# Chapter 13. Security Considerations

* Privacy
* Confidentiality
* Authentication
* Replay Protection
* Denial of Service
* Mixers and Translators
* Active Content
* Other Considerations

Until now we have considered an RTP implementation only in isolation. In the real world, however, there are those who seek to eavesdrop on sessions, impersonate legitimate users, and "hack into" systems for malicious purposes. A correctly implemented RTP system is secure against such attackers and can provide privacy, confidentiality, and authentication services to its users. This chapter describes the features of RTP that enhance user privacy, and outlines various potential attacks and how they may be prevented.

## Privacy

The obvious privacy issue is how to prevent unauthorized third parties from eavesdropping on an RTP session. This issue is discussed in the next section, HTUConfidentialityUTH. Confidentiality is not the only privacy issue, though; users of an RTP implementation may want to limit the amount of personal information they give out during the session, or they may want to keep the identities of their communication partners secret.

Information regarding a source, possibly including personal details, is communicated in RTCP source description packets, as noted in HTUChapter 5UTH, RTP Control Protocol. This information has several uses, most of which are legitimate, but some may be regarded as inappropriate by some users. An example of legitimate use might be a teleconferencing application that uses source description packets to convey the names and affiliations of participants, or to provide other caller identification information. Inappropriate use might include providing personal details via RTCP while the user is listening to an online radio station, hence allowing the station and its advertisers to track their audience. Because of these concerns, it is recommended that applications not send source description packets without first informing the user that the information is being made available.

The exception is the canonical name (CNAME), which is mandatory to send. The canonical name includes the IP address of the participant, but this should not

represent a privacy issue, because the same information is available from the IP header of the packet (unless the packets pass through an RTCP-unaware Network Address Translation (NAT) device, in which case CNAME packets will expose the internal address, which some sites might want to hide). The canonical name also exposes the user name of the participant, which may be a greater privacy issue. Applications may omit or rewrite the user name to obscure this information, provided that this is done consistently by any set of applications that will associate sessions via the CNAME.

The CNAME provides a mostly unique identifier, which could be used to track participants. This issue is avoided if the participant is joining from a machine with a temporary IP address (for example, a dial-up user whose IP address is assigned dynamically). If the system has a static IP address, little can be done to protect that address from being tracked, but the CNAME provides little more information than can be obtained from the IP headers (especially if the user name portion of the CNAME is obscured).

Some receivers may not want their presence to be visible at all. It may be acceptable for those receivers not to send RTCP, although doing so prevents the sender from using the reception quality information to adapt its transmission to match the receiver (and it may cause a source to assume that the receiver has failed, and stop transmission). Furthermore, some content providers may require RTCP reports, to gauge the size of their audience.

A related privacy concern involves being able to keep the identities of the partners in a conversation secret. Even if the confidentiality measures described in the next section are used, it is possible for an eavesdropper to observe the RTP packets in transit and gain some knowledge of the communication by looking at the IP headers (for example, your boss might be interested in knowing that your voice-over-IP call is destined for an IP address owned by a recruitment agency). This problem cannot easily be solved with IP, but it can be mitigated if traffic is routed through a trusted third-party gateway that acts as an intermediary with a neutral IP address. (Traffic analysis of the gateway might still reveal communication patterns, unless the gateway is both busy and designed to withstand such analysis.)

## Confidentiality

One of the key security requirements for RTP is confidentiality, ensuring that only the intended receivers can decode your RTP packets. RTP content is kept confidential by encryption, either at the application level—encrypting either the entire RTP packet or just the payload section—or at the IP layer.

Application-level encryption has advantages for RTP, but also some disadvantages. The key advantage is that it allows header compression. If only the RTP payload is

encrypted, then header compression will work normally, which is essential for some applications (for example, wireless telephony using RTP). If the RTP header is encrypted too, the operation of header compression is disrupted to some extent, but it is still possible to compress the UDP/IP headers.

The other advantage of application-level encryption is that it is simple to implement and deploy, requiring no changes to host operating systems or routers.

Unfortunately, this is also a potential disadvantage because it spreads the burden of correct implementation to all applications. Encryption code is nontrivial, and care must be taken to ensure that security is not compromised through poor design or flaws in implementation.

