Secure VoIP Using a Bare PC

Gholam H. Khaksari
Pennsylvania State University
PA, USA
Email: ghk10@psu.edu

Alexander L. Wijesinha, and Ramesh K. Karne
Towson University
MD, USA
Email: {awijesinha, rkarne}@towson.edu}

***Abstract*—Bare PC applications do not use an operating system or hard disk. We present a lightweight VoIP security scheme for a bare PC softphone that consists of an RSA-based key exchange, AES voice encryption, and SHA-1 data integrity and authentication. The scheme is easily extended to incorporate replay protection and a key derivation function as specified in SRTP for example. Experimental results comparing this security scheme on bare PC and Windows softphones show that the bare PC softphone has packet inter-arrival times (delta) that are closer to the ideal value, and intrinsic jitter values that are smaller and less variable. Moreover, the total time to complete the key exchange, and the time to generate RSA keys of various sizes or a number of RSA keys of a given size are significantly less on the bare PC softphone.**

Keywords-bare PC; softphone; VoIP; security; performance

I. INTRODUCTION

A bare PC [1] is an ordinary desktop or laptop that runs self-contained applications directly over the hardware without using an operating system (OS) or a hard disk. It may be used for protecting applications or devices against attacks that target OS vulnerabilities. Other advantages of bare computing include the elimination of OS overhead present in a conventional system, and the ability to run self-executing bare applications, which may be carried on a portable storage medium such as a USB flash drive, on a variety of devices without requiring OS support.

In this paper, we present a lightweight security scheme for a bare softphone application that provides key exchange, user authentication, voice encryption, and data integrity. We determine the overhead due to this scheme by measuring the maximum packet interarrival gap (delta) and intrinsic jitter in a LAN environment. We also measure the time for key generation and individual operations performed during the exchange of keys, and compare the performance of bare PC and Windows softphones equipped with this security scheme.

Since this paper focuses on protecting the voice stream between bare PC softphones, we do not address call setup, which may be handled by a standard VoIP infrastructure using SIP [2] and associated user agent clients and servers as in conventional systems. We also do not discuss communication between bare PC softphones behind NAT/firewalls since this may be achieved if needed using external servers and well-known techniques that are used by VoIP systems today [3].

The rest of this paper is organized as follows. In Section 2, we briefly summarize related work. In Section 3 we describe the implementation of secure VoIP on a bare PC. In Section 4, we present the experimental results. Section 5 contains the conclusion.

II. RELATED WORK

The design, implementation and performance of a bare PC softphone with no security features are described in [4] and [5]. Siscophone [6] is a softphone incorporating several optimizations similar to those implemented in the bare PC softphone. However, it runs on a conventional OS and does not appear to incorporate any security mechanisms. While softphones used by VoIP providers, including Skype [7], provide secure communication, they also require an OS. Some softphones use SRTP [8] proposed by the IETF for securing RTP media streams. SRTP may be used with a variety of key exchange methods, including ZRTP [9]. SRTP performance for a bare PC softphone and OS-based softphones is compared in [10].

III. BARE PC VOIP SECURITY

A bare PC VoIP softphone application can run on any PC such as a desktop or laptop with an Intel Architecture 32-bit (IA 32) CPU. The absence of an OS enables optimization of a bare PC application through streamlined protocol design, zero copy buffering, and novel task scheduling as described in [5]. The softphone application includes drivers for the relevant network (Ethernet) interface and audio hardware, a G.711 codec, and a simple (non-graphical) user interface.

To make a call, the peers establish a TCP connection using a pre-determined port and exchange the AES key. The key exchange is followed by the transfer of encrypted authenticated voice packets over RTP/UDP. It is also possible to transfer a master secret during the TCP connection from which keys needed for encryption and authentication may be derived. The key exchange can also be done over UDP with acknowledgments and retransmission.

