This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2806741, IEEE Transactions on Information Forensics and Security

RNN-SM: Fast Steganalysis of VoIP Streams Using Recurrent Neural Network

Zinan Lin, Yongfeng Huang, *Senior Member, IEEE*, and Jilong Wang

***Abstract*—Quantization index modulation (QIM) steganography makes it possible to hide secret information into VoIP streams, which could be utilized by unauthorized entities to set up covert channels for malicious purpose. Detecting short QIM steganography samples, as is required by real circumstances, remains an unsolved challenge. In this paper, we propose an effective online steganalysis method to detect QIM steganography. We find out four strong codeword correlation patterns in VoIP streams, which will be distorted after embedded with hidden data. To extract those correlation features, we propose Codeword Correlation Model (CCM), which is based on recurrent neural network (RNN). Furthermore, we propose Feature Classification Model (FCM) to classify those correlation features into cover speech and stego speech categories. The whole RNN Based Steganalysis Model (RNN-SM) is trained in supervised learning framework. Experiments show that on full embedding rate samples, RNN-SM is of high detection accuracy, which remains over 90% even when sample is as short as 0.1s, and is significantly higher than other state-of-the-art methods. For the challenging task of conducting steganalysis towards low embedding rate samples, RNN-SM also achieves a high accuracy. The average testing time for each sample is below 0.15% of sample length. These clues show that RNN-SM meets the short sample detection demand and is a state-of-the-art algorithm for online VoIP steganalysis.**

***Index Terms*—steganalysis, steganography, information hiding, covert channel, recurrent neural network.**

I. Introduction

Steganography is the technique that hides secret information into digital carriers in undetectable ways. It can be used for setting up covert channels and sending concealed information over the Internet between two parties whose connection is being restricted or monitored. The carriers could be any kind of data streams transferred over the Internet, such as images [1], texts [2], [3], and protocols [4]. In recent years, Voice-over IP (VoIP) [5], a protocol for making high quality calls via the Internet, facilitates the popularity of a number of voice-based applications such as mobile VoIP (mVoIP) and voice over instant messenger (VoIM), which drives many researches on VoIP-based steganograpy [6], [7], [8], [9], [10], [11], [12], [13]. Compared with traditional carriers, VoIP has many essential advantages. Its massive payloads provide great information hiding capacity and high covert bandwidth. Its instantaneity enables real-time steganography. And its widespread popularity makes it possible to be deployed in many different scenes. Therefore, VoIP-based steganography turns out to be a good option for secure communication. However, hackers, terrorists, and other lawbreakers may use this technique for malicious intents. For example, they can smuggle unauthorized data or send virus control instructions without being detected by network surveillance. Hence, it is important to develop countermeasures to effectively detect steganography. And this technique is called steganalysis.

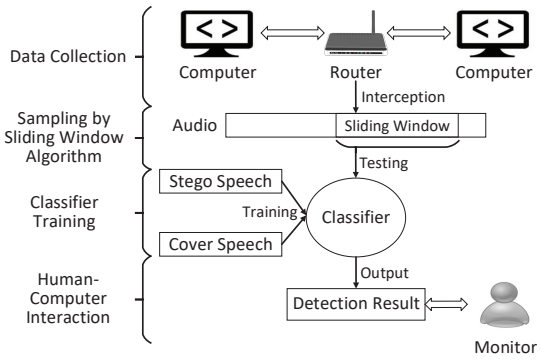


Fig. 1. VoIP Steganalysis Scenario

There are two types of speech coders in VoIP scenarios: waveform coders (e.g. G.711, G.726) and vocoders (e.g. G.723, G.729, iLBC). Compared with waveform coders which are based on quantization values of the original speech signal, vocoders try to minimize the decoding error by analysis-by-synthesis (AbS) framework and can achieve high compression ratio while preserving superb voice quality. Therefore, vocoders have been widely used in VoIP applications and their related steganography techniques are among research focuses. For example, based on quantization index modulation (QIM) [14], researchers proposed algorithms to embed secret information in vocoder streams by changing the process of vector quantization of linear predictive coding (LPC) [11], [12]. The resultant error is theoretically bounded and experiments show that QIM based steganography can achieve state-of-the-art results [11], [12]. In this paper, we focus on detecting QIM based steganography.

The classic VoIP steganalysis scenario is shown in Figure 1. Two suspect entities are communicating through a VoIP channel (e.g. making a VoIP phone call). We set up a traffic monitor on the router that the communication must go through. The collected network packets are being assembled into VoIP streams in real time. At the same time, we use sliding window algorithm [15] with a window of length l and step s to sample the latest segment, which is sent to the pre-trained classifier to get the online detection results. The online detection results are sent to the monitor for further actions (e.g. reporting to administrators and cutting off the connection).

All the above steganalysis actions must be done in real time for the following reasons. First, to minimize losses

from potential malicious actions, we need to cut off the covert channel as soon as possible if it exists. The essential step is to know whether there is steganography happening and the detection delay determines how soon we can react. Online detection is therefore a must. Second, because of the popularity of VoIP applications, there are a large volume of VoIP connections on the Internet. For each connection, the size of the whole VoIP stream is unpredictable. Therefore, it is impractical to cache the data streams and do offline detection. When deploying online steganalyis, we can not only react to malicious steganography more quickly, but also save memory resources. To enable online detection, the time for classifying sample of length l need to be shorter than step s. Taking overheads into account, the time for classification must be as short as possible. This is the first requirement for VoIP steganalysis algorithms.

We should also notice that, to avoid being detected, steganography applications do not embed secret data into VoIP streams all the time. Instead, in many circumstances, they only do information embedding in short periods and keep inactive for most of the time. If the sample we extract for classification is too long, it will be filled with a mixture of embedding and non-embedding frames, which impairs detection accuracy. To achieve successful detection, the window length l must be as short as possible. This poses the second requirement for VoIP steganalysis that it must be able to detect short samples. However, existing steganalysis methods towards QIM based steganography [16], [17] cannot achieve effective detection results when samples are short.

In this paper, we design a recurrent neural network (RNN) based model for steganalysis tasks. The contributions of this work are:

- We conduct a detailed analysis of codeword correlation in VoIP streams by summarizing correlations into four categories and proposing a metric to evaluate their existence and importance, which provides helpful evidence for steganalysis.
- To the best of our knowledge, we are the first to introduce RNN into VoIP steganalysis task. Experiment results verify the practicability of this mechanism and indicate that RNN is a powerful alternative to traditional methods when solving similar problems.
- The detection accuracy of our proposed steganalysis method is above 90% even if the sample is as short as 0.1s, and its accuracy is significantly higher than other state-of-the-art methods on short samples. In addition, the average detection time for each sample is below 0.15% of the sample length. These features indicate that our method can be effectively deployed for online VoIP steganalysis.

The rest of the paper is structured as follows. In Section II, we introduce some background knowledge. Related work is introduced in Section III. In Section IV, our proposed steganalysis method is presented. Experiments and discussions are shown in Section V. Finally, we give the conclusion and the future work in Section VI.

II. Background

In this section, we introduce some preliminary knowledge for our algorithm: QIM based steganography and LPC.

A. QIM Based Steganography

QIM was first proposed by Chen and Wornell [14]. It embeds data by changing the quantization process when encoding a digital media such as image, text, audio, and video.

During the encoding process, there are many coefficients that need to be quantized. In the normal procedure, for the coefficient vector $\boldsymbol{x}$, we will choose the closest vector from a codebook D as its representative:

$$\mathrm{Q}(\boldsymbol{x}) = \arg\min_{\boldsymbol{y}\in D} \|\boldsymbol{x}-\boldsymbol{y}\| \tag{1}$$

QIM modifies this procedure. It first divides the codebook D into sub-codebooks $C = \{C_1, C_2, ..., C_n\}$, which satisfies

$$D = \bigcup_{i=1}^{n} C_i$$

and

$$\forall i \neq j, \quad C_i \cap C_j = \emptyset$$

Assume that the secret information we want to transfer is from the set $S = \{s_1, s_2, ..., s_n\}$. We further define an embedding projection function f as a one-to-one mapping from S to C, and f^{-1} is its inverse function. When we want to quantize coefficient vector $\boldsymbol{x}$ and hide secret information s_k at the same time, we just use the sub-codebook $\mathrm{f}(s_k)$ instead of the whole codebook D:

$$\mathrm{Q}'(\boldsymbol{x}, s_k) = \arg\min_{\boldsymbol{y}\in \mathrm{f}(s_k)} \|\boldsymbol{x}-\boldsymbol{y}\| \tag{2}$$

The receiver can recover the secret information by judging to which sub-codebook the quantitative vector belongs:

$$\mathrm{R}(y) = \mathrm{f}^{-1}(C_k) \text{ where } y \in C_k \tag{3}$$

The core problem of QIM based steganography is the codebook partitioning strategy. The simplest way is to divide the codebook randomly. However, it will lead to large additional quantization distortion. Xiao proposed Complementary Neighbor Vertices (CNV) algorithm [11]. It can guarantee that every codeword and its nearest neighbor be in different sub-codebooks, so the additional quantization distortion can be bounded.

In this paper, we will take CNV algorithm as our test target, while our algorithm can be directly applied to other QIM steganalysis algorithm.

B. Linear Predictive Coding

LPC [18] has been widely used to model speech signal, and is the essential part of vocoders such as G.723 and G.729. It is based on the physical process of speech signal generation.

Speech signal is generated by organs in respiratory tract. The organs involved are lung, glottis, and vocal track. When passing through glottis, the exhaled breath from lung would turn to a periodic excitation signal. The excitation signal would then go through vocal track. We can divide vocal track into

cascaded segments, whose functions can be modeled as one-pole filters. Therefore, the function of vocal tracker can be modeled as an all-pole filter, i.e. LPC filter:

$$H(z) = \frac{1}{A(z)} = \frac{1}{1 - \sum_{i=1}^{n} a_i z^{-i}} \tag{4}$$

where a_i is the i-th order coefficient of LPC filter. Because speech signal has short-time stationarity, we can assume that LPC coefficients a_i would not change in short time. Therefore, we can divide the speech into short frames and compute the LPC coefficients respectively. Vocoders only encode the deduced LPC coefficients and excitation signals to achieve high compression ratio.

In LPC encoding, the LPC coefficients are first converted into Line Spectrum Frequency (LSF) coefficients. And the LSFs are encoded by vector quantization. Specifically, G.729 and G.723 quantizes LSFs into three codewords l_1, l_2, and l_3 using codebooks L_1, L_2, and L_3 respectively.

QIM steganography could be performed while quantizing LSFs [11]. Altered after QIM steganography, LSF quantization vectors serve as clues for steganalysis. In this paper, we propose an algorithm to detect QIM steganography on LSFs. Moreover, it is also possible to apply our algorithm on steganography on other quantization processes such as pitch period prediction [13], since pitch period prediction-based steganography uses similar way to hide data (changing quantization vectors).