It's worth saying again: Encryption code is nontrivial, and care must be taken to ensure that security is not compromised through poor design or flaws in implementation. I strongly recommend that you study the relevant standards in detail before building a system using encryption, and make sure that you use well-known, publicly analyzed, cryptographic techniques.

Another potential disadvantage of application-level encryption is that it leaves some header fields unencrypted. In some cases, the lack of encryption might reveal sensitive information. For example, knowledge of the payload type field may allow an attacker to ascertain the values of parts of the encrypted payload data, perhaps because each frame starts with a payload header with a standard format. This should not be a problem, provided that an appropriate encryption algorithm is chosen, but it has the potential to compromise an already weak solution.

As an alternative, encryption can be performed at the IP layer—for example, using the IP security (IPsec) protocols. This approach has the advantage of being transparent to RTP, and of providing a single—presumably well-tested—suite of encryption code that can be trusted to be correct. The disadvantages of IP-layer encryption are that it disrupts the operation of RTP header compression and its deployment requires extensive changes to host operating systems.

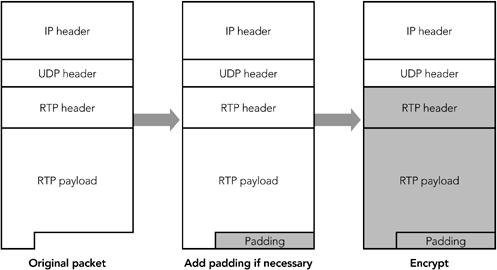
##### Confidentiality Features in the RTP Specification

The RTP specification provides support for encryption of both RTP data packets (including headers) and RTCP packets.

All octets of RTP data packets—including the RTP header and the payload data—are encrypted. Implementations have a choice of the encryption schemes they support. Depending on the encryption algorithm used, it may be necessary to append padding octets to the payload before encryption can be performed. For example, DES encryptionHTPU56UTPH operates on blocks of 64 bits, so payloads will need to be padded if they are not multiples of eight octets in length. HTUFigure 13.1UTH illustrates the process.

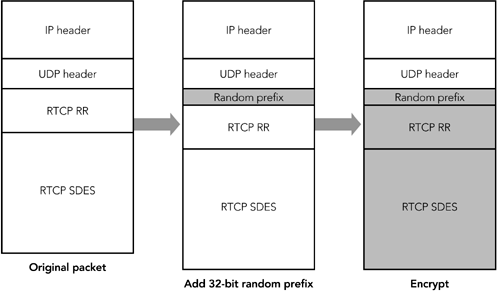
When padding is used, the P bit in the RTP header is set, and the last octet of the padding indicates the number of padding octets that have been appended to the payload.

###### Figure 13.1. Standard RTP Encryption of a Data Packet



When RTCP packets are encrypted, a 32-bit random number is inserted before the first packet, as shown in HTUFigure 13.2UTH. This is done to prevent known plain-text attacks. RTCP packets have a standard format with many fixed octets; knowledge that these fixed octets exist makes a wily cracker's work easier because he knows part of what he is looking for in a decrypted packet. The cracker could employ a brute-force key guessing, using the fixed octet values in the decryption attempt to determine when to stop.

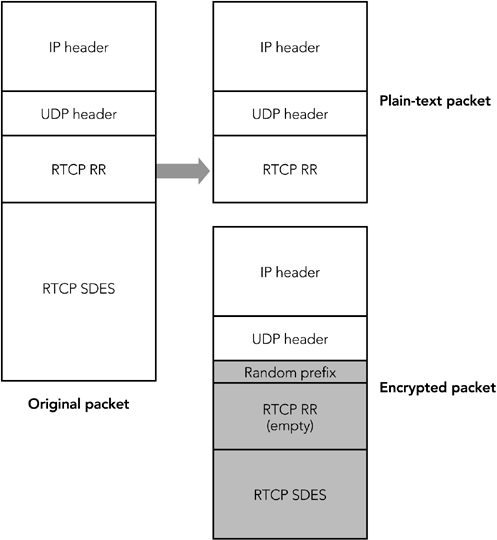
###### Figure 13.2. Standard RTP Encryption of a Control Packet



The insertion of the prefix provides initialization for the cipher, which effectively prevents known plain-text attacks. No prefix is used with data packets because there are fewer fixed header fields: The synchronization source is randomly chosen, and the sequence number and timestamp fields have random offsets.

