CHAPTER 4

TRIPLE HASH AUTHENTICATION SECURITY MECHANISM

4.1 INTRODUCTION

The next step to further improve the security in ad hoc networks is to provide an efficient security mechanism based on Ad hoc On-demand Distance Vector (AODV) routing protocol and Dynamic Source Routing (DSR) protocol. The THT Authentication is developed as security mechanism to protect the routing information. THT Authentication is an extension of Dual Hash Authentication which was discussed in the previous chapter. It includes HMAC Authentication as the third part.

4.2 ABOUT HMAC

4.2.1 HMAC AUTHENTICATION

In recent years, there has been increased interest in developing a MAC derived from a cryptographic hash code, such as MD5, SHA-1, or RIPEMD-160.

The motivations for this interest are:

- Cryptographic hash functions generally execute faster in software than symmetric block ciphers such as DES. Library code for cryptographic hash functions is widely available.
There are no export restrictions for cryptographic hash functions, whereas symmetric block ciphers, even when used for MACs, are restricted.

A hash function such as MD5 was not designed for use as a MAC and cannot be used directly for that purpose because it does not rely on a secret key. There have been a number of proposals to incorporate a secret key into an existing hash algorithm. HMAC received the most support. HMAC has been chosen as the mandatory-to-implement MAC for IP Security, and is used in other Internet protocols, such as Transport Layer Security (TLS, soon to replace Secure Sockets Layer) and Secure Electronic Transaction (SET).

4.2.2 HMAC-Design Objectives

The design objectives for HMAC include:

1. To use, without modifications, available hash functions. In particular, hash functions that perform well in software, and for which code is freely and widely available.

2. To allow for easy replaceability of the hash function in case there are more secure hash functions found or required.

3. To maintain the original performance of the hash function without a significant degradation.

4. Handling and usage of key is simple.

5. To have a well understood cryptographic analysis of the strength of the authentication mechanism based on reasonable assumptions on the embedded hash function.
4.2.3 The HMAC Algorithm

The Hash-based Message Authentication Code (HMAC) algorithm is implemented using binary operations and hash functions. To compute MAC over the data ‘text’ using the HMAC function, the following operation is performed:

\[
\text{HMAC}(K, \text{text})^t = H((K0 \oplus \text{opad}) || H((K0 \oplus \text{ipad}) || \text{text}))^t
\]

**HMAC Parameters and Symbols**

- **B** Block size (in bytes) of the input to the Approved hash function.
- **H** An Approved hash function.
- **Ipad** Inner pad; the byte \( x'36' \) repeated \( B \) times.
- **K** Secret key shared between the originator and the intended receiver(s).
- **K0** The key \( K \) after any necessary pre-processing to form a \( B \) byte key.
- **L** Block size (in bytes) of the output of the Approved hash function.
- **Opad** Outer pad; the byte \( x'5c' \) repeated \( B \) times.
- **t** The number of bytes of MAC.
- **Text** The data on which the HMAC is calculated; text does not include the padded key. The length of text is \( n \) bits, where \( 0 \leq n < 2B - 8B \).
- **x’N’** Hexadecimal notation, where each symbol in the string ‘N’ represents 4 binary bits.
- **||** Concatenation
- **(+)** Exclusive-Or operation.
**STEP-BY-STEP DESCRIPTION**

Step 1: If the length of $K = B$: set $K_0 = K$. Go to step 4.

Step 2: If the length of $K > B$: hash $K$ to obtain an $L$ byte string, then append $(B-L)$ zeros to create a $B$-byte string $K_0$ (i.e., $K_0 = H(K) \| 00...00$). Go to step 4.

Step 3: If the length of $K < B$: append zeros to the end of $K$ to create a $B$-byte string $K_0$ (e.g., if $K$ is 20 bytes in length and $B = 64$, then $K$ will be appended with 44 zero bytes 0x00).

Step 4: Exclusive-Or $K_0$ with $ipad$ to produce a $B$-byte string: $K_0 (+) ipad$.

Step 5: Append the stream of data 'text' to the string resulting from step 4: $(K_0 (+) ipad) \| text$.

Step 6: Apply $H$ to the stream generated in step 5: $H((K_0 (+) ipad) \| text)$.

Step 7: Exclusive-Or $K_0$ with $opad$: $K_0 (+) opad$.

Step 8: Append the result from step 6 to step 7: $(K_0 (+) opad) \| H((K_0 (+) ipad) \| text)$.

Step 9: Apply $H$ to the result from step 8: $H((K_0 (+) opad) \| H((K_0 (+) ipad) \| text))$.

Step 10: Select the leftmost $t$ bytes of the result of step 9 as the MAC.

