网络语音流隐写分析全流程 (Steganalysis of VoIP Speech Streams)

原创

欢迎访问我的个人博客：https://hi.junono.com/

AMR隐写数据集地址(Kaggle)

网络语音流隐写分析全流程

隐写分析流程介绍：

基本知识

基于网络语音（VoIP）流的隐写术及隐写分析

1. 数据集准备(语音编码参数)

1.1 使用现成的语音编码参数

1.2 自己准备编码参数

1.2.1 wav->pcm

1.2.2 pcm->dat

1.2.3 dat->txt

2. 隐写分析

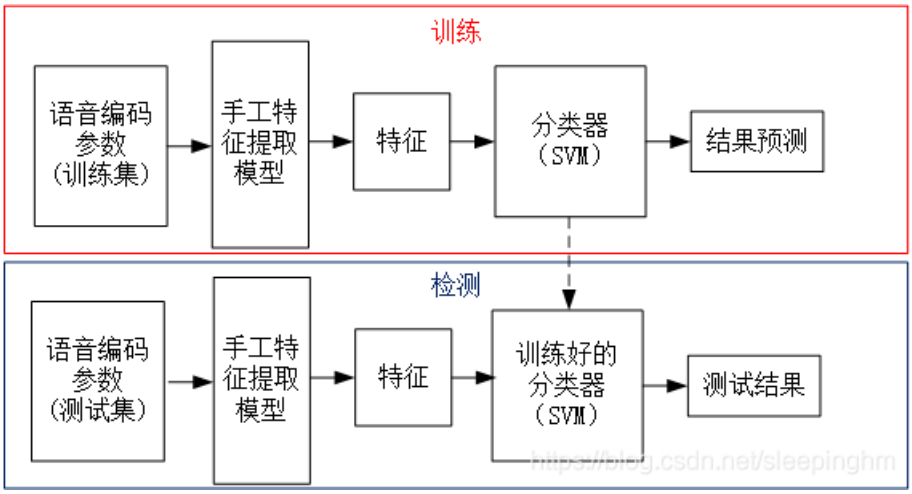
2.1 RSM

3. AMR隐写分析数据集—Kaggle

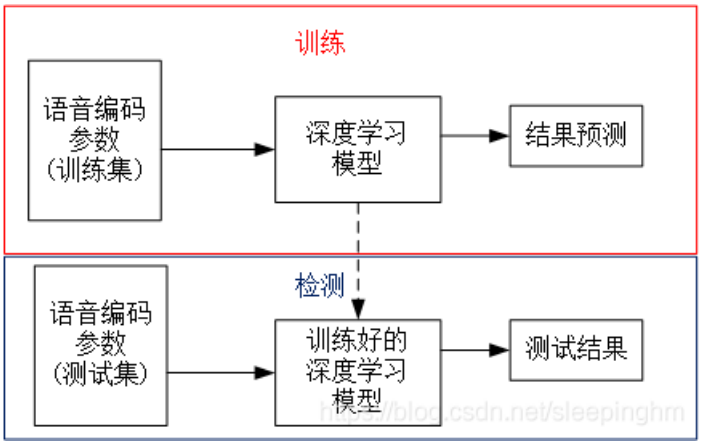
Reference:

隐写分析流程介绍：

基于“手工特征+分类器”的隐写分析方法：



基于深度学习的隐写分析方法：深度学习模型自动提取特征，经过分类层给出分类结果。



基本知识

隐写(Steganography)：在不影响载体质量的情况下将秘密信息隐藏在多媒体载体（图像，音频，视频等）中并通过公共信道进行安全传播，除了发送方和接收方，第三者无法察觉秘密信息的存在。

隐写分析(Steganalysis)：检测这段多媒体中是否有秘密信息的存在，为二分类任务。

影响隐写分析检测性能的两个重要指标：嵌入率和样本长度。

1. **嵌入率(Embedding Rate)**：$EMR=$ 秘密信息比特数 ，有100%到0%不等。其中0%嵌入率为载体样本(co
2. **样本长度（Sample Length)**: 语音样本时长，有10秒，1秒不等。