III. Related Work

There has been some effort in steganalysis of digital audio. The most common way was to directly extract statistical features from the audio and then conduct classification. Mel-cepstrum was one of the statistical features that steganalysis algorithms used [19], [20]. Liu *et al.* improved this method by discovering that high-frequency components were more effective for classification [21]. The three papers above used Support Vector Machine (SVM) classifier. Other statistical features were also used. For example, Dittmann *et al.* combined features such as mean value, variance, LSB-ratio, and histogram altogether to classify the audio [22]. Avcibas *et al.* used a series of audio quality measures such as signal-to-noise ratio (SNR) and log-likelihood ratio (LLR) to detect steganography [23]. These two papers used threshold classifier.

At the observation that marginal distortion decreases under repeated embedding, Altun *et al.* watermarked the audio sample for another two times and fed the additional distortion into a neural network classifier [24]. Similarly, Ru *et al.* discovered that the variations of statistical features such as mean, variance, skewness, and kurtosis were different when conducting steganography on stego object and cover object. Therefore, they embedded random message on the audio sample and put the increment of statistical features into kernel SVM classifier [25]. Huang *et al.* applied a second steganography on compressed speech to estimate the embedding rate [26].

Neural network models were also introduced into speech steganalysis tasks. Paulin *et al.* employed deep belief networks to solve this problem [27]. They calculated Mel Frequency Cepstrum Coefficient (MFCC) and deep belief networks (DBN) served as a classifier. In another work, Paulin *et al.* used Evolutionary Algorithms (EAs) to train a Restricted Boltzmann Machines (RBMs), which classified stego and cover speech [28]. The input to RBMs was still MFCC features. Rekik *et al.* first introduced Time Delay Neural Networks (TDNN) to detect stego-speech [29]. They extracted LSF from the original audio and did the classification with TDNN. Those methods were partly inspired by the good performance of Artificial Neural Network (ANN) in other fields. However, they all firstly extracted hand-crafted features and then used ANN as classifier, which could not fully exploit ANN's capability in feature extraction. Chen *et al.* used Convolutional Neural Network (CNN) to do steganalysis tasks, and raw audio streams served as input [30].

The above speech steganalysis algorithms were universal. They extracted features from the original audio streams and therefore could be applied to almost all kinds of steganography algorithms. The weakness was that their accuracies on specific steganography were usually lower than other targeted steganalysis algorithm, for example, steganalysis towards QIM based steganography.

QIM steganography algorithms only modifies specific codewords to achieve information hiding. Extracting only those modified bits, instead of the whole audio stream, will certainly benefit the detection accuracy. The QIM steganalysis algorithms [17], [16] utilized this intuition. In [17], the author extracted the modified codewords into a data stream, and used Markov chain to model the transition pattern between successive codewords. In [16], the author further took the transition probability within a frame into consideration. Those two steganalysis algorithms achieved state-of-the-art detection results. However, in the codeword sequence, there were other correlation relationships that those two methods did not consider. The algorithm proposed in this paper has better ability to model correlation patterns by utilizing RNN model and achieves better results.

Since our proposed speech steganalysis methods involve neural network models, we are also interested in image steganalysis algorithms that use neural networks. Actually there has been a long history of utilizing neural networks for image steganalysis. However, earlier works all used hand-crafted features, and neural networks only served as classifiers [31], [32], [33], which could not make full use of the power of neural networks. Qian *et al.* first utilized CNN for image steganalysis, and proposed a unified neural network model for both feature extraction and classification [34]. Xu *et al.* later proposed another CNN-based image steganalysis model [35] by incorporating more domain knowledge. Chen *et al.* extended this work from spatial domain to JPEG domain [36]. Ye *et al.* further proposed a new CNN-based image steganalysis model with some novel ideas. They used precomputed weights in the first layer for faster convergence, introduced truncated linear unit (TLU) in the network, and used selection channel in training. The proposed method achieved state-of-the-art results.

IV. Steganalysis Using Recurrent Neural Network

For normal speech encoding, there exist strong correlation patterns in codewords. The correlation patterns would likely be weakened if original codewords are embedded with hidden data. Correlation patterns are consequently regarded as an indicator of steganography and could be extracted for steganalysis. RNN is supposed to be capable of exploiting codeword correlations, as its current output always takes a reference of earlier input data.

Our solution for steganalysis is applying RNN to detecting the disparities in codeword correlations. It takes the advantage that RNN could not only show temporal behavior, but integrate a variety of correlation patterns which are drawn from our analysis (Section IV-A). We propose a Codeword Correlation Model (CCM) to delineate correlations in codewords (Section IV-B). We then put forward a Feature Classification Model (FCM) for RNN to decide judge threshold of cover speech and stego speech (Section IV-C). Finally, we suggest how the two models above should be cascaded in order to construct our RNN Based Steganalysis Model (RNN-SM) (Section IV-D).

A. Codeword Correlation Analysis

First we clarify what codeword correlation is. We define $x_{i,j}$ as the i-th codeword at frame j, where $j \in [1, T]$ and T is the time duration. For G.729 and G.723, $i \in [1, 3]$, and the three codewords are from codebook L_1, L_2, and L_3 respectively. When all codewords are uncorrelated, their appearances are independent. Therefore, we have

$$\begin{aligned} \mathrm{P}(x_{i,j} = & u \text{ and } x_{k,l} = v) = \mathrm{P}(x_{i,j} = u) \cdot \mathrm{P}(x_{k,l} = v), \\ & \forall i, k \in [1, 3], j, l \in [1, T], u \in L_i, v \in L_k \end{aligned} \tag{5}$$

When the two sides of the equation are not equal, certain correlation pattern exists. For example, when the left side of the equation is of higher value than right, it means that u and v are more likely to appear in pair in the given positions. Otherwise, u and v are less likely to appear in pair in the given positions. Larger imbalance of the two sides indicates stronger correlation.

However, given only one codeword sequence, we cannot estimate the three involved possibility items. More observations are required so that we can accurately estimate those items. One solution is to consider the possibilities for multiple frame pairs where j and l have a fixed distance, instead of taking j and l as fixed frames. Specifically, we need to estimate the following three possibility items:

$$\mathrm{P}(x_{i,j} = u \text{ and } x_{k,l} = v \big| \forall l - j = \delta) \tag{6}$$

$$\mathrm{P}(x_{i,j} = u \big| \forall l - j = \delta) = \mathrm{P}(x_{i,j} = u \big| \forall j \leqslant T - \delta) \tag{7}$$

$$\mathrm{P}(x_{k,l} = v \big| \forall l - j = \delta) = \mathrm{P}(x_{k,l} = v \big| \forall l \geqslant \delta + 1) \tag{8}$$

We denote the possibility estimated from observations as $\tilde{\mathrm{P}}$. Thus, the following equation can be used to evaluate correlation:

$$\begin{aligned} & \tilde{\mathrm{P}}(x_{i,j} = u \text{ and } x_{k,l} = v \big| \forall l - j = \delta) \\ & - \tilde{\mathrm{P}}(x_{i,j} = u \big| \forall j \leqslant T - \delta) \cdot \tilde{\mathrm{P}}(x_{k,l} = v \big| \forall l \geqslant \delta + 1) \end{aligned} \tag{9}$$

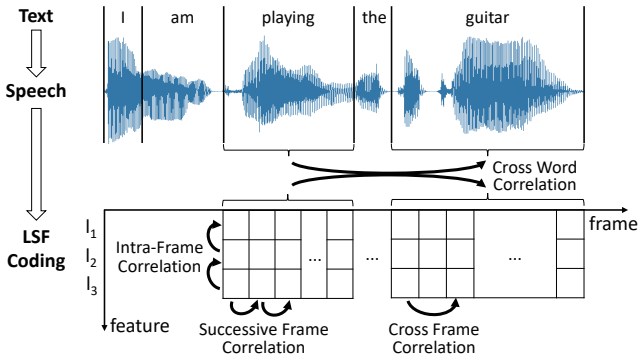


Fig. 2. Correlations Between Codewords

The state-of-the-art steganalysis algorithms [17], [16] shared the same pattern: extracting correlation features from the codewords and then feeding the features to SVM classifiers. In [17], the author modeled the sequence of codewords as a Markov chain, and transition probability from one codeword to the one that was the most likely to appear immediately behind was selected as feature in this model. In [16], the author extended the method by taking the transition probabilities between l_1, l_2, and l_3 in one frame into consideration. And the features were selected by principal component analysis (PCA).

These feature selection strategies had limitations. They only considered the codeword connections in one frame and between successive two frames. However, speech signals are highly correlated in a long time interval. Current codeword is not only determined by the previous codeword, but also influenced by the codewords appeared long before. Figure 2 explains the four kinds of correlations between codewords:

- **Successive frame correlation**
  Each codeword is computed on a short time frame (10ms for G.729, 30ms for G.723), which is comparable to the length of a phoneme in a word. The successive phonemes in a word are correlated, so that the successive codewords in the coding streams are correlated. We name this kind of correlation as successive frame correlation. To model successive frame correlation, [17], [16] both used the deduced features from transition probabilities between any two codewords, i.e. $\frac{\tilde{\mathrm{P}}(x_{i,j}=u \text{ and } x_{i,l}=v \big| \forall l-j=1)}{\tilde{\mathrm{P}}(x_{i,j}=u \big| \forall j \leqslant T-1)}$ for all i, u and v.
- **Intra-frame correlation**
  In each frame, there are three codewords: l_1, l_2, and l_3. l_1 and l_2 together compose the first five LSFs, while l_1 and l_3 together compose the last five LSFs. Therefore, l_1, l_2, and l_3 are also correlated within a frame. We name the correlations between l_1, l_2, and l_3 as intra-frame correlation. In [16], the author used the transition probabilities of $l_1 \to l_2$, $l_1 \to l_3$, and $l_2 \to l_3$ to model intra-frame correlation, i.e. $\frac{\tilde{\mathrm{P}}(x_{i,j}=u \text{ and } x_{k,j}=v \big| \forall j)}{\tilde{\mathrm{P}}(x_{i,j}=u \big| \forall j)}$ for all u, v, and for (i, k) in $\{(1, 2), (1, 3), (2, 3)\}$.
- **Cross frame correlation**
  There are multiple phonemes in a word. Different words have different phoneme transition patterns. Therefore,

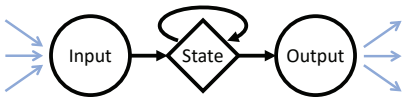


Fig. 3. The Structure of RNN Unit

current phoneme cannot be fully determined by the previous phoneme. Instead, we should take all previous appeared phonemes in this word into consideration. Cross frame correlation means the correlations between non-adjacent codewords in a word.

- **Cross word correlation**
  Codeword streams are essentially generated from sentences. It is known to all that words are highly correlated with each other on the sentence level. Therefore, their corresponding codewords are also correlated. In other words, a codeword from a word is not only determined by other codewords from the same word, but also determined by codewords from other words in the whole context. We name the correlation of codewords from different words as cross word correlation.

The first two correlations explain the local features while the last two correlations describe the global features. In [17], [16], the authors simplified the problem by only keeping local features, i.e. successive frame correlation and intra-frame correlation, and omitting global ones, i.e. cross frame correlation and cross word correlation, which would harm the detection accuracy to some extent.

