Perseus on VoIP

Development and Implementation of VoIP Platforms

Saran Chiwtanasuntorn, Bhume Bhumiratana
Computer Engineering
King Mongkut's University of Technology Thonburi
Bangkok, Thailand
saranchiw@gmail.com, csbhume@gmail.com

Eric Filiol
Operational Cryptology and Virology Lab
France
ffiliol@gmail.com

***Abstract*—Nowadays, voice over IP (VoIP) is rapidly replacing standard phone line as a telephony service of choices. However, VoIP can be exploited by an attacker using method such as eavesdropping, hijacking and etc. To ensure confidentiality and integrity of the conversation or messages sent in the network, a more secure protocol is needed. This research proposes a solution to enhance the security of VoIP by using PERSEUS library. PERSEUS is a technique that encodes data with punctured convolutional codes and adds noise in a suitable way so any cryptanalytic must use supercomputers during a significant, scalable amount of time to decode without the key.**

Keywords—PERSEUS; VoIP; Voice over IP; Security

I. INTRODUCTION

Voice over IP (VoIP) also called IP Telephony or Internet Telephony is widely used and is rapidly replacing the traditional phone line. VoIP is very popular for many people who use softphone applications and services such as Hangouts, Line, Skype, and many more. It is used to transmit voice communication over a data network using Internet Protocol (IP). Not only can VoIP deliver voice communication, but the users can also have multimedia sessions such as video conferencing, instant chat messaging, and fax data over the IP network. The primary reason that many people prefers to use VoIP than PSTN (Public Switched Telephone Network) is the lower cost compare to the telephone call. In the beginning, there were some disadvantage of VoIP such as the voice quality and latency but these issues have been improved consecutively. Today the performance of VoIP is not different from based phone line. However, because the VoIP shares the network infrastructure with other IP network services, it is vulnerable to being exploited by malicious adversary. There are vulnerabilities in the standard deployment of VoIP protocols suite that can be attacked by the attacker such as hijacking, eavesdropping [6] and etc. [7]

While strong encryptions do not allowed the necessary action of states to detect illegal things, any PERSEUS-protected data can be broken by using supercomputer with a large amount of time and a resource which only state's organization can afford. This limits states' to focus and spy only on important suspects who may be involved in terrorism, mafia activities or child pornography due to resource limitation and are unable to afford to spy on innocent people. In other word, State can investigate a questionable data but they have not enough time to investigate every data. PERSEUS is not as decoding resilience as symmetric encryption, because it uses a different kind of technology, thus enabling law-enforcement the need to decode given court permission (unlike encryption, which cannot be decrypted) and also PERSEUS can be used in place where encryption of voice communication is deemed illegal.

This research proposes a solution to enhance the security of VoIP using PERSEUS technology [1]. PERSEUS provides protection from eavesdropping by encoding the messages and conversation inside channel generated using punctured convolutional code.

The rest of this proposal is organized as follows. Part II is related works. In part III, the implementation is described in more detail. Part IV is the result and finally, this research is concluded in part V.

II. BACKGROUND & RELATED WORK

A. PERSEUS Technology

PERSEUS is a technology that helps to protect the data when two people have to communicate to each other. PERSEUS encodes data with punctured convolutional codes instead of encrypting it using an encryption scheme. These codes have very high encoding speed and high correcting rate. Before the communication begins, the PERSEUS parameters are randomly generated such as polynomial size constraint, encoding rate, matrix puncturing, noise parameter, encoder polynomials, etc. Then a short HTTPS initial session with these parameters will be sent from the sender to the receiver. The receiver will receive the artificial deterministic noise and set up the suitable algorithm for data encoding. There are many reasons that make PERSEUS more appropriate than strong encryption.

It is very easy to detect encrypted data because by nature it has a high entropy profile. On the other side, the encoded data has a lower entropy profile closer to the plain data. This lower statistical profile enables the encoded data to pass any detection that used an entropy test or any other statistical detection while encrypted data cannot. From TABLE I. you can see that the PERSEUS-protected data entropy profile is lower than encrypted data entropy profile [1].

TABLE I. AVERAGE ENTROPY PROFILE FOR PLAIN, PERSEUS-PROTECTED AND AES ENCRYPTED DATA.