评价指标(Evaluation Metrics)：精度ACC；虚警率FPR；漏检率FNR；检测时长。
一般而言，检测精度越高，虚警率漏检率越低，检测速度越快越好。

1. $ACC=$
2. $FPR=$
3. $FNR=$
4. 检测时长：平均测试时间= (average testing time over sample length)。要给出具体的电脑配置。

基于网络语音（VoIP）流的隐写术及隐写分析

VoIP 编码为压缩编码， 在编码后语音文件会变成相应参数， 而参数编码过程中的隐写方法又可以根据隐写方式分为最低有效位(least significant bit， LSB)隐写和量化索引调制(quantized index modulation，QIM)隐写。

1. **LSB 隐写**：基于 VoIP 编码器编码过程中的冗余提出的隐写方案。在编码中检测某些参数属于最低有效位， 并修改为秘密信息。
2. **QIM 隐写**：利用 VoIP 编码器的码本编码特性划分码本来隐藏信息的方法，即动态选择最佳激励向量的量化索引值，同时尽可能地最小化合成语音的失真，以实现信息隐藏。

VoIP 的编码器大多使用码本激励线性预测编码 (code excited linear prediction，CELP)编码规则编码。编码器根据实现分为：线性预测分析、基音搜索和固定码本搜索。

因此编码过程中最重要的参数为：**线性预测(linear predictivecoefficients，LPC) 参数， 固定码本(fixed codebook，FCB)参数，自适应码本(adaptive codebook，ACB)参数**和Gain 参数。其中Gain参数冗余太小不适于隐写， 在参数隐写的部分基本上是前面 3 种或者混合。

LPC，FCB，ACB 参数编码都是使用码本的参数编码， 所以既可以使用**LSB 方式隐写**又可以根据**QIM方式隐写**， 对于隐写，提出的隐写方案更加侧重于隐写方法， 其目的在于如何使提出的方法能够提高隐蔽性和隐藏容量。所以按照隐写手段来分类。但是对于检测， 不论使用的是何种方法， 重要的是将其检测出来并且最好知道隐写的位置， 因此， 隐写分析会按照参数位置来分类。