The simple AES key exchange between bare PC softphones is shown in Fig. 1. The caller first generates a random AES key (128-bit default) and its SHA-1 hash value. The latter is signed (encrypted) using the caller's private RSA key. The public RSA keys of users can be exchanged by out-of-band means (or downloaded from a secure server)

and stored on the portable medium carried by users. The AES key, its signature, and a field indicating the length of the signature are then encrypted with the callee's public RSA key and a single message consisting of the encrypted content is sent to the callee. If it is necessary to ensure liveness, additional messages and timestamps/random values can be used. The callee decrypts the message using its private RSA key and recovers the AES key and its signature using the signature length field. After the signature is decrypted using the public RSA key of the caller and verified by re-computing the SHA-1 hash value using the recovered AES key and comparing it with the decrypted signature, the TCP connection is closed.

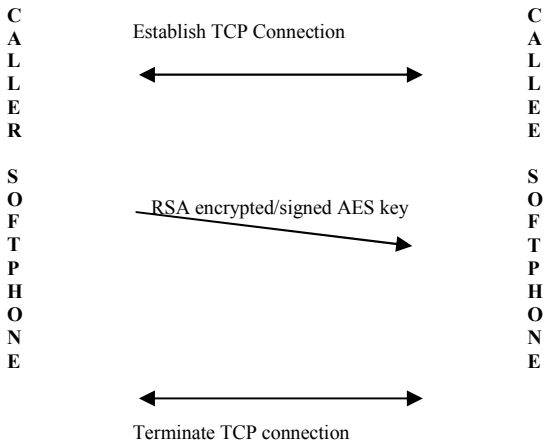


Figure 1. Secure exchange of the AES key

If the callee is unable to verify the signature, the call connection attempt fails and the users are notified prior to closing the TCP connection. The key exchange ensures that only the callee with knowledge of its private RSA key can recover the AES key by decryption. Moreover, successful verification of the signature using the caller's RSA public key implies that only the caller could have generated the AES key and that it has not been altered. This simple AES key exchange is vulnerable to a denial-of-service (DoS) attack that opens a large number of TCP connections to the callee and forces it to perform decryption using RSA keys. Such attacks can be handled by an adaptation of known techniques to combat SYN floods and delaying the RSA computations until the caller returns a random value chosen by the callee. We do not consider the prevention of DoS attacks in this paper. To protect private keys on a bare PC softphone, a standard technique such as encryption with an AES key (derived from a password-based SHA-1 hash) could be used.

If the key exchange is successful, each side is able to send and receive encrypted authenticated voice messages over UDP using a pre-determined port. First, voice is sampled at 8000 samples/sec for 20 ms, using 16-bit samples, which produces 160 bytes of G.711-compressed voice data as described in [5]. Next, a 20-byte SHA-1 hash of the voice data and RTP header is computed, and the voice data and hash is encrypted using the AES key. Before encryption, padding bytes are added since AES requires that the plaintext length be a multiple of 16 bytes. The padding field, pad count and padding indicator in the RTP header are used for this purpose.

The receiver decrypts the encrypted content using the AES key and verifies the hash in the usual manner. Since the AES key is only known by the communicating entities, only the receiver can decrypt voice messages, and their integrity and the sender's authenticity are guaranteed. The voice packet is then played back from a jitter buffer after G.711 decompression. Details concerning the design and implementation of a bare PC softphone are given in [5]. Once the conversation is finished, a call on a bare PC softphone is terminated manually by each side.

By including the RTP header (which includes a timestamp and sequence number) in the hash, additional protection is afforded for voice packets. Although we only use an AES-encrypted hash to authenticate the voice packets and the sender, if a master secret is exchanged originally instead of an AES key, a secret value derived from the master secret can be used with HMAC/SHA-1. The RTP packet could also be protected as in SRTP, in which case only the voice data, RTP padding and pad count fields are encrypted, and the encrypted portion plus the RTP header is authenticated. Additional enhancements specified in SRTP such as replay protection using a rollover counter and replay list, and AES counter mode, which involves the encryption of successive integers, could be easily added to a bare PC softphone if needed.

