problem in the case of interactive traffic and real-time traffic. In addition, such disruptions can result in changing the traffic shape, hence, revealing the existence of the traffic disruptors to the covert communicating parties. Another approach in defending covert channels is to passively monitor network traffic in order to detect such channels. This is done using different statistical tests including shape tests, regularity tests and entropy tests [3, 2, 7].

To thwart these detection mechanisms Gianvecchio et al. devise a *model-based* covert timing channel that models the network traffic in order to generate the covert traffic [8]. The scheme, however, requires frequent communication of traffic model parameters from the sender to the receiver. Recent research has considered simple encoding of messages into the inter-packet delays in order to increase the data rate of covert communication. Sellke et al. use a simple *Geometric code* in conjunction with pseudo random generators to build a low-rate undetectable covert timing channel for i.i.d. traffic [20]. The scheme is limited by the strong assumption of the additive jitter being bounded. Liu et al. in [14] also modulate covert messages in the IPDs of the traffic by using spreading codes in order to increase the robustness of the covert channel to the network perturbations. A similar covert timing channel is proposed in [15] for i.i.d. traffic that provides *provable* undetectability for the covert message.

3 Preliminaries

3.1 System model

A covert channel consists of a *sender (S)* sending a *covert message* to a *receiver (R)* through a steganographic channel. In this paper, we consider the design of a covert timing channel for computer networks, i.e., the steganographic communication is established through the packet timing information of a network flow, the *cover flow*. The designed covert channel is *active*, in a sense that it generates the timings of the network traffic used for embedding the covert message. CoCo is based on modulating the covert message into the inter-packet delays (IPD) of network flows. Some secret information, the *secret key*, is shared between the sender and the receiver so that a third-party is not able to either extract the covert message, or detect the presence of the covert communication channel.

3.2 Adversary model

We assume that the CoCo covert traffic is monitored by an adversary that tries to detect the presence of covert timing channels. This requires the covert channel to be *undetectable*, as defined later. Also, we consider *active* adversaries that perturbs network traffic in order to destroy the possibly existing covert timing channels. An effective covert timing channel should be *robust* to the *channel noise* which is combined of the adversarial perturbations and the natural noise of the network.

The adversarial perturbations are constrained in two ways; first of all, they should not interfere with the benign traffic, as such perturbations are usually performed by entities providing services to legitimate users, e.g., intrusion detection systems. Secondly, the adversarial perturbations should be concealed from covert parties: having knowledge of the active adversaries the covert parties can evade the detection by taking alternative covert mechanisms which are not affected by the known perturbations. Based on this, we assume that the adversarial perturbations can amplify the effect of the network noise in the channel noise, but they do not change the behavior of the channel noise drastically. In our simulations throughout this paper we model the channel noise as an amplified version of the network noise.

3.3 Features of the covert timing channel

Undetectability We define a covert channel to be *undetectable* by a test algorithm A with a *Cross-Over Error Rate (COER)* of C if the algorithm A is not able to distinguish between a legitimate flow and a covert flow with a COER smaller than C. A test A has a COER of C if it returns the same rates of false alarms and misses in detecting a covert traffic. This is further elaborated in Section 6.

Robustness As mentioned above, a cover flow containing the covert message is perturbed by the channel noise which is composed of the network noise and the adversary noise. We define a covert timing channel to be *robust* to the channel noise with a bit error rate (BER) of ϵ if the receiver is able to extract the covert message with a BER of at most ϵ. BER is defined as the number of errored covert bits at the receiver divided by the total number of covert bits.

Rate Another feature of a covert channel is the *rate* of the covert communication. We define the rate of the CoCo covert channel to be

$$r = \lim_{N \to \infty} \frac{K}{N} \tag{1}$$

where K is the number of covert message bits sent using $N + 1$ packets of a cover flow. Under given undetectability and robustness requirements, a covert channel designer aims in maximizing the rate of the covert channel. As will be discussed later, the choice of the coding algorithm of CoCo trades off the rate for the robustness of the covert channel.

4 The CoCo scheme

In this section we describe the scheme of the CoCo covert timing channel.