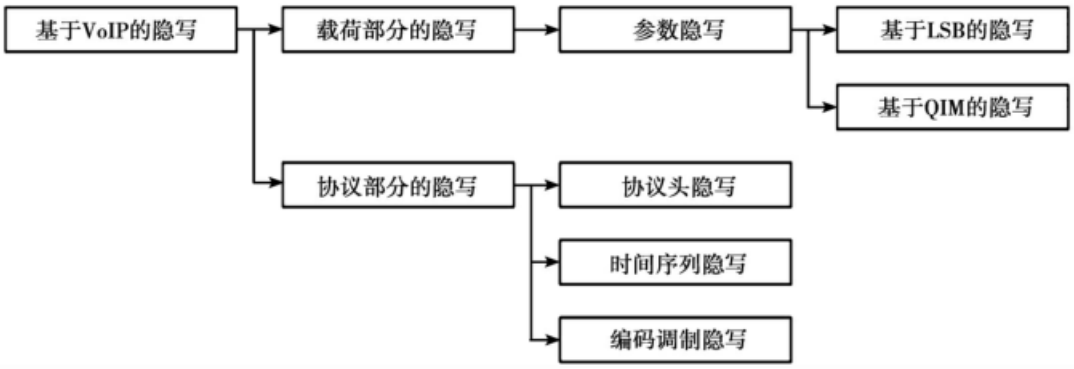


图 1　基于 VoIP 的隐写分类

Fig.1　Steganography classification based on VoIP

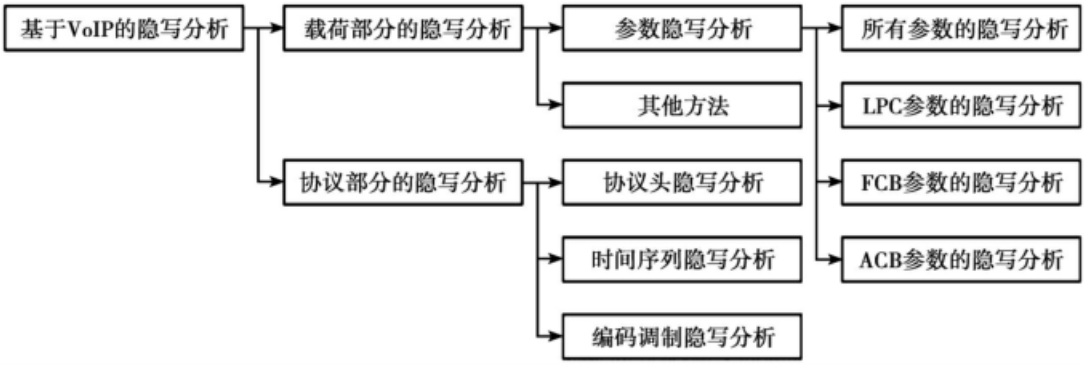


图 2　基于 VoIP 的隐写分析分类

Fig.2　Steganalysis classification based on VoIP

1. 数据集准备(语音编码参数)

为了进行隐写分析，第一步就是要提取语音编码参数，我们需要对音频进行处理，例如wav处理，隐写处理(编码)和提取参数(解码)，提取好编码参数后，再进行隐写分析。

1.1 使用现成的语音编码参数

只有特定时长嵌入率的参数，如果需要准备更多的数据，参考1.2步。

下载链接

Train Set：https://pan.baidu.com/share/init?surl=dJtBXQuZnG2eba13tbmnOA (a1xd)
Test Set：https://pan.baidu.com/share/init?surl=MREI-doUf2MG4-BuE91P0w (levg)

数据集介绍

编码器	隐写方法	使用参数	时长	嵌入率	样本数量

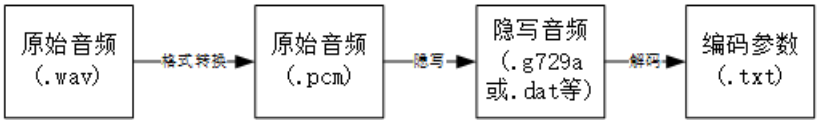
编码器	隐写方法	使用参数	时长	嵌入率	样本数量
G.729a	CNV-QIM	LPC，线性预测参数	1s	0%~40%	45000~50000
G.729a	PMS	ACB，基音延迟参数	1s	0%~40%	45000~50000

CNV-QIM:Xiao B, Huang Y, Tang S. An approach to information hiding in low bit-rate speech stream[C]//IEEE GLOBECOM 2008-2008 IEEE Global Telecommunications Conference. IEEE, 2008: 1-5.

PMS: Huang Y, Liu C, Tang S, et al. Steganography integration into a low-bit rate speech codec[J]. IEEE transactions on information forensics and security, 2012, 7(6): 1865-1875.

有了这些数据，我们直接进入第2步。

1.2 自己准备编码参数



需要得到的就是编码参数(txt)

1.2.1 wav->pcm

wav语音数据集
中文:Chinese.tar.gz,wav 格式的 160 条语音,40+小时。
英语:English.tar.gz, wav 格式的 160 条语音,70+小时。