IV. RESULTS

We conducted several experiments to evaluate the performance of the lightweight security scheme on a bare PC softphone and to compare the results with a WinRTP softphone [11] running on Windows with the same security scheme. We use a softphone running on Windows instead of Linux for comparison with the bare PC softphone since it illustrates the worst-case overhead of security operations due to an OS. WinRTP was chosen over other softphones running on Windows since its source code was readily available at the time the experiments were done. We measured (1) maximum packet inter-arrival time (delta); (2) maximum and mean jitter with and without security; (3) internal timings to complete the each step of the key exchange; and (4) times to generate RSA keys of various sizes and a large number of RSA keys of a given size.

The test LAN consisted of PCs connected to a 100 Mbps Ethernet. The bare PC and Windows softphones ran on identical 2.4 GHz Dell Optiplex GX260 PCs with 512MB of memory, an external 3Com 905CX network interface card, and onboard audio chip AD1981B.

Each softphone collects its own timing data, and a Wireshark packet analyzer [12] running on a separate Windows PC is used to measure jitter and delta values. Since

there is no other traffic on the LAN, there is no packet loss or significant end-to-end network delay. The values of jitter and delta are therefore intrinsic and can be used to estimate the overhead due to processing the voice packets with and without security. Calls were made between two bare PC softphones and between two WinRTP softphones (results for calls between a bare PC and a WinRTP softphone are similar and are not included). Data was collected for approximately 20 seconds and each experiment was repeated three times to ensure that the results were consistent. For each run, we first averaged the results for the voice streams in both directions, and then computed the average over the three runs.

Fig. 2 shows the maximum packet inter-arrival time (delta), and maximum and mean jitter when transferring 20-ms of voice data between two bare PC softphones and two WinRTP softphones. Results without and with security are given. For the bare PC softphone, security consists of adding a 20-byte SHA-1 hash and encrypting the voice data and hash with a 128-bit AES key (this is the key size commonly used for AES). The increase in maximum delta due to security is about 170 µs, while there is no increase in maximum or mean jitter due to security.

We were unable to successfully modify the WinRTP softphone code to add a SHA-1 hash to the voice data and encrypt the payload and hash, so the results for the WinRTP softphone in Fig. 2 are for voice payload encryption only. The increase in maximum delta due to security is 50 µs, while there is a negligible decrease in both maximum and mean delta with security. Compared to the bare PC softphone, maximum delta for the WinRTP softphone is about 730µs more, maximum jitter is about 20 µs more, but mean jitter is 20 µs less (possibly due to the reduced overhead in WinRTP from excluding the SHA-1 hash). The results indicate that addition of security mechanisms to a bare PC softphone or to a WinPC softphone has no significant impact on the intrinsic delta or jitter values. However, the bare PC softphone has slightly lower values of maximum delta and maximum jitter even though the overhead of adding and encrypting the SHA-1 hash is not included in WinRTP.

Fig. 3 compares the results for the same experiment for bare PC and WinRTP softphones with AES keys sizes of 128, 192, and 256 bits. Due to the additional processing required with a larger key size, there is a constant insignificant increase of about 20 µs in maximum delta for the bare PC softphone between the 128 and 256-bit key sizes; the corresponding difference for the WinRTP softphone is 180 µs. Maximum delta for the WinRTP softphone for 128, 192 and 256-bit keys is respectively 560, 620 and 760 µs larger than for the bare PC softphone. Maximum and mean jitter for the bare PC softphone remains nearly constant across all key sizes. In contrast, maximum and mean jitter for the WinRTP softphone increases by 220 and 30 µs respectively between the 128 and 256-bit key sizes. Maximum jitter for the WinRTP softphone for 128, 192 and 256-bit keys is respectively 20, 30 and 100 µs more than for the bare PC softphone. However, mean jitter for the WinRTP softphone for 128 and 192-bit keys is respectively 20 and 40 µs less than for the bare PC softphone, but 20 µs more for 256-bit keys.