### 4.2.4 HMAC Implementation

The HMAC algorithm is specified for an arbitrary approved cryptographic hash function, $H$. With minor modifications, a HMAC implementation can easily replace one hash function, $H$, with another hash function, $H’$. 
Conceptually, the intermediate results of the compression function on the B-byte blocks \((K0 (+) \text{ipad})\) and \((K0 (+) \text{opad})\) can be precomputed once, at the time of generation of the key \(K\), or before its first use. These intermediate results can be stored and then used to initialize \(H\) each time that a message needs to be authenticated using the same key. For each authenticated message using the key \(K\), this method saves the application of the hash function of \(H\) on two B-byte blocks (i.e., on \((K (+) \text{ipad})\) and \((K (+) \text{opad})\)). This saving may be significant when authenticating short streams of data. These stored intermediate values shall be treated and protected in the same manner as secret keys.

### 4.2.5 HMAC Security

The security of any MAC function based on an embedded hash function depends in some way on the cryptographic strength of the underlying hash function. The appeal of HMAC is that its designers have been able to prove an exact relationship between the strength of the embedded hash function and the strength of HMAC. The security of a MAC function is generally expressed in terms of the probability of successful forgery with a given amount of time spent by the forger and a given number of message-MAC pairs created with the same key. In essence, it can be shown that, for a given level of effort (time, message-MAC pairs), on messages generated by legitimate users and seen by attackers, the probability of a successful attack on HMAC is equivalent to one of the following attacks on the embedded hash function:

- Attackers are able to compute an output of the compression function even with an Initial Value (IV) that is random and secret to attackers.
• Attackers find collisions in the hash function even when the IV is random and secret.

4.3 TRIPLE HASH AUTHENTICATION IN AODV AND DSR

4.3.1 Triple Hash Technique

The Fast and Efficient Hash function is adopted to authenticate routing information instead of digital signatures under the reasonable assumption that no two compromised nodes are colluding and are within two hops of each other. In this Triple Hash authentication, one is used to authenticate the received routing packets and the other is used to prevent the current nodes modifying the routing information themselves and the third is used to authenticate the source at the destination. If some compromised node modifies the routing information, its neighbouring nodes can detect the misbehavior immediately. In an initial phase each node makes use of the management of local node group to distribute the common secret with its two-hop node group.

4.3.1.1 Distribution of Secret Key

In this technique each node needs to distribute a common secret by its two-hop node group. This secret key is kept secret against its one-hop node group. Each node has a pair of private and public keys (widely known). The source node generates random key $K_s$ and encrypts it with the public key of the nodes within two-hop. On receiving the encrypted key each node decrypts it with the corresponding private key and gets the common secret key $K_s$. Due to the mobility, ad hoc network can result in the change of the local node groups and the distribution of the common secrets should be adjusted timely. When some new nodes join in two-hop node group, the source node needs to distribute $K_s$ to those new nodes and if some nodes within two-hop becomes
the member of its one-hop node group (due to roaming) the source needs refreshing and redistribution of \( K_s \).

### 4.3.1.2 Triple Hash Algorithm

The Public one way hash function \( H \) and HMAC are used to authenticate the RREQ thrice, so that the routing packets includes not only the RREQ but also three hash values (\( H_1, H_2, H_3 \)), where \( H_2 \) is used to check whether the received routing packet has been modified, \( H_1 \) is used to prevent the current node modifying the packet and \( H_3 \) is used to check the routing packet at the destination. The algorithm is as follows.

1. Generate RREQ from source node \( RREQ = \{ S, L, H, R \} \)
   a. \( S \)- Source Identity; \( L \)- sequence number (RREQ)
   b. \( R \)- Routing information. \( H \)- Hop count.
2. Source multicasts \( \{ S, L+1, H, R, H1, 0, H3 \} \) to its multicasts group.
3. Any intermediate node within this group can verify the authenticity of packet. \( H2=0 \) (from source node); \( H1=H(S\backslash L+1\backslash H\backslash R\backslash K) \)
   \( K \)- Secret key shared by two-hop node and source.
4. Before forwarding the packet increment the hop count by 1 and copy \( H1 \) to \( H2 \) and calculate the new \( H1 \). i.e. \( H1=H(S\backslash L+1\backslash H+1\backslash R\backslash Ki) \);
   \( H2=H(S\backslash L+1\backslash H\backslash R\backslash K) \) where \( Ki \) - common secret key between intermediate node and two hop node.
5. Forward the Routing packet to its Multicast group.
6. On receiving \{S, L+1, H+1, R, H2\} nodes within the group can use \{S, L+1, H+1, R\} and public hash function to calculate H(S\L+1\H\R\K).

7. Compare this value with H2 and validate whether routing packet is modified by intermediate node.

8. If intermediate node wants to modify the packet it has to forge the H2 value before forwarding the packet.

9. Finally, at the destination, HMAC value is calculated and compared with H3 value in the received packet.

The same concept is applied for RREP from destination to source.

4.4 SIMULATION ENVIRONMENT

The simulation environment to carry out the triple hash authentication mechanism in the protocols AODV and DSR is similar to that of Dual hash authentication as mentioned in section 3.3 of the Chapter 3.