In some cases it is desirable to encrypt only part of the RTCP packets while sending other parts in the clear. The typical example would be to encrypt the SDES items, but leave reception quality reports unencrypted. We can do this by splitting a compound RTCP packet into two separate compound packets. The first includes the SR/RR packets; the second includes an empty RR packet (to satisfy the rule that all compound RTCP packets start with an SR or RR) and the SDES items. (For a review of RTCP packet formats, see HTUChapter 5UTH, RTP Control Protocol.) HTUFigure 13.3UTH illustrates the process.

###### Figure 13.3. Using Standard RTP Encryption to Partially Encrypt a Control Packet



There is an exception to the rule that all compound RTCP packets must contain both an SR/RR packet and an SDES packet: When a compound packet is split into two separate compound packets for encryption, the SDES packet is included in only one compound packet, not both.

The default encryption algorithm is the Data Encryption Standard (DES) in cipher block chaining mode.HTPU56UTPH When RTP was designed, DES provided an appropriate level of security. However, advances in processing capacity have rendered it weak, so it is recommended that implementations choose a stronger encryption algorithm where possible. Suitable strong encryption algorithms include Triple DESHTPU57UTPH and the Advanced Encryption Standard (AES).HTPU58UTPH To maximize interoperability, all implementations that support encryption should support DES, despite its weakness.

The presence of encryption and the use of the correct key are confirmed by the receiver through header or payload validity checks, such as those described in the Packet Validation sections of HTUChapter 4UTH, RTP Data Transfer Protocol, and HTUChapter 5UTH, RTP Control Protocol.

The RTP specification does not define any mechanism for the exchange of encryption keys. Nevertheless, key exchange is an essential part of any system, and it must be performed during session initiation. Call setup protocols such as SIPHTPU UTPH and RTSPHTPU UTPH are expected to provide key exchange, in a form suitable for RTP.

28

14

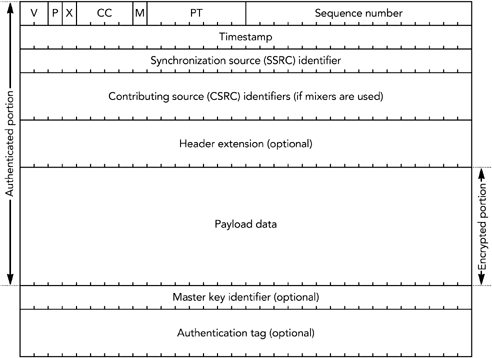
##### Confidentiality Using the Secure RTP Profile

An alternative to the mechanisms in the RTP specification is provided by the Secure RTP (SRTP) profile.HTPU UTPH This new profile, designed with the needs of wireless telephony in mind, provides confidentiality and authentication suitable for use with links that may have relatively high loss rate, and that require header compression for efficient operation. SRTP is a work in progress, with the details of the protocol still evolving at the time of this writing. After the specification is complete, readers should consult the final standard to ensure that the details described here are still accurate.

55

SRTP provides confidentiality of RTP data packets by encrypting just the payload section of the packet, as shown in HTUFigure 13.4UTH.

###### Figure 13.4. Secure RTP Encryption of a Data Packet



The RTP header, as well as any header extension, is sent without encryption. If the RTP payload format uses a payload header within the payload section of the RTP packet, that payload header will be encrypted along with the payload data. The authentication header is described in the section titled HTUAuthenticationUTH Using the Secure RTP Profile later in this chapter. The optional master key identifier may be used by the key management protocol, for the purpose of rekeying and identifying a particular master key within the cryptographic context.

When using SRTP, the sender and receiver are required to maintain a cryptographic context, comprising the encryption algorithm, the master and salting keys, a 32-bit rollover counter (which records how many times the 16-bit RTP sequence number has wrapped around), and the session key derivation rate. The receiver is also expected to maintain a record of the sequence number of the last packet received, as well as a replay list (when using authentication). The transport address of the RTP session, together with the SSRC, is used to determine which cryptographic context is used to encrypt or decrypt each packet.

The default encryption algorithm is the Advanced Encryption Standard in either counter mode or f8 mode,HTPU UTPHP HPTPU UTPH with counter mode being mandatory to implement. Other algorithms may be defined in the future.

