SRTP+: An Efficient Scheme for RTP Packet Authentication

Sachin Garg, Navjot Singh, and Timothy Tsai
Avaya Labs Research
Basking Ridge, NJ 07920 USA
{sgarg,singh,ttsai}@research.avayalabs.com
January 9, 2004

Abstract

The Real-time Transport Protocol (RTP) is susceptible to several security attacks, including third-party snooping of private conversations, injection of forged content, and introduction or modification of packets to degrade voice quality. The Secure Real-time Transport Protocol (SRTP) provides confidentiality, message authentication, and replay protection for RTP traffic. However, SRTP incurs an additional overhead to verify the HMAC-SHA1 message authentication code for each packet. SRTP+ is a set of proposed schemes that significantly decrease the verification overhead compared to SRTP and thereby increases the number of faked packets required to mount a successful denial of service attack. SRTP+ provides packet authentication but not integrity. However, SRTP+ is compatible with SRTP.

1 Introduction

Telephony via Voice over IP (VoIP) offers tremendous potential in rich features and cost savings. However, the leveraging of data networks and their corresponding communication protocols also carries attendant security vulnerabilities. Furthermore, VoIP protocols for signaling and media transport themselves present additional vulnerabilities that might lead to toll fraud, privacy compromise, voice quality degradation, or denial of service (DoS). In particular, the Real-time Transport Protocol (RTP) [1] used as the basis for media transport is susceptible to several attacks, including third-party snooping of private conversations, injection of forged content, and introduction or modification of packets to degrade voice quality.

In response to concerns about these RTP vulnerabilities, the Internet Engineering Task Force (IETF) Audio/Visual Transport Working Group is working on an Internet Draft for the Secure Real-time Transport Protocol (SRTP) [2], which provides confidentiality, message authentication, and replay protection for RTP traffic.[1] The SRTP draft specifies AES [3] encryption of the RTP payload and a message authentication hash of the header and the encrypted payload using HMAC-SHA1[2] [4] to achieve enhanced security. Some Avaya products utilize AEA [5]encryption as a temporary solution until full AES support is implemented.

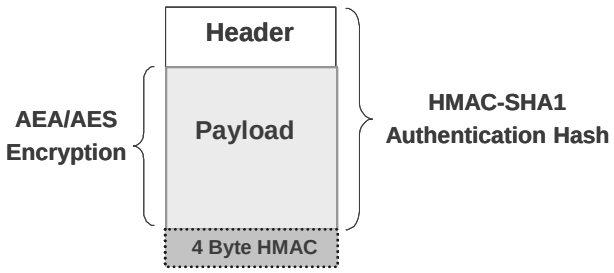

Figure 1 SRTP Packet

One potential concern for SRTP is the overhead imposed by the authentication hash. As seen in Figure 1, the entire header and encrypted payload are hashed via the HMAC-SHA1 algorithm to produce an 80-bit or 32-bit message authentication tag.[3] The tag must be computed for each packet before authentication can be determined.

[1] As with SRTP, SRTCP provides similar security for RTP Control Protocol (RTCP) [1] traffic.
[2] HMAC-SHA1 is the default message authentication code, and its implementation is mandatory. However, optional message authentication codes are permitted.
[3] HMAC-SHA1 produces a 160-bit digest and recommends that no more than 80 bits to be truncated from the least-significant end. However, 128 bits can be truncated (resulting in a 32-bit authentication tag) for bandwidth

Thus, one possible denial of service attack is to bombard a target with a series of forged packets, each of which contains an improper authentication tag. Figure 2 shows the steps for computing an authentication tag using HMAC-SHA1. The main computation includes hashing N=L+3 blocks of 512 bits, where L=ceil(M/64) and M is the number of bytes in the RTP header and payload covered by the authentication tag. The full computation also includes a few XOR, padding, copying operations, but the total execution time is heavily dependent on the hashing operations. The main processing for one SHA-1 hash consists of four rounds of 20 steps each. These four rounds contain a total of 740 32-bit logical and arithmetic operations (AND, OR, NOT, XOR, modulo addition). In addition to the hash operations, the two 512-bit XOR operations require 32 32-bit logical XOR operations. However, these two XOR operations can be optimized away.

