Internet Engineering Task Force
Audio/Video Transport Working Group
INTERNET-DRAFT
Expires in May, 2001

David A. McGrew
David Oran
Cisco Systems, Inc.
November, 2000

The Secure Real Time Protocol
<draft-mcgrew-avt-srtp-00.txt>

Status of this Memo

Table of Contents

1. Abstract

This document describes the Secure Real Time Protocol (SRTP), a profile of the Real Time Protocol (RTP) which provides privacy, message authentication, and replay protection.

SRTP achieves high throughput and low packet expansion by using an additive stream cipher for encryption, a universal hashing based function for message authentication, and an `implicit' index for sequencing based on the RTP sequence number.

2. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [B97].

3. Goals

The security goals for SRTP are to ensure:

* the privacy of the RTP payload, the RTP contributing source identifiers, and any RTP header extensions, if present,

* the authentication of the RTP header, payload, and header extensions, if present, and

* replay protection.

Source origin authentication (e.g., digitally signed packets) may be desirable in some situations, but is deferred from consideration in this document. See Section 10.3 for a discussion on this point.

Other goals for the protocol are:

* a low computational cost,

* a low footprint (i.e., small code size and data memory for key

schedules and replay lists),

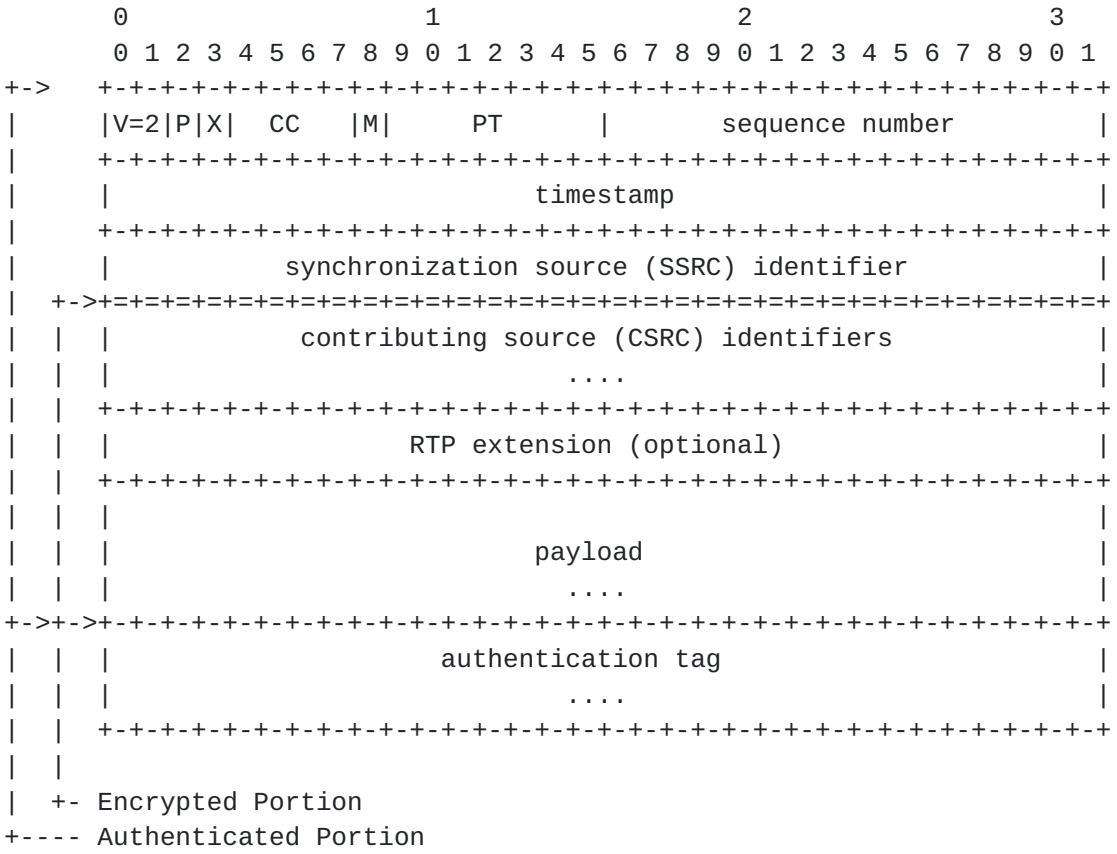
* limited packet expansion, and

* preservation of RTP header compression efficacy.