4.1 The CoCo sender scheme

The sender of the covert channel, S, inserts the covert message into the inter-packet delays of the covert traffic. Figure 1 illustrates the scheme of the CoCo sender that consists of the following components:

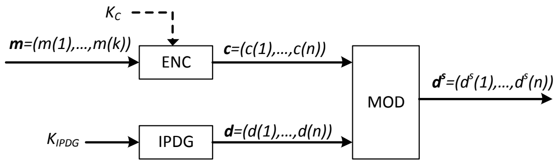
K_C
$\mathbf{m}$=(m(1),...,m(k))
ENC
$\mathbf{c}$=(c(1),...,c(n))
MOD
$\mathbf{d}^s$=(d^s(1),...,d^s(n))
K_{IPDG}
IPDG
$\mathbf{d}$=(d(1),...,d(n))

Fig. 1. The block diagram of the CoCo sender.

- IPD Generator (IPDG): pseudo-randomly generates the inter-packet delays (IPD) according to a given traffic model and using a secret key.
- Message Encoder (ENC): encodes the covert message bits using a coding algorithm in order to make the covert channel robust to the channel noise.
- IPD Modulator (MOD): modulates the output of the ENC encoder into the IPDs generated by the IPDG block.

A covert flow is generated in three steps:

A. Encoding the message An encoder ENC is called an (n, k) coding algorithm if for any block of k message bits, $\mathbf{m} = (m(1), \ldots, m(k))$, it generates a block of n encoded bits, $\mathbf{c} = (c(0), \ldots, c(n))$. Such an encoder has a *coding rate* of k/n which also determines the rate of CoCo, i.e., $r = k/n$. A CoCo sender divides a covert message into blocks of k bits and encodes each block using the ENC (n, k) coding algorithm. The ENC encoder accepts inputs in the binary format and generates output in the bipolar format, i.e., $m(i) \in \{0, 1\}$, and $c(j) \in \{-1, 1\}$ $(1 \leq i \leq k, 1 \leq j \leq n)$. Note that the ENC encoder may be protected by an encoding key K_c in order to protect the message confidentiality in case of a compromise. This is not necessary as the covert message $\mathbf{m}$ can be encrypted before encoding. In Section 5, we investigate the use of different coding algorithms as the ENC encoder of CoCo and compare the functionality of CoCo for different schemes.

B. Generating the inter-packet delays The IPDG block generates sequences of inter-packet delays pseudo-randomly according to a given traffic model. The IPDG takes a traffic model and a secret key, K_{IPDG}, as input and generates sequences of IPDs according to the given traffic model. The secret key K_{IPDG} is only shared between the sender S and the receiver R, that enable S and R to generate the same pseudo-randomly generated sequences of IPDs. In Section 4.3 we design an IPDG for generating i.i.d. traffic. In fact, the i.i.d. assumption is for the sake of simplicity and the IPDG can be designed in a similar manner for any type of network traffic.

C. Modulating the IPDs with the encoded message Once the coded message $\mathbf{c}$ is generated (step A) it is embedded within the inter-packet delays generated by the IPDG (step B). In order to send a length n encoded message $\mathbf{c} = (c_1, ..., c_n)$, the sender requires a length n IPD sequence $\mathbf{d} = (d(1), ..., d(n))$. The modulated IPDs are given by:

$$d^S(i) = d(i) + a \times c(i) \qquad 1 \leq i \leq n \tag{2}$$

where $d^S(i)$ is the i^{th} modulated IPD and a is the *amplitude* of the covert channel. We define the *gain* of the covert channel, γ, as:

$$\gamma = \log_2 \frac{a}{\sigma} \tag{3}$$

where σ is the standard deviation of the channel *jitter* (channel jitter is the effect of the channel noise on the inter-packet delays). In fact, γ represents the signal-to-noise ratio of the covert communication and is used in our evaluations in the following sections. Finally, the timings of the covert flow, $\mathbf{t^S} = (t^S(1), ..., t^S(n+1))$, are evaluated as:

$$t^S(i) = \sum_{j=1}^{i-1} d^S(j) + t^S(1) \qquad (2 \leq i \leq n+1) \tag{4}$$

where $t^S(1)$ is the timing of the first packet which is selected at random from the interval $[0, 2]$ seconds.