In recent years, stimulated by big data, ANN was successfully used in many pattern recognition and artificial intelligence tasks. It is composed of a network of neuron-like units. At any time step, each non-input neuron computes its current output as a nonlinear function of the weighted sum of the activations of all units from which it receives inputs. Many ANNs, like CNN and multi-layer perceptron (MLP), are in a feedforward structure, which means the output at a time is only determined by its current input. RNN, on the other hand, is able memorize the past inputs by an internal state in the neuron as shown in Figure 3. The memory ability makes RNN very suitable for modeling long time series like audio. RNN has been widely and successfully used in many audio related tasks, such as speech recognition [37], natural language processing [38], phoneme classification [39], etc. But to the best of our knowledge, RNN has never been used in audio steganalysis tasks.

Because RNN can generate outputs with not only the information of the latest two frames, but also the information of all past frames, it is possible for RNN to consider all the four kinds of correlations at the same time. Long-Short Term Memory (LSTM) [40] is a refined version of RNN. It is capable of learning long-term dependencies in time series. This feature suits our task well. We use it to model the correlations of speech codewords. The model is further explained in the next subsection.

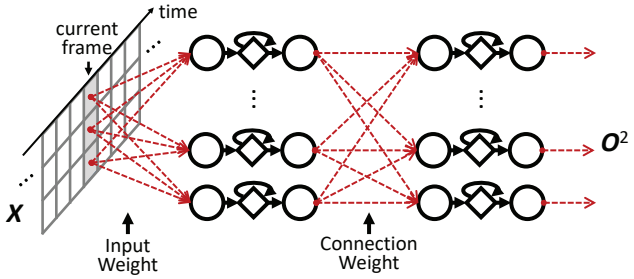


Fig. 4. Codeword Correlation Model

B. Codeword Correlation Model

For simplicity, we first introduce some notations. Assume $\boldsymbol{M}$ is a matrix and $m_{i,j}$ is its element. We define $\boldsymbol{M}_{i,a:b}$ as the row vector composed by the elements at row i and column a to b of $\boldsymbol{M}$, i.e.

$$\boldsymbol{M}_{i,a:b} = [m_{i,a}, m_{i,a+1}, ..., m_{i,b}]$$

and $\boldsymbol{M}_{a:b,i}$ as the column vector composed by the elements at column i and row a to b of $\boldsymbol{M}$, i.e.

$$\boldsymbol{M}_{a:b,i} = [m_{a,i}, m_{a+1,i}, ..., m_{b,i}]^{\mathrm{T}}$$

and $\boldsymbol{M}_{a:b,c:d}$ as the matrix composed by the elements at row a to b and column c to d of $\boldsymbol{M}$, i.e.

$$\boldsymbol{M}_{a:b,c:d} = [\boldsymbol{M}_{a:b,c}, \boldsymbol{M}_{a:b,c+1}, ..., \boldsymbol{M}_{a:b,d}]$$

Assume $\boldsymbol{V}$ is a vector and v_i is its elements. We define $\boldsymbol{V}_{a:b}$ as the row vector composed by a-th to b-th elements of $\boldsymbol{V}$, i.e.

$$\boldsymbol{V}_{a:b} = [v_a, v_{a+1}, ..., v_b]$$

We pack all codewords of a speech sample which has T frames into a codeword matrix $\boldsymbol{X}$ as

$$\boldsymbol{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & ... & x_{1,T} \\ x_{2,1} & x_{2,2} & ... & x_{2,T} \\ x_{3,1} & x_{3,2} & ... & x_{3,T} \end{bmatrix} \tag{10}$$

where $x_{1,i}$, $x_{2,i}$, $x_{3,i}$ stand for l_1, l_2, l_3 coefficients of the i-th frame respectively. For G.729 vocoder, $x_{1,i}$, $x_{2,i}$, and $x_{3,i}$ are of 7 bits, 5 bits, and 5 bits respectively. For G.723 vocoder, $x_{1,i}$, $x_{2,i}$, and $x_{3,i}$ are all of 8 bits. Because steganography only changes l_1, l_2, and l_3, $\boldsymbol{X}$ contains the full information for steganalysis. It is presented as the input of our CCM.

As stated before, LSTM has good ability to model time series. We use LSTM to build our CCM. We denote the transfer function of LSTM units by f. In other words, when the input sequence is $\boldsymbol{Q} = [q_1, q_2, ..., q_t]$, the output sequence $\boldsymbol{R} = [r_1, r_2, ..., r_t]$ satisfies

$$r_i = \mathrm{f}(\boldsymbol{Q}_{1:i})$$

The whole structure of CCM is shown in Figure 4. CCM contains two layers of LSTM units. The first layer has n_1 LSTM units and the second layer has n_2 LSTM units. We name the set of LSTM units in the first layer as $U_1 = \{u_{1,1}, u_{1,2}, ..., u_{1,n_1}\}$ and the set of LSTM units in the second layer as $U_2 = \{u_{2,1}, u_{2,2}, ..., u_{2,n_2}\}$.

Between input codewords and LSTM units in the first layer, there are Input Weights (IW) which define how much we should value each codeword. IW is presented in a $3 \times n_1$ matrix $\boldsymbol{A}$:

$$\boldsymbol{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & ... & a_{1,n_1} \\ a_{2,1} & a_{2,2} & ... & a_{2,n_1} \\ a_{3,1} & a_{3,2} & ... & a_{3,n_1} \end{bmatrix} \tag{11}$$

For each LSTM unit $u_{1,i}$, there are three associated weights: $a_{1,i}$, $a_{2,i}$, and $a_{3,i}$, which will be multiplied to the three input codewords respectively to formulate the final input value at each time step. To be more specific, the input value for $u_{1,i}$ at time t is

$$e^1_{i,t} = a_{1,i}x_{1,t} + a_{2,i}x_{2,t} + a_{3,i}x_{3,t} \tag{12}$$

We define $\boldsymbol{E}^1$ as the matrix packing all $e^1_{i,t}$ together:

$$\boldsymbol{E}^1 = \begin{bmatrix} e^1_{1,1} & e^1_{1,2} & ... & e^1_{1,T} \\ e^1_{2,1} & e^1_{2,2} & ... & e^1_{2,T} \\ ... & & & \\ e^1_{n_1,1} & e^1_{n_1,2} & ... & e^1_{n_1,T} \end{bmatrix} \tag{13}$$

Then the output value of $u_{1,i}$ at time t is

$$\begin{aligned} o^1_{i,t} &= \mathrm{f}(\boldsymbol{E}^1_{i,1:t}) \\ &= \mathrm{f}(a_{i,1}\boldsymbol{X}_{1,1:t} + a_{i,2}\boldsymbol{X}_{2,1:t} + a_{i,3}\boldsymbol{X}_{3,1:t}) \end{aligned} \tag{14}$$

And we define $\boldsymbol{O}^1$ as the matrix gathering all first-layer outputs from start to end, i.e.

$$\boldsymbol{O}^1 = \begin{bmatrix} o^1_{1,1} & o^1_{1,2} & ... & o^1_{1,T} \\ o^1_{2,1} & o^1_{2,2} & ... & o^1_{2,T} \\ ... & & & \\ o^1_{n_1,1} & o^1_{n_1,2} & ... & o^1_{n_1,T} \end{bmatrix} \tag{15}$$

At every time step, each unit will give a separate output based on all codewords in the past. This first layer serves as the step of extracting preliminary features $\boldsymbol{O}^1$.

Inspired by the common sense that a deeper network usually yields a better modeling ability, we stack the network with another layer of LSTM units. Between the two layers of LSTM units, there are Connection Weights (CW) which recompose preliminary features. CW is represented as an $n_1 \times n_2$ matrix $\boldsymbol{B}$:

$$\boldsymbol{B} = \begin{bmatrix} b_{1,1} & b_{1,2} & ... & b_{1,n_2} \\ b_{2,1} & b_{2,2} & ... & b_{2,n_2} \\ ... & & & \\ b_{n_1,1} & b_{n_1,2} & ... & b_{n_1,n_2} \end{bmatrix} \tag{16}$$

For each LSTM unit $u_{2,i}$, there are n_1 associated weights: $b_{1,i}$, $b_{2,i}$, ..., $b_{n_1,i}$, which will be multiplied to the outputs of previous layer to form the final input. To be more specific, the input value for $u_{2,i}$ at time t is

$$\begin{aligned} e^2_{i,t} &= \sum_{j=1}^{n_1} o^1_{j,t} b_{j,i} \\ &= {\boldsymbol{O}^1_{1:n_1,t}}^{\mathrm{T}} \boldsymbol{B}_{1:n_1,i} \end{aligned} \tag{17}$$

We define $\boldsymbol{E}^2$ as the matrix packing all $e^2_{i,t}$ together:

$$\boldsymbol{E}^2 = \begin{bmatrix} e^2_{1,1} & e^2_{1,2} & ... & e^2_{1,T} \\ e^2_{2,1} & e^2_{2,2} & ... & e^2_{2,T} \\ ... & & & \\ e^2_{n_2,1} & e^2_{n_2,2} & ... & e^2_{n_2,T} \end{bmatrix} \tag{18}$$

Then the output of $u_{2,i}$ at time t is:

$$\begin{aligned} o^2_{i,t} &= \mathrm{f}(\boldsymbol{E}^2_{i,1:t}) \\ &= \mathrm{f}({\boldsymbol{B}_{1:n_1,i}}^{\mathrm{T}} \boldsymbol{O}^1_{1:n_1,1:t}) \end{aligned} \tag{19}$$

The final output matrix $\boldsymbol{O}^2$

$$\boldsymbol{O}^2 = \begin{bmatrix} o^2_{1,1} & o^2_{1,2} & ... & o^2_{1,T} \\ o^2_{2,1} & o^2_{2,2} & ... & o^2_{2,T} \\ ... & & & \\ o^2_{n_2,1} & o^2_{n_2,2} & ... & o^2_{n_2,T} \end{bmatrix} \tag{20}$$

contains the final correlation features.

CCM has the potential of modeling all four types of correlations for the following reasons. First, IW combines l_1, l_2, and l_3 together into a value which is propagated in the whole network. Different weights on l_1, l_2, and l_3 indirectly determine what combinations of l_1, l_2, and l_3 can lead to the activation of LSTM units. Intra-frame correlation is therefore taken into account. Second, with LSTM's ability of memorizing the past, every output is deduced from all past codewords. The LSTM units in first layer can directly memorize the original codewords. The LSTM units in the second can further memorize more complicate past features by receiving information from the first layer. Thus, CCM has strong ability to model patterns over time. Successive frame correlation, cross frame correlation, and cross word correlation are just correlations on different time spans. Definitely they can be modeled by CCM.