4. SRTP Overview

RTP is the Real Time Protocol [SCFJ96]. We define SRTP as a profile of RTP, in an analogous way to RFC1890 which defines the audio/video profile for RTP. Conceptually, we consider a `bump in the stack' implementation which resides between the RTP application and the transport layer, which intercepts RTP packets and then forwards an equivalent SRTP packet on the sending side, and which intercepts SRTP packets and passes an equivalent RTP packet up the stack on the receiving side.

Figure 1. The format of an SRTP packet.



The format of an SRTP packet is illustrated in Figure 1. The

authentication tag is the only field defined by SRTP that is not in RTP. It provides data origin authentication of the header and payload, and it indirectly provides replay protection by authenticating the sequence number. The Encrypted Portion of an SRTP packet consists of the contributing source identifiers, the RTP extension (if present), and the RTP payload, of the equivalent RTP packet. The Authenticated Portion of an SRTP packet consists of the entire equivalent RTP packet.

4.1 SRTP Cryptographic Contexts

Each SRTP session requires the sender and receiver to maintain cryptographic state information. This information is called the cryptographic context, and it consists of:

* an encryption key k_e,

* a message authentication key k_a,

* a 32-bit rollover counter r (which records how many times the 16-bit RTP sequence number has been reset to zero after passing through 65,535),

* a sequence number s_l (which is the last received and authenticated sequence number for the receiver, and is the last sequence number sent for the sender), and

* a replay list L (maintained by the receiver only).

4.2 Mapping SRTP Packets to Cryptographic Contexts

The RTP synchronization source (SSRC) identifier is used by a receiver to identify the proper cryptographic context for each packet.

An SSRC identifier is unique for a given session, and all packets with the same SSRC form part of the same timing and sequence number space. Thus, the SSRC field can be used by an SRTP receiver (or by a bump in the stack implementation on the sender's side) to identify the proper cryptographic context.

The authentication key and encryption key of each context MUST remain fixed for the duration of that context. This ensures that incorrect keys will not be used by the receiver due to a synchronization error.

4.3 SRTP Packet Processing

To construct a proper SRTP packet, given an RTP packet, the sender does the following:

1. Determine which cryptographic context to use by checking the SSRC field of the RTP packet.

2. Determine the index of the SRTP packet from the rollover counter in the cryptographic context and the sequence number from the RTP packet, as described in Section 5.

3. Encrypt the Encrypted Portion of the packet, as described in Section 7, using the index determined in Step 2 and the encryption key in the context found in Step 1.

4. Compute the authentication tag for the Authenticated Portion of the packet, as described in Section 8, using the index determined in Step 2 and the authentication key in the context found in Step 1. Note that the Encrypted Portion is encrypted before the authentication tag is computed.

To authenticate and decrypt an SRTP packet, the receiver does the following:

1. Determine which cryptographic context to use by checking the SSRC field of the RTP packet.

2. Determine the index of the SRTP packet from the rollover counter in the cryptographic context and the sequence number from the RTP packet.

3. Check the Replay List to ensure that no packet with that index has been received and authenticated before, as described in Section 6. If that index is in the list, then the packet has been replayed and is invalid. It MUST be discarded, and the event SHOULD be logged.

4. Compute the authentication tag for the Authenticated Portion of the packet, as described in Section 8, using the index determined in Step 2 and the authentication key in the context found in Step 1. Note that the Encrypted Portion is not decrypted before the authentication tag is computed.

   If the authentication tag that is computed matches that in the SRTP packet, then the packet is accepted and the index is added to the Replay List. Otherwise, the packet is invalid: it MUST be discarded, and the event SHOULD be logged.

5. Decrypt the Encrypted Portion of the packet, as described in Section 7, using the index determined in Step 2 and the encryption key in the context found in Step 1.

The processing has been chosen to maximize resistance to denial of service attacks (i.e., to minimize the receiver's effort in processing spurious packets).

4.4 Cryptographic Algorithms

Default encryption and authentication algorithms are specified in Sections 7.1 and 8.1. While there are numerous encryption and message authentication algorithms that can be used in SRTP, we define default algorithms in order to avoid the complexity of specifying the encodings for the signaling of algorithm and parameter identifiers.