58 , 59

The encryption process consists of two steps:

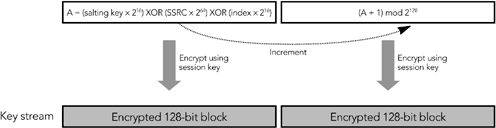
1. The system is supplied with one or more master keys via a non-RTP-based key exchange protocol, from which ephemeral session keys are derived. Each session key is a sampling of a pseudorandom function, redrawn after a certain number of packets have been sent, with the master key, packet index, and key derivation rate as inputs. The session key can potentially change for each packet sent, depending on the key derivation rate in the cryptographic context. The master key identifier may be used by the key management protocol, to signal which of the preexchanged master keys is to be used, allowing changes in the master key to be synchronized.
2. The packet is encrypted via the generation of a key stream based on the packet index and the salting and session keys, followed by computation of the bitwise exclusive-OR (XOR) of that key stream with the payload section of the RTP packet.

In both steps, the packet index is the 32-bit extended RTP sequence number. The details of how the key stream is generated depend on the encryption algorithm and mode of operation.

If AES in counter mode is used, the key stream is generated in this way: A 128-bit integer is calculated as follows: (2P16P x the packet index) XOR (the salting key x 2P16P) XOR (the SSRC x 2P64P). The integer is encrypted with the session key, resulting in the first output block of the key stream. The integer then is incremented modulo 2P128P,

and the block is again encrypted with the session key. The result is the second output block of the key stream. The process repeats until the key stream is at least as long as the payload section of the packet to be encrypted. HTUFigure 13.5UTH shows this key-stream generation process.

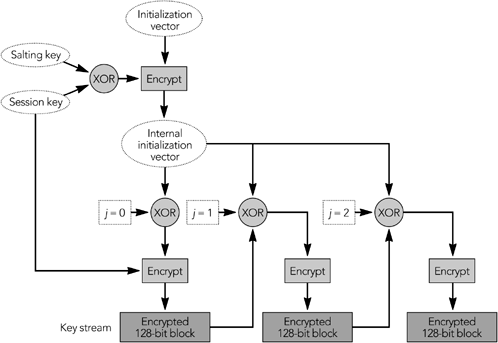
###### Figure 13.5. Key-Stream Generation for SRTP: AES in Counter Mode



When implementing AES in counter mode, you T*must*T ensure that each packet is encrypted with a unique key stream (the presence of the packet index and SSRC in the key stream derivation function ensures this). If you accidentally encrypt two packets using the same key stream, the encryption becomes trivial to break: You simply XOR the two packets together, and the plain text becomes available (remember from the discussion of parity FEC in HTUChapter 9UTH, Error Correction, that TA XOR B XOR B = AT).

If AES in f8 mode is used, the key stream is generated in this way: The XOR of the session key and a salting key is generated, and it is used to encrypt the initialization vector. If the salting key is less than 128 bits in length, it is padded with alternating zeros and ones (0x555...) to 128 bits. The result is known as the T*internal initialization vector*T. The first block of the key stream is generated as the XOR of the internal initialization vector and a 128-bit variable (T*j*T = 0), and the result is encrypted with the session key. The variable T*j*T is incremented, and the second block of the key stream is generated as the XOR of the internal initialization vector, the variable T*j*T, and the previous block of the key stream. The process repeats, with T*j*T incrementing each time, until the key stream is at least as long as the payload section of the packet to be encrypted. HTUFigure 13.6UTH shows this key-stream generation process.

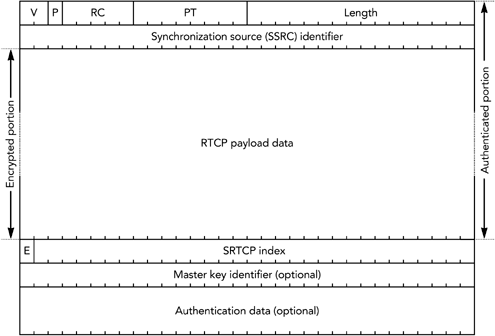
###### Figure 13.6. Key-Stream Generation for SRTP: AES in f8 Mode



The default encryption algorithm and mode is AES in counter mode; use of AES f8 mode can be negotiated during session initiation.