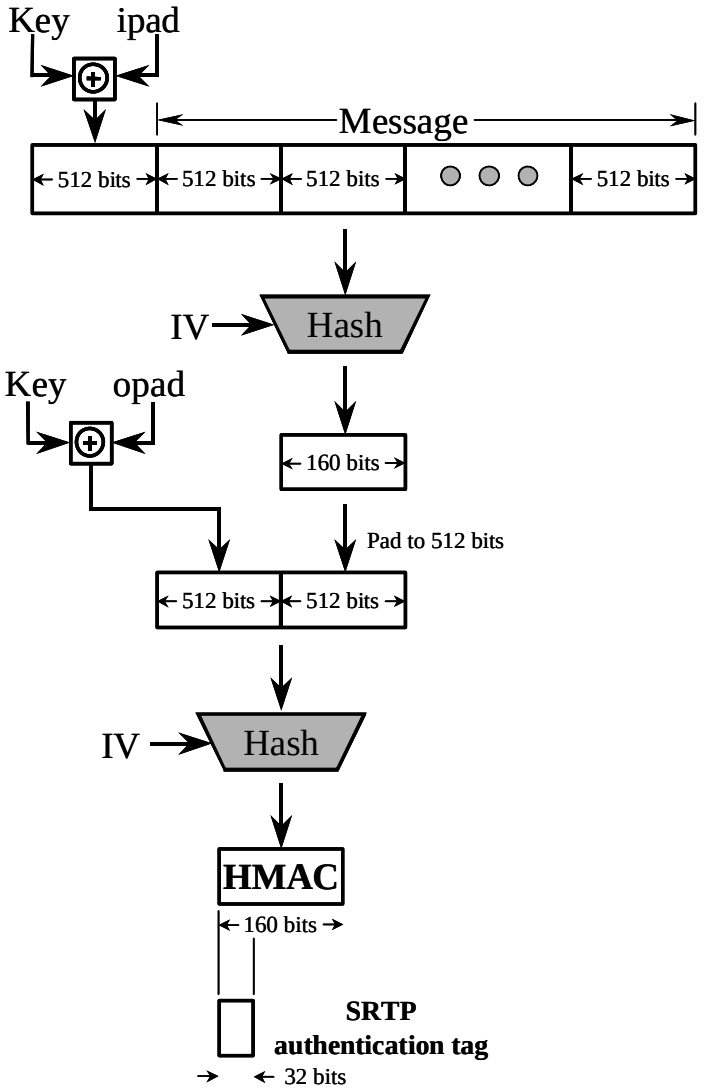

Figure 2 SRTP Authentication Tag Generation using HMAC-SHA1

An optimization is possible for small values of M. Each XOR operation in Figure 2 is followed by a hash operation. The result of the XOR and hash operations can be precomputed and used to replace the original initialization vector (IV) corresponding to that hash operation. These two values only need to be calculated once because the Key and ipad/opad values never change after the initial SRTP exchange of keys. Thus, two hashes can

---

efficiency if the following conditions are met: (1) the RTP payload is stateless, (2) an attacker is unlikely to be able to intelligently modify the SRTP ciphertext, (3) and no data forwarding or access control decisions are made based on the RTP data.

be avoided in the steady state, which reduces the computational overhead to hashing L+1 blocks. For G.711 speech coding at 8000Hz, each RTP packet payload contains 160 8-bit samples or 160 bytes. The RTP header is 12 bytes long, assuming no CSRC fields. Thus, each RTP packet contains 172 bytes, which means L=ceil(172/64)=3 and N=L+3=6. With the optimization described above, N=L+1=4.

The authentication for each SRTP packet requires 6 SHA1 hash operations, which equates to approximately 740*6=4560 logical operations, which takes 76µs on a 60MHz processor, not counting control flow instructions. An attacker can take advantage of this computational overhead by bombarding a victim with forged packets that invoke the authentication process with the sole intent to consume processor cycles. Section 2 describes three schemes, collectively called SRTP+, that significantly reduce the maximum processor cycles consumed by forged packets. Each scheme utilizes a message authentication scheme that does not require authentication of the entire header and payload. This is a departure from SRTP, which performs both message authentication and integrity checking with a single message authentication tag. In contrast, SRTP+ focuses only on authentication, with the intent of identifying forged packets quickly. Thus, in the presence of many forged packets, SRTP+ performs well. However, if all packets are legitimate, a minimal, but constant overhead is imposed. Section 0 presents the expected performance of SRTP+.