Noise probability	Plain data average entropy	PERSEUS-protected data	Encrypted data
5%	4.21	4.96	8.00
10%	4.21	6.19	8.00
15%	4.21	6.46	8.00
20%	4.21	7.11	8.00
25%	4.21	7.39	8.00
30%	4.21	7.45	8.00
35%	4.21	7.71	8.00

There are two main steps in the implementation structure of prototype PERSEUS library. The first step is communication setup step. The sender will randomly generate encoder, noise generator and noise generator secret key. Then, send the secret elements to the receiver through a HTTPS session. The receiver receives the HTTPS session.

The corresponding data structures are then initialized. The second step is processing data step. The sender encodes data and sends it to the receiver. The encoding steps are character to binary encoding, the punctured convolutional code coding itself, data puncturing right after the encoding, the binary to hex nibbles encoding and the addition of deterministic noise. The receiver decodes data, the decoding steps are removing the deterministic noise, hex nibble to binary transcoding, data unpuncturing and decoding.

B. *Session Initiation Protocol*

The Session Initiation Protocol (SIP) [2] is a signaling communication protocol for setting up connections across the network, normally used with multimedia communication sessions such as voice and video over IP networks. It can be used to establish, modify and terminate multimedia sessions between two-party or multiparty.

SIP is a peer-to-peer protocol which has been designed similar to the HTTP request and response transaction model and reuses most of the HTTP's header fields. Each SIP user in a SIP network has a unique SIP addresses. User registers to a server using their assigned SIP addresses. SIP user sends a SIP request to a server when the user establishes a call. For example, user A can call user B by using SIP URI. The user end point is called SIP User Agent (UA). A user agent can perform one of these two roles at a time. First, User Agent Client (UAC) is a user agent who is a SIP request sender. Second, User Agent Server (UAS) who is a user agent who is a SIP request receiver and returns SIP response.

C. *Real-time Transport Protocol*

The Real-time Transport Protocol (RTP) [3] is a primary standard packet for delivering data which has a real-time property such as voice or video over IP networks. It is one of the technical foundation and basic implementation of VoIP. RTP is an end-to-end, real-time protocol which involve streaming data. RTP protocol allows data transfer for both unicast and multicast sessions. It also provides facility for detection of some packet loss or out of sequence arrival data. This protocol can be secured with the Secure Real-time Transport Protocol (SRTP) [4].

D. *ZRTP*

ZRTP [5] is a cryptographic key-agreement protocol using Diffie-Hellman key exchange algorithm based on RTP protocol and use SRTP protocol for encryption. ZRTP protocol begins after two endpoints have established a signaling protocol, such as SIP, and is ready to exchange key. ZRTP starts with both endpoints send Hello packet to each other and a response of Hello packet is HelloACK packet. Next, one of the endpoints sends Commit packet, the one who sends Commit packet called the initiator and another one called the responder. After that, the responder sends DHPart1 packet and the initiator sends DHPart2 packet to perform Diffie-Hellman key-agreement. Now, both endpoints generate SRTP session key. Then, the responder and initiator sends encrypted session key via Confirm1 and Confirm2 packet respectively. Last, the responder send Conf2ACK packet to finish the ZRTP protocol and begin SRTP protocol.

E. *Secure Real-time Transport Protocol*

The Secure Real-time Transport Protocol (SRTP) is an extension from RTP protocol. The purpose is to make RTP more secure to ensure the confidentiality and integrity of RTP payloads. SRTP also has low bandwidth and low computational cost. By default, SRTP protocol uses AES in counter mode as an encryption scheme and HMAC-SHA1 as a message authentication and integrity. After the ZRTP connection is finished, the server and client will change normal RTP packets into SRTP packets which the payload is encrypted by using AES and append the packets with authentication tag.

III. IMPLMENTATION

This research merges PERSEUS and VoIP technology therefore the implementation must be done on both the server and the client. This research uses FreeSWITCH [9] as a prototype server software and Twinkle [10] (version 1.4.2) as a client software because both support ZRTP protocol. PERSEUS implementation for both the server and the client has to be configured in some part such as the parameters and encoding/decoding part to make PERSEUS library (version 1.0.10) appropriate with VoIP platform. Before the VoIP connection starts, Twinkle must use PERSEUS library to

initialize the parameters. Next, FreeSWITCH which act as a prototype server will exchange all parameters between two end points via ZRTP protocol. We modified Twinkle to replace AES encryption scheme in the SRTP protocol with PERSEUS encoding/decoding scheme by using PERSEUS library before transmitting it via SRTP protocol. FreeSWITCH uses libZRTP SDK library for key-agreement and twinkle uses GNU ccRTP [16] (ccrtp-1.8.0) and GNU ZRTP [15] (libzrtpcpp-1.4.6) for encryption and key-agreement respectively.