C. Feature Classification Model

We can use the features collected in $\boldsymbol{O}^2$ to classify whether the original speech has hidden data. A basic idea is to calculate the linear combination of all features. To be more specific, we define the Detection Weight (DW) as matrix $\boldsymbol{C}$ which is of $n_2 \times T$ size and the linear combination is calculated as

$$y = \sum_{i=1}^{n_2} \sum_{j=1}^{T} \boldsymbol{O}^2_{i,j} \boldsymbol{C}_{i,j} \tag{21}$$

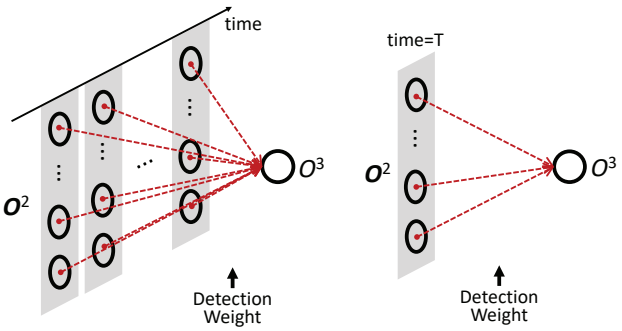
To get normalized output between $[0, 1]$, we put the value through a sigmoid function S

$$\mathrm{S}(x) = \frac{1}{1 + e^{-x}}$$

and the final output is

$$\begin{aligned} O^3 &= \mathrm{S}(y) \\ &= \mathrm{S}(\sum_{i=1}^{n_2} \sum_{j=1}^{T} \boldsymbol{O}^2_{i,j} \boldsymbol{C}_{i,j}) \end{aligned} \tag{22}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2806741, IEEE Transactions on Information Forensics and Security



(a) Full Model (b) Pruned Model

Fig. 5. Feature Classification Models

If we set the detection threshold at 0.5, the final detection result can be expressed as

$$\text{Detection Result} = \begin{cases} \text{Stego Speech} & (O^3 \geqslant 0.5) \\ \text{Normal Speech} & (O^3 < 0.5) \end{cases} \tag{23}$$

In other words, the model tries to predict the label (0 for normal, 1 for stego) for a given speech. In Section V-D, we will further discuss how the threshold will influence the results.

We name this model as full FCM. The structure is shown in Figure 5(a).

However, when the speech sequence is long, DW matrix will grow large. The training and testing process of the model will be slowed down as a result. In addition, too many coefficients will raise the possibility of overfitting. Moreover, the size of model is dependent on the length of input sequence and it will severely limit the model's practicability.

To solve these problems, we propose a pruned FCM model as shown in Figure 5(b). Notice that the final outputs at the end time T have already included all outputs at all time steps from the first layer because of LSTM's memorizing ability. Therefore, it is fair to only use $\boldsymbol{O}^2_{1:n_2,T}$ for detection and cast away all past outputs $\boldsymbol{O}^2_{1:n_2,1:T-1}$. DW now shrinks to a n_2-dimensional vector and the size of model is independent to the length of input sequence.

To be more specific, we define DW as a vector $\boldsymbol{C}$ which contains n_2 coefficients:

$$\boldsymbol{C} = [c_1, c_2, ..., c_{n_2}]^{\mathrm{T}} \tag{24}$$

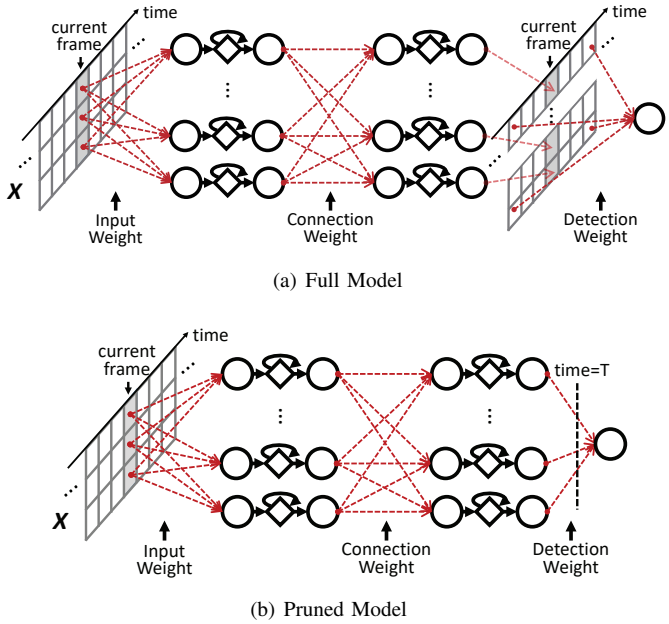
The final output is

$$\begin{aligned} O^3 &= \mathrm{S}(\sum_{i=1}^{n_2} \boldsymbol{O}^2_{i,T} c_i) \\ &= \mathrm{S}({\boldsymbol{O}^2_{1:n_2,T}}^{\mathrm{T}} \boldsymbol{C}) \end{aligned} \tag{25}$$

We will make a comparison of the full and the pruned model in Section V-C.

D. RNN Based Steganalysis Model

The final RNN-SM is constructed by cascading CCM and FCM together. Full RNN-SM and pruned RNN-SM are shown in Figure 6(a) and Figure 6(b) respectively. At each time step,



(a) Full Model

(b) Pruned Model

Fig. 6. RNN Based Steganalysis Model

we input the new l_1, l_2, and l_3 coefficients to the network. Starting from the left to the right, each LSTM unit upgrades its internal state according to the current input and outputs with a new value. For pruned RNN-SM, at the end of the sequence, the outputs from the second layer of LSTM are being forwarded to the final output node. For full RNN-SM, all outputs from the second layer of LSTM are being forwarded to the final output node. The output node gives the final detection value which is between $[0, 1]$. The final detection result can then be decided according to (23).

In RNN-SM, there are three sets of undetermined weights: IW, CW, and DW, which are presented in matrix $\boldsymbol{A}$, matrix $\boldsymbol{B}$, and matrix/vector $\boldsymbol{C}$ respectively. They need to be determined before being used for steganalysis.

To determine the weights, we follow a supervised learning framework as shown in Figure 7. First we collect a number of normal speech samples which make up the cover speech set. Each sample is further encoded with G.729 vocoder with or without QIM steganography. And then LSF codewords are extracted from the speech coding streams. For codeword segments with secret information, we assign a label 1 to them. For codeword segments without secret information, we assign a label 0 to them. Those segments will be randomly grouped into mini-batches. Each mini-batches will be inputed to RNN-SM whose weights are randomly initialized and the deviations between RNN-SM's outputs and true labels will be back-propagated to optimize the weights using Adam algorithm [41].

During the testing stage, the untested samples are being processed by similar procedure: G.729 encoding, LSF coefficient extraction, and being inputed to RNN-SM. And the final detection result is given according to (23).

Our implementation of RNN-SM can be found on https://github.com/fjxmlzn/RNN-SM/, which is based on Keras library.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2806741, IEEE Transactions on Information Forensics and Security

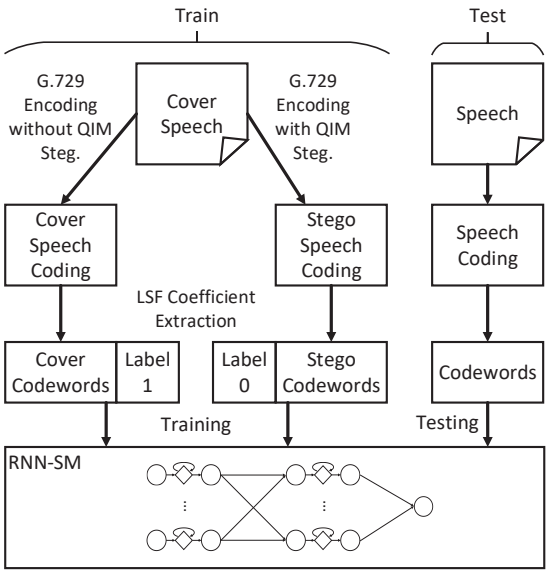


Fig. 7. Steganalysis Framework

V. Experiments and Discussion

In this section, we do some experiments to show the high accuracy and efficiency of RNN-SM.

As discussed in Section IV-C, pruned RNN-SM is more efficient and has better usability than full RNN-SM. In Section V-C, we compare their performance. In other sections, RNN-SM stands for pruned RNN-SM.

In Section V-A, we introduce the dataset and the performance evaluation metric we use. In Section V-B, we introduce how we determine the model size parameters, i.e. n_1 and n_2. In Section V-C, we compare the performance of full RNN-SM and pruned RNN-SM. In Section V-D, we discuss how the classification threshold will influence the results. In Section V-E, we evaluate the importance of four kinds of codeword correlations. In Section V-F, we present the accuracy testing results of RNN-SM and compare it with other state-of-the-art methods. In Section V-G, we test the time consuming performance of RNN-SM and other state-of-the-art methods.

A. Dataset and Metrics

To the best of our knowledge, there is no public steganography/steganalysis dataset available for our evaluation. To test our algorithm, we need to construct our own dataset, which includes a cover speech dataset and a stego speech dataset. We publish the speech dataset on https://github.com/fjxmlzn/RNN-SM/.

We collected 41 hours of Chinese speech and 72 hours of English speech in PCM format with 16 bits per sample from the Internet. The speech samples are from different male and female speakers. Those speech samples make up the cover speech dataset.

For each sample in cover speech dataset, we embed random 01 bit streams using CNV-QIM steganography proposed in [11]. Embedding rate is defined as the ratio of the number of embedded bits to the whole embedding capacity. Lower embedding rate indicates fewer changes to the original data streams, and therefore it is harder to detect low embedding rate steganography. CNV-QIM is a 100% embedding algorithm and it embeds data in every frame. To further test the ability of our algorithm, we extend CNV-QIM by enabling low embedding rate steganography. When conducting $a\%$ embedding rate steganography, we embed each frame with $a\%$ probability. We perform 10%, 20%, ..., 100% embedding rate CNV-QIM to each sample in cover speech dataset, and the generated speech samples make up the stego speech dataset.

In addition to embedding rate, sample length is another factor that influences detection accuracy. Usually when the sample length decreases, the detection accuracy decreases. However, as explained in Section I, steganalysis algorithm should be able to detect short samples. Therefore, we test the algorithms' performance on detecting samples of different lengths. We cut the samples in cover speech dataset and stego speech dataset into 0.1s, 0.2s, ..., 10s segements. Segments of the same length are successive and nonoverlapped. Those segments make up the cover segment dataset and stego segment dataset respectively.

For each test on RNN-SM, we pick up the positive and negative samples from stego segment dataset and cover segment dataset according to the required language, embedding rate and sample length. The ratio of the number of positive samples to the number of negative samples is 1 to 1. We randomly pick up four fifths of the samples as training set and the rest as testing set.

In order to compare RNN-SM to other methods, we also conduct tests on two state-of-the-art methods: IDC [17] and SS-QCCN [16]. Those two methods are based on SVM. SVM has quadratic time complexity. Therefore, it is impractical to utilize all samples in stego segment dataset and cover segment dataset when evaluating IDC and SS-QCCN. According to experimental settings in [16], for each test on IDC and SS-QCCN, we randomly pick up 2000 samples from stego segment dataset and 2000 samples from cover segment dataset. Those 4000 samples form the training set. In addition, we randomly pick up 1000 samples from stego segment dataset and 1000 samples from cover segment dataset. Those 2000 samples form the testing set.

We use three metrics to evaluate the performance. The first metric we use is classification accuracy, which is defined as the ratio of the number of samples that are correctly classified to the total number of samples. The second metric we use is false positive rate, which is defined as the ratio of cover segments that are classified as stego segments. The third metric we use is false negative rate, which is defined as the ratio of stego segments that are classified as cover segments.

B. Determining Model Size

There are two parameters in RNN-SM that are not yet determined: n_1 and n_2, which are the numbers of RNN units in the first layer and in the second layer. Generally, increasing the number of RNN units will enhance network's representation ability. However, it may increase the possibility of overfitting and slow down the training and testing process.