wavsplit.py:
需要将wav分割成10s,9s,8s,...时长，之后再转成pcm。

```
import os
import wave
import numpy as np

path = "E:/wav_path/"
pcm_path = "E:/pcm_path/"
CutTime=10 # 剪裁的时间
files = os.listdir(path)
files = [path + "\\" + f for f in files if f.endswith('.wav')]

if __name__ == '__main__':
    for i in range(len(files)):
        FileName = files[i]
        f = wave.open(r"" + FileName, 'rb')
        params = f.getparams()  # 读取音频文件信息
        nchannels, sampwidth, framerate, nframes = params[:4]  # 声道数, 量化位数, 采样频率, 采样点数
        str_data = f.readframes(nframes)
        f.close()
        # 根据声道数对音频进行转换
        wave_data = np.frombuffer(str_data, dtype=np.short)
        if nchannels > 1:
            wave_data.shape = -1, 2
            wave_data = wave_data.T
            temp_data = wave_data.T
        else:
            wave_data = wave_data.T
            temp_data = wave_data.T
        CutFrameNum = framerate * float(CutTime)
        Cutnum = nframes / CutFrameNum  # 音频片段数
        StepNum = int(CutFrameNum)
        origin_name, _ = os.path.basename(FileName).split(".wav")
        for j in range(int(Cutnum)):
            FileName = pcm_path + os.path.basename(origin_name) + "_" + str(j) + ".wav"
            print(FileName)
            temp_dataTemp = temp_data[StepNum * j:StepNum * (j + 1)]
            StepTotalNum = (j + 1) * StepNum
            temp_dataTemp.shape = 1, -1
            temp_dataTemp = temp_dataTemp.astype(np.short)  # 打开WAV文档
            f = wave.open(FileName, 'wb')
            # 配置声道数、量化位数和取样频率
            f.setnchannels(nchannels)
            f.setsampwidth(sampwidth)
            f.setframerate(framerate)
            f.writeframes(temp_dataTemp.tostring())  # 将wav_data转换为二进制数据写入文件
            f.close()
```

wav2pcm.py:

```
#-*-coding:utf-8 -*-

import os
import numpy as np

path = "E:/wav_path/"
pcm_path = "E:/pcm_path/"

for file in os.listdir(path):
 f = 0
 data = 0
 f = open(os.path.join(path,file))
 f.seek(0)
 f.read(44)
 data = np.fromfile(f, dtype=np.int16)
 dataname = file.rstrip('wav')+'pcm'
 data.tofile(os.path.join(pcm_path, dataname))
```

或者使用ffmpeg工具，并将ffmpeg添加到系统变量中，使用以下代码：

```
import os

path = "E:/wav_path/"
pcm_path = "E:/pcm_path/"
files = os.listdir(path)
files = [pcm_path + "\\" + f for f in files if f.endswith('.wav')]

if __name__ == '__main__':
    j = 0
    for i in range(len(files)):
        in_wav = files[i]
        origin_name, _ = os.path.basename(in_wav).split(".wav")
        out_pcm = os.path.join(pcm_path, "prefix_%05d.pcm" % (noise, j))
        j += 1
        cmd = "ffmpeg -i %s -f s16le %s" % (in_wav, out_pcm)
        os.system(cmd)
```

1.2.2 pcm->dat

可以使用多种**编码器**,iLBC,G.723.1,G.729a,AMR等，并根据隐写论文的方法进行复现，通过找到编码过程中对应的参数进行修改，例如LPC参数为LSF搜索函数中VQ量化后的的incidence，ACB参数就是闭环基音延迟搜索中的pitch。

这里介绍开源的隐写工具：https://github.com/YangzlTHU/VStego800K/blob/main/Steganography/Stega_tool/StegCoder.exe

提供了两种隐写方法：CNV和PMS，CNV是修改了LPC参数（VQ量化索引值），PMS修改了ACB参数（基音延迟整数参数）。

> **CNV-QIM**:Xiao B, Huang Y, Tang S. An approach to information hiding in low bit-rate speech stream[C]//IEEE GLOBECOM 2008-2008 IEEE Global Telecommunications Conference. IEEE, 2008: 1-5.
>
> **PMS**: Huang Y, Liu C, Tang S, et al. Steganography integration into a low-bit rate speech codec[J]. IEEE transactions on information forensics and security, 2012, 7(6): 1865-1875.