Thus on the bare PC softphone, there is no significant difference in maximum delta or maximum and mean jitter values due to encryption even when increasing the AES key size. This stability is a result of streamlined processing and efficient task scheduling on the bare PC softphone. The implication is that higher levels of security may be achieved by increasing increasing the AES key size on the bare PC softphone with no difference in cost. The values of maximum delta and maximum jitter for the WinRTP softphone were higher than for the bare PC softphone regardless of the key size. The increased mean jitter values seen for the bare PC softphone with the smaller key sizes are likely due to the extra processing associated with the SHA-1 hash.

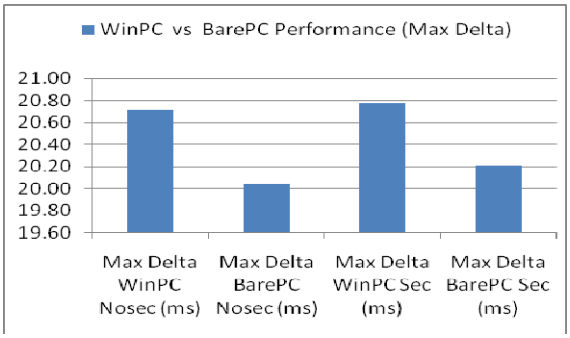
In Fig. 4, we compare key exchange performance for bare PC softphones and WinRTP softphones when exchanging 128, 192 and 256-bit AES keys. The steps in the key exchange as described earlier involve AES key generation, generation of a 20-byte SHA-1 hash, signing the hash, and encrypting the AES key and signature using a 256-bit RSA key. Although the total time increases by about 20 ms for each increase in the AES key size, the time for the bare PC softphone is about 238 ms less than for the WinRTP softphone for all key sizes. This constant reduction in overhead may be attributed to the processing efficiency on the bare PC softphone.

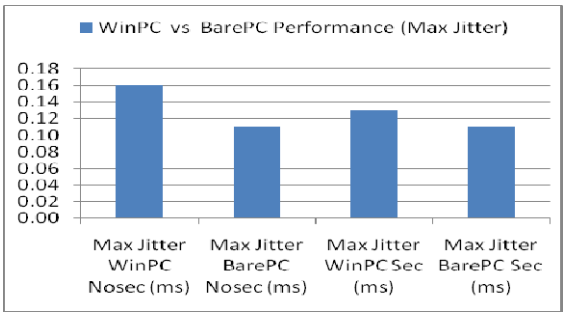
Table I provides more insight into the observed performance difference between the bare PC and WinRTP softphones with respect to the AES key exchange. It shows the time for the various components of the key exchange for the two softphones at the sender and the receiver using a 256-bit RSA key, a 256-bit AES key and a 20-byte SHA-1 hash. While AES key generation and computing the SHA-1 hash take minimal time, the bulk of processing time involves RSA encryption and decryption. These results imply that the performance impact on the key exchange of increasing the RSA key size (to increase security strength) will be less on a bare PC softphone.

In Fig. 5, we compare the time to generate 256, 512 and 1024-bit RSA keys on bare PC and WinRTP softphones. The benefit of using a barePC softphone is greatest when generating 1024-bit keys. However, these results indicate that even with optimized processing on the bare PC softphone, RSA key generation overhead with larger key sizes is significant and may not be acceptable unless a high level of security is desired.

Fig. 6 compares the total time for generating between 10 to 100 keys on bare PC and WinRTP softphones. The time to generate 100 keys on a bare PC softphone is about 3 minutes less than on a WinRTP softphone. Generation of multiple keys could be done in the background and the faster completion time of the bare PC softphone suggests that key

generation will have a lesser effect on voice calls and other applications running simultaneously.