To determine how n_1 and n_2 will influence the accuracy, training time, and prediction time, we enumerate n_1 and n_2 to be 25, 50, 75, and test all 9 combinations on pruned RNN-SM. The tests are done on all 0.1s 100% embedding rate Chinese samples in cover segment dataset and stego segment dataset. Specifically, the training set contains 1,243,240 stego segments and 1,243,240 cover segments. The testing set contains 310,810 stego segments and 310,810 cover segments. We run each test for 30 epochs, and report: (1) the accuracy on testing set, (2) the average training time for each epoch, and (3) the total prediction time for all samples in training set and testing set. The training process was done on a single GeForce GTX 1080 GPU and the prediction process was done on "Intel(R) Xeon(R) CPU E5-2683 v3 @ 2.00GHz". Table I shows the results.[1]

		$n_2 = 25$	$n_2 = 50$	$n_2 = 75$
$n_1 = 25$	Accuracy (%)	89.11	90.43	91.48
	Training time (s)	2974.72	2476.45	2637.33
	Prediction time (s)	255.79	352.29	480.43
$n_1 = 50$	Accuracy (%)	90.70	91.29	91.31
	Training time (s)	2518.60	3251.67	3432.27
	Prediction time (s)	334.60	432.56	594.89
$n_1 = 75$	Accuracy (%)	90.63	91.31	92.00
	Training time (s)	2587.19	3289.05	3444.77
	Prediction time (s)	445.75	600.52	644.80

TABLE I
GRID SEARCH FOR MODEL SIZE (100% EMBEDDING RATE, 0.1S CHINESE SAMPLES)

As we can see, when the model size increase from $n1 = 25$ and $n2 = 25$ to $n1 = 50$ and $n2 = 50$, the accuracy increases from 89.11% to 92.00%, but the training time and prediction time also increase. When $n1 = 50$ and $n2 = 50$, the training time and prediction time is reasonable, and the accuracy is also satisfactory. In the following tests, we just empirically set $n_1 = 50$ and $n_2 = 50$. It should be noted that n_1 and n_2 could be further tuned when one wants to get a better balance between accuracy and time cost.

C. Comparing Pruned RNN-SM and Full RNN-SM

Following the same experiment settings as Section V-B except for setting $n_1 = 50$ and $n_2 = 50$, we test the accuracy and efficiency of pruned RNN-SM and full RNN-SM. The results are shown in Table II.

	Full RNN-SM	Pruned RNN-SM
Accuracy (%)	91.88	91.29
Training time (s)	3737.55	3251.67
Prediction time (s)	450.38	432.56

TABLE II
COMPARING FULL RNN-SM AND PRUNED RNN-SM

Compared with pruned RNN-SM, full RNN-SM's accuracy is slightly higher, but training time is significantly longer. The training time is even longer than the largest model ($n_1 = n_2 = 75$) in Table I. Considering the efficiency as well as other benfits of pruned RNN-SM mentioned in Section IV-C, we choose to use pruned RNN-SM in following tests.

[1]The results in Table I are based on an different run with the results in Table III, so the accuracy for $n1 = 50$ and $n2 = 50$ in Table I (91.29%) is slightly different from the one in Table III (90.91%).

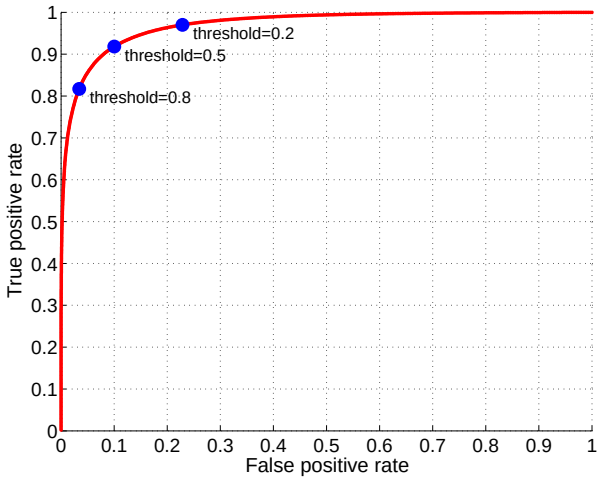


Fig. 8. ROC curve of RNN-SM (Tested on 100% embedding rate, 0.1s Chinese samples)

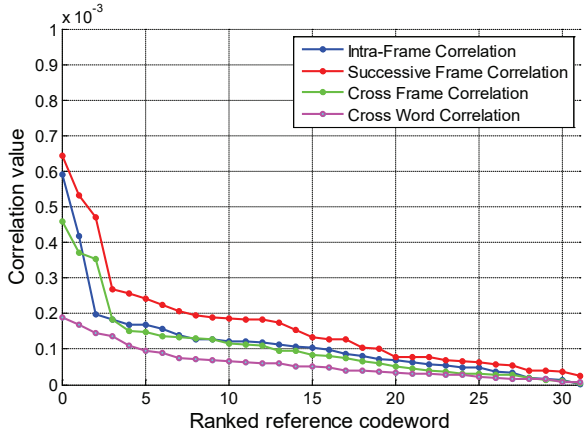
D. Discussion of Classification Threshold

In (23), we define the detection threshold as 0.5. If the output of FCM is larger than 0.5, we will classify the speech as stego speech; otherwise, we will classify the speech as normal speech. In fact, this threshold can be adjusted for a desired balance between true positive rate and true negative rate. We change the threshold and plot an example of ROC curve in Figure 8. In this test case, when the threshold is 0.5, the true positive rate and the true positive rate are very close. If we want to decrease the false positive rate with some sacrifice of true positive rate, we can increase the threshold. RNN-SM provides a very easy way for users to adjust their desired working point by simply changing the threshold. For simplicity, we set threshold to be 0.5 in the following tests.

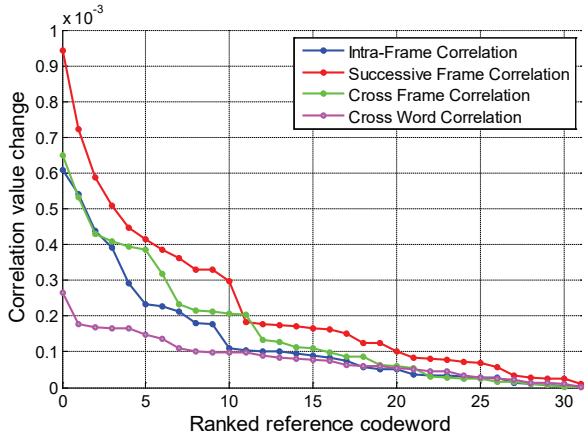
E. Codeword Correlation Testing

There are four kinds of codeword correlations discussed in the paper: successive frame correlation, intra-frame correlation, cross frame correlation, and cross word correlation. To show the importance of them, we do some analyses. We collect a G.729 coding stream with 180,000 frames and evaluate the codeword correlations according to (9). We fix $u = 15$ and enumerate reference codeword v from 0 to 31. Other parameters are set as follows: (1) For successive frame correlation, we set $\delta = 1$, $i = 2$, $k = 2$; (2) For intra-frame correlation, we set $\delta = 0$, $i = 2$, $k = 3$; (3) For cross frame correlation, we set $\delta = 2$, $i = 2$, $k = 2$; (4) For cross word correlation, we set $\delta = 100$, $i = 2$, $k = 2$. For each type of correlation, we take the absolute value of the results and rank them in descending order. The result is presented in Figure 9(a). Larger value indicates stronger correlation. As the figure shows, in this example, successive frame correlation

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2806741, IEEE Transactions on Information Forensics and Security



(a) Ranked Absolute Correlation Values



(b) Ranked Absolute Correlation Change

Fig. 9. Evaluation of the Four Correlations

is the strongest one. Intra-frame correlation and cross frame correlation are tying with each other. Cross word correlation is the weakest one.

To further evaluate how the four kinds of correlations would change after embedded with hidden data, we embed the speech coding stream with hidden data (100% embedding rate) and rank the absolute value of correlation change for all v from 0 to 31 in descending order, as shown in Figure 9(b). The correlation with larger change is a better indication for steganalysis. As the figure shows, the importance of the four correlations in this example can be roughly ranked as: successive frame correlation $>$ cross frame correlation $>$ intra-frame correlation $>$ cross word correlation.

The method proposed in [17] only considered successive frame correlation. The method proposed in [16] only considered successive frame correlation and intra-frame correlation. Cross frame correlation and cross word correlation were omitted in those two methods. However, in the example we present, cross frame correlation is more important than intra-frame correlation. Moreover, even though cross word correlation is the weakest, it can still provide classification clues. RNN-SM has the potential to consider all four correlations at the same time, and therefore it is more likely for RNN-SM to have better results.

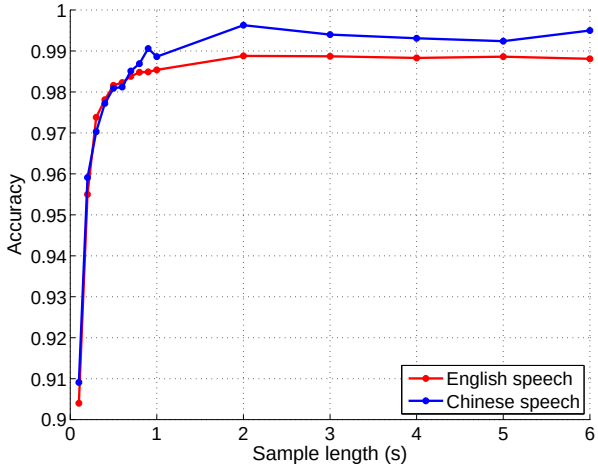


Fig. 10. RNN-SM's Detection Accuracy of 100% Embedding Rate Samples at Different Lengths

F. Accuracy Testing

In this section, we test and compare RNN-SM's accuracy with other state-of-the-art methods: IDC [17] and SS-QCCN [16]. For each embedding rate, sample length, and language, we train a separate model for all three algorithms. The code of RMM-SM and our implementations of IDC and SS-QCCN can be found on https://github.com/fjxmlzn/RNN-SM/.

1) Influence of Sample Length: Detection of short steganography samples is challenging. To test the performance of our RNN-SM algorithm towards different sizes of samples, we fix the embedding rate at 100%. As for sample length, we first test 10 samples whose lengths are equally spaced in the range of 0.1s to 1s. We then increase step size to 1s and test another 5 samples, which lie between 2s and 6s. English and Chinese speech are tested separately. The result is shown in Table III and Figure 10.

As we see, when the sample length increases, the accuracy also increases. This phenomenon is easy to explain. Longer sequence provides more observations on codeword correlations, which can therefore be modeled more accurately. Thus, the difference between the codeword correlation patterns of stego speech and cover speech is more distinct, leading to easier classification.

Moreover, when the sample length is small, increasing sample length significantly benefits the accuracy. As the sample length increases, the benefit of increasing sample length diminishes. When the sample length is longer than 2s, accuracy starts to stabilize at around 99%. This observation indicates that the sample length as short as 2s is totally enough for RNN-SM in full embedding scenario. We should also notice that even when the sample is of only 0.1s (10 frames), the detection accuracy is above 90%, which is an acceptable accuracy for steganalysis task. These clues indicate that RNN-SM can effectively detect both short samples and long samples.