使用方法

1. 将需要隐写的pcm文件放在一个命名为input的文件夹内
2. 新建一个output文件夹，用于存放隐写后的文件
3. 如果使用CNV-QIM隐写，隐写嵌入率为100%，则在命令提示符内输入
   `StegCoder.exe -i input -o output -a cnv -r 100`
   如果使用PMS基音隐写方法，隐写嵌入率为100%，则在命令提示符内输入
   `StegCoder.exe -i input -o output -a pitch -r 100`
4. 将嵌入率100换成其他嵌入率可以生成不种嵌入率的隐写样本。

1.2.3 dat->txt

根据自己使用的编码器iLBC,G.723.1,G.729a,AMR等，所对应的**解码器**的解码步骤中提取相应的编码参数。

或者使用G729A编码文件参数提取工具：https://github.com/YangzlTHU/VStego800K/blob/main/Steganalysis/RSM/G729PreProcessor_CNV%2BPMS.py

```
parser.add_argument("--input", default="inputdir", help="输入文件/文件夹", type=str)
parser.add_argument("--output", default="outputdir", help="输出文件夹", type=str)
```

可以直接在代码中改default后直接运行，其中 `outputdir` 是提取特征(txt)的文件夹， `inputdir` 是待提取的编码音频(.g729a)文件夹。

或者在命令提示符内输入 `python G729PreProcessor_CNV+PMS.py --input inputdir --output outputdir`

最后会输出特征txt，每帧包含5个编码参数，其中前3个是LPC参数，后两个是ACB参数。

```
def extract_frame(content):
    if type(content) == str:
        content_t = [int(item.encode('hex'), 16) for item in content]
    else:
        content_t = content
    a = content_t[0] & 0x7f
    b = (content_t[1] >> 3) & 0x1f
    c = ((content_t[1] << 2) & 0x1c) | ((content_t[2] >> 6) & 0x03)
    d = ((content_t[2] << 2) & 0xfc) | ((content_t[3] >> 6) & 0x03)
    e = content_t[6] & 0x1f
    return [a, b, c, d, e]
```

2. 隐写分析

https://github.com/YangzlTHU/VStego800K/tree/main/Steganalysis提供了以下五种隐写分析方法。

SS-QCCN:Yang, H., Yang, Z., Bao, Y., & Huang, Y. (2019, December). Hierarchical representation network for steganalysis of qim steganography in low-bit-rate speech signals. In International Conference on Information and Communications Security (pp. 783-798). Springer, Cham.

CCN:Li, S. B., Jia, Y. Z., Fu, J. Y., & Dai, Q. X. (2014). Detection of pitch modulation information hiding based on codebook correlation network. Chinese Journal of Computers, 37(10), 2107-2117.

RSM:Lin, Z., Huang, Y., & Wang, J. (2018). RNN-SM: Fast steganalysis of VoIP streams using recurrent neural network. IEEE Transactions on Information Forensics and Security, 13(7), 1854-1868.

FSM:Yang, H., Yang, Z., Bao, Y., Liu, S., & Huang, Y. (2019). Fast steganalysis method for voip streams. IEEE Signal Processing Letters, 27, 286-290.

SFFN:Hu, Y., Huang, Y., Yang, Z., & Huang, Y. Detection of heterogeneous parallel steganography for low bit-rate VoIP speech streams. Neurocomputing, 419, 70-79.

https://github.com/fjxmlzn/RNN-SM/blob/master/algorithms/IDC.py提供了**IDC**方法

- **IDC**:：LI Song-Bin, HZ Tao, YF Huang. Detection of quantization index modulation steganography in G.723.1 bit stream based on quantization index sequence analysis[J]. Journal of Zhejiang University-Science C(Computers & Electronics), 2012, 13(8):624-634.