4.2 The CoCo receiver scheme

After being perturbed by the channel noise, the covert traffic receiver, R, receives the packets of the covert flow at times $\mathbf{t^R} = (t^R(1), ..., t^R(n+1))$. The IPDs of the received flow, $\mathbf{d^R} = (d^R(1), ..., d^R(n))$, are evaluated as:

$$d^R(i) = t^R(i+1) - t^R(i) \qquad (1 \leq i \leq n) \tag{5}$$

The receiver intends to extract the covert message form the received IPDs $\mathbf{d^R}$ which is a noisy version of the sent IPDs $\mathbf{d^S}$. The CoCo receiver scheme is shown in Figure 2 which consists of three components:

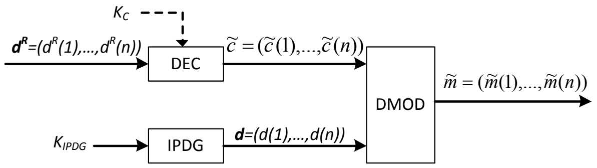
K_C
$\mathbf{d^R}=(d^R(1),\ldots,d^R(n))$
DEC
$\tilde{c}=(\tilde{c}(1),\ldots,\tilde{c}(n))$
DMOD
$\tilde{m}=(\tilde{m}(1),\ldots,\tilde{m}(n))$
K_{IPDG}
IPDG
$\mathbf{d}=(d(1),\ldots,d(n))$

Fig. 2. The CoCo receiver block diagram.

- IPD Generator (IPDG): the same IPD generator used by the CoCo sender.
- Message Decoder (DEC): a decoder for the ENC encoded messages. A key might be shared between ENC and DEC algorithms.
- IPD Demodulator (DMOD): extracts the modulated bits from the received IPDs.

The CoCo receiver scheme works in three steps in order to extract the covert message bits from a received covert flow.

A. Re-generating the IPDs The receiver uses the same IPD generator used by the sender in order to re-generate the IPDs used by the sender to modulate the covert message, i.e., **d**. The IPDG block uses a key K_{IPDG} which is only shared between the sender and receiver and is essential for the covert channel undetectability. In Section 4.3, we describe the design of the IPDG for an i.i.d. traffic model.

B. Demodulating the encoded message The received IPDs sequence $\mathbf{d^R}$ can be represented as:

$$
\begin{aligned}
d^R(i) &= d^S(i) + \delta(i) &\quad (6)\\
&= d(i) + a \times c(i) + \delta(i) &\quad (7)
\end{aligned}
$$

As mentioned in part A, the IPD sequence **d** can be regenerated by the receiver using some shared information. So, the receiver is able to evaluate $\tilde{\mathbf{c}} = (\tilde{c}(1), \ldots, \tilde{c}(n))$ which is a noisy version of the encoded message **c**:

$$
\begin{aligned}
\tilde{c}(i) &= \frac{d^R(i) - d(i)}{a} &\quad (8)\\
&= c(i) + \delta(i)/a &\quad (9)
\end{aligned}
$$

C. Decoding the covert message The final step in extracting the covert message is to decode the noisy encoded message, $\tilde{\mathbf{c}}$. The DEC algorithm is used to decode the length-n $\tilde{\mathbf{c}}$ derived in the previous step into a length-k *recovered message* $\tilde{\mathbf{m}} = (\tilde{m}(1), \ldots, \tilde{m}(k))$. The goal of the DEC algorithm is to minimize the bit-error rate (BER) which is defined as:

$$
BER = \frac{\sum_{i=1}^{k} e(m(i), \tilde{m}(i))}{k} \quad (10)
$$

where $e(x, y) = 1$ for $x \neq y$, and $e(x, y) = 0$ for $x = y$.

4.3 IPD Generator for an i.i.d. traffic model

The IPDG is responsible for generating IPDs according to a given traffic model in a pseudo-random manner. Using a secret key, K_{IPDG}, the sender and receiver are able to use IPDG to generate the same IPDs sequences, while it is cryptographically infeasible for a third party to generate the same sequences of IPDs. The IPDG can be set up to generate any kind of traffic in a random manner that allows the CoCo covert channel to be implemented for different types of traffic. For the sake of simplicity we design an IPDG for generating i.i.d. network traffic. In fact, i.i.d. traffic models are used in many analysis of network traffic and they constitute fundamental elements of many advanced network traffic models.