We also notice that the accuracies of English and Chinese speech are very close. Although the accuracy of Chinese speech starts to be a little higher than that of English speech when the sample length is greater than 0.8s, the accuracy

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2806741, IEEE Transactions on Information Forensics and Security

Method	Language	Metric	Sample Length (s) 0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	2	3	4	5	6
IDC	English	Acc (%)	85.40	88.00	88.50	89.25	90.10	91.45	91.40	92.40	92.95	93.70	96.20	96.95	97.15	97.65	97.15
		FP (%)	14.60	11.40	10.70	11.60	8.90	8.00	8.30	8.60	7.60	6.10	3.10	2.00	2.20	1.30	1.60
		FN (%)	14.60	12.60	12.30	9.90	10.90	9.10	8.90	6.60	6.50	6.50	4.50	4.10	3.50	3.40	4.10
	Chinese	Acc (%)	86.80	88.65	90.20	90.50	91.20	92.25	93.10	94.25	94.70	94.05	96.80	97.20	98.15	97.75	97.95
		FP (%)	13.60	11.40	9.40	9.30	9.90	7.80	6.70	6.70	6.10	7.10	3.80	2.20	1.40	2.00	1.40
		FN (%)	12.80	11.30	10.20	9.70	7.70	7.70	7.10	4.80	4.50	4.80	2.60	3.40	2.30	2.50	2.70
SS-QCCN	English	Acc (%)	82.00	88.85	92.15	95.00	95.70	96.15	96.25	96.90	96.90	98.00	98.80	**99.00**	98.80	98.50	**98.95**
		FP (%)	8.00	8.90	5.80	4.60	4.20	2.90	3.70	1.80	1.70	0.60	0.00	0.00	0.00	0.00	0.00
		FN (%)	28.00	13.40	9.90	5.40	4.40	4.80	3.80	4.40	4.50	3.40	2.40	2.00	2.40	3.00	2.10
	Chinese	Acc (%)	81.20	90.05	93.75	95.25	96.50	97.45	97.60	98.30	98.10	98.50	**99.70**	**99.65**	**99.80**	**99.70**	99.45
		FP (%)	9.50	8.30	7.10	4.80	3.40	1.90	2.70	1.80	1.40	0.90	0.00	0.00	0.00	0.00	0.00
		FN (%)	28.10	11.60	5.40	4.70	3.60	3.20	2.10	1.60	2.40	2.10	0.60	0.70	0.40	0.60	1.10
RNN-SM	English	Acc (%)	**90.40**	**95.50**	**97.38**	**97.81**	**98.16**	**98.23**	**98.38**	**98.48**	**98.49**	**98.54**	**98.88**	98.87	**98.83**	**98.86**	98.81
		FP (%)	8.16	3.72	1.42	1.30	0.81	0.93	0.74	0.49	0.39	0.49	0.09	0.13	0.31	0.10	0.22
		FN (%)	11.05	5.29	3.83	3.10	2.87	2.61	2.50	2.55	2.62	2.42	2.15	2.13	2.02	2.18	2.16
	Chinese	Acc (%)	**90.91**	**95.91**	**97.03**	**97.72**	**98.09**	**98.12**	**98.51**	**98.69**	**99.06**	**98.86**	99.63	99.40	99.31	99.24	**99.50**
		FP (%)	10.02	4.66	2.46	1.77	1.57	1.76	1.06	1.04	0.85	0.81	0.17	0.53	0.36	0.16	0.54
		FN (%)	8.16	3.53	3.48	2.80	2.24	2.01	1.91	1.59	1.04	1.47	0.57	0.68	1.02	1.35	0.47

TABLE III
DETECTION ACCURACY OF 100% EMBEDDING RATE SAMPLES UNDER DIFFERENT LENGTHS

difference is still smaller than 1%. This means that the characteristic difference between two languages has little effect in full embedding situations.

And we can see that the accuracy on Chinese speech does not increase consistently with sample length. There are some peaks in the results (e.g. at 0.9s). This may due to the variance resulted from the randomness during training (e.g. randomly initialized neural network parameters, random mini-batch).

We also compare the results with IDC and SS-QCCN. Full results are shown in Table III. As you can see, when sample length is longer than 2s, all three methods almost converge to their own saturation accuracy. SS-QCCN and RNN-SM have similar saturation accuracy, which is slightly higher than IDC's saturation accuracy. However, when sample length is shorter than 2s, their accuracies are very different. To further compare their performance on short samples, we draw their accuracy on sample length between 0.1s and 2s in Figure 11 (Chinese) and Figure 12 (English). Obviously, RNN-SM outperforms other two methods on short samples. This phenomenon is easy to explain. SS-QCCN and IDC are based on intra-frame correlation and successive frame correlation. When the sample is short, information from those two correlations is limited. RNN-SM has the potential of exploiting correlations between frames of longer distance. Therefore, it can detect short samples better.

2) Influence of Embedding Rate: To avoid being easily detected, steganography algorithms often adopt low embedding rate strategy, which poses a challenge to steganalysis. In this test, we fix the sample length at 10s, and change embedding rate from 10% to 100% with step size of 10%. English and Chinese speech are tested separately. The result on RNN-SM is shown in Table IV and Figure 13.

As the figure shows, when the embedding rate is low, the accuracy increases remarkably with the increase of embedding rate. When the embedding rate is above 30%, the detection accuracies of English speech samples and Chinese speech samples are both above 90%.

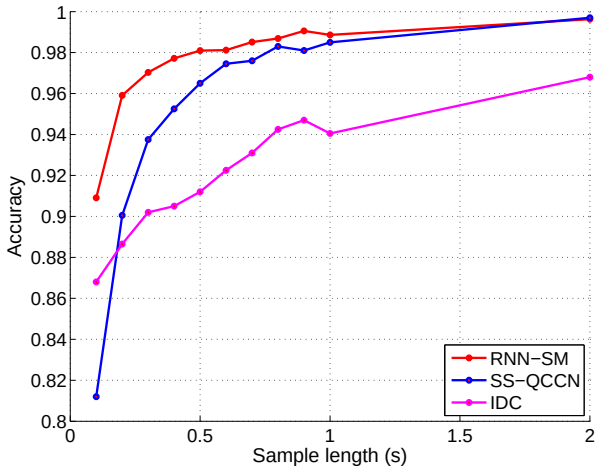


Fig. 11. Comparison on Detection Accuracy of 100% Embedding Rate Chinese Samples at Different Lengths

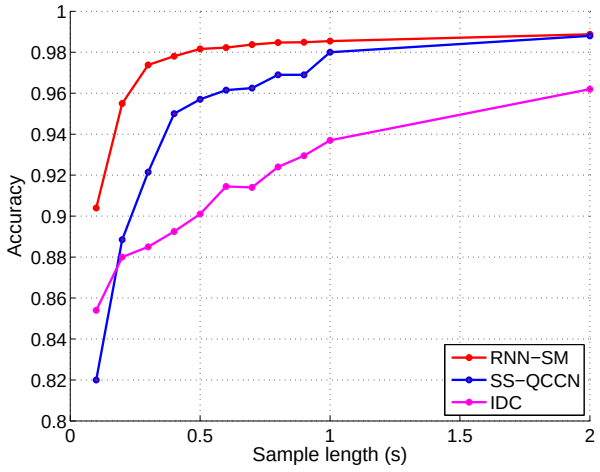


Fig. 12. Comparison on Detection Accuracy of 100% Embedding Rate English Samples at Different Lengths

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2806741, IEEE Transactions on Information Forensics and Security

Method	Language	Metric	Embedding Rate (%) 10	20	30	40	50	60	70	80	90	100
IDC	English	Acc (%)	51.60	58.55	63.65	71.50	76.25	83.50	87.25	91.60	95.55	97.20
		FP (%)	51.50	44.70	39.60	31.10	28.30	19.60	14.10	9.00	4.50	2.40
		FN (%)	45.30	38.20	33.10	25.90	19.20	13.40	11.40	7.80	4.40	3.20
	Chinese	Acc (%)	52.75	59.25	65.55	71.40	78.50	82.60	89.15	93.60	96.05	98.05
		FP (%)	47.30	45.20	39.80	34.10	26.40	21.20	13.00	8.00	5.50	2.10
		FN (%)	47.20	36.30	29.10	23.10	16.60	13.60	8.70	4.80	2.40	1.80
SS-QCCN	English	Acc (%)	54.40	75.45	92.45	**97.35**	**99.15**	**99.60**	**100.00**	**100.00**	**99.95**	**99.30**
		FP (%)	55.50	32.30	10.80	4.10	1.20	0.60	0.00	0.00	0.00	0.00
		FN (%)	35.70	16.80	4.30	1.20	0.50	0.20	0.00	0.00	0.10	1.40
	Chinese	Acc (%)	**57.35**	**75.00**	**92.00**	**98.25**	**99.50**	**99.85**	**100.00**	**99.95**	**99.90**	**99.75**
		FP (%)	45.50	29.60	12.10	3.00	0.90	0.20	0.00	0.10	0.00	0.00
		FN (%)	39.80	20.40	3.90	0.50	0.10	0.10	0.00	0.00	0.20	0.50
RNN-SM	English	Acc (%)	**59.64**	**92.44**	**94.56**	96.90	97.76	98.77	99.24	99.71	99.79	98.78
		FP (%)	38.07	9.18	5.73	3.95	1.98	2.12	0.84	0.31	0.29	0.04
		FN (%)	42.64	5.94	5.16	2.24	2.50	0.34	0.68	0.27	0.13	2.39
	Chinese	Acc (%)	55.14	74.19	90.12	95.24	98.05	98.25	99.09	99.51	99.76	99.55
		FP (%)	71.18	33.27	10.71	6.97	2.07	1.59	0.22	0.52	0.06	0.26
		FN (%)	19.87	18.66	9.05	2.60	1.83	1.91	1.61	0.45	0.43	0.66

TABLE IV
DETECTION ACCURACY OF 10S SAMPLES UNDER DIFFERENT EMBEDDING RATE

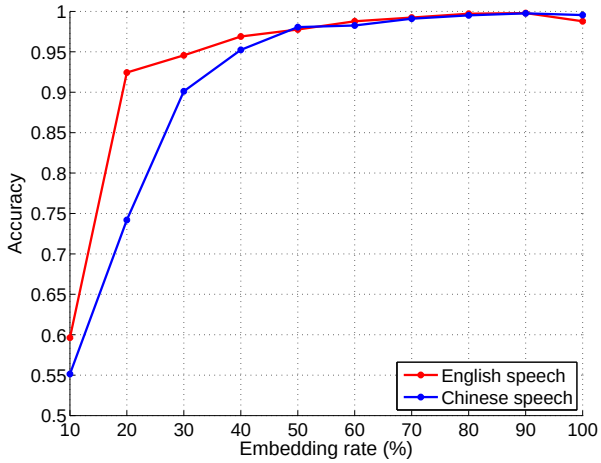


Fig. 13. RNN-SM's Detection Accuracy of 10s Samples at Different Embedding Rates

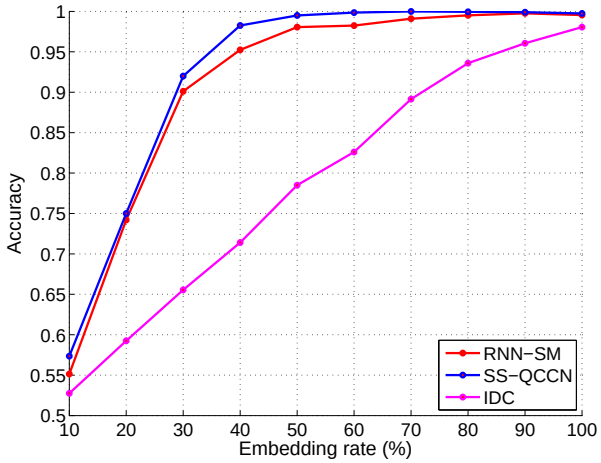


Fig. 14. Comparison on Detection Accuracy of 10s Chinese Samples at Different Embedding Rate

We also notice that, when the embedding rate is low, the accuracy of English speech samples is higher than that of Chinese speech samples. However, when the embedding rate is high, the accuracies of two languanges are close. This phenomenon may be explained by the different characteristics of the two languages. English is composed by 20 vowels and 28 consonants. However, in Chinese, there are 412 kinds of syllables. The diversity makes correlation model for Chinese language more complicated and therefore it is more difficult to detect steganography in Chinese speech, especially when embedding rate is low. When the embedding rate increases, the detection difficulty decreases and impact resulted from language characteristics goes down. Therefore, the two accuracy curves both converge to the same high level.