5. Synchronization

SRTP implementations use an `implicit' packet index for sequencing. Receiver-side implementations use the RTP sequence number to reconstruct the correct index (that is, location in the sequence of all RTP packets). The index is defined as s + r * 65,536, where the sequence number is s and the rollover counter is r.

A robust approach for the proper use of a rollover counter requires that its handling and use be well defined. In particular, out-of-order RTP packets with sequence numbers close to 65,536 or zero must be properly dealt with.

A receiver reconstructs the index i of a packet with sequence number s using the estimate

i = 65,536 * t + s,

where t is chosen from the set { r-1, r, r+1 } such that i is closest to the value 65,536 * r + s_l. If the value r+1 is used, then the rollover counter r in the cryptographic context is incremented by one.

The pseudocode for the algorithm to process a packet with sequence number s follows:

```
if (s_l < 32,768)
   if (s - s_l > 32,768)
      set i to s + 65,536 * (r-1)
   else
```

```
      set i to s + 65,536 * r
   endif
else
   if (s_l - 32,768 > s)
      set r to r + 1
   endif
   set i to s + r * 65,536
endif
set s_l to s
```

The index i is used in replay protection (Section 6), in encryption (Section 7), and in message authentication (Section 8).

As the rollover counter is 32 bits long, the maximum number of packets in any given SRTP session is 2^48 = 281,474,976,710,656. After that number of SRTP packets have been sent, the sender MUST not send any more packets with that cryptographic context. This limitation enforces a security benefit by providing an upper bound on the amount of traffic that can pass before cryptographic keys are changed.

Other approaches to sequencing were considered and rejected; please see Section 10.1 for our rationale.

6. Replay Protection

A packet is `replayed' when it is stored by an adversary, then re-injected onto the network. SRTP provides protection against such attacks by requiring the storage of the indices of the most recently received and authenticated packets.

Each SRTP receiver maintains a Replay List, which contains the indices of the packets which have been received and authenticated which are no less than s_l * 65,536 - SRTP_WINDOW_SIZE, where SRTP_WINDOW_SIZE is a parameter that MUST be at least 64, and which MAY be set to a higher value. In this `sliding window' approach, a fixed amount of storage is used for replay protection.

The Replay List can be efficiently implemented by using a bitmap to represent which packets have been received, as described in the Security Architecture for IP [KA98a].

7. Encryption

Encryption uses a `seekable' additive stream cipher, following the Stream Cipher ESP [sc-esp]. The stream ciphers that can be used

must be able to efficiently seek to arbitrary locations in their keystream. Ciphers that can do this include SEAL [RC94, RC98], LEVIATHAN [MF00b], and any block cipher run in counter mode [LRW00, M00]. In particular, AES in counter mode will provide good security, reasonable performance, and conform to emerging U.S. Federal standards, and is thus defined as the default cipher.

SRTP encryption consists of generating a keystream segment corresponding to the index of the packet, and then bitwise exclusive-oring that keystream segment into the RTP packet, starting at bit number 96 (the first bit in the first contributing source identifier, if present). Decryption is the done the same way, but swapping the roles of the plaintext and ciphertext. The definition of how keystream is generated, given the index, depends on the cipher.

7.1 Default Cipher: Counter Mode AES

AES will be used with a 128 bit key size and a 128 bit block size, using the Segmented Integer Counter Mode [M00].

In Counter Mode AES, keystream for the packet with index i is defined as the concatenation of the outputs of the AES cipher with the inputs

i*4096, i*4096 + 1, i*4096 + 2, ... , (i+1)*4096 - 1.

The AES has a block size of 128 bits, so 4096 output blocks are sufficient to generate the 8 * 64,536 = 524,288 bits of keystream needed to encrypt the largest possible RTP packet (actually, any IPv4 or IPv6 packet except for IPv6 `jumbograms' [rfc2675], which are not likely to be used for RTP-based multimedia traffic).

8. Message Authentication

Message integrity authentication can be provided by any message authentication code, though the default value is UMAC [KBHHKR00].

The authentication tag is computed by applying the UMAC function to the Authenticated Portion of the SRTP packet.

8.1 Default MAC: UMAC

The default message authentication code is UMAC [KBHHKR00], which has proven security properties and is quite fast. Furthermore, it

can be used with short (e.g., two or four byte) authentication tags, as well as larger tags.