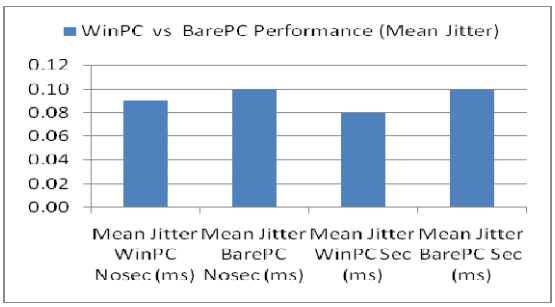
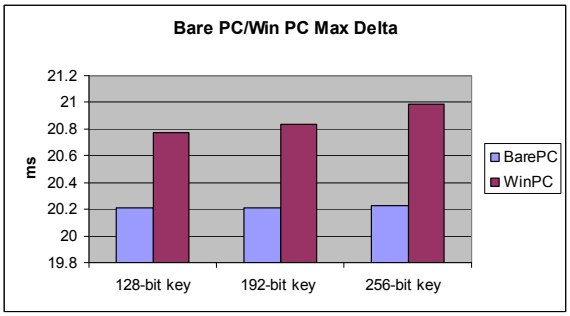
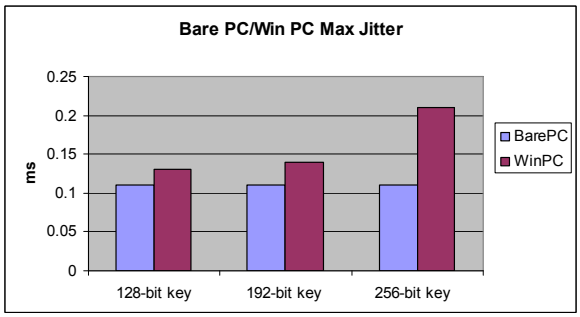


Figure 2. Maximum delta, maximum and mean jitter for the bare PC and WinRTP softphones with and without security (AES encryption with a 128-bit key and a 20-byte SHA-1 hash).





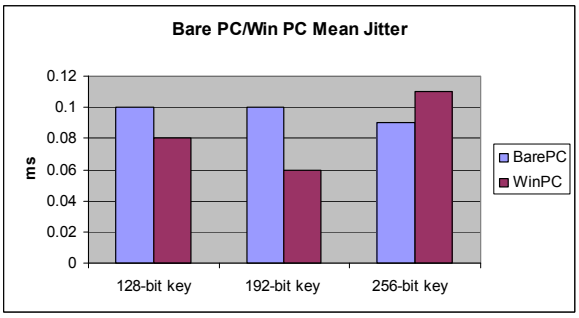


Figure 3. Maximum delta, maximum jitter, and mean jitter for barePC to barePC and WinRTP to WinRTP calls for various AES key sizes.

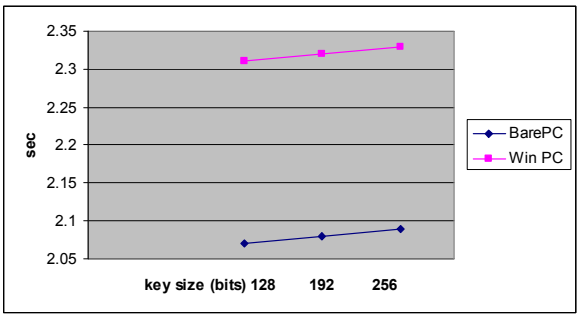


Figure 4. Total processing time for the key exchange for bare PC softphones and WinRTP softphones for various AES key sizes.

TABLE I. TIME FOR VARIOUS STEPS OF THE AES KEY EXCHANGE FOR BARE PC SOFTPHONES AND WINRTP SOFTPHONES USING 256-BIT RSA KEYS AND A 256-BIT AES KEY.

KEY EXCHANGE STEP	BARE PC (MS)	WIN PC (MS)
CALLER		
COMPUTE SHA-1	0	0
ENCRYPT SHA-1	736	782
RSA ENCRYPTION	554	588
CALLEE		
DECRYPT AES	195	254
RE-COMPUTE SHA-1	0	0
DECRYPT SHA-1	589	688
COMPARE SHA-1	0	0

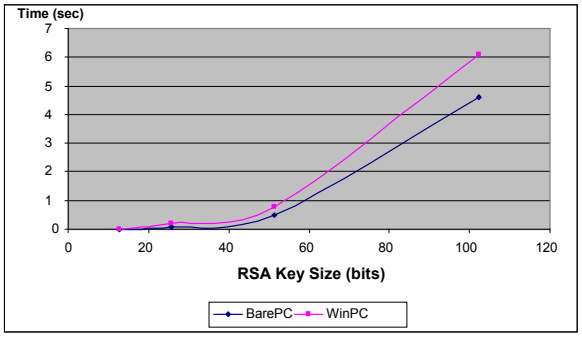


Figure 5. RSA key generation time for various RSA key sizes on barePC and WinRTP softphones.