A. *Key Exchange Module*

Every connection must begin with a key exchange module. Both end points will generate their own parameters before setting up the connection such as encoder, noise generator and noise generator secret key. After that, client A (the client who connect the server via ZRTP first) will start the connection with the server via ZRTP protocol and send his parameters to the server. Server will forward these parameters to client B (The recipient of the VoIP call), who will then use the A's parameter to initialize his PERSEUS decoding engine. Fig. 1. shows the original and the additional parts in ZRTP of the implementation.

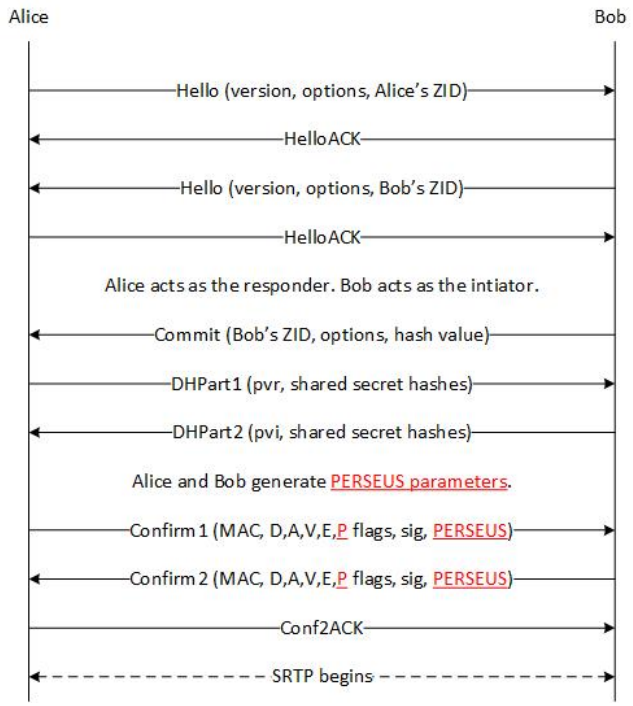

Fig. 1. The original ZRTP in black and the additional PERSEUS in red.

There are two steps of FreeSWITCH ZRTP connection. First, the connection between the server and the client A and the second, the connection between the server and the client B. These two steps but we must append the PERSEUS parameters to the messages sent confirming the correct DH key exchange. In the first step, after both clients generate PERSEUS parameters, the client A sets the PERSEUS flag as true and sends the parameters to the server. The server then forwards these parameters to the client B. In each step, after Initiator and responder finish sharing Diffie-Hellman secret key in DHPart1 and DHPart2 packet. PERSEUS parameters will be appended to Confirm1 or Confirm2 packet depends on the role of PERSEUS sender. Note that if there are PERSEUS parameters within the Confirm packet, the PERSEUS (P) flag will be set.

libZRTP SDK library has to be modified as follows:

- PERSEUS flag addition and management.
- Synchronization Source Identifier (SSRC) management.
- PERSEUS parameters forwarding.

GNU ZRTP library has to be modified as follows:

- PERSEUS flag addition and management.
- Appending PERSEUS parameters and converting (from data to encoder).

B. *Encoding and Decoding Module*

After the key exchange session (ZRTP) is finished, the communication (SRTP) will start immediately. The sender side will encode all of the packets by encoding it in the noise channel before transmits to the receiver side then the receiver will decode the packet by removing the noise. This implementation is an end-to-end encoding so there is no plain data sending around the network.

The original SRTP uses four AES secret keys (first is server - client A, second is client A - server, third is server - client B and fourth is client B - server). The server will decrypt the received message from client A with 1st secret key (shared secret key from server to client A) and encrypt the message with third secret key (shared secret key from server to client B) and vice versa. The server also computes a new authentication tag and appends it at the end of SRTP packet.

Instead of four AES secret keys, this implementation use only one set of PERSEUS parameters, so the server only computes new authentication tag and it can forward any encoded messages to another client, there is no need to encrypt or decrypt the message at the server node.

libZRTP SDK library has to be modified as follows:

- Disable encryption and decryption scheme in SRTP module.

GNU ccRTP library has to be modified as follows:

- Encode and decode the message part using PERSEUS.