The IPDG uses a keyed cryptographically secure pseudo-random number generator (CSPRNG), being known to anyone including the attacker. However, the key of this SCPRNG, K_{IPDG}, is only shared between the sender S and the receiver R. Also, S and R agree on the cumulative density function (CDF) of the legitimate traffic, $g(\cdot)$. To generate the i^{th} IPD value, $d(i)$, the IPDG uses the CSPRNG along with its key K_{IPDG} to draw a number $u(i)$ *uniformly* at random from the range of $[0, 1]$. The IPD value $d(i)$ is then generated as:

$$d(i) = g^{-1}(u(i)) \tag{11}$$

where $g^{-1}(\cdot)$ is the inverse of the CDF function $g(\cdot)$. Note that since $g(\cdot)$ is a one-to-one function with output range of $[0, 1]$ its inverse function g^{-1} is also a one-to-one function with input domain of $[0, 1]$. It is easy to show that the IPD sequence $\mathbf{d}$ generated in this manner has an empirical distribution according to the CDF function $g(\cdot)$. A more elaborated approach can be taken to generate IPDs according to non-i.i.d. traffic models.

5 CoCo performance for different coding schemes

A covert timing channel can be considered as a noisy communication channel for the cover message. The use of the ENC/DEC encoding algorithms is to improve the detection performance of the covert message at the receiver. In this paper, we consider the use of different types of linear encoding schemes. The linear codes are classified into two main groups: *block codes* and *convolutional codes.* A third class of linear codes is derived by combining block codes and convolutional codes, the *Turbo codes*, which are known to approach the channel capacity. We also consider *Low-density parity check* (LDPC) codes, another class of capacity-approaching codes.

Simulation methodology In each of the simulations a random covert message is generated with a length appropriate for the selected ENC algorithm. The random message is, then, used to generate the covert traffic using the CoCo sender scheme described in Section 4.1. In order to simulate the effect of the channel noise we randomly select samples of network jitter from a large database and apply them to the IPDs of the covert traffic (note that as discussed in Section 3.2 we consider the channel noise to be an amplified version of the natural network noise). The jitter database is collected over the Planetlab [1] and contains 100000 packets (jitters have an average standard deviation of approximately $12msec$). Finally, a receiver R uses the receiver scheme mentioned in Section 4.2 to extract the covert message bits from the perturbed covert traffic. Note that for the sake of simulations we do not need to generate and add the IPDs to the encoded message, as the IPDs are regenerated by the receiver and are canceled out from the received noisy message before performing the decoding algorithm.

5.1 Block codes

Reed-Solomon (RS) Codes Reed-Solomon (RS) codes [13] are a class of linear block correcting-codes that are maximum distance separable (MSD), e.g., they meet the equality criteria of the *singleton bound* [13]. RS codes have been used in satellite communications for many years because of their strength regarding bursty errors. For an (n, k) RS code, each code symbol is m bits, where $n = 2^m - 1$ is the size of the coded message (e.g., an n-bit RS code consists of $m \times n$ binary bits).

Table 1 shows the detection performance of the CoCo scheme using RS encoders with different parameters (each simulation is run 1000 times and the gain of the covert scheme is $\gamma = 0$). Instead of BER, for evaluating RS codes we use a similar metric, *Block Error Rate* (BLER). This is because each error in an RS code affects a whole block of data. As can be seen from the table, decreasing the rate of the RS encoder improves the detection performance of the covert channel, as more redundant bits are inserted to compensate for the channel noise. Note that larger symbol size m requires more processing resources for the encoder and decoder; hence for similar BLER and rate a code with smaller m is preferred. To illustrate the effect of covert gain on the detection performance we also run the simulations for different values of γ. Table 2 shows the detection results for a $(7, 3)$ RS code. As can be seen, increasing γ significantly improves the detection performance. In Section 6, we investigate the effect of the gain on the undetectability of the CoCo covert channel.

Golay codes Golay codes [13] are one of the few existing *perfect codes*, i.e., they meet the equality criteria of the *hamming bound* [13]. Unfortunately, there are only two instances of the Golay codes: a binary $(23, 12)$ code, and a ternary $(11, 6)$ code. We use the binary Golay code of $(23, 12)$ which is able to correct 3 errors in a block of 23 encoded bits. Considering the high running speed of Golay codes we concatenate them with simple *Hamming* codes [13] in order to improve the CoCo robustness. Tables 3 and 4 illustrate the detection performance achieved by concatenating the binary Golay code with 2-bit and 3-bit parity check codes, respectively. A 2-bit parity check reduces the rate from 12/23 to 10/23, however

Table 1. RS codes for different parameters (1000 runs, $\gamma = 0$).

rate (r)	m	k	n	BLER
0.57	3	4	7	0.363
0.43	3	3	7	0.136
0.29	3	2	7	0.135
0.14	3	1	7	0.029
0.26	4	4	15	0.155
0.20	4	3	15	0.074
0.13	4	2	15	0.060
0.07	4	1	15	0.018
0.16	5	5	31	0.087

Table 2. BLER of the (7,3) RS code for different gains (1000 runs) ($r = 0.43$).