2 Design and Implementation

This section describes the three proposed schemes for SRTP+. The schemes differ in the manner in which they generate and verify an authentication tag for each RTP packet. However, all three schemes append a tag to each packet. Figure 3 displays the fields of an RTP header. The RTP specification allows header extensions if the Extension (labeled as "X") field is set to 1. The actual header extension is appended to the original header and consists of an extension length as well as the extension data itself. Use of this second location is convenient and simply takes advantage of the existing RTP specification. However, some transport mechanisms might compress the header to minimize bandwidth utilization, in which case header extensions might not be allowed. For that situation, the tag may be attached to the RTP payload immediately after the SRTP HMAC tag, which is shown in Figure 1.

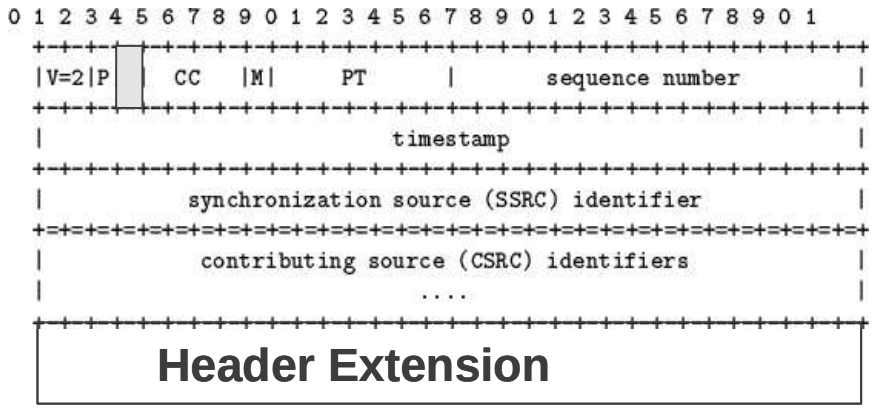

Figure 3 Proposed SRTP+ Authentication Tag Location

2.1 Secret pseudorandom number sequence (Scheme 1)

In this scheme the authentication tag is a number in a cryptographically secure pseudorandom number sequence. Each packet is assigned a pseudorandom number based on its sequence number, as shown in Table 1. For use in Avaya products, the pseudorandom number generator (PRNG) must be FIPS-140 [6][7] approved. One such approved PRNG [8] is based on a cryptographic hash, such as SHA1, as shown in Figure 4.

Table 1 Assignment of Pseudorandom Number Based on Sequence Number

Sequence #	1	2	…	1002	1003
Pseudorandom number	R1	R2	…	R1002	R1003

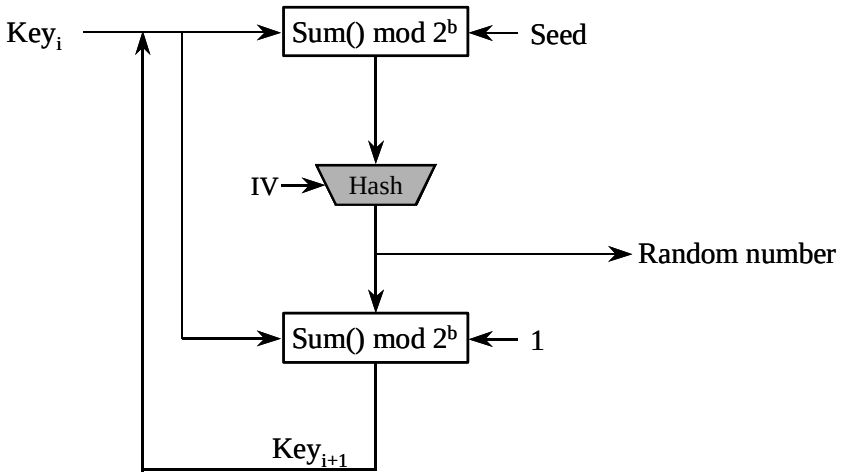

Figure 4 FIPS 140 Approved Cryptographically Secure Pseudorandom Number Generator