IV. RESULTS

This implementation has been tested on Ubuntu 12.04 [8], 512 MB RAM virtual machine which has 4 GB RAM, Intel Core i5-2400 CPU (3.10 GHz) as a server; 4 GB

RAM, Intel Core 2 Duo CPU P8600 (2.40 GHz) and 2 GB RAM, Pentium Dual-Core CPU E5400 (2.70 GHz) as clients. The results can be classified in four types: delay of time, bandwidth used, voice performance and entropy profile.

A. Latency

There is no perceptible latency difference between the standard VoIP implementation and the PERSEUS augmented VoIP implement. However, there is a significant delay in connection initialization when the client registers to the SIP server. Due to the way PERSEUS library in implemented, it takes about 6 - 7 minutes to generate PERSEUS parameters depend on the level of noise probability. This is the minimum requirement for any PERSEUS-protected connection. On the other hand, the process of encoder establishment or encoding and decoding data is very fast that means there is no additional delay of time after the conversation started.

This problem can be solved by parallelising the parameter setup and noise generation alongside the conversation to lower than initiate setup latency.

B. Bandwidth

We use Wonder Shaper [14] to limit the bandwidth to measure minimum required bandwidth for a usable connection. From our test, the PERSEUS-protected VoIP connection requires 768/768 Kbps (download/upload speed) for each pair of connection. Which means both clients are required to have 768/768 Kbps connection for uninterrupted conversation, while the server requires 1.5/1.5 Mbps to connect and route traffic between both clients.

C. Voice Performance

Voice performance is very good. There is no perceptible difference between using PERSEUS-protected, AES-protected or unprotected VoIP conversation. This shows that PERSEUS can losslessly hide the content of the communication. Fig. 2. shows the spectrogram and the additional distortion from the listener side (lower spectrogram) compare to the speaker side (upper spectrogram) of RTP protocol and Fig. 3. for PERSEUS implementation. At the distortion point, the voice will be more bass than the original voice. We directly capture voice before (speaker) and after (listener) VoIP engine and plot spectrogram by using Audacity [13]. The results are same as other VoIP protocol such as RTP or AES encryption on SRTP which have the distortion on the listener side.

D. Entropy Profile

This implementation uses entropy profile as a factor to ensure that it contains the original PERSEUS property. The entropy of PERSEUS-protected data should be lower than AES encrypted data.

This implementation tested the entropy profile of plain data from RTP protocol. We captured the data by using

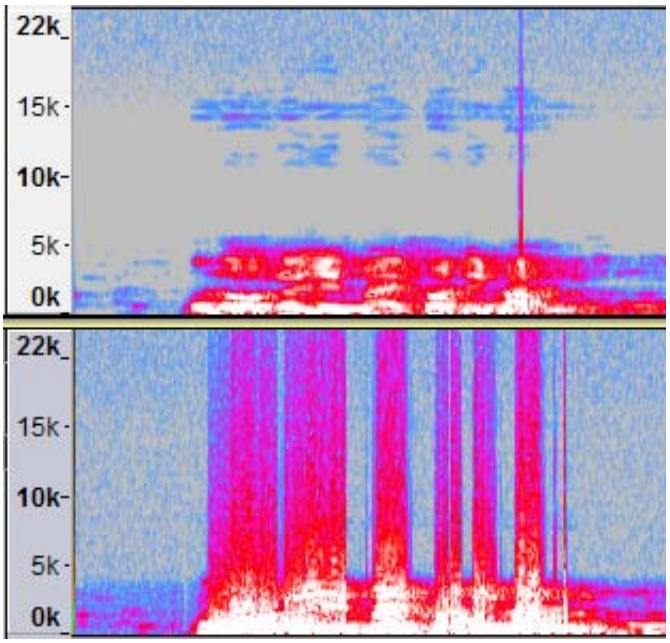

Fig. 2. Spectrogram and distortion of the RTP speaker (top) and listener (bottom).

