CHAPTER 8

Security Features of IPv6[1]

TCP/IP networks based on IPv4 are plagued with security problems because they are designed to work in a friendly environment and with physically secure connections. When these assumptions are no longer valid—as they are nowadays—the many security weaknesses of IPv4 become manifest and can be easily exploited.

In general, IP communications are exposed to several types of attack:

[1]This chapter was written by Antonio Lioy, professor at Politecnico di Torino, who, besides being one of the greatest Italian experts in the security field, is a good friend of mine. I sincerely thank him for his important contribution to my book.

- *packet sniffing:* Due to network topology, IP packets sent from a source to a specific destination can also be read by other nodes, which can then get hold of the payload (for example, passwords or other private information).
- *IP spoofing:* IP addresses can be very easily spoofed both to attack those services whose authentication is based on the sender's address (as the `rlogin` service or several WWW servers) and to supply wrong information to subvert the logical organization of the network (for example, by forging false ICMP messages of the type "destination unreachable" or "redirect").
- *connection hijacking:* Whole IP packets can be forged to appear as legal packets coming from one of the two communicating partners, to insert wrong data in an existing channel.

Solutions to these and other attacks are not always available. When countermeasures do exist, they are usually placed at the application level. As a consequence, solutions are usually not interoperable, and several functions are duplicated inside different applications. The development of a new version of the IP protocol has offered a chance to insert some basic security mechanisms at the network level so that they can be available to all the layered applications. The security techniques adopted in IPv6 have been designed to be easily inserted also in IPv4, as detailed in RFC 1825[1], which introduces IPSEC, the new generic security architecture at the IP level. However, because the IPv4 protocol also suffers from other problems, it is unlikely that current network stacks and applications will be modified only to implement IPSEC. On the contrary, it is very likely—and probably will even be required for standard's compliance—that the IPSEC security features be implemented in IPv6.

We might question whether locating the security functions at the IP level is appropriate. Obviously, no definitive answer exists because, generally, the security of a system is not based on a single element; rather it is the result of a combination of several elements. The IP level is surely the right one to block many low-level attacks, as those mentioned at the beginning of this section, which account for a large percentage of all the network attacks due to their simple implementation. On the other hand, IPSEC is not a complete solution when the applications to be protected are user-oriented (as in the case of electronic mail) rather than network-oriented. Last but not least, the IPv6 security features are implemented by extension headers (see Section 3.2) so that they can be easily turned off when security aspects are not relevant and network throughput is of paramount importance.

8.1 Security Features

Security features in IPv6 have been introduced mainly by way of two dedicated extension headers: the *Authentication Header* (AH) and the *Encrypted Security Payload* (ESP), with complementary capabilities.

The AH header was designed to ensure authenticity and integrity of the IP packet. Its presence guards against two threats: illegal modification of the fixed fields and packet spoofing. On the other hand, the ESP header provides data encapsulation with encryption to ensure that only the destination node can read the payload conveyed by the IP packet. The two headers can be used together to provide all the security features simultaneously.

Both the AH and the ESP headers exploit the concept of security association (SA) to agree on the security algorithms and parameters between the sender and the receiver. In general, each IPv6 node manages a set of SAs, one for each secure communication currently active. The *Security Parameters Index* (SPI) is a parameter contained in both the AH and ESP headers to specify which SA is to be used in decrypting and/or authenticating the packet.

In unicast transmissions, the SPI is normally chosen by the destination node and sent back to the sender when the communication is set up. In multicast transmissions, the SPI must be common to all the members of the multicast group. Each node must be able to identify the right SA correctly by combining the SPI with the multicast address.

The negotiation of an SA (and the related SPI) is an integral part of the protocol for the exchange of security keys.

8.1.1 Authentication Header (AH)

The Authentication Header[2] is one of the general extension headers defined for IPv6; it is identified by the value 51 in the Next Header field (see Table 3-2) of the previous header. Normally, it is inserted between the IPv6 header and the upper level payload, as shown in Figure 8-1.