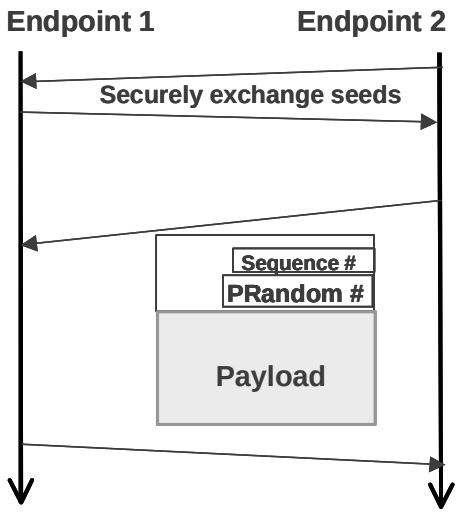

Figure 5 SRTP+ Communications for Pseudorandom Number Sequence Scheme (Scheme 1)

Both endpoints of the RTP communication must know the pseudorandom number corresponding to each sequence number, both for generating the authentication tag for the outgoing packet and also for verifying the tag of an incoming packet. Figure 5 shows the communications between two endpoints. Before any RTP packet is sent, the seed for the PRNG is exchanged in a secure manner. If SRTP is used, then the exchange of SRTP+ PRNG keys can be piggybacked on top of the secure exchange of keys for SRTP. After the keys are exchanged, a 32-bit SRTP+ authentication tag is attached to each RTP packet. This authentication tag can be transmitted in cleartext, since knowledge of the authentication tag for one packet cannot yield the tag for succeeding packets.

For SRTP, a 32-bit rollover counter effectively extends the width of the sequence number to 48 bits. Thus, the sequence numbers in Table 1 can be based on the full 48-bit sequence number. However, in practice no confusion should result even if only the actual 16-bit sequence numbers are considered. For example, if packets are sent at the rate of a packet every 20ms, then 1310 seconds elapse before the sequence numbers roll over and start to repeat.

If packets arrive out of order or are dropped, the sequence number of the packet will not be the expected number. In that case, the PRNG must iterate multiple times until the pseudorandom number for the correct sequence number is calculated. For example, if the last received sequence number was 1000 and the sequence number of the newly received packet is 1004, then the PRNG must iterate four times to calculate the pseudorandom number corresponding to sequence number 1004. If the packets have indeed arrived out of order, the pseudorandom

numbers for the sequence numbers 1001, 1002, and 1003 must be remembered because the PRNG can only calculate numbers going forward from the current number. Thus, a sliding window of calculated pseudorandom number must be maintained. When the authentication tag for an incoming packet matches the pseudorandom number for the lowest sequence number in the window, the sliding window can be shifted forward to discard the verified packet. The size of the sliding window should be based on the largest forward skip in sequence numbers that can be tolerated by the underlying media codec. One advantage of using a sliding window is that pseudorandom numbers for future packets can be calculated in advance, which allows the computationally intensive portion of the authentication process to be shifted away temporally from the time when a packet arrives.

2.2 Sequence number/timestamp hash (Scheme 2)

An alternative to the scheme in Section 2.1 avoids the need to maintain a sliding window to remember previously calculated pseudorandom numbers. Instead of using a PRNG, the sequence number and timestamp in the RTP header are hashed using a cryptographically secure hash, such as HMAC-SHA1 to obtain the authentication tag. Thus, each packet is independently verified. Out-of-order and dropped packets do not affect the authentication process. Figure 6 shows the between the two endpoints. This scheme is similar to SRTP, except that the message authentication code only authenticates part of the header instead of the entire header and payload. Thus, it should come as no surprise that such a performance savings is realized by applying the same secure hash to a smaller set of bits.

As with the scheme in Section 2.1, the SRTP+ keys can be exchanged along with the SRTP keys. In fact, the same SRTP keys can even be used as the SRTP+ keys. The SRTP+ authentication tag can be transmitted in cleartext.