γ	BLER
-1	0.129
0	0.036
1	0.001
1.6	0.008
2	0.0002

significantly improves the BER. As an example, for $\gamma = 0$ the the BER is reduced from 0.13 to 0.02. The results are even better using 3 bits of parity.

Table 3. Binary $(23, 12)$ Golay code and 2-bits parity check ($r \approx 0.43$)

γ	Correct blocks	Detected errors	BER
-1	0.4712	0.8078	0.1016
0	0.8637	0.8437	0.0213
1	0.9827	0.8959	0.0018
1.6	0.9950	0.9400	0.0003
2	0.9989	1.0000	0.0000

Table 4. Binary $(23, 12)$ Golay code and 3-bits parity check ($r \approx 0.39$)

γ	Correct blocks	Detected errors	BER
-1	0.4712	0.9030	0.0504
0	0.8639	0.9213	0.0107
1	0.9853	0.9863	0.0002
1.6	0.9964	0.9722	0.0001
2	0.9989	1.0000	0.0000

5.2 Convolutional codes

Convolutional codes are another class of linear error-correcting codes that have use in many different applications [13]. An (n, k) convolutional code is a device with k inputs and n outputs. The input stream of a message $\mathbf{m}$ is split into k streams entering the inputs of the encoder, and each of the n output streams is evaluated by convolving some of the input streams with a generator sequence $\mathbf{G}$. The length of the generator function is called the *constraint length* v, and $u = v - 1$ is the memory of the encoder. An easy to implement decoder for convolutional codes is an ML decoder based on the Viterbi algorithm [13].

The convolutional codes simulated in this paper use a constraint length of $v = 7$. This is a popular value which is also used in the Voyager program and also the IEEE 802.16e standard. Larger v results in more powerful codes but is only used in space missions because of the decoder's high complexity. Convolutional codes also use *puncturing* which is a method to make a k_2/n_2-rate code out of a k_1/n_1 code by deleting some of the encoded bits based on a puncturing matrix.

An M/N puncture means that out of M code bits only N bits are used. Table 5 shows the BER performance of CoCo using different convolutional codes, and Table 6 shows the results for a specific code, but for different γ.

Table 5. Average BER of Convolutional code (1000 runs, each 10000 bits) for different rates ($\gamma = 0$)

Rate (r)	k/n	puncture	BER
0.67	1/3	6/3	0.2029
0.6	1/2	6/5	0.1653
0.57	1/2	8/7	0.1514
0.5	1/2	1/1	0.1098
0.5	1/3	6/4	0.1414
0.4	1/3	6/5	0.0815
0.33	1/3	1/1	0.0413
0.25	1/4	1/1	0.0351
0.2	1/5	1/1	0.0200

Table 6. Average BER of CoCo using a Convolutional code with $k/n = 1/2$ and puncturing of 1/1 (1000 runs, each 10000 bits) ($r = 0.5$)

γ	BER
-1	0.3825
0	0.1095
1	0.0076
1.6	0.0006
2	0.0001

5.3 Turbo codes

Turbo codes are a class of high-performance error correction codes and are the first practical capacity-approaching codes [16]. A turbo code is generated by concatenating two or more *constituent* codes, where each constituent code can be a convolutional or a block code. Usually some *interleaver* reorders the data at the input of the inner encoders. Turbo codes are decoded through iterative schemes. There are two types of Turbo codes: *Block Turbo Codes* (BTC), and *Convolutional Turbo codes* (CTC). In Figure 3 we draw the BER of the CoCo covert channel using BTC and also CTC codes. The simulated BTC and CTC codes are used in the IEEE 802.16e standard. Due to the space constraints we leave the use of other convolutional codes for the future research.