The format of the AH header (depicted in Figure 8-2) is simple; it is composed of a 64-bit fixed part followed by a variable number of 32-bit blocks. The fixed part contains the following:

- The value of the next type of payload in the daisy chain of headers (8 bits)
- The Payload Length—that is, the total length of the authentication data expressed as a multiple of 32-bit words (8 bits)

- A reserved field (16 bits)
- The SPI used by this header (32 bits)

The variable part of the AH header is composed of a variable number of 32-bit blocks, which contain the actual authentication data. Because the Payload Length is expressed as an 8-bit number, a maximum of 255 32-bit blocks can be used—that is, 1020 bytes. As a consequence, the exact length of this header depends on the selected authentication algorithm.

When the destination node receives a packet with an AH header, the packet's authenticity and integrity can be checked by using the procedure illustrated in Figure 8-3. For the preliminary step, care should be taken in normalizing the received packet, to eliminate all the variable parts and correctly compute the authentication value only on the fixed parts. Figure 8-4 illustrates the procedure to normalize the packet and to compute the authentication value.

8.1.2 Authentication Techniques

Data integrity in telecommunication systems is normally ensured by computing and checking the value of a suitable function of the data, often named *Message Digest* (MD). Among the most frequently used algo-

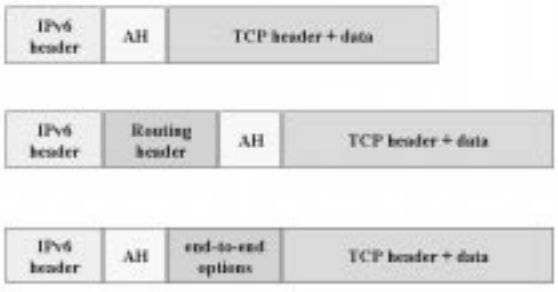

Figure 8-1
Examples of use of the AH header

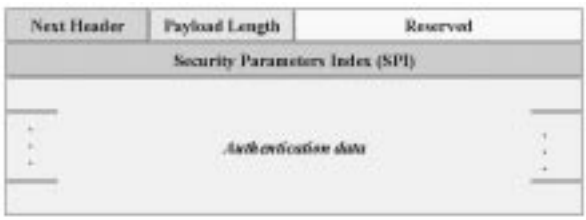

Figure 8-2
Structure of the AH header

Figure 8-3
Procedure to verify the authenticity of a packet protected by the AH header

1. Clear the Hop Count field.
2. If the packet contains a Routing Header, then do the following:
   - **2.1.** Set the Destination Address field to the address of the final destination.
   - **2.2.** Set the Routing Header field to the value that it will have at the final destination.
   - **2.3.** Set the Address Index field to the value that it will have at the final destination.
3. Clear all the options that have the C bit (change en route) active.

Figure 8-4
Procedure for packet normalization to compute the authentication value

rithms are CRC-16 and CRC-32 (see *Applied Cryptography* [3]).

These functions effectively perform their tasks when data modifications are caused by random errors, but they are completely inadequate to protect the packets against deliberate modifications. In this case, a reasonable degree of protection can be ensured only by better digest algorithms, such as MD5[4] or SHA[5].

We should note that data integrity without origin authentication is completely useless. Therefore, digest algorithms are normally applied in a way to include some parameters that can be used to provide proof of the sender's identity simultaneously. Often this result is achieved by using public key encryption algorithms; unfortunately, they are computationally much heavier than digest algorithms. Because speed is a premium in computer networks, the default authentication technique chosen for IPSEC is a simpler one, named *keyed MD5*[6]. Briefly, the technique calls for com-

puting the MD5 digest on the data to be protected, preceded and followed by a *key* (a secret string of bits). The exact sequence of operations to compute this type of digest is shown in Figure 8-5.

The keyed-MD5 algorithm must be provided by any standard implementation of IPv6. However, the MD5 algorithm has been recently shown to be attackable, so it is highly likely that in the near future other authentication techniques will be standardized for use in IPv6. For example, the *keyed-SHA* technique has been proposed in RFC 1852[7]. It is based on the SHA[5] message digest algorithm, which exhibits better security properties than MD5 because it produces a 160-bit digest rather than a 128-bit digest.

