Interim Report

P2P Systems and Security

Group 09 - Ankit Bahuguna and Zeeshan Nasir

In this report you should give the explain the design you adopted for your module and your reasons for doing so. It should cover the following questions:

1. *What type of process architecture you have? (single threaded/multi-threaded/ multi-process, etc.) Do you use event loops or not?*
   A: We will be working on a single threaded architecture; possibly in the future, will try out multi-threaded scenarios in our project only if time persists. We will not be using event loops in the single threaded version, although we may consider the same for the multi-threaded architecture one we may consider it.

2. *Document the inter-module protocol, i.e, the protocol used for communication between your modules. This should cover the following points:*

- *Explain the message formats in the protocol (similar to the way they are explained in this document) and why they are needed.*

Overview: VoIP Application Working

1. For each pseudo-identity each peer periodically chooses a random peer as its exchange point and builds an onion tunnel to that peer.
2. They then store their respective exchange points signed by their corresponding pseudo-identity into the DHT with the public key of that pseudo-identity as a key.
3. A peer wishing to call will then perform a lookup in the DHT for the exchange point of the caller. The lookup is performed by searching for the public key of the pseudo-identity of the caller.
4. If the peer is unable to nd the exchange point, the call fails.
5. If the exchange point is found, the peer then builds an onion tunnel to that exchange point.
6. Call signalling happens via this onion tunnel until the exchange point, and from the exchange point via the onion tunnel the receiving peer has built.
7. After successful call parameter negotiation, the voice stream transmission starts.

- **STEP 1:** How to find pid_callee?:

    - The callee's VoIP must choose a random exchange point using DHT TRACE and
    - Then callee does a DHT PUT with its pseudo-identity, the address information of the exchange point and the port number it is going to listen for incoming connections.
- **STEP 2:** Caller(pid_callee):
    - DHT GET to get KX exchange point of callee
    - DHT REPLY
        - Negative: ERROR
        - Positive: Retrieve the address information of the exchange point and ask the KX to build outgoing tunnel to the exchange point.

- **STEP 3:** Afterwards, the callee's VoIP must ask its KX to build an incoming tunnel from the exchange point. When the tunnel is built, the VoIP must listen on the tunnel IP address given by the KX and the port number it advertised in the DHT

- **STEP 4:** MSG_KX_TN_READY : Tunnel is ready - Send to both caller and callee

- **STEP 5:** Establish session between caller and callee. Session keys can be derived using DH. This will encrypt all our messages for data and control.

- **STEP 6:** Caller Sends the voip-control messages via TCP first then data message containing the audio message via UDP.

    - **Control Phase based on TCP**
        - MSG_VOIP_CALL_INITIATE

size (16 bits)	**MSG_VOIP_CALL_INITIATE** (16 bits)
pseudo_identity (Callee) 256 bit	
NoT(Number of Tries) 8 bits	reserved

        - MSG_VOIP_CALL_INITIATE_REPLY

size (16 bits)	**MSG_VOIP_CALL_INITIATE_REPLY** (16 bits)
pseudo_identity (Caller) 256 bit	
ok_bit (boolean bit)	reserved

ok_bit : TRUE: signifies: Ok to call initiate;
FALSE signifies: Not Ok

- MSG_VOIP_CALL_BUSY

If ok bit is TRUE in previous message reply, then we send this message to seek if the callee is busy with another caller peer.

size (16 bits)	**MSG_VOIP_CALL_BUSY** (16 bits)
pseudo_identity (Callee) 256 bit	

- MSG_VOIP_CALL_BUSY_REPLY

size (16 bits)	**MSG_VOIP_CALL_BUSY_REPLY** (16 bits)
pseudo_identity (Caller) 256 bit	
ok_bit (boolean bit)	reserved

ok_bit : TRUE signifies: Callee is busy; Initiate - MSG_VOIP_CALL_WAITING
FALSE: Retry MSG_VOIP_CALL_INITIATE [To performed till some threshold of NoT is reached]

- MSG_VOIP_CALL_WAITING

size (16 bits)	**MSG_VOIP_CALL_INITIATE** (16 bits)
pseudo_identity (Callee) 256 bit	

- MSG_VOIP_CALL_WAITING_REPLY

size (16 bits)	**MSG_VOIP_CALL_WAITING_REPLY** (16 bits)	
pseudo_identity (Caller) 256 bit		
ok_bit (boolean bit)	TTW (Time To Wait) 8 bits	reserved

- MSG_VOIP_CALL_TERMINATE

size (16 bits)	**MSG_VOIP_CALL_TERMINATE** (16 bits)
pseudo_identity (Callee) 256 bit	

- MSG_VOIP_CALL_TERMINATE_REPLY

size (16 bits)	**MSG_VOIP_CALL_TERMINATE_REPLY** (16 bits)
pseudo_identity (Caller) 256 bit	
ok_bit (boolean bit)	reserved

- Data based on UDP
  - MSG_VOIP_CALL_DATA

size	**MSG_VOIP_CALL_DATA**
pseudo_identity (Callee) 256 bit	
Audio _Data* (16 KB)	
MD5 hash of Audio Data (128 bits)	

- *We will try to investigate what is the optimum audio data packet size for best quality of audio transmission on our network only if time persists; else default size right now is 16 KB; may change in the final implementation.

- *How do your modules running on different peers authenticate? This is needed because there could be another service running on the port allocated for your module on the remote peer and your module needs to verify that it is talking to that correct module on the remote peer.*

  A: We can authenticate the VoIP module running on other peer by sending the VoIP heartbeat messages before even sending the call initiation messages. Format of the heartbeat message will be as follows:

size (16 bits)	**MSG_VOIP_HEART_BEAT** (16 bits)
pseudo_identity (Callee) 256 bit	

  After sending the heartbeat message, we can expect from the VoIP module running on other peer to reply back the message according to the format specified below.

size (16 bits)	**MSG_VOIP_HEART_BEAT_REPLY** (16 bits)
pseudo_identity (Caller) 256 bit	
ok_bit (boolean bit)	reserved

- *How does your protocol handle exceptions, i.e peers going down, connection breaks, corrupted data, etc.*

  A: We will be incorporate exception handling in our code. Some of the prominent scenarios are as follows:
  - **Peers going down**: We will be sending heartbeat messages periodically once some predefined time has elapsed.
  - **Connection breaks:** We will send certain number of heartbeat messages to check whether connection is lost or not.
  - **Corrupted data:** To maintain the integrity of the audio packets sent , the caller will also send the hash (MD5 or any other) within the UDP packet. Thus the callee can perform the same hash and compare with the hash provided by the caller, to verify that data is corrupted or not.

- **Misbehaving/Malicious peers:** Misbehaving/Malicious peers can be identified if a peer does not reply back the **heartbeat messages** periodically as well as messages sent by the peer does not fulfill the **message protocols** defined by VoIP. So, once such peers are identified then we will drop the call immediately. We are maintaining a session between caller and callee, so this way we can protect our data from malicious peers who are trying to snoop between caller and callee.