The authentication tag is appended to the RTP packet. This expansion of the RTP packet may cause the packet size to exceed the Maximum Transmission Unit (MTU) of a network interface on its path, especially in circumstances when the application is attempting to `optimize' the size of packets. MTU path discovery SHOULD be used to avoid this problem.

UMAC is a parameterized algorithm (see Section 2.1 of [KBHHKR00]). The default selection of UMAC parameters for SRTP are:

WORD-LEN	2
UMAC-OUTPUT-LEN	4
L1-KEY-LEN	128
UMAC-KEY-LEN	16
ENDIAN-FAVORITE	BIG
L1-OPERATIONS-SIGN	SIGNED

This choice of parameters is intended to work well on low-power processors, to minimize packet expansion, and to minimize the size of the cryptographic context. The WORD-LEN of two will work well on 16 bit and higher processors. The packet expansion is determined by the UMAC-OUTPUT-LEN to be only four bytes. The storage requirement, per cryptographic context, is 144 bytes. These parameters ensure a forgery probability of no greater than 1/2^30 for each individual packet. Please see the security considerations section in [KBHHKR00] and the references therein for a more detailed discussion.

9. SRTP Parameters

The SRTP_WINDOW_SIZE (Section 6) is the only currently defined parameter. Other parameters may be added in the future.