4.5 PERFORMANCE ANALYSIS – THT Vs AODV

4.5.1 Packet Delivery Ratio

Table 4.1 shows Packet Delivery Ratio of AODV, Triple Hash Technique. The packets may travel through malicious node in AODV. Hence the delivery ratio of THT is better than that of AODV. The packet delivery ratio of THT is 2% higher than that of AODV.
Table 4.1 Nodes Vs Packet Delivery Ratio

<table>
<thead>
<tr>
<th>Number of nodes</th>
<th>Packet Delivery Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AODV</td>
</tr>
<tr>
<td>10</td>
<td>94.58</td>
</tr>
<tr>
<td>15</td>
<td>86.45</td>
</tr>
<tr>
<td>20</td>
<td>86.01</td>
</tr>
<tr>
<td>25</td>
<td>88.27</td>
</tr>
<tr>
<td>30</td>
<td>87.12</td>
</tr>
<tr>
<td>40</td>
<td>85.97</td>
</tr>
<tr>
<td>50</td>
<td>83.14</td>
</tr>
<tr>
<td>60</td>
<td>83.9</td>
</tr>
</tbody>
</table>

4.5.2 Control Overhead

Figure 4.1 shows the control overhead with respect to number of nodes. When the number of active mobile nodes in a network increases, the control overhead also increases in both AODV and AODV with THT. The control overhead is decreased by 9% in AODV-THT when compared to AODV.

![Figure 4.1 Effect of No. of nodes on Control Overhead](image-url)

Figure 4.1 Effect of No. of nodes on Control Overhead
4.5.3 Packet Loss

Table 4.2 shows the effect of pause time on Packet Loss. Packet Loss decreases as the pause time increases. The loss of packets decreases by 0.4% in THT when compared to AODV.

Table 4.2 Pause Time Vs Packet Loss

<table>
<thead>
<tr>
<th>Pause Time</th>
<th>Packet Loss</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>AODV</td>
<td>THT</td>
</tr>
<tr>
<td>0</td>
<td>842</td>
<td>780</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>730</td>
<td>640</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>619</td>
<td>582</td>
<td></td>
</tr>
<tr>
<td>300</td>
<td>552</td>
<td>538</td>
<td></td>
</tr>
<tr>
<td>400</td>
<td>513</td>
<td>492</td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>470</td>
<td>449</td>
<td></td>
</tr>
<tr>
<td>600</td>
<td>435</td>
<td>411</td>
<td></td>
</tr>
<tr>
<td>700</td>
<td>398</td>
<td>372</td>
<td></td>
</tr>
</tbody>
</table>

4.5.4 Throughput

Figure 4.2 shows Nodes Vs Throughput for AODV and THT. The figure shows that throughput has been increased when the number of nodes is increased. The throughput has been increased by 1% in the case of THT when compared to AODV.
Figure 4.2 Nodes Vs Throughput

4.5.5 Route Acquisition Time

Figure 4.3 Nodes Vs Route Acquisition Time

The above figure 4.3 shows that route acquisition time is increasing when the number of nodes increases. Here, the route acquisition time has been decreased by 4% in the case of AODV with THT when compared to AODV when the number of nodes is increased from 10 to 70.
4.6 PERFORMANCE ANALYSIS – THT Vs DSR

4.6.1 Packet Delivery Ratio

Figure 4.2 shows Packet Delivery Ratio of DSR and Triple Hash Technique. The packets may travel through malicious node in DSR and hence the delivery ratio of THT is better than that of DSR. The packet delivery ratio of THT is 4% more than that of DSR.

![Packet Delivery Ratio Graph]

**Figure 4.4 Nodes Vs Packet Delivery Ratio**

4.6.2 Control Overhead

Figure 4.3 shows the Node Vs Control Overhead for DSR and Triple Hash Technique. Here the number of control packets increases when the number of nodes increases. Triple Hash Technique has 6% less control packets compared to that of DSR due to the secure algorithm.
Figure 4.5 Nodes Vs Control Overhead

4.6.3 Packet Loss

Figure 4.4 shows the Pause Time Vs Packet Loss for DSR and Triple Hash Technique. When the pause time increases, the node movement reduces from one place to other. Due to this, the packet loss is reduced. Triple Hash Technique has reduced packet loss by 5% compared to DSR.
4.6.4 Throughput

Figure 4.5 shows Nodes Vs Throughput for DSR and THT. The throughput has been increased moderately when the number of nodes is increased. The throughput has been increased by 1% only for DSR with THT when compared to DSR.

![Figure 4.7 Number of nodes Vs Throughput](image)

4.6.5 Route Acquisition Time

The route acquisition times for DSR and THT are shown in the figure 4.6. The route acquisition time for DSR is less by 2% compared to DSR-THT when the number of nodes is increased from 10 to 70.
4.7 SUMMARY

In this chapter, another security mechanism namely Triple Hash Authentication is developed to have secure routing. The THT has been applied to the routing protocols AODV and DSR. The performance of these protocols has been analysed using simulation. The performance of the DSR protocol is better than that of AODV for the parameters Packet Delivery Ratio, Packet Loss, Throughput and Route acquisition time. The performance of AODV is better in terms of control overhead.