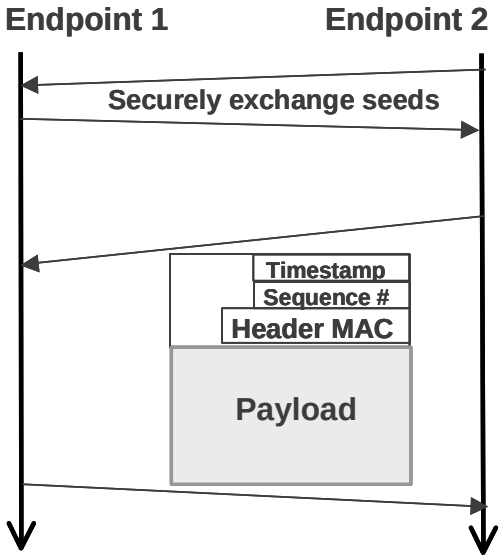

Figure 6 SRTP+ Communications for Sequence Number/Timestamp Hash (Scheme 2)

2.3 Secret random number chaining (Scheme 3)

Each of the schemes in Sections 2.1 and 2.2 depends on SHA1 to either generate a pseudorandom number or to calculate a message digest. Thus, the performance savings realized with either of these two schemes is limited by the need to perform a SHA1 hash operation on at least one block. A third scheme avoids the need for SHA1 altogether, but must be used in conjunction with SRTP encryption. In this scheme the sender calculates in advance a series of random numbers and uses a number as the authentication tag for each packet. The authentication tag is sent in cleartext, but the random numbers for the next N packets are encrypted (e.g., included in the SRTP encrypted payload). The receiver stores the N random numbers after decrypting the payload. These random numbers correspond to the sequence numbers for the next N expected packets and are compared to the authentication tags for succeeding packets for authentication. Before the first packet can be authenticated, the first N random numbers

must be sent to the receiver, possibly during the SRTP key exchange. Figure 7 shows the communications between the two endpoints.

The main advantage of this scheme is the extremely low overhead of verifying authentication, which requires a simple arithmetic comparison. However, additional bytes need to be encrypted at the sender and decrypted at the receiver.[4] Furthermore, these encrypted bytes increase the size of the RTP payload. A further disadvantage is the need to transmit the full set of N random numbers with each packet, which carries a great deal of redundancy. One possible mitigation is to minimize the width of the random numbers. Since the numbers can be truly random, as opposed to pseudorandom, the width of the numbers can be very small, since truly random numbers cannot be predicted based upon knowledge of previous numbers. Perhaps, the width can be as small as 4 bits for each number. Also, the value of N can be limited by the maximum forward skip in sequence numbers allowed by the media codec.

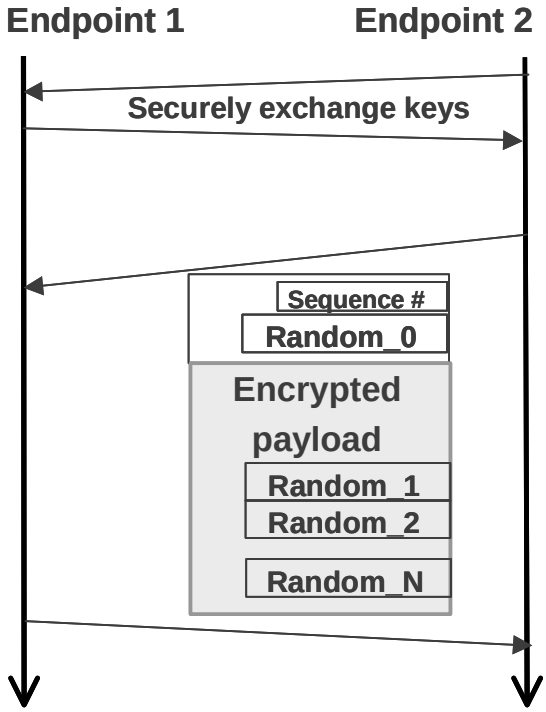

Figure 7 SRTP+ Communications for Secret Random Number Chaining Scheme (Scheme 3)

3 Performance Analysis

To gain an understanding of the anticipated performance improvements from each of the three SRTP+ schemes, we implemented a version of the schemes on several systems. The systems were (1) a 1.5 GHz Pentium IV system with 256 MB of RAM running Linux 2.4.7-10, (2) an 866 MHz 2-CPU Pentium III system with 512 MB of RAM running Linux 2.4.20-20.9, and (3) a 450 MHz 4-CPU Sparc system with 4 GB RAM. All test programs executed on a single CPU, and memory contention was not an issue. The open-source beecrypt-3.1.0 package was used for the SHA1 and HMAC-SHA1 implementations.