SRTP also provides confidentiality of RTP control packets. The entire RTCP packet is encrypted, excluding the initial common header (the first 64 bits of the packet) and several additional fields that are added to the end of each RTCP packet, as shown in HTUFigure 13.7UTH. The additional fields are an SRTCP (Secure RTCP) index, a bit to indicate if the payload is encrypted (the E bit), an optional master key identifier, and an authentication header (described in the section titled HTUAuthenticationUTH Using the Secure RTP Profile later in this chapter).

###### Figure 13.7. Secure RTP Encryption of a Control Packet



Encryption of RTCP packets proceeds in much the same way as encryption of RTP data packets does, but using the SRTCP index in place of the extended RTP sequence number.

The encryption prefix applied during standard RTCP encryption is not used with SRTP (the differences in encryption algorithm mean that the prefix offers no benefit). It is legal to split RTCP packets into encrypted and unencrypted packets, as can be done with standard RTCP encryption, indicated by the E bit in the SRTCP packet.

As with the RTP specification, the SRTP profile defines no mechanism for exchange of encryption keys. Keys must be exchanged via non-RTP means—for example, within SIP or RTSP. The master key identifier may be used to synchronize changes of master keys.

##### Confidentiality Using IP Security

In addition to the application-level security provided by standard RTP and Secure RTP, it is possible to use IP-level securityHTPU UTPHP HPTPU UTPH with RTP. IPsec is implemented as part of the operating system network stack or in a gateway, not by applications. It provides security for all communications from a host, and it is not RTP-specific.

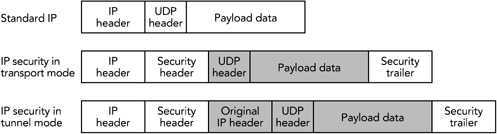
17 , 110

IP security (IPsec) has two modes of operation: transport mode and tunnel mode.

T*Transport mode*T inserts an additional header between IP and the transport header,

providing confidentiality of the TCP or UDP header and payload, but leaving the IP header untouched. T*Tunnel mode*T encapsulates the entire IP datagram inside a security header. HTUFigure 13.8UTH illustrates the differences between the two modes of operation. IP security in tunnel mode is most commonly used to build virtual private networks, tunneling between two gateway routers to securely extend an intranet across the public Internet. Transport mode is used when end-to-end security between individual hosts is desired.

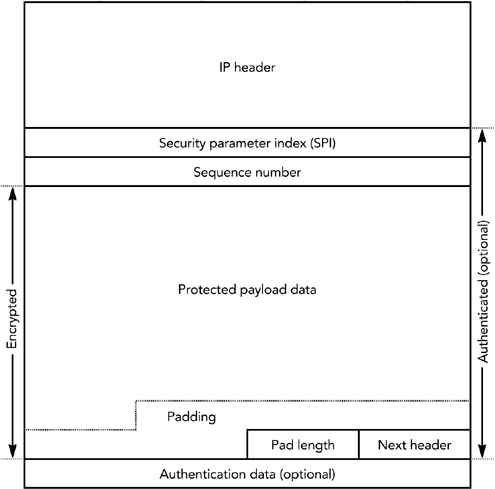
###### Figure 13.8. Transport Mode versus Tunnel Mode IPsec (Shaded Data Is Protected)



Both tunnel mode and transport mode support confidentiality and authentication of packets. Confidentiality is provided by a protocol known as the Encapsulating Security Payload (ESP).HTPU21UTPH ESP comprises an additional header and trailer added to each packet. The header includes a security parameter index and sequence number; the trailer contains padding and an indication of the type of the encapsulated data (TCP or UDP if transport mode is used, IP-in-IP if tunnel mode is used).

Encapsulated between the header and trailer is the protected data, including the remaining headers. HTUFigure 13.9UTH shows the encapsulation process, header, and trailer.

###### Figure 13.9. An Encapsulating Security Payload Packet



The security parameter index (SPI) and sequence number are 32-bit fields. The SPI is used to select the cryptographic context, and hence the decryption key to be used. The sequence number increments by one with each packet sent and is used to provide replay protection (see the section titled HTUReplay ProtectionUTH later in this chapter). Following these two header fields is the encapsulated payload: a UDP header followed by an RTP header and payload if transport mode is used; IP/UDP/RTP headers and payload if tunnel mode is used.