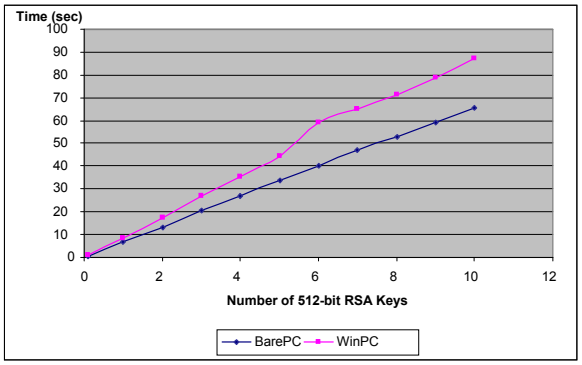


Figure 6. Total time for generating 10-100 RSA keys on barePC and WinRTP softphones.

V. CONCLUSION

We presented a lightweight security scheme for VoIP using bare PC softphones. A simple key exchange based on RSA is used to securely transfer an AES session key. Then the voice data is encrypted using AES, and a SHA-1 hash is computed for authentication and integrity of voice data. Comparison of bare PC and WinRTP softphone performance with security indicates that the bare PC has less variation in packet interarrival time and more stable intrinsic jitter than the WinRTP softphone. Furthermore, the total time to exchange AES keys of various sizes is less on a bare PC softphone than a WinRTP softphone. Bare PC softphones can use also use larger RSA key sizes with less overhead than a WinRTP softphone, and generation of multiple RSA keys is faster on bare PC softphones. The results indicate that bare PC softphones can provide secure VoIP with better performance than compatible OS-based softphones.

REFERENCES

[1] R. K. Karne, K. V. Jaganathan, and T. Ahmed, "DOSC: Dispersed Operating System Computing," 20th Annual ACM Conference on Object Oriented Programming, Systems, Languages, and Applications, Onward Track (OOPSLA '05), San Diego, CA, October 2005, pp. 55-61.

[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] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing. "Session Traversal Utilities for NAT (STUN)," RFC 5389, October 2008.

[4] G. H. Khaksari, A. L. Wijesinha, R. K. Karne, L. He and S. Girumala, "A Peer-to-Peer Bare PC VoIP Application," Consumer Communications & Networking Conference (CCNC 2007), pp. 803-807, Jan 2007.

[5] G. H. Khaksari, A. L. Wijesinha, R. K. Karne, Q. Yao, and K. Parikh, "A VoIP Softphone on a Bare PC," International Conference on Embedded Systems and Applications (ESA '07), June 2007.

[6] O. Hagsand, I. Marsh and K. Hanson. Sicsophone: A low-delay Internet telephony tool," 29th EUROMICRO Conference "New Waves in System", 2003

[7] S. A. Baset and H. Schulzrinne, "An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol," INFOCOM '06, Barcelona, Spain, April 2006.

[8] M. Baugher, D. McGrew, M. Naslund, E. Carrara and K.Norrman. "The secure real-time transport protocol (SRTP)," RFC 3711, March 2004.

[9] P. Zimmermann, A. Johnston, and J. Callas, "ZRTP: Media Path Key Agreement for Secure RTP," Internet-Draft, March 2009.

[10] A. L. Alexander, A. L. Wijesinha, R. K. Karne, "An Evaluation of Secure Real-Time Protocol (SRTP) for VoIP," 3rd International Conference on Network & System Security (NSS 2009), in press.

[11] WinRTP, http://www.pbx.in/voip-info/wiki/view/VOCAL.html.

[12] Wireshark, http://www.wireshark.org.