Table 2 shows the measured performance for each of the three SRTP+ schemes as well as the performance for SRTP. These numbers only show the performance for the portion of the run-time overhead imposed by the random number generation for Scheme 1, the HMAC-SHA1 code generation for Scheme 2, the AES encryption of one 128-bit block[5] for Scheme 3, the HMAC-SHA1 code generation for SRTP with a 172-byte RTP packet, which is common for VoIP applications. Although the actual platforms on which real VoIP systems are deployed may differ

[4] If the last block of the block cipher encryption involves a number of cleartext bits that is less than the block size, the unused bits can be used to hide the N random numbers. Unfortunately, the default 160-byte payload for G.711 encoding leaves no unused bits when used with 128-bit AES blocks.

[5] The number of 128-bit AES blocks depends on the size and number of random numbers needed for Scheme 3. The assumption for these experiments is that 32 4-bit random numbers are sufficient.

from these test systems, the relative performance of SRTP+ on most systems should be similar to the results presented in Table 2.

Table 2 Measured Performance for SRTP+ Schemes

	1.5 GHz Pentium IV			866 MHz Pentium III			450 MHz Sparc		
	μs	Steady-state overhead	Attack speedup	μs	Steady-state overhead	Attack speedup	μs	Steady-state overhead	Attack speedup
Scheme 1	2.04	26%	3.8	2.59	28%	3.5	5.95	26%	3.8
Scheme 2	5.28	68%	1.5	6.53	71%	1.4	16.31	72%	1.4
Scheme 3	0.47[6]	6%	16.5	0.80[7]	9%	11.5	2.83[8]	12%	8.0
SRTP	7.75			9.19			22.68		

Several observations can be made from Table 2. The most significant observation is that all three schemes offer a performance improvement relative to SRTP. However, the actual speedup is less than an order of magnitude, except for Scheme 3. The relative numbers are intuitive. Table 3 shows the time on each of our test systems to calculate the SHA1 digest for a 512-bit message block. Looking again at the numbers in Table 2, we see that Scheme 1 spends most of its time calculating a single SHA1 hash. For Scheme 2, L=ceil(4/64)=1, and N=L+1=2, which means that Scheme 2 must calculate the SHA1 digest for two 512-bit blocks and therefore requires a little more than twice the run-time compared to Scheme 1. For SRTP and a 172-byte RTP packet, L=ceil(172/64)=3, and N=L+1=4. As expected, the run-times for SRTP are approximately 4 times that for Scheme 1. Scheme 3 is much faster than the other schemes largely because the 128-bit AES block cipher encryption operates on a smaller input block compared to the 512-bit SHA1 hash operation.

In most cases, one of the SRTP+ schemes will be used along with SRTP. Thus, two types of overheads can be understood from Table 2. First, the steady-state overhead is incurred when all incoming packets are successfully authenticated. In that case, the total overhead consists of the SRTP overhead in addition to the SRTP+ overhead. For example, on the first experimental system, the steady-state overheads for Schemes 1, 2, and 3 are 26%, 68%, and 6%, respectively. The second type of overhead occurs when a faked packet is detected. A denial of service attack might involve a barrage of faked packets. In such a scenarios, the total overhead on the receiver side is just the SRTP+ overhead, since SRTP+ will detect the fake packet and avoid the need to incur the SRTP overhead. Again, for the first experimental system, the fake packet detection speedups for Schemes 1, 2, and 3 are 16.5, 11.5, and 8.0 respectively.

Table 3 Time to Apply SHA1 to One 512-bit Message Block

	Time (μs)
1.5 GHz Pentium IV	1.40
866 MHz Pentium III	1.64
450 MHz Sparc	3.96

It should be noted that the run-times for Scheme 1 in Table 2 were based on 160-bit keys to generate the random numbers. FIPS 186-2 [8] specifies that the key size must be 160 to 512 bits inclusive. Figure 8 shows the time required to generate one pseudorandom number for various key sizes. The key size does not affect the time, except at a key size of 448 bits, when the time suddenly doubles. This increase occurs at 448 bits because SHA1 requires a 64-bit message length to be appended to the message. Hence at 448 bits, an extra 512-bit block needs to be processed.