Following the payload data is padding, if required, and a "next header" field. This last field determines the type of the encapsulated header. Its name is somewhat misleading, given that the header to which this field refers is actually sent earlier in the packet. Finally, optional authentication data completes the packet (see the section titled HTUAuthentication Using IP SecurityUTH later in this chapter).

ESP encrypts the protected data section of the packet, using a symmetric algorithm (DES is mandatory to implement; other algorithms may be negotiated). If ESP is

used with RTP, the entire RTP header and payload will be encrypted, along with the UDP headers—and IP headers if tunnel mode is used.

It is not possible to use header compression with IP security in transport mode. If tunnel mode is used, the inner IP/UDP/RTP headers may be compressed before encryption and encapsulation. Doing so largely removes the bandwidth penalty due to the IPsec headers, but it does not achieve the efficiency gains expected of header compression. If bandwidth efficiency is a goal, application-level RTP encryption should be used.

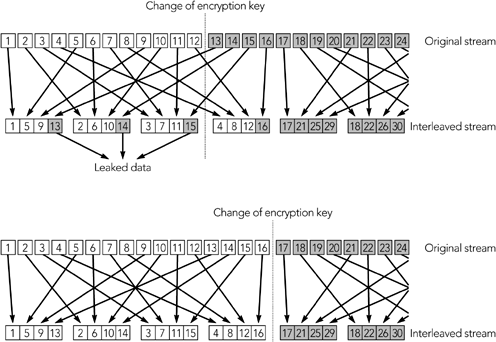
IP security may also cause difficulty with some firewalls and Network Address Translation (NAT) devices. In particular, IP security hides the TCP or UDP headers, replacing them with an ESP header. Firewalls are typically configured to block all unrecognized traffic, in many cases including IPsec (the firewall has to be configured to allow ESP [IP protocol 50], in addition to TCP and UDP). Related problems occur with NAT because translation of TCP or UDP port numbers is impossible if they are encrypted in an ESP packet. If firewalls and NAT boxes are present, application-level RTP encryption may be more successful.

The IP security protocol suite includes an extensive signaling protocol, the Internet Key Exchange (IKE), used to set up the necessary parameters and encryption keys. The details of IKE are beyond the scope of this book.

##### Other Considerations

Several RTP payload formats provide coupling between packets—for example, when interleaving or forward error correction is being used. Coupling between packets may affect the operation of encryption, restricting the times when it is possible to change the encryption key. HTUFigure 13.10UTH shows an example of interleaving that illustrates this problem.

###### Figure 13.10. Interactions between Encryption and Interleaving



The confidentiality mechanisms available for RTP can use a range of encryption algorithms, but they define a "must implement" algorithm to ensure interoperability. In many cases the mandatory algorithm is the Data Encryption Standard (DES).

Advances in computational power have made DES seem relatively weak—recent systems have demonstrated brute-force cracking of DES in less than 24 hours—so it is appropriate to negotiate a stronger algorithm if possible. Triple DESHTPU UTPH and the recently announced Advanced Encryption Standard (AES)HTPU UTPH are suitable possibilities.

57

58

The use of end-to-end encryption to ensure confidentiality in RTP is effective at preventing unauthorized access to content, whether that content is pay-per-view TV or private telephone conversations. This protection is generally desirable, but it does have some wider implications. In particular, there is an ongoing debate in some jurisdictions regarding the ability of law enforcement officials to wiretap communications. Widespread use of encryption as a confidentiality measure in RTP makes such wiretaps—along with other forms of eavesdropping—more difficult. In addition, some jurisdictions restrict the use, or distribution, of products that include encryption. You should understand the legal and regulatory issues regarding use of encryption in your jurisdiction before implementing the confidentiality measures described in this chapter.

## Authentication

There are two types of authentication: proof that the packets have not been tampered with, known as T*integrity protection*T, and proof that the packets came from the correct source, known as T*source origin authentication*T.

Integrity protection is achieved through the use of message authentication codes. These codes take a packet to be protected, and a key known only to the sender and receivers, and use these to generate a unique signature. Provided that the key is not known to an attacker, it is impossible to change the contents of the packet without causing a mismatch between the packet contents and the message authentication code. The use of a symmetric shared secret limits the capability to authenticate the source in a multiparty group: All members of the group are able to generate authenticated packets.