5.4 Low density parity-check codes (LDPC)

First designed by Gallager in 1962, LDPC codes are linear error correcting codes which are considered as another class of capacity-approaching codes [16]. Compared with the turbo codes the LDPC codes outperform for high code rates while the turbo codes are better suited for lower code rates. The LDPC encoders are represented by randomly generated sparse parity-check matrices and their decoding is performed iteratively using message-passing decoders [16].

We simulate the CoCo covert channel using the LDPC codes used in the IEEE 802.16e standard. In particular, we use a rate 0.5 LPDC code with $k = 384$. The BER performance of this covert channel is illustrated in Figure 3; we leave the simulation of other LDPC codes for the future research due to the space constraints.

5.5 Comparisons and tradeoffs

Figure 3 shows the average BER of the CoCo covert scheme using different coding algorithms and for different values of γ. The coding schemes are selected such that they result in a covert channel with a rate close to $r = 0.5$ (note that not all the codes can be designed for an exact rate of $r = 0.5$). As can be seen, the BER performance depends on the type of the coding scheme used by the CoCo. As an example, the CoCo covert channel using "Goaly(23,12)-Parity 2" outperforms the one using "RS(7,3)" code, even though they both have a rate of $r = 0.43$. We observe that in all of the cases, for a given rate, increasing γ reduces the BER of the CoCo. We also observe that for a given rate the choice of the coding algorithm depends on the gain γ. As an instance, the "Conv(2,1)-no Punc" code outperforms all of the 0.5-rate codes for gains larger than 1, but is outperformed by all of them for $\gamma \leq 0$.

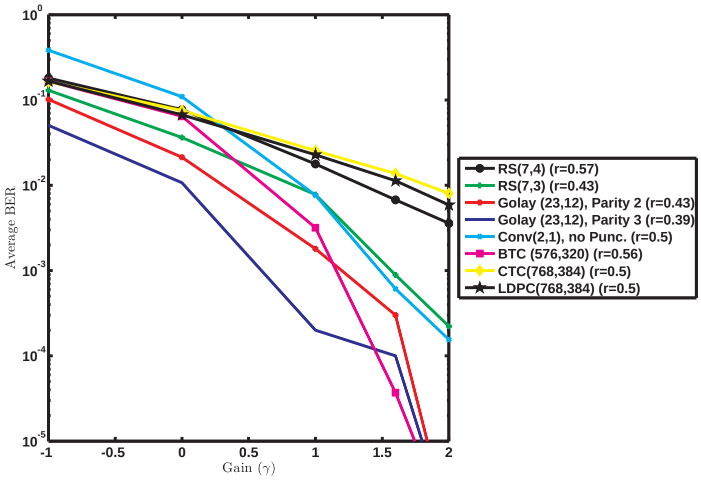

Fig. 3. The average BER of the CoCo using different coding schemes.

Unlike previous covert channels that the BER performance depends on the noise power [20, 14, 15], the BER performance of the CoCo only depends on the signal-to noise ratio of the covert channel, i.e., γ (see equation (9)). This, unlike the other schemes, enables CoCo to provide similar performance for different noise powers by adjusting the covert channel amplitude a. In contrast, other schemes lose performance as the channel noise power increases [20, 14, 15]. For a covert rate of $r = 0.5$ the covert channel of [15] results in BER rates varying from 0.04 to 0.16 for a channel with Gaussian noise $N(m_N, \sigma_N)$ where σ_N varies between $50ms$ and $500ms$. The BER performance is even worse for the covert channel of [20], as compared in [15]. [14] also results in BERs of approximately 0.07 to 0.32 for a gaussian noise with $1ms \leq \sigma_N < 20ms$. On the other side, as mentioned above the BER performance of CoCo does not depend on the noise power, but on the gain of the covert channel. For a similar rate of $r = 0.5$ CoCo can achieve BER rates less than 10^{-4} for a gain parameter of 2. In fact, the gain parameter makes a tradeoff between the robustness and undetectability of the CoCo covert scheme. Larger γ reduces the BER, hence improves the robustness of the CoCo, while degrades the undetectability of CoCo as discussed in Section 6. In fact the CoCo covert channel sacrifices the provable undetectability achieved by recent research [15] for a better robustness/rate performance and a practical undetectability.

Another feature of the CoCo covert channel is being adjustable: based on the application of the covert channel and the adversarial model one can tradeoff undetectability, rate, and robustness of the CoCo covert channel. As discussed in Section 6 the choice of γ trades off the undetectability and robustness of the CoCo covert channel. Also, for a specific coding scheme reducing the rate improves the covert channel robustness.