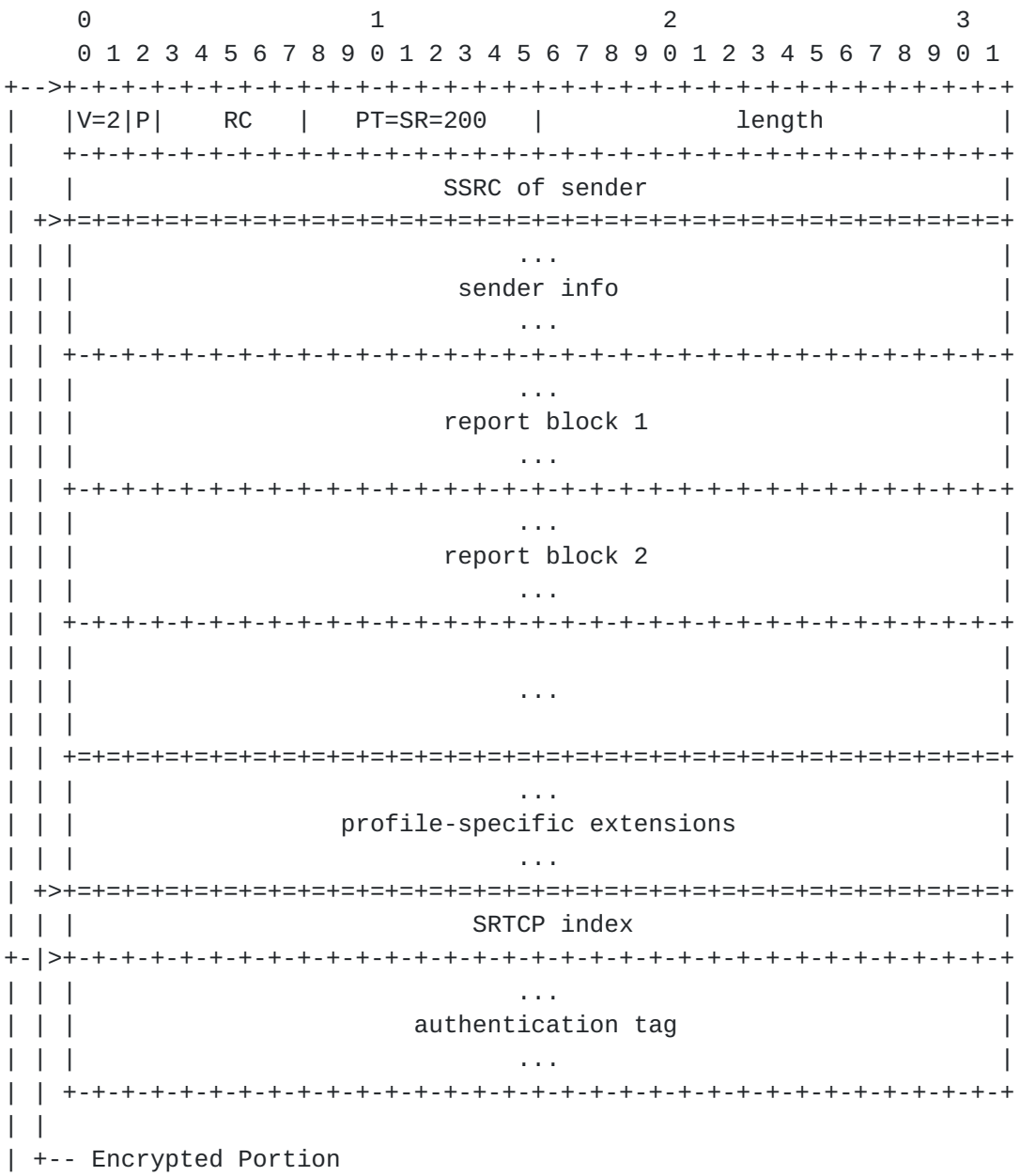
10. Secure RTCP

Secure RTCP follows the definition of Secure RTP, but defines the index differently. In order to differentiate this quantity, we refer to it as the SRTP index.

Each sender must use a distinct cryptographic context, as there is no way to synchronize sequencing information among senders. Therefore, each SSRC corresponds to a distinct SRTCP cryptographic context (and to a distinct SRTP context as well).

SRTCP is defined as a profile of RTCP, and it adds two new fields to the RTCP packet definition, the SRTCP index and the authentication tag. Those fields are appended to an RTCP packet in order to form an equivalent SRTP packet, so that they follow any other profile-specific extensions. An SRTCP packet is illustrated in Figure 2.

Figure 2. The format of a Secure RTCP packet, after Section 6.3.1 of [SCFJ96]. In this case, the underlying RTCP packet is a sender report packet; the SRTP format is identical for other RTCP packet types.



+ ---- Authenticated Portion

The SRTCP index is a 32-bit value, which is set to zero before the first SRTCP packet is sent, and is incremented by one after each SRTCP is sent. This index is explicitly included in each packet, in contrast to the `implicit' index approach used for SRTP.

SRTCP packet processing is identical to that of SRTP packet processing, with the following changes:

* SRTCP replay protection is as defined in Section 6, but using the the SRTCP index as the index i.

* SRTCP encryption is as defined in Section 7, but using the definition of the SRTCP Encrypted Portion as defined in this section, and using the SRTCP index as the index i.

* The SRTCP authentication tag is defined as in Section 8, but applying the UMAC function to the Authenticated Portion of the SRTCP packet as defined in this section, and using the SRTCP index as the index i.

The encryption prefix (Section 6.1 of [SCFJ96]), which is a random 32-bit quantity intended to improve privacy, SHOULD NOT be used. This is because SRTP encryption uses an additive stream cipher, and thus the prefix offers no benefit.

The maximum number of SRTCP packets is limited to 2^32 = 4,294,967,296. The last RTCP packet MUST contain an RTCP BYE. SRTCP senders MUST send an RTCP BYE in the final packet, if the maximum number of SRTCP packets is reached. Similarly, SRTCP receivers MUST act as though the last RTCP packet included a BYE, even if no BYE was included in the packet, if the maximum number of SRTCP packets is reached.

11. Rationale

SRTP achieves high throughput and low packet expansion by using fast stream ciphers for encryption, universal hash functions for message authentication, and an implicit index for synchronization.

Only a single header extension may be appended to the RTP data header, so the use of a header extension for SRTP was avoided. SRTP and SRTCP are defined as profiles of RTP and RTCP, respectively.

10.1 Synchronization

RTP runs over unreliable transport. Thus, maintaining synchronization of the cryptographic context between the sender and receiver is a conspicuous challenge. Because of the requirement to minimize packet expansion, no explicit sequencing information should be added. RTP packets contain two fields for synchronization purposes, the timestamp and the sequence number. The timestamp field could be used for cryptographic synchronization in some circumstances. However, this field is not appropriate for such use; from [SCFJ96]:

> Several consecutive RTP packets may have equal timestamps if they are (logically) generated at once, e.g., belong to the same video frame. Consecutive RTP packets may contain timestamps that are not monotonic if the data is not transmitted in the order it was sampled, as in the case of MPEG interpolated video frames.

The RTP sequence number cannot be directly used as a unique identifier for SRTP packets. It has only sixteen bits, which would limit the duration of an SRTP security association to only 64,536 packets.