We also compare the results with IDC and SS-QCCN. Full results are shown in Table IV. Results on Chinese and English are plotted in Figure 14 and Figure 15 respectively. For Chinese speech, RNN-SM and SS-QCCN have very close accuracy, which is much better than IDC's accuracy. For English speech, when embedding rate is smaller than 30%, RNN-SM has better accuracy than SS-QCCN. When embedding rate is greater than 40%, RNN-SM and SS-QCCN have close accuracy, which is still better than IDC's accuracy. These results indicate that compared with other state-of-the-art methods, RNN-SM can provide competitive accuracy in low embedding rate samples.

3) Simultaneous Influence of Sample Length and Embedding Rate: To further evaluate how sample length and embedding rate would influence the detection accuracy, we test a set of samples with multiple lengths and multiple embedding rates. Specifically, we test with 3 different sample lengths, which are 0.5s, 2s, and 6s, respectively; and with 5 embedding rates from 20% to 100%, increasing by 20%. Our experimental goal is to determine detection accuracy of all 15 combinations. English and Chinese speech are tested separately. The results are listed in Table V.

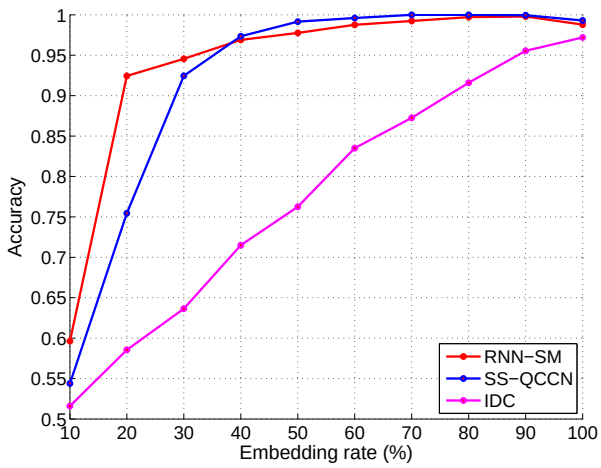


Fig. 15. Comparison on Detection Accuracy of 10s English Samples at Different Embedding Rate

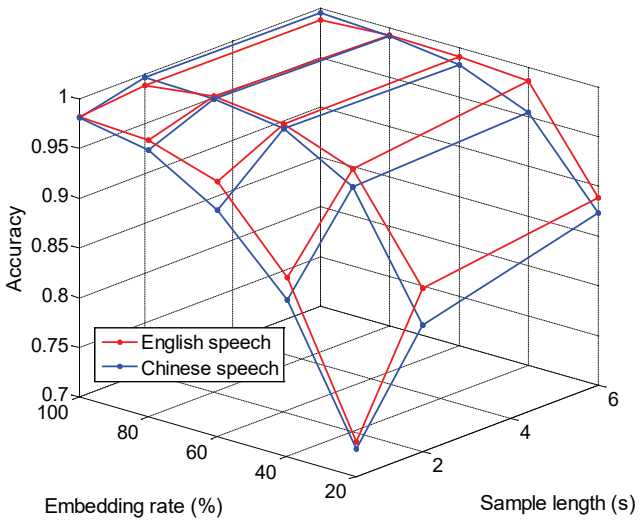


Fig. 16. RNN-SM's Detection Accuracy under Different Sample Lengths and Different Embedding Rates

We first look at results of RNN-SM. We plotted its results in Figure 16. As the figure shows, the accuracy plane is in a convex shape: decreasing in embedding rate or sample length will result in more detection errors, and the impact is bigger when embedding rate and sample length are small. When the sample is longer than $2s$ and the embedding rate is higher than 40%, the accuracies of Chinese speech and English speech are both above 90%.

We also notice that, the accuracy of English speech is slightly higher than that of Chinese speech at most of the points. This observation accords with what we discovered in the previous test and can be explained in the same way.

Now let's compare the results with IDC and SS-QCCN. As Table V shows, RNN-SM outperforms other two methods in all 0.5s tasks, most of the 2s tasks and half of the 6s tasks. For all tasks that RNN-SM does not have the best accuracy, the results of RNN-SM are actually very close to the best results.

Sam. Len.	Lang.	Method	Metric	Embedding Rate (%) 20	40	60	80	100
0.5s	English	IDC	Acc (%)	60.05	69.20	78.30	86.80	90.10
			FP (%)	34.40	30.80	23.30	13.10	8.90
			FN (%)	45.50	30.80	20.10	13.30	10.90
		SS-QCCN	Acc (%)	60.95	75.15	87.30	92.75	95.70
			FP (%)	41.10	24.90	12.60	6.40	4.20
			FN (%)	37.00	24.80	12.80	8.10	4.40
		RNN-SM	Acc (%)	**73.49**	**87.98**	**95.67**	**97.84**	**98.16**
			FP (%)	27.58	10.68	4.30	1.66	0.81
			FN (%)	25.44	13.35	4.37	2.65	2.87
	Chinese	IDC	Acc (%)	58.30	71.30	80.80	85.75	91.20
			FP (%)	38.80	26.80	19.30	13.40	9.90
			FN (%)	44.60	30.60	19.10	15.10	7.70
		SS-QCCN	Acc (%)	59.60	75.65	86.20	93.05	96.50
			FP (%)	44.40	22.00	16.90	6.00	3.40
			FN (%)	36.40	26.70	10.70	7.90	3.60
		RNN-SM	Acc (%)	**72.75**	**85.75**	**92.78**	**96.84**	**98.09**
			FP (%)	25.98	14.42	6.94	3.15	1.57
			FN (%)	28.51	14.09	7.51	3.17	2.24
2s	English	IDC	Acc (%)	67.85	81.75	88.95	94.50	96.20
			FP (%)	30.40	18.40	10.90	5.90	3.10
			FN (%)	33.90	18.10	11.20	5.10	4.50
		SS-QCCN	Acc (%)	65.75	88.10	96.70	98.75	98.80
			FP (%)	37.80	11.20	2.90	0.40	0.00
			FN (%)	30.70	12.60	3.70	2.10	2.40
		RNN-SM	Acc (%)	**86.42**	**96.39**	**98.97**	**99.79**	**98.88**
			FP (%)	12.54	5.04	0.85	0.17	0.09
			FN (%)	14.62	2.20	1.21	0.25	2.15
	Chinese	IDC	Acc (%)	64.30	81.65	88.70	94.20	96.80
			FP (%)	33.90	20.20	12.50	5.40	3.80
			FN (%)	37.50	16.50	10.10	6.20	2.60
		SS-QCCN	Acc (%)	65.50	87.75	97.70	99.10	**99.70**
			FP (%)	34.10	13.30	1.30	0.50	0.00
			FN (%)	34.90	11.20	3.30	1.30	0.60
		RNN-SM	Acc (%)	**82.73**	**94.58**	**98.52**	**99.49**	99.63
			FP (%)	14.75	4.73	1.87	0.46	0.17
			FN (%)	19.78	6.12	1.08	0.56	0.57
6s	English	IDC	Acc (%)	61.70	72.70	85.75	94.70	97.15
			FP (%)	37.90	32.10	16.90	6.70	1.60
			FN (%)	38.70	22.50	11.60	3.90	4.10
		SS-QCCN	Acc (%)	73.95	95.00	98.90	**99.80**	**98.95**
			FP (%)	29.00	7.10	1.70	0.00	0.00
			FN (%)	23.10	2.90	0.50	0.40	2.10
		RNN-SM	Acc (%)	**88.83**	**98.58**	**99.08**	99.23	98.81
			FP (%)	10.42	1.99	0.88	0.86	0.22
			FN (%)	11.93	0.85	0.97	0.68	2.16
	Chinese	IDC	Acc (%)	62.40	75.20	85.15	95.20	97.95
			FP (%)	42.20	27.90	16.90	7.10	1.40
			FN (%)	33.00	21.70	12.80	2.50	2.70
		SS-QCCN	Acc (%)	71.75	**97.00**	**99.55**	**99.90**	99.45
			FP (%)	33.50	4.00	0.50	0.10	0.00
			FN (%)	23.00	2.00	0.40	0.10	1.10
		RNN-SM	Acc (%)	**87.30**	95.45	98.26	99.12	**99.50**
			FP (%)	13.81	4.26	2.11	0.86	0.54
			FN (%)	11.60	4.84	1.36	0.91	0.47

TABLE V
DETECTION ACCURACY UNDER DIFFERENT SAMPLE LENGTHS AND DIFFERENT EMBEDDING RATES

Again, these results show that RNN-SM can effectively detect samples of various lengths and various embedding rates.

G. Efficiency Testing

a) Testing time: To enable online steganalysis, the time for testing each sample must be as short as possible. We collect the average detecting time for samples of 0.1s and 0.5s and samples whose lengths lie between 1s and 10s with a step of

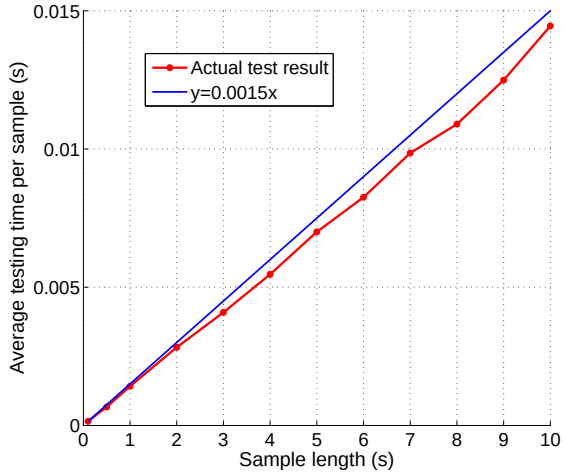


Fig. 17. Time to Perform RNN-SM

Method	Average testing time over sample length (%)
IDC	0.79
SS-QCCN	1.86
RNN-SM	0.14

TABLE VI
TESTING TIME COMPARISON

1s. This experiment is conducted on a computer whose CPU is "Intel(R) Xeon(R) CPU E5-2683 v3 @ 2.00GHz".