其中CCN和SS-QCCN是基于“手工特征+分类器”的隐写分析方法,IDC和SS-QCCN用于检测CNV隐写，CCN检测PSM隐写。RSM、FSM、SFFN是基于深度学习的隐写分析方法，可用于检测CNV和PMS隐写。

2.1 RSM

https://github.com/YangzlTHU/VStego800K/blob/main/Steganalysis/RSM/RSM.py

RNN-SM网络，是第一个深度学习网络应用在网络语音流隐写分析任务中。

直接修改FOLDERS部分就可以运行，其中类别1代表的是隐写(stego)参数的数据集目录，0是正常编码即嵌入率为0(cover)的参数的数据集目录：

```
FOLDERS = [
    {"class": 1, "folder": "/data/train/g729a_Steg_feat"},
    # The folder that contains positive data files.
    {"class": 0, "folder": "/data/train/g729a_0_feat"}
    # The folder that contains negative data files.
]
```

这里可改可不改:提取的每帧包含5个编码参数，其中前3个是LPC参数，后两个是ACB参数。如果使用的是CNV隐写，因为这个隐写方法只改变了LPC参数，所以只需要使用每帧的前三个LPC参数。如果使用的是PMS隐写，因为这个隐写方法只改变了ACB参数，所以只需要使用每帧的后两个ACB参数。

```
'''for CNV'''
x_train,x_test=x_train[:,:,:3],x_test[:,:,:3]
#然后可以将input_dim的5改成3
model.add(LSTM(50, input_length=int(SAMPLE_LENGTH / 10), input_dim=3, return_sequences=True))
'''for PMS'''
x_train,x_test=x_train[:,:,3:],x_test[:,:,3:]
#然后可以将input_dim的5改成2
model.add(LSTM(50, input_length=int(SAMPLE_LENGTH / 10), input_dim=2, return_sequences=True))
```

RSM网络由双层LSTM组成，分类层由带有sigmoid激活函数的全连接层组成。使用adam优化器和二分类交叉熵作为损失函数进行训练。

```
model = Sequential()
model.add(LSTM(50, input_length=int(SAMPLE_LENGTH / 10), input_dim=5, return_sequences=True))  # first layer，默认input_dim是5， 可以修改为3或者2
model.add(LSTM(50, return_sequences=True))  # second layer
model.add(Flatten())  # flatten the spatio-temporal matrix
model.add(Dense(1))  # output layer
model.add(Activation('sigmoid'))  # activation function
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=["accuracy"])
```

训练部分：

```
for i in range(ITER):
    model.fit(x_train, y_train, batch_size=BATCH_SIZE, nb_epoch=1, validation_data=(x_test, y_test))
    model.save('full_model_%d.h5' % (i + 1)) # 模型保存
```

可在训练部分加入ACC,FPR,FNR评价指标：

```
y_pred = np.argmax(model.predict(x_test), axis=1)
accuracy = accuracy_score(y_test, y_pred)
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
fpr = fp / (fp + tn)
fnr = fn / (fn + tp)
```

其余评价指标：

```
from sklearn.metrics import confusion_matrix, f1_score, precision_score, recall_score, accuracy_score

y_pred = np.argmax(model.predict(x_test), axis=1)
accuracy = accurac y_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred, average='binary')
f1score = f1_score(y_test, y_pred, average='binary')
MiF1 = f1_score(y_test, y_pred, average='micro')
MaF1 = f1_score(y_test, y_pred, average='macro')
```

3. AMR隐写分析数据集—Kaggle

- Kaggle数据集地址：Multiple Voip Steganography Datasets
  包含1秒下AMR的隐写数据集用于测试隐写分析性能，例如LPC,FCB,ACB的隐写数据。
- 隐写分析Demo地址1：keras-rnn-sm
- 隐写分析Demo地址2：pytorch-rnn-sm
- 隐写分析Demo地址3：MLSteg-msdpd
  只要根据demo的代码完整运行一遍就看以了解隐写分析流程。

Reference:

https://github.com/YangzlTHU/VStego800K

https://github.com/fjxmlzn/RNN-SM

刘小康, 田晖, 刘杰,等. IP语音隐写及隐写分析研究[J]. 重庆邮电大学学报:自然科学版, 2019.