Source origin authentication is a much harder problem for RTP applications. It might first be thought equivalent to the problem of generating message authentication codes, but it is more difficult because a shared secret between sender and receiver is not sufficient. Rather, it is necessary to identify the sender in the signature, meaning that the signature is larger and more expensive to compute (in much the way public key cryptography is more expensive than symmetric cryptography). This requirement often makes it infeasible to authenticate the source of each packet in an RTP stream.

Like confidentiality, authentication can be applied at either the application level or the IP level, with much the same set of advantages and disadvantages. Both alternatives have been developed for use with RTP.

##### Authentication Using Standard RTP

Standard RTP provides no support for integrity protection or source origin authentication. Implementations that require authentication should either implement secure RTP or use a lower-layer authentication service such as that provided by the IP security extensions.

##### Authentication Using the Secure RTP Profile

SRTP supports both message integrity protection and source origin authentication. For integrity protection, a message authentication tag is appended to the end of the packet, as was shown in HTUFigure 13.4UTH. The message authentication tag is calculated over the entire RTP packet and is computed after the packet has been encrypted. At the time of this writing, the HMAC-SHA-1 integrity protection algorithm is specified

for use with SRTP. Other integrity protection algorithms may be defined in the future, and negotiated for use with SRTP.

Source origin authentication using the TESLA (Timed Efficient Stream Loss-tolerant Authentication) authentication algorithm has been considered for use with SRTP, but TESLA is not fully defined at the time of this writing. You are advised to consult the SRTP specification, when it is completed, for details.

The authentication mechanisms of SRTP are not mandatory, but I T*strongly*T recommend that all implementations use them. In particular, you should be aware that it is trivially possible for an attacker to forge data encrypted using AES in counter mode unless authentication is used. The secure RTP profile specification describes this issue in detail.

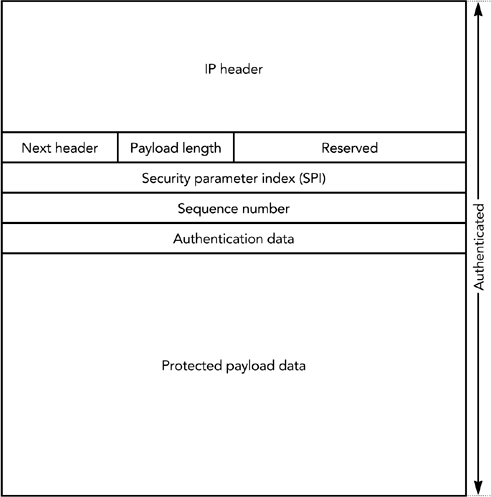
##### Authentication Using IP Security

The IP security extensions can provide integrity protection and authentication for all packets sent from a host. There are two ways this can be done: as part of the encapsulating security payload, or as an authentication header.

The Encapsulating Security Payload was described earlier, in the section titled HTUConfidentiality Using IP SecurityUTH. As shown in HTUFigure 13.9UTH, ESP includes an optional authentication data section as part of the trailer. If present, the authentication provides a check on the entire encapsulated payload, plus the ESP header and trailer. The outer IP header is not protected.

An alternative to ESP is the Authentication Header (AH) defined in RFC 2402HTPU18UTPH and shown in HTUFigure 13.11UTH. The fields in this header are much like their counterparts in ESP, and the AH can be used in both tunnel mode and transport mode. The key difference is that the entire packet is authenticated, including the outer IP header.

###### Figure 13.11. The IPsec Authentication Header



If the requirement is to provide confidentiality as well as authentication, then ESP is appropriate (using ESP to encrypt and AH to authenticate would lead to excessive bandwidth overheads). If the only requirement is authentication—not confidentiality—then the choice between ESP and AH depends on whether it is considered necessary to authenticate the outer IP header. Authentication of the outer header provides some additional security, by ensuring that the source IP address is not spoofed. It also has the side effect of rendering Network Address Translation impossible.

A range of authentication algorithms may be used with IP security. If authentication is used, the mandatory algorithms for both ESP and AH are HMAC-MD5-96 and HMAC-SHA-96,HTPU19UTPHP,HPTPU20UTPH which provide integrity protection only. Algorithms for source origin authentication may be defined in the future.