8.1.3 Encrypted Security Payload (ESP)

The Encrypted Security Payload[8], which is one of the general extension headers defined in IPv6, is identified by the value 52 in the Next Header field (see Table 3-2) of the preceding header. When used, this block must always be the last one in the header chain because it completely hides both the upper level payload and all the next headers (see Figure 8-6).

Even the ESP header itself is only partly in the clear (see Figure 8-7); it consists of an integer number of 32-bit blocks, with the first one containing the SPI to select the SA to be used in decrypting all other blocks in the packet.

The exact format of the encrypted part depends on the encryption algorithm used. The default encryption technique in IPv6 is DES-CBC[9], which is the DES algorithm applied in *Cipher Block Chaining* (CBC) mode. DES is a private key encryption algorithm that is normally applied to 64-bit data blocks with a 56-bit key (extended to 64 bits by adding one parity bit for each 7 bits of the key). Various techniques have been proposed to apply the DES transformation to blocks bigger than 64 bits. The CBC mode divides the data stream into a sequence of 64-bit blocks, and

Figure 8-5
Algorithm to generate a keyed MD5 digest

1. Given a message M to protect, normalize it (M′).
2. Pad the message M′ by adding as many zero bytes as necessary to align the message to a multiple of 128 bits (message M'_p).
3. Pad the key K by adding as many zero bytes as necessary to align the key to a multiple of 128 bits (message K_p).
4. Compute the authentication value as the result of the MD5 function applied to the argument given by the concatenation K_p, M'_p, K_p.

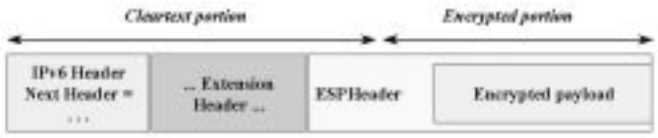

Figure 8-6
IPv6 packet with an ESP header

Figure 8-7
Structure of the ESP header

each block is EX-ORed with the result of the previous encryption before being encrypted itself. Let *E(d,k)* be the encryption operation applied to the data block *d* with key *k*; then the CBC mode can be described by the following transform to generate the i-th encrypted block:

$$c_i = \boldsymbol{E}\,(d_i \oplus c_{i\text{-}1},\, k)$$

Obviously, the encryption of the first data block d^1 requires an initial value $c^{0,}$ commonly called the *Initialization Vector* (IV). The initialization vector must not be null and must be carefully chosen to insert a random factor in the encryption process. This is needed to avoid cryptographic attacks based on partial knowledge of the data being encrypted, such as the *known-plaintext* attacks that can be led against the fixed header of some common files (for example, the data files of various office automation tools). Normally, the IV value is either a 64-bit number generated by a pseudo-random number generator, or the value is a 32-bit number generated in a similar way and is then extended to 64 bits by concatenating it to its complement.

In the DES-CBC mode, the encrypted portion of the ESP header (see Figure 8-8) begins with an initialization vector composed of an integer number of 32-bit words. In general, the exact length of the IV depends on the security association being used; however, RFC 1829[9] provides specification only for vectors of 32 or 64 bits.

The IV is followed by the encrypted payload that is padded with blocks to ensure that the total dimension of the ESP header is a multiple of 64 bits. The next-to-last byte in the ESP header contains the padding length (expressed in bytes), whereas the last byte contains the payload type. The minimum length of the padding varies between 0 and 7 bytes, but using a longer padding (up to 255 bytes) to hide the real length of the encrypted data is legal.