Figure 17 shows the testing time of RNN-SM. As the figure shows, the testing time approximately increases linearly with respect to the sample length, and is below 0.15% of sample length. This result demonstrates that RNN-SM is highly efficient and has no problem being deployed in online steganalysis tasks.

We also compare the testing time with IDC and SS-QCCN. The results are shown in Table VI. Because SS-QCCN computes a high dimensional feature vector and needs to perform PCA reduction, its overhead is distinctly higher than the other two methods.

b) Training time: SS-QCCN and IDC depends on SVM algorithm, which has quadratic time complexity during training, whereas RNN-SM's training time is linear with respect to the number of training samples. Therefore, RNN-SM has the ability to scale up to large dataset whereas the other two methods do not. In practice, we can generate large training dataset, and usually large training dataset can cover more data modes and improve classifier's generalization capability.

VI. Conclusion And Future Work

In this paper, we design a novel VoIP steganalysis algorithm called RNN-SM which can effectively detect QIM steganography in VoIP streams. Compared with previous state-of-the-art algorithms, our method has higher accuracy for short sample steganography detection and achieves accuracy above 90% even when the sample is of $0.1s$. The average testing time for each sample is only 0.15% of sample length. These features demonstrate that RNN-SM is a state-of-the-art algorithm for short sample detection problem and can be effectively used for online VoIP steganalysis. Moreover, we are the first to introduce RNN into VoIP steganalyis field and our work shows its practicability.

In the future, we will further excavate the advantages of RNN and work on tasks that are temporarily unsolved with traditional steganalysis method, such as predicting the positions of embedding bits.

Acknowledgements

We thank Yubo Luo, Wenhui Que, and Huaizhou Tao for helpful discussions on the algorithm, and thank Wenyu Wang for useful suggestions on the paper.

This work is supported by the National Key Research and Development Program of China (No. 2016YFB0800402) and the National Natural Science Foundation of China (No. U1636113, No. U1536207, No. U1536115, No. U1536113).

References

[1] A. Cheddad, J. Condell, K. Curran, and P. Mc Kevitt, "Digital image steganography: Survey and analysis of current methods," *Signal Processing*, vol. 90, no. 3, pp. 727–752, 2010.

[2] M. H. Shirali-Shahreza and M. Shirali-Shahreza, "A new approach to persian/arabic text steganography," in *Computer and Information Science, 2006 and 2006 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse. ICIS-COMSAR 2006. 5th IEEE/ACIS International Conference on*. IEEE, 2006, pp. 310–315.

[3] Y. Luo and Y. Huang, "Text steganography with high embedding rate: Using recurrent neural networks to generate chinese classic poetry," in *Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security*. ACM, 2017, pp. 99–104.

[4] N. B. Lucena, J. Pease, P. Yadollahpour, and S. J. Chapin, "Syntax and semantics-preserving application-layer protocol steganography," in *International Workshop on Information Hiding*. Springer, 2004, pp. 164–179.

[5] B. Goode, "Voice over internet protocol (VoIP)," *Proceedings of the IEEE*, vol. 90, no. 9, pp. 1495–1517, 2002.

[6] M. Hamdaqa and L. Tahvildari, "ReLACK: a reliable VoIP steganography approach," in *Secure Software Integration and Reliability Improvement (SSIRI), 2011 Fifth International Conference on*. IEEE, 2011, pp. 189–197.

[7] H. Tian, K. Zhou, H. Jiang, Y. Huang, J. Liu, and D. Feng, "An adaptive steganography scheme for voice over IP," in *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*. IEEE, 2009, pp. 2922–2925.

[8] E. Xu, B. Liu, L. Xu, Z. Wei, B. Zhao, and J. Su, "Adaptive VoIP steganography for information hiding within network audio streams," in *Network-Based Information Systems (NBiS), 2011 14th International Conference on*. IEEE, 2011, pp. 612–617.

[9] D. M. Ballesteros L and J. M. Moreno A, "Highly transparent steganography model of speech signals using efficient wavelet masking," *Expert Systems with Applications*, vol. 39, no. 10, pp. 9141–9149, 2012.

[10] Y. F. Huang, S. Tang, and J. Yuan, "Steganography in inactive frames of VoIP streams encoded by source codec," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 2, pp. 296–306, 2011.

[11] B. Xiao, Y. Huang, and S. Tang, "An approach to information hiding in low bit-rate speech stream," in *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*. IEEE, 2008, pp. 1–5.

[12] H. Tian, J. Liu, and S. Li, "Improving security of quantization-index-modulation steganography in low bit-rate speech streams," *Multimedia Systems*, vol. 20, no. 2, pp. 143–154, 2014.

[13] Y. Huang, C. Liu, S. Tang, and S. Bai, "Steganography integration into a low-bit rate speech codec," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 6, pp. 1865–1875, 2012.

[14] B. Chen and G. W. Wornell, "Quantization index modulation: A class of provably good methods for digital watermarking and information embedding," *IEEE Transactions on Information Theory*, vol. 47, no. 4, pp. 1423–1443, 2001.

[15] Y. Huang, S. Tang, and Y. Zhang, "Detection of covert voice-over Internet protocol communications using sliding window-based steganalysis," *IET Communications*, vol. 5, no. 7, pp. 929–936, 2011.
[16] S. Li, Y. Jia, and C.-C. J. Kuo, "Steganalysis of QIM steganography in low-bit-rate speech signals," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2017.
[17] S.-b. Li, H.-z. Tao, and Y.-f. Huang, "Detection of quantization index modulation steganography in G. 723.1 bit stream based on quantization index sequence analysis," *Journal of Zhejiang University-Science C*, vol. 13, no. 8, pp. 624–634, 2012.
[18] D. O'Shaughnessy, "Linear predictive coding," *IEEE Potentials*, vol. 7, no. 1, pp. 29–32, 1988.
[19] C. Kraetzer and J. Dittmann, "Mel-cepstrum-based steganalysis for VoIP steganography," in *Electronic Imaging 2007*. International Society for Optics and Photonics, 2007, pp. 650 505–650 505.
[20] ——, "Pros and cons of mel-cepstrum based audio steganalysis using SVM classification," in *International Workshop on Information Hiding*. Springer, 2007, pp. 359–377.
[21] Q. Liu, A. H. Sung, and M. Qiao, "Temporal derivative-based spectrum and mel-cepstrum audio steganalysis," *IEEE Transactions on Information Forensics and Security*, vol. 4, no. 3, pp. 359–368, 2009.
[22] J. Dittmann, D. Hesse, and R. Hillert, "Steganography and steganalysis in voice-over IP scenarios: operational aspects and first experiences with a new steganalysis tool set," in *Electronic Imaging 2005*. International Society for Optics and Photonics, 2005, pp. 607–618.
[23] I. Avcıbas, "Audio steganalysis with content-independent distortion measures," *IEEE Signal Processing Letters*, vol. 13, no. 2, 2006.
[24] O. Altun, G. Sharma, M. U. Celik, M. Sterling, E. L. Titlebaum, and M. Bocko, "Morphological steganalysis of audio signals and the principle of diminishing marginal distortions," in *ICASSP (2)*, 2005, pp. 21–24.
[25] X.-m. Ru, Y.-t. Zhuang, and F. Wu, "Audio steganalysis based on negative resonance phenomenon caused by steganographic tools," *Journal of Zhejiang University-Science A*, vol. 7, no. 4, pp. 577–583, 2006.
[26] Y. Huang, S. Tang, C. Bao, and Y. J. Yip, "Steganalysis of compressed speech to detect covert voice over internet protocol channels," *IET Information Security*, vol. 5, no. 1, pp. 26–32, 2011.
[27] C. Paulin, S.-A. Selouani, and E. Hervet, "Audio steganalysis using deep belief networks," *International Journal of Speech Technology*, vol. 19, no. 3, pp. 585–591, 2016.
[28] C. Paulin, S.-A. Selouani, and É. Hervet, "Speech steganalysis using evolutionary restricted Boltzmann machines," in *Evolutionary Computation (CEC), 2016 IEEE Congress on*. IEEE, 2016, pp. 4831–4838.
[29] S. Rekik, S. Selouani, D. Guerchi, and H. Hamam, "An autoregressive time delay neural network for speech steganalysis," in *Information Science, Signal Processing and their Applications (ISSPA), 2012 11th International Conference on*. IEEE, 2012, pp. 54–58.
[30] B. Chen, W. Luo, and H. Li, "Audio steganalysis with convolutional neural network," in *Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security*. ACM, 2017, pp. 85–90.
[31] L. Shaohui, Y. Hongxun, and G. Wen, "Neural network based steganalysis in still images," in *Multimedia and Expo, 2003. ICME'03. Proceedings. 2003 International Conference on*, vol. 2. IEEE, 2003, pp. II–509.
[32] Y. Q. Shi, G. Xuan, D. Zou, J. Gao, C. Yang, Z. Zhang, P. Chai, W. Chen, and C. Chen, "Image steganalysis based on moments of characteristic functions using wavelet decomposition, prediction-error image, and neural network," in *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*. IEEE, 2005, pp. 4–pp.
[33] V. Sabeti, S. Samavi, M. Mahdavi, and S. Shirani, "Steganalysis and payload estimation of embedding in pixel differences using neural networks," *Pattern Recognition*, vol. 43, no. 1, pp. 405–415, 2010.
[34] Y. Qian, J. Dong, W. Wang, and T. Tan, "Deep learning for steganalysis via convolutional neural networks." *Media Watermarking, Security, and Forensics*, vol. 9409, pp. 94 090J–94 090J, 2015.
[35] G. Xu, H.-Z. Wu, and Y.-Q. Shi, "Structural design of convolutional neural networks for steganalysis," *IEEE Signal Processing Letters*, vol. 23, no. 5, pp. 708–712, 2016.
[36] M. Chen, V. Sedighi, M. Boroumand, and J. Fridrich, "Jpeg-phase-aware convolutional neural network for steganalysis of jpeg images," 2017.
[37] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 6645–6649.
[38] R. Socher, C. C. Lin, C. Manning, and A. Y. Ng, "Parsing natural scenes and natural language with recursive neural networks," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 129–136.
[39] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602–610, 2005.
[40] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[41] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

Zinan Lin received the B.E. degree in electronic engineering from Tsinghua University, Beijing, China, in 2017. He is currently working toward the Ph.D. degree in the Department of Electrical and Computer Engineering, Carnegie Mellon University. He has broad interests in machine learning and information security.

Yongfeng Huang received the Ph.D. degree in computer science and engineering from the Huazhong University of Science and Technology, in 2000. He is a professor in the Department of Electronic Engineering, Tsinghua University, Beijing, China. His research interests include cloud computing, data mining, and network security. He is a senior member of the IEEE.

Jilong Wang received the Ph.D. degree in computer science from Tsinghua University, Beijing, China, in 1996. He is a professor in the School of Information Science and Technology, Tsinghua University, Beijing, China. His research interests include network architecture and network management.