The earlier section titled HTUConfidentialityUTH Using IP Security noted potential interactions between IP security and header compression, firewalls, and NAT boxes. These issues also apply when IP security is used to provide authentication.

The IP security protocol suite includes an extensive signaling protocol, the Internet Key Exchange (IKE), which is used to set up the necessary parameters and authentication keys. The details of IKE are beyond the scope of this book.

## Replay Protection

Related to authentication is the issue of replay protection: stopping an attacker from recording the packets of an RTP session and reinjecting them into the network later for malicious purposes. The RTP timestamp and sequence number provide limited replay protection because implementations are supposed to discard old data. However, an attacker can observe the packet stream and modify the recorded packets before playback such that they match the expected timestamp and sequence number of the receiver.

To provide replay protection, it is necessary to authenticate messages for integrity protection. Doing so stops an attacker from changing the sequence number of replayed packets, making it impossible for old packets to be replayed into a session.

## Denial of Service

A potential denial-of-service threat exists with payload formats using compression that has nonuniform receiver-end computational load. If the payload format has such properties, an attacker might be able to inject pathological packets into a media stream, which are complex to decode and cause the receiver to be overloaded. There is little a receiver can do to avoid this problem, short of stopping processing packets in overload situations.

Another problem that can cause denial of service is failure to implement the RTCP timing rules correctly. If participants send RTCP at a constant rate rather than increasing the interval between packets as the size of the group increases, the RTCP traffic will grow linearly with the size of the group. For large groups, this growth can result in significant network congestion. The solution is to pay close attention to the correct implementation of RTCP.

Similarly, failure to implement the RTCP reconsideration algorithms can result in transient congestion when rapid changes in membership occur. This is a secondary concern because the effect is only transitory, but it may still be an issue.

Network congestion can also occur if RTP implementations respond inappropriately to packet loss. HTUChapter 10UTH, Congestion Control, describes the issues in detail; for our purposes here, suffice it to say that congestion can cause significant denial of service.

## Mixers and Translators

It is not possible to use an RTP mixer to combine encrypted media streams unless the mixer is trusted and has access to the encryption key. This requirement typically prevents mixers from being used if the session needs to be kept confidential.

Translation is also difficult, although not necessarily impossible, if the media streams are encrypted. Translation that involves recoding of the media stream is typically impossible unless the translator is trusted. Some translations, however, do not require recoding of the media stream—for example, translation between IPv4 and IPv6 transport—and these may be provided without affecting the encryption, and without having to trust the translator.

Source origin authentication is similarly difficult if the mixer or translator modifies the media stream. Again, if the mixer or translator is trusted, it may re-sign the modified media stream; otherwise the source cannot be authenticated.

## Active Content

TMost audiovisual content is passiveT: It comprises encoded data that is passed through a static decryption algorithm to produce the original content. With the exception of the issues due to nonuniform processing requirements noted earlier, such content is not dangerous.

There is, however, another class of content: that which contains executable code—for example, Java applets or ECMAScript, which can be included with MPEG-4 streams. Such T*active content*T has the potential to execute arbitrary operations on the receiving system, unless carefully restricted.

## Other Considerations

The text of SDES items is not null-terminated, and manipulating SDES items in languages that assume null-terminated strings requires care. This is a particular problem with C-based implementations, which must take care to ensure that they use lengthchecking string manipulation functions—for example, Tstrncpy()T rather than Tstrcpy()T. Careless implementations may be vulnerable to buffer overflow attacks.

The text of SDES items is entered by the user, and thus it cannot be trusted to have safe values. In particular, it may contain metacharacters that have undesirable side effects. For example, some user interface scripting languages allow command substitution to be triggered by metacharacters, potentially giving an attacker the means to execute arbitrary code.

Implementations should not assume that packets are well formed. For example, it might be possible for an attacker to produce packets whose actual length does not correspond to the expected length. Again, there is a potential for buffer overflow attacks against careless implementations.

## Summary

This chapter has outlined the features of RTP that provide privacy and confidentiality of communication, as well as authentication of participants and the media stream. It has also described various potential attacks on a system, and how they can be prevented.

These are not the only issues that a secure implementation needs to consider, though; it is also necessary to secure the session initiation and control protocols. This chapter has highlighted some issues relating to the security of these initiation and control protocols, but a full treatment of this topic is outside the scope of this book.