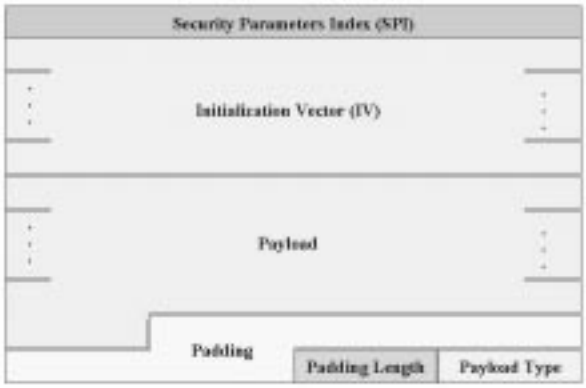

Figure 8-8
Structure of the ESP header in the DES-CBC case

The DES-CBC algorithm must be available in all IPv6 standard implementations. Because the DES algorithm can be regarded at best as a moderately difficult algorithm to be broken, it is very likely that in the near future other algorithms will be standardized for use in IPv6. For example, the 3DES-CBC algorithm is proposed in RFC 1851[10]. This technique is based on the repeated application of the DES transformation to the same data block with three different keys, and it is cryptographically stronger than plain DES because it is equivalent to an encryption algorithm that uses a 112-bit key (rather than the 56-bit key used by DES).

8.2 Key Management

Correct application of the AH and ESP headers requires that all the communicating parties agree on a common key to be used in forming and checking the security headers. IPv6 allows for key management to occur either out-of-band or with specifically crafted protocols. However, no general agreement has yet been reached on this subject within the Internet community, with different groups stressing different needs: fast key exchange, strong authentication, lightweight protocols, and others. Key management is the area that is still mostly unsettled within the whole IPSEC architecture.

8.2.1 Manual Key Management

IPv6 requires every implementation to allow for manual setting of the se-

curity keys, in case no in-line key management technique is adopted or human-based security is desired. Obviously, manual keying is possible only if the security operators have separately agreed out-of-band on the keys to be used—for example, at a reserved meeting.

This solution exhibits high personnel costs and does not scale well because it requires personal action of an operator on each network device taking part in the secure channel. Additionally, it can generate a false sense of security. Remember that human intervention does not automatically ensure a higher level of security, due to untrusted operators and residual problems related to hardware and software integrity of the device where the key is set.

However, in spite of these disadvantages, manual key management finds application in restricted environments, with a small number of devices physically secured that, according to the security policy, can operate only when explicitly enabled by human intervention.

8.2.2 Automatic Key Management

Within the IPSEC, key management is surely the area that is less settled and the area in which much work has yet to be done before arriving at a set of protocols that completely meet the security needs at the IP level. The only decision that has already been made is that, for sake of generality, the key management protocol (IKMP, Internet Key Management Protocol) will be placed at the application layer, and it will be independent of the protocols at the lower layers.

A first proposal is to base IKMP on the coupling of the ISAKMP[11] and Oakley[12] protocols, as described in the IEFT Draft, *The Resolution of ISAKMP with Oakley*[13].

Internet Security Association and Key Management Protocol (ISAKMP) defines a generic architecture for authenticated SA setup and key exchange, without specifying the actual algorithms to be used. In this way, it can be used with different key exchange techniques.

Oakley is a key-exchange protocol, based on a modified version of the Diffie-Hellman algorithm (see[3]). Therefore, it is one of the natural partners for ISAKMP.

However, in addition to the ISAKMP-Oakley couple, different solutions are being proposed. Currently, the major competitor is *Simple Key-management for Internet Protocols* (SKIP)[14], which bases its operations on the Diffie-Hellman algorithm. SKIP is simple and addresses several problems of key management in high-speed networks, such as zero-message key

setup and updates that permit fast dynamic rekeying (that is, frequent in-line change of the security keys to avoid analytic attacks based on accumulation of cyphertext encrypted with the same key). Moreover, although SKIP is not yet standardized, it already features many commercial-level implementations, both for UNIX workstations and personal computers.

So the war of the key-management protocols is raging, and the likely outcome is that more than one protocol will attain RFC status because these protocols exhibit different merits that are valuable in different application environments.

8.3 Application of IPv6 Security Features

The AH and ESP headers can be used in different ways to protect IP communications. In the following subsections, we will briefly review some of the most interesting applications, with references to the corresponding weaknesses in IPv4.

8.3.1 Private Virtual Networks

Nowadays, technical and economical reasons are pushing implementation of corporate wide area networks to migrate from dedicated links and proprietary network technologies to solutions based on public shared links and open network architectures. This migration creates several advantages but currently exhibits a serious drawback: There is a drastic reduction in intrinsic system security, due to the use of shared channels and devices.