6 Undetectability analysis

We use the two-sample Kolmogorov-Smirnov (K-S) test [6] to evaluate the undetectability of the CoCo covert channel. We simulate the CoCo covert channel for sending SSH covert traffic and use the K-S test to distinguish between covert SSH traffic and legitimate SSH traffic. To model the legitimate traffic we use SSH traces collected by the CAIDA project from its equinix-chicago monitor — an OC192 link of a Tier 1 ISP — in January 2009 [22]. Our evaluations show that 84.6% of the SSH flows have a flow rate almost uniformly distributed between $0.2pps$ and $4.2pps$. We select 100 SSH flows with rates uniformly distributed within this range to represent our sample for the legitimate traffic, as required by the two-sample K-S tests. Each of the selected flows have at least 100 packets.

We then use CoCo to generate the covert traffic. The IPDG of CoCo simply models each SSH flow as a Poisson process with a rate randomly selected from the range of the samples, i.e., $[0.2pps, 4.2pps]$. Note that a more complicated traffic model can be generated in order to better match the behavior of a certain traffic, e.g., by matching the statistical behavior of the legitimate traffic [14]. Each flow is then generated as described in Section 4.3 and is used to modulate the covert message. Also, we use the same IPDG to generate legitimate traffic for the target traffic.

For different values of γ we run the two-sample K-S test between the traffic sample and the CoCo covert flows. Also, we run the same K-S tests between the legitimate flows and the traffic sample. We use the K-S test to distinguish between the legitimate traffic and the CoCo covert traffic by setting up a threshold for the K-S test, η_{KS}. If the K-S test of a flow passes η_{KS} the flow is declared covert, otherwise legitimate. The test produces a *false alarm* if the K-S test result is higher than the η_{KS} for a legitimate traffic, and produces a *miss* if a test value is smaller than η_{KS} for a covert flow. For some value of η_{KS}, the K-S test results in the same rates of false alarms and misses; we call this error rate as the *Cross-Over Error Rate (COER)* of the K-S test. A good test should result in very small COER rates, e.g., orders of 10^{-2}, while a bad test has COER values close to 0.5 (a random guess has a COER of 0.5). Table 7 shows the COER of the K-S test for our simulations. As can be seen, even for $\gamma = 2$ the K-S test is very poor in distinguishing between legitimate traffic and the CoCo covert traffic. In fact, this gives a *practical* undetectability as compared to the *provable* undetectability provided by [15]. In many applications of the covert channels a practical undetectability is sufficient. By adjusting the γ parameter the CoCo covert channel can be designed such that it achieves the undetectability requirements for a specific application .

Table 7. COER of the K-S in detecting the CoCo covert flows.

γ	-1	0	1	1.6	2
K-S test	0.4690	0.4660	0.3390	0.3480	0.3700

7 Conclusions

In this paper, we design CoCo, an adjustable framework for covert timing channels. Using efficient coding algorithms we show that CoCo can reliably transfer covert messages with bit error rates as low as 10^{-4}, while remaining practically undetectable. Also, the robustness of the CoCo covert channel depends on the signal-to-noise ratio, not the noise power, making it suitable for establishing very noisy covert channels.

References

1. Bavier, A., Bowman, M., Chun, B., Culler, D., Karlin, S., Muir, S., Peterson, L., Roscoe, T., Spalink, T., Wawrzoniak, M.: Operating systems support for planetary-scale network services. In: Morris, R., Savage, S. (eds.) Symposium on Networked Systems Design and Implementation. pp. 253–266. USENIX (Mar 2004)
2. Berk, V., Giani, A., Cybenko, G.: Detection of covert channel encoding in network packet delays. Tech. Rep. TR2005-536, Dartmouth College, Computer Science, Hanover, NH (Aug 2005), `http://www.cs.dartmouth.edu/reports/TR2005-536-rev1.pdf`
3. Cabuk, Brodley, Shields: IP covert timing channels: Design and detection. In: SIGSAC: 11th ACM Conference on Computer and Communications Security. ACM SIGSAC (2004)
4. Cabuk, S.: Network covert channels: Design, analysis, detection, and elimination (Jan 2006), `http://docs.lib.purdue.edu/dissertations/AAI3260014`
5. Department of Defense: DoD 5200.28-STD: Department of Defense (DoD) Trusted Computer System Evaluation Criteria (TCSEC) (1985)
6. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. Wiley (2001), `http://www.rii.ricoh.com/~stork/DHS.html`