[6] 0.47μs is the time for a 128-bit key. For 192-bit and 256-bit keys, the times are 0.56μs and 0.66μs.
[7] 0.80μs is the time for a 128-bit key. For 192-bit and 256-bit keys, the times are 0.96μs and 1.10μs.
[8] 2.83μs is the time for a 128-bit key. For 192-bit and 256-bit keys, the times are 3.23μs and 3.62μs.

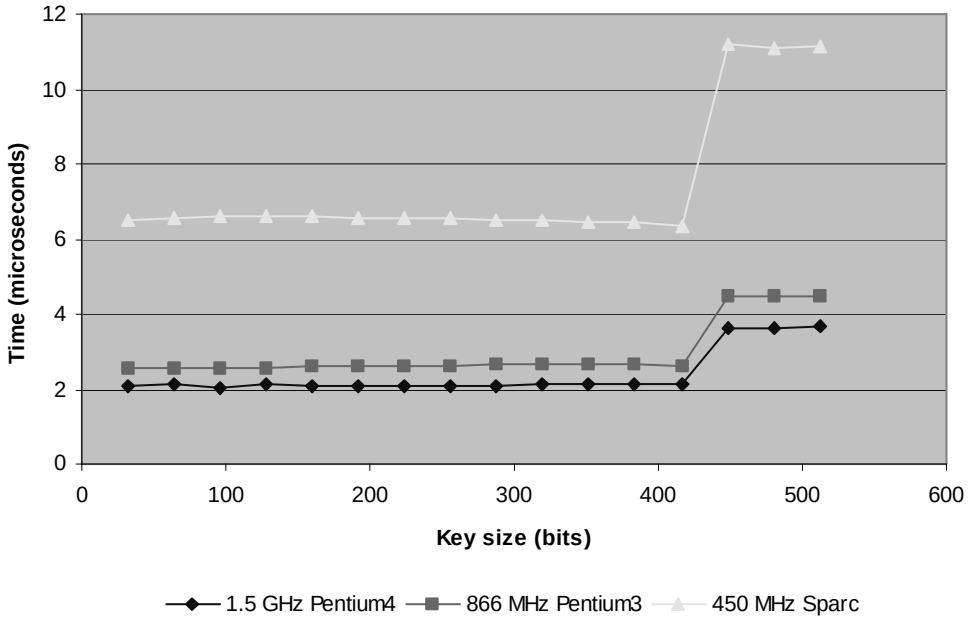

Figure 8 Time to Generate One Scheme 1 Pseudorandom Number Based on Key Size

4 Conclusion

The three SRTP+ schemes present techniques for tolerating a denial of service attack based on a barrage of faked RTP packets. Such a barrage of packets might demand sufficient computing resources to prevent the system from delivering the expected service. The SRTP+ schemes allow quick authentication of packets and thus decrease the potential load on the processor.

In steady-state operation, SRTP+ imposes an additional overhead. Fortunately, such an overhead is less onerous in steady-state operation, i.e., in the absence of a denial of service attack. On the other hand, the reduction of processor load for detecting fake packets is very important in a denial of service situation.

References

[1] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," IETF RFC 3550, July 2003, <http://www.ietf.org/rfc/rfc3550.txt?number=3550>.

[2] M. Baugher, D. McGrew, E. Carrara, M. Naslund, and K. Norrman, "The Secure Real-time Transport Protocol," IETF Internet Draft, <http://www.ietf.org/internet-drafts/draft-ietf-avt-srtp-09.txt>.

[3] National Institute for Standards and Technology (NIST), "Advanced Encryption Standard (AES)," FIPS Pub 197, <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>.

[4] H. Crawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," IETF RFC 2104, February 1997, <http://www.ietf.org/rfc/rfc2104.txt?number=2104>.

[5] R. Gilman, "An Efficient Encryption Algorithm for Audio Version 1.2," Avaya COMPAS document 93923, July 3, 2002.

[6] National Institute for Standards and Technology (NIST), "Security Requirements for Cryptographic Modules," FIPS Pub 140-2, <http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf>.

[7] National Institute for Standards and Technology (NIST), "Annex C: Approved Random Number Generators," FIPS Pub 140-2 Annex C, <http://csrc.nist.gov/publications/fips/fips140-2/fips1402annexc.pdf>.

[8] National Institute for Standards and Technology (NIST), "Digital Signature Standard (DSS)," FIPS Pub 186-2, <http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf>.