To regain the same previous level of network security while maintaining the economic advantages offered by public networks, an organization has to succeed in separating and protecting its own data packets within the crowd of packets traveling across the public links. Usually, this result is achieved by establishing a *Virtual Private Network* (VPN). In IPv4, this is done by using the *IP tunneling* technique: IP packets to be protected are wrapped in a security envelope and encapsulated inside normal IP packets that are used just to transport the original packets across the public network to their final destination. Often, the endpoints of an IP tunnel are not the hosts wanting to exchange the data; rather they are

two *firewalls* that protect the LANs from external attacks. This setup is shown in Figure 8-9.

In IPv6, creating a VPN is easier and more standard than in IPv4, thanks to the AH and ESP headers. As an example, with reference to Figure 8-9, let's suppose that a TCP channel between host H1 in network N1 and host H2 in network N2 has to be protected only against data manipulation and origin falsification, while data privacy is not required. In this case, the AH header can be exploited in the following way. The FW1 firewall gets the IP packet shown in Figure 8-10 and modifies it by adding an AH header before sending it to its partner, FW2, as shown in Figure 8-11.

When this packet is received from the FW2 firewall, the firewall checks the packet for integrity and origin authentication by using the data in the AH header. If the check is successful, then the IP header and the AH header are removed, and the remaining data (that is, the original packet) are sent to the final destination, as shown in Figure 8-12.

If the VPN is implemented by using only the AH header, then attackers can neither alter the transmitted packets nor insert forged packets in the channel. However, they can still read the content of the packets. To prevent disclosure of the payload, the ESP header has to be used, too. Even the use of AH in conjunction with ESP does not completely protect the traffic; packets

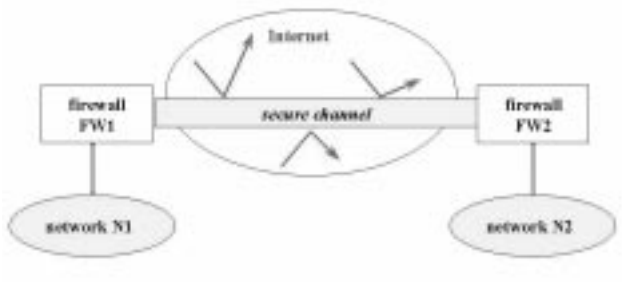

Figure 8-9
Example of a tunnel between two firewalls

IPv6 header (src=H1, dest=H2, Next Header=TCP)

TCP payload

Figure 8-10
IPv6 packet sent from H1 to FW1

IPv6 header (src=FW1, dest=FW2, Next Header=AH)

AH header (Next Header=IPv6)

IPv6 header (src=H1, dest=H2, Next Header=TCP)

TCP payload

Figure 8-11
IPv6 packet sent from FW1 to FW2

can be deleted by intermediate nodes or recorded and later replayed. These attacks cannot be easily contrasted at the IP level; appropriate defenses (such as the use of unique packet identifiers and the generation of heartbeat packets) are usually placed at some upper level in the network stack. A partial solution at the IP level is likely to be offered by the new format and algorithms that are going to replace the current ones in the AH header.

Comparing this method of creating a VPN with the one usually adopted in IPv4 by many firewall suppliers that also offer secure tunnels is interesting. The basic architecture is the same as that used in IPv6 (refer to Figure 8-9), but, because IPv4 does not allow for multiple headers, the tunnel has to be implemented by some form of encapsulation, such as IP in IP[15]. Obviously, this solution raises problems of compatibility between the firewalls of different vendors as well as fragmentation problems. If the packet to be transmitted already has the maximum dimension allowed for an IP packet, then encapsulating it inside another IP packet is not possible; fragmentation and reassembling must take place at the two endpoints of the tunnel. As a consequence, the performance of the virtual channel can degrade down to 50 percent of the normal throughput. The worst case takes place for larger packets, which are typically used in transferring large data sets that, by contrast, would need no fragmentation to achieve maximum speed. On the other hand, the best case occurs for small packets, such as those used in interactive applications that, ironically, would better accept even a larger performance penalty, as long as the total throughput remains compatible with the reaction time of the human operator.