7. Gianvecchio, S., Wang, H.: Detecting covert timing channels: an entropy-based approach. In: Ning, P., di Vimercati, S.D.C., Syverson, P.F. (eds.) ACM Conference on Computer and Communications Security. pp. 307–316. ACM (2007), http://doi.acm.org/10.1145/1315245.1315284
8. Gianvecchio, S., Wang, H., Wijesekera, D., Jajodia, S.: Model-based covert timing channels: Automated modeling and evasion. In: Lippmann, R., Kirda, E., Trachtenberg, A. (eds.) Recent Advances in Intrusion Detection, 11th International Symposium, RAID 2008, Cambridge, MA, USA, September 15-17, 2008. Proceedings. Lecture Notes in Computer Science, vol. 5230, pp. 211–230. Springer (2008), http://dx.doi.org/10.1007/978-3-540-87403-4_12
9. Giffin, Greenstadt, Litwack, Tibbetts: Covert messaging through TCP timestamps. In: International Workshop on Privacy Enhancing Technologies (PET), LNCS. vol. 2 (2002)
10. Giles, J., Hajek, B.: An information-theoretic and game-theoretic study of timing channels. IEEE Transactions on Information Theory 48(9), 2455–2477 (2002)
11. Girling, C.G.: Covert channels in LAN's. IEEE Transactions in Software Engineering SE-13(2), 292–296 (Feb 1987)
12. Handel, Sandford: Hiding data in the OSI network model. In: IWIH: International Workshop on Information Hiding (1996)
13. van Lint, J.H.: Introduction to Coding Theory (third edition). Springer Verlag, Berlin (D) (1998)
14. Liu, Y., Ghosal, D., Armknecht, F., Sadeghi, A.R., Schulz, S., Katzenbeisser, S.: Hide and seek in time - robust covert timing channels. In: Backes, M., Ning, P. (eds.) ESORICS. Lecture Notes in Computer Science, vol. 5789, pp. 120–135. Springer (2009), http://dx.doi.org/10.1007/978-3-642-04444-1
15. Liu, Y., Ghosal, D., Armknecht, F., Sadeghi, A.R., Schulz, S., Katzenbeisser, S.: Robust and undetectable steganographic timing channels for i.i.d. traffic. In: Böhme, R., Fong, P.W.L., Safavi-Naini, R. (eds.) Information Hiding. Lecture Notes in Computer Science, vol. 6387, pp. 193–207. Springer (2010), http://dx.doi.org/10.1007/978-3-642-16435-4
16. MacKay, D.: Information Theory, Inference, and Learning Algorithms (Sep 2003)
17. Murdoch, S.J., Lewis, S.: Embedding covert channels into TCP/IP. In: Barni, M., Herrera-Joancomartí, J., Katzenbeisser, S., Pérez-González, F. (eds.) Information Hiding. Lecture Notes in Computer Science, vol. 3727, pp. 247–261. Springer (2005), http://dx.doi.org/10.1007/11558859_19
18. Padlipsky, M., Snow, D., Karger, P.: Limitations of end-to-end encryption in secure computer networks. Tech. Rep. ESD TR-78-158,, Mitre Corporation (1978)
19. Rowland, C.H.: Covert channels in the TCP/IP protocol suite. First Monday 2(5) (1997), http://firstmonday.org/issues/issue2_5/rowland/index.html
20. Sellke, S.H., Wang, C.C., Bagchi, S., Shroff, N.B.: Tcp/ip timing channels: Theory to implementation. In: INFOCOM. pp. 2204–2212. IEEE (2009)
21. Shah, G., Molina, A., Blaze, M.: Keyboards and covert channels. In: Proceedings of the 15th conference on USENIX Security Symposium - Volume 15. USENIX Association, Berkeley, CA, USA (2006), http://portal.acm.org/citation.cfm?id=1267336.1267341
22. Walsworth, C., Aben, E., kc claffy, Andersen, D.: The CAIDA anonymized 2009 Internet traces—January. http://www.caida.org/data/passive/passive_2009_dataset.xml (Mar 2009)