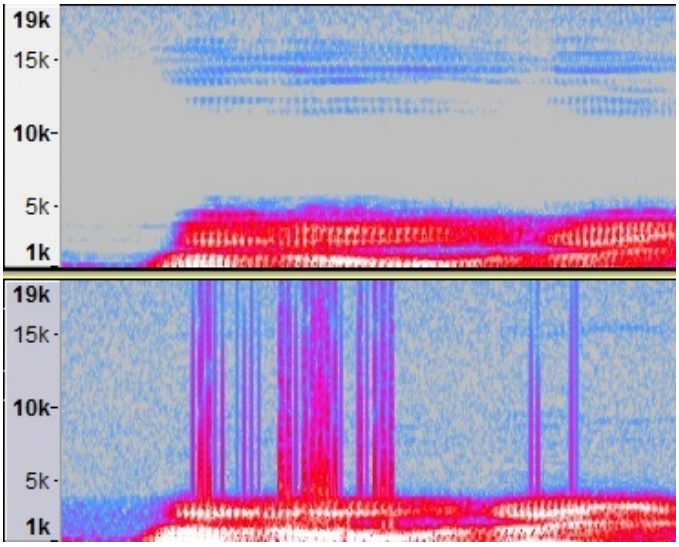

Fig. 3. Spectrogram and distortion of the PERSEUS speaker (top) and listener (bottom)

Wireshark [11] and calculated the entropy profile by MATLAB [12]. The plain data of voice (RTP) is high because of the voice property. The pressure and noise from your voice are not stable so the encoded data from the voice is various. The AES-encrypted data has the highest entropy profile because of the encryption property which is seems to be completely random. The entropy profile of PERSEUS-protected data is similar to plain data which is almost high compare to other plain data that is not a voice. The similarity of both entropy profile makes the man-in-the-middle cannot differentiate between unprotected and PERSEUS-protected VoIP.

TABLE II. AVERAGE ENTROPY PROFILE FOR PLAIN, PERSEUS-PROTECTED AND ENCRYPTED DATA IN THIS IMPLEMENTATION.

Plain data (RTP)	PERSEUS-protected data	Encrypted data (SRTP)
7.13	7.17	8.00

V. CONCLUSION

This research integrates PERSEUS with VoIP to increase security of voice communication, without requiring the use of Cryptography techniques such as encryption. PERSEUS is not as strong as traditional encryption scheme such as AES, thus allowing law enforcements to monitor for illegal activity, yet because of the computation cost required to decode PERSEUS is high, the law enforcement agency can only monitor the truly suspicious connections and not every connection, thus providing privacy and security for the legitimate users. We believe this is a good balance between law enforcement need to monitor illegal activities such as those used by terrorist, and protecting privacy of legitimate citizen use.

There remain some issue with using PERSEUS with VoIP which is the initial setup time for communication, but for limited setup, this problem can be solved by pre-generating noise parameter before the conversation ahead of time. The actual noise encoding and decoding is very fast, and there is no perceptible difference compare to AES encryption.

We implemented and tested the implementation and present the result in this paper and report that PERSEUS can be used with VoIP with little performance impact, and the strength of PERSEUS is preserved.

REFERENCES

[1] E. Filiol, “PERSEUS Technology: New Trends in Information and Communication Security,” in iAWACS 2010 conference, 2010. Technical paper available on http://arxiv.org/abs/1101.0057

[2] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol," RFC 3261, June 2002.

[3] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," STD 64, RFC 3550, July 2003.

[4] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)," RFC 3711, March 2004.

[5] P. Zimmermann, A. Johnston, Ed., and J. Callas, "ZRTP: Media Path Key Agreement for Unicast Secure RTP," RFC 6189, April 2011.

[6] Symantec Connect Community, “Two attacks against VoIP,” *http://www.symantec.com/connect/articles/two-attacks-against-voip.*

[7] S. McGann, D. C. Sicker, “An Analysis of Security Threats and Tools in SIP-Based VoIP Systems,” Second VoIP Security Workshop, 2005.

[8] Ubuntu, The World’s Most Popular Free OS, *http://www.ubuntu.com.*

[9] FreeSWITCH, The World’s First Cross-Platform Scalable FREE Multi-Protocol Soft Switch, *http://www.freeswitch.org.*

[10] Twinkle, *http://www.twinklephone.com.*

[11] Wireshark, *http://www.wireshark.org*

[12] MATLAB, The Language of Technical Computing, *http://www.mathswork.com/products/matlab.*

[13] Audacity, Free, Open Source, Cross-Platform Software for Recording and Editing Sounds, *http://www.audacity.sourceforge.net.*

[14] Wonder Shaper, *http://www.lartc.org/wondershaper.*

[15] GNU ZRTP distributed as libzrtpcpp, *http://www.gnutelephony.org/index.php/GNU_ZRTP.*

[16] GNU ccRTP, *http://www.gnu.org/software/ccrtp.*