In IPv6, the situation is completely inverted; because the overhead is fixed in size (the dimension of AH, or that of AH plus ESP) and independent of the dimension of the original packet, the applications that suffer the highest overhead are the interactive ones, which are the applications with better resistance properties.

Anyway, in both cases, the performance penalty is definitely lower for the VPN implemented in IPv6 compared to those built in IPv4.

Last but not least, it is interesting to realize that this VPN technique can be adopted even between a firewall and a single external host (see Figure 8-13). Obviously, this case is of particular relevance to guaranteed security when a mobile host is used outside the protected network perimeter, and it is a perfect complement to the mobility support features of IPv6 (see Chapter 10). The firewall will act as home agent for H_M in the Neigh-

IPv6 header (src=H1, dest=H2, Next Header=TCP)

TCP payload

Figure 8-12
IPv6 packet sent from FW2 to H2

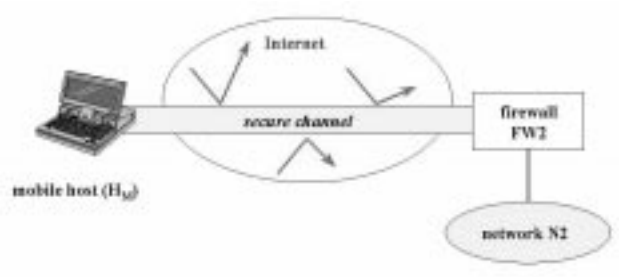

Figure 8-13
Tunnel between a firewall and a single host

bor Discovery procedure. H_M will be assigned two different IP addresses: one when it is connected inside the security perimeter and the other one when it is outside the perimeter. In this last case, the firewall will also act as a relay, by redirecting packets coming from inside the corporate network to the external address, after adding the required headers (AH only, or AH plus ESP).

8.3.2 Application-Level Security

Networked applications executing on top of an IPv6 stack may choose to require the use of a communication channel with specific features. To avoid duplication of functionality (and hence performance degradation), being able to specify, at the transport layer, the security attributes of the channel being created is useful. In the first BSD-UNIX implementations of IPv6, this effect can be obtained by properly using the **`setsocketoption()`** system call.

Anyway, this solution is not complete for application-level security because only partial protection is obtained. AH provides host-based authentication only; whereas applications usually require user-based authentication. Moreover, AH and ESP protect the data only during their transmission along the channel. After the data have been received, they are no longer protected in any way. This fact may not be relevant if the receiving host is a secure one, but there is the additional implication that origin authentication and data integrity properties are lost as well, so formal nonrepudiation cannot occur after the data have been extracted from the secure channel.

We can therefore draw the conclusion that the security features of IPv6 do not eliminate the need for other security mechanisms, which will probably be better placed at the application level.

8.3.3 Routing Security

Because IP addresses in IPv6 are quite often dynamically assigned, it is of the utmost importance that this process be done in a secure fashion. Moreover, as different security properties are available through a proper combination of AH and ESP headers, it is highly desirable that they be applied to the messages exchanged by routers to prevent attacks aiming to subvert the logical architecture of the network.

The following types of communications should be protected:

- The routing advertisement messages, to ensure that they are originated by an authorized router
- The neighbor advertisement messages, to ensure that they come from authorized hosts and to avoid the risk of somebody attaching a new host to the network without proper authorization
- The ICMP messages related to an unreachable host or network (*destination unreachable*) or to a better route (*redirect*), to ensure that these messages come from hosts or routers that were on the original path of the packets

Securing these types of messages is surely not trivial. For example, the routing advertisements are sent to a multicast group; therefore, all the routers in the group must know the (common) secret key to be used to verify and/or decrypt the messages. In turn, this fact implies that they can forge messages and impersonate any router in the group!

