# 树莓派项目

## 项目简介

使用python代码编写一个可抗密钥丢失的点对点通信协议

## 已完成工作

1. 对整个项目进行仿真测试

|  |  |  |
| --- | --- | --- |
| 仿真测试目标 | 仿真方式 | 仿真结果 |
| 仿真sender---packet---->>receiever丢包 | 通过netsim代码进行综合仿真，即在netsim中，在一次正常交流中（sender---<----->netsim<------->receiever），上述各类丢包、错包情况可能会交错出现。（包括下面的断电仿真） | 正常 |
| 用于仿真sender---packet---->>receiever数据被篡改或数据传输时数据错误 | 正常 |
| 仿真receiever---ack----->>sender丢包 | 正常 |
| 仿真receiever---ack----->>sender数据出错 | 正常 |
| 仿真sender中途断电重启 | 通过ctrl＋c进行断电仿真 | 正常 |
| 仿真Receiever中途断电重启 | 正常 |
| 仿真sender，reveiever均断电重启 | 正常 |

意外问题：

由于我在appRec进行一个消息结束状态判定，即当Rec收到end数据时，自动关闭socket，表示消息结束。但是，若Rec对end数据签名返回的ack在传回Sender时，丢包或者数据错误，那么sender永远以为Rec还未接受到end包，会一直重传。最后，导致该数据一直发送失败。尚未想到解决方案。（简而言之，终止包丢包问题）

1. 实现当前所述状态机工作流程，并实现了netsim网络仿真
2. 把加解密操作封装在同一个文件中，对外呈现为一个加密函数和解密函数
3. 将文件读写操作封装在file文件中

## 项目文档目录

项目目录结构：

.  
├── appRec.py <----外部应用，用于接收  
├── appSend.py <----外部应用，用于发送  
├── netsocket <----整个系统封装在该文件下  
│   ├── Setdefault.py <----重置文件信息  
│   ├── \_\_init\_\_.py <----初始化文件  
│   ├── comlib <----加密相关信息  
│   │   ├── Common.py <----用于分割数据（尚未使用）  
│   │   ├── Cryption.py <----用于所有加密处理  
│   │   ├── File.py <----用于文件操作  
│   │   └── \_\_init\_\_.py <----初始化文件  
│   ├── communication.py <----整个系统封装在该文件下  
│   ├── file <----储存为文件  
│   │   ├── Rec <----接收端储存的文件  
│   │   │   ├── config.json <----JSON文件  
│   │   │   ├── key <----储存key的文件  
│   │   │   ├── msg <----储存接收信息的文件  
│   │   │   ├── seq <----储存接收端序列的文件  
│   │   │   └── state <----储存接收端状态的文件  
│   │   ├── Send   
│   │   │   ├── config.json <----JSON文件  
│   │   │   ├── key <----储存key的文件  
│   │   │   ├── msg <----储存发送信息的文件  
│   │   │   ├── seq <----储存发送端序列的文件  
│   │   │   └── state <----储存发送端状态的文件  
│   │   └── readme.md  
│   ├── netSim.py <----测试文件  
│   └── readme.md  
└── readme.md

使用方法

1，第一次运行时运行Setdefault.py重置文件信息。

2. 接下来，运行appRec.py 监听端口

3. 然后，运行appSend.py 发送数据

4. 若上述配置，正确就能正常运行

## 状态机



图1:发送者状态机



图2:接受者状态机

## 打包过程

第一步

message

message

Rand

Len8

Seq8

AESOUT

HMAC（32）

PACKET(992位)

包填充

AES加密

中1/3key

HAMC

前1/3key

打包

exchangge

ACK

data1

data2

第二步

HAMC

后1/3key

data3

HMAC（32）

第一步：

消息message已经打包生成data1，data1前8为储存序列seq，接下来8位储存message长度，接下来是message，最后随机填充长度为0-256位的字符串。

第二步：

AES处理date1使用中1/3key加密生成AESOUT，HMAC 处理date1使用前1/3key生成HMAC。使用AESOUT和HMAC生成date2。

使用exchangge函数处理date2，交换了date2前后部分，HMAC 处理交换后的部分得到ACK。

最后date2已经打包处理成1024的供发送的数据包date3。

# 六． 未解决问题