The `implicit index' approach works as long as the reorder and loss of the packets is not too great. In particular, 32,768 packets would need to be lost, or a packet would need to be 32,768 packets out of sequence in order for synchronization to be lost. Such drastic loss or reorder is likely to disrupt the RTP application itself.

10.2 Replay Protection

Replay protection is undoubtedly important for multimedia data. Otherwise, it would be possible for an adversary to perform simple manipulations on data that subverted security. For example, in a voice application, the phrase ``yes'' could be substituted for ``no'' if replay protection were not present.

10.3 Source Origin Authentication

`Source origin authentication' was listed as an option in the security goals, not because it not an appropriate goal, but because it may not be achievable. This goal may be desirable in some circumstances, such as multicast environments in which the sender is more trusted than the receivers, or when translators or mixers (Section 2.3 of [SCFJ96]) are used. However, it is not clear that this capability can always be provided, as mixers and translators can change the payload. Furthermore, this security

service essentially requires digital signatures (at least if collusion resistance is required [BF00]).

Two examples of the multicast scenario mentioned above are a military commander addressing his troops over RTP, and financial market data sent over RTP. In these situations, a `stream signing' method can provide digital signatures on the entire RTP packets. An extensive literature on such methods is developing, and it is reasonable to expect that one of these methods can be reduced to practice and specified for RTP. This suggests that it should be left as an option in the current specification. A future effort can define a stream signing method as an authentication type for RTP, which could be used as a replacement for a message integrity transform.

Examples of the mixer and translator scenarios include a translator re-encoding data at a lower rate or in a different encoding, and a mixer combining the audio streams of multiple speakers in a teleconference. In these cases, it is not clear that meaningful source origin authentication is possible, as the data that is received is not the same as the data that is signed. If the translator is trusted by the receivers, then it could sign or re-sign the data streams, but this scenario may not be prevalent. It may be possible to devise a signing scheme that authenticates the source but not the content (enabling the receivers to know that ``John is one of the people talking'', but not providing authentication on who said what) by signing the concatenation of the Contributing source (CSRC) field and some sequencing information (e.g., a timestamp or sequence number), but such schemes require synchronization between the senders. This synchronization is not required by the RTP protocol itself, and may be difficult or impossible to arrange.

11. Security Considerations

The security of UMAC is well understood, and is described in [KBHHKR00].

Additive ciphers do not provide any security service other than privacy. In particular, they do not provide message authentication (see [RK99] or [S96] for a discussion of this security service). However, SRTP uses a message authentication code to provide that security service.

By using `seekable' stream ciphers, SRTP avoids the denial of service attacks that are possible on stream ciphers that lack this property (these attacks are described in Section 3.4 of [B96]).

No bit of keystream in an additive stream cipher should ever be used to encrypt multiple distinct plaintext bits. Such keystream reuse (jokingly called a `two-time pad' system by cryptographers), can seriously compromise security. The NSA's VENONA project [C99] provides a historical example of such a compromise. In SRTP, a `two-time pad' is avoided by requiring that both keys and indices be unique.

If manual keying is used, two different cryptographic contexts might accidentally use the same encryption key with non-negligible probability, through manual error or procedural inadequacies. Thus, manual keying SHOULD NOT be used for SRTP (or SRTCP).

An additive stream cipher is vulnerable to attacks that use statistical knowledge about the plaintext source to enable key collision and time-memory tradeoff attacks [MF00,H80,Bi96]. These attacks take advantage of commonalities among plaintexts, and provide a way for a cryptanalyst to amortize the computational effort of decryption over many keys, thus reducing the effective key size of the cipher. A detailed analysis of these attacks and their applicability to the encryption of Internet traffic is provided in [MF00]. In summary, the effective key size of SRTP when used in a security system in which m distinct keys are used, is equal to the key size of the cipher less the logarithm (base two) of m. Protection against such attacks can be provided simply by increasing the size of the keys used.

In order to provide an effective key size of n bits in a deployment in which 2^m SRTP/SRTCP cryptographic contexts will be created, the true key size will need to be n+m bits. The value of m SHOULD be 32 bits for networks with 50,000 connections (fully meshed networks with up to 200 devices), and SHOULD be 64 bits for networks with 49e+12 connections (fully meshed networks with up to 7,000,000 devices). These choices of m ensures that key collision attacks amortized over a ten year period offer no advantage over exhaustive search, when new SC/ESP keys are established for every connection every hour (note that such an attack requires the storage of all network traffic over the ten year period). These choices will suffice for many networks, though SRTP deployments with more stringent security requirements will need to make a detailed assessment of those requirements with respect to the attacks described in [MF00].