Protection of the neighbor advertisements poses a serious problem; these messages can be protected only after an SA has been created between the host and the address distribution center. On the other hand, this SA can be created only after an address has been assigned to the host, so we can conclude that this is the typical "chicken-and-egg" problem, which has no correct solution. To break the loop, partial solutions are possible. For example, priority can be given to the address assignment phase, and SA setup can be permitted only subsequently, but in this way the address assignment phase is not protected. Alternatively, public key authentication can be used. Each host is assigned a key pair (private and public key) and has to be preconfigured with the public key of the authority that signs the certificates of the routers and the address distribution centers. The last alternative is to configure routers so that they do not advertise local prefixes; in this way, each host is forced to contact a router first.

Protection against false ICMP messages requires that they be protected by an AH header, but this approach has the drawback of requiring the establishment of an SA with each router and host on the path between

the source and the destination of the packets.

With respect to the security of the messages used by the various routing protocols, they should always be exchanged only within the frame of an SA and be protected by AH. For the sake of generality, this solution is highly preferable to using authentication mechanisms specific for each routing protocol.

Based on the preceding analyses, we can conclude that routing security is apparently still a problem in IPv6, but chances of solving the problem are higher than in IPv4.

8.4 Future Directions

Security is one of the fastest moving areas in computer networks because protecting data and computer resources is vital, as is enabling economic exploitation through electronic commerce. IPv6 security is not the exception to the rule; although this area is new, it is already undergoing a redesign to better achieve its objectives.

Currently, AH and ESP headers are being modified along the following guidelines:

- The AH format is substantially changing to accommodate new and stronger authentication algorithms (HMAC[16]) that support prevention of packet replay and cancellation. (RFC 2085[17] describes this format when used with the MD5 digest algorithm.)
- The ESP specification is only marginally changing to achieve a better orthogonality with algorithms, to simplify application of different encryption algorithms.

The net benefit of these changes will be that more security will be available at the network level; hence, applications will be able to concentrate on different security aspects, such as authorizations and nonrepudiation.

REFERENCES

[1]R. Atkinson, *RFC 1825: Security Architecture for the Internet Protocol*, August 1995.

[2]R. Atkinson, *RFC 1826: IP Authentication Header*, August 1995.

[3]B. Schneier, *Applied Cryptography*, John Wiley & Sons, New York, 1996.

[4]R. Rivest, *RFC 1321: The MD5 Message-Digest Algorithm*, April 1992.

[5]*Secure Hash Standard*, Document FIPS-180-1, National Institute of

Standards and Technology, U.S. Department of Commerce, April 1995.

[6]P. Metzger and W. Simpson, *RFC 1828: IP Authentication using Keyed MD5*, August 1995.

[7]P. Metzger and W. Simpson, *RFC 1852: IP Authentication using Keyed SHA*, September 1995.

[8]R. Atkinson, *RFC 1827: IP Encapsulating Security Payload (ESP)*, August 1995.

[9]P. Karn, P. Metzger, and W. Simpson, *RFC 1829: The ESP DES-CBC Transform*, August 1995.

[10]P. Karn, P. Metzger, and W. Simpson, *RFC 1851: The ESP Triple DES Transform*, September 1995.

[11]D. Maughhan, M. Schertler, M. Schneider, and J. Turner, *Internet Security Association and Key Management Protocol (ISAKMP)*, IEFT Draft (**`draft-ietf-ipsec-isakmp-*.txt`**).

[12]H. Orman, *The Oakley Key Determination Protocol*, IEFT Draft (**`draft-ietf-ipsec-oakley-*.txt`**).

[13]D. Harkins and D. Carrel, *The Resolution of ISAKMP with Oakley*, IEFT Draft (**`draft-ietf-ipsec-isakmp-oakley-*.txt`**).

[14]A. Aziz, T. Markson, and H. Prafullchandra, *Simple Key-Management For Internet Protocols (SKIP),* IEFT Draft (**`draft-aziz-skip-*.txt`**).

[15]W. Simpson, *RFC 1853: IP in IP Tunnelling*, October 1995.

[16]H. Krawczyk, M. Bellare, and R. Canetti, *RFC 2104: HMAC: Keyed-Hashing for Message Authentication*, February 1997.

[17] M. Oehler and R. Glenn, *RFC 2085: HMAC-MD5 IP Authentication with Replay Prevention*, February 1997.