Implementations SHOULD use keys that are as large as possible. Please note that in many cases increasing the key size of a cipher does not affect the throughput of that cipher.

It is an important point that the m bits of `extra' key provided to

thwart these attacks need not be private. In jurisdictions with mandated limits on the length of a secret key, the additional key bits could be made public. This is because those bits are functionally equivalent to the `salt' that is used to protect passwords from dictionary attacks. The fact that the `extra' key bits are distinct for many different keys defeats the key collision and time-memory tradeoff attacks by reducing the number of keys over which cryptanalytic computation can be amortized.

Note that other security protocols which use additive ciphers for the encryption of Internet traffic (e.g., SSL, TLS, SSH, IPSEC) are also vulnerable to the attacks described in [MF00]. Those attacks are generic to additive encryption of redundant plaintext, and are not particular to SRTP.

12. Contact Information

Questions and comments about this memo can be directed to:

David A. McGrew
David Oran
Cisco Systems, Inc.
San Jose, CA 95134-1706 USA
mcgrew@cisco.com, oran@cisco.com

13. References

[B97] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

[KA98a] Kent, S., and R. Atkinson, "Security Architecture for IP", RFC 2401, November 1998.

[BF00] Boneh, D., and Franklin, M., "Message Authentication in a Multicast Environment", the Proceedings of the Seventh Annual Workshop on Selected Areas in Cryptography (SAC 2000), Springer-Verlag.

[C99] Crowell, W. P., "Introduction to the VENONA Project", http://www.nsa.gov:8080/docs/venona/index.html.

[H80] Hellman, M. E., "A cryptanalytic time-memory trade-off", IEEE Transactions on Information Theory, July 1980, pp. 401-406.

[KBHHKR00] Krovetz, T., Black, J., Halevi, S., Hevia, A., Krawczyk, H., Rogaway, P., "UMAC: Message Authentication Code using Universal Hashing", Internet Draft, October 2000,

<draft-krovetz-umac-01.txt>.

[LRW00] Lipmaa, H., Rogaway, P., and Wagner, D., "Comments to NIST Concerning AES Modes of Operation: CTR-Mode Encryption", NIST Workshop on AES Modes of Operation, http://csrc.nist.gov/encryption/aes/modes/lipmaa-ctr.pdf

[M00] McGrew, D., "Segmented Integer Counter Mode: Specification and Rationale", NIST Workshop on AES Modes of Operation, http://www.mindspring.com/~dmcgrew/sic-mode.pdf

[MF00] McGrew, D., and Fluhrer, S., "Attacks on Encryption of Redundant Plaintext and Implications on Internet Security", the Proceedings of the Seventh Annual Workshop on Selected Areas in Cryptography (SAC 2000), Springer-Verlag.

[MF00b] McGrew, D., and Fluhrer, S., "The Stream Cipher LEVIATHAN: Specification and Supporting Documentation", Submission to the New European Schemes for Signatures, Integrity, and Encryption (NESSIE) Process, October, 2000. http://www.cryptonessie.org/.

[R92] Rueppel, R., "Stream Ciphers", Chapter 2 of Simmons, G., "Contemporary Cryptology: the Science of Information Integrity," 1992, IEEE Press.

[RC94] Rogaway, P. and Coppersmith, D., "A Software-Optimized Encryption Algorithm", Proceedings of the 1994 Fast Software Encryption Workshop, Lecture Notes In Computer Science, Volume 809, Springer-Verlag, 1994, pp. 56-63.

[RC98] Rogaway, P. and Coppersmith, D., "A Software-Optimized Encryption Algorithm", Journal of Cryptology, Volume 11, Number 4, Springer-Verlag, 1998, Pages 273-287. Also available on the Internet at http://www.cs.ucdavis.edu/~rogaway/papers/seal-abstract.html.

[RK99] Rescorla, E., and Korver, B., "Guidelines for Writing RFC Text on Security Considerations," draft-rescorla-sec-cons-00.txt

[S96] Schneier, B. "Applied Cryptography: Protocols, Algorithms, and Source Code in C", Wiley, 1996.

[SCFJ96] Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V., "RTP: A Transport Protocol for Real-Time Applications", IETF Request For Comments RFC 1889.