Network Security Assignment-5

1) **a)** The protocol has the following properties:
- Provides Perfect Forward Secrecy
- Provides Escrow foilage against passive attacks
- Provides Escrow foilage against active attacks
- No Identity hiding
-No PFS for Identity hiding.

b)
-Provides PFS
-Provides Escrow foilage against passive attacks
-No escrow foilage against active attacks
- Provides identity hiding
- No PFS for identity hiding.

c)
-Provides PFS
-Provides Escrow foilage against passive attacks
-Provides escrow foilage against active attacks (unless signature key is escrowed)
-Provides identity hiding
-Provides PFS for identity hiding, an active attacker can discover Alice's identity.

d)
-Provides PFS
-Provides escrow foilage against passive attacks
- No escrow foilage against active attacks
- No identity hiding.
-No PFS for identity hiding

e) To establish mutual authentication, Alice sends a message to Bob that includes a message "Alice" and a message authentication code (MAC) that is calculated using S as the shared key. Bob can verify Alice's identity by verifying the MAC using the shared key S. Similarly, Bob can send a message to Alice that includes his identity "Bob" and a MAC that is calculated using S as the shared key. Alice can verify Bob's identity by verifying the MAC using the shared key S. The use of a new shared secret each time the protocol is run provides Perfect Forward Secrecy (PFS), since if one shared secret is compromised, the attacker will not be able to use it to decrypt previous or future communications.

f)
Alice and Bob share a secret key K. Alice selects a random number, r, and computes R = r * G, where G is a number known to everyone. Alice sends R to Bob.
Bob selects a random number s and computes S = s * G. Bob sends S to Alice.
Alice computes H = Hash(R, S, K, ‘Alice’), where Hash is a hash function; Alice sends H to Bob.
Bob computes H' = Hash(R, S, K, ‘Bob’) and verifies that H' matches the value received from Alice. If the verification fails, Bob aborts the protocol.
Bob computes T = r * S and sends T to Alice.
Alice computes T' = s * R and verifies that T' matches the value received from Bob. If the verification fails, Alice aborts the protocol.
Alice and Bob can now use K to exchange messages securely.

g)
If Bob changes his secret periodically and uses a stateless crypto cookie mechanism, he can ensure that a connection attempt will still succeed even if he changed his secret between the time the initiator asked for the cookie and returned, by including a timestamp in the crypto cookie. When the initiator returns the crypto cookie to Bob, Bob can verify that the timestamp is within an acceptable time range based on the maximum time difference between his and the initiator's clock.

2)

a) Yes, it is possible to encapsulate AH packet within ESP packet and vice versa.
i) If AH is encapsulated within ESP, then the packet is first applied with AH protocol and this ensures authentication of the packet. If ESP protocol is then applied on it, the confidentiality of the newly formed packet is ensured. Vice versa, if ESP is encapsulated within AH packet. First the packet is being applied with ESP which ensures confidentiality and the newly formed packet is benign applied with AH protocol which ensure authentication and data integrity.

b)No, because the retransmitted packet is considered as new IPsec.

c)

(i) A to F1:

IP header: SOURCE=A, DESTINATION=F1

TCP header: SOURCE=A, DESTINATION=B

TCP payload: data being transmitted

IPsec header: inserted by A before the TCP header, including the ESP or AH protocol, the SPI (Security Parameter Index), and other security-related fields

NEXT HEADER: set to TCP

(ii) F1 to F2:

IP header: SOURCE=F1, DESTINATION=F2

IPsec header: inserted by F1 before the IP header, including the ESP or AH protocol, the SPI, and other security-related fields

NEXT HEADER: set to IP

Inner IP header: the original IP header from A to B, with SOURCE=A and DESTINATION=B

TCP header: included in the inner IP header

TCP payload: included in the inner IP header

(iii) F2 to B:

IP header: SOURCE=F2, DESTINATION=B

IPsec header: inserted by F2 before the inner IP header, including the ESP or AH protocol, the SPI, and other security-related fields

NEXT HEADER: set to TCP

Inner IP header: the original IP header from A to B, with SOURCE=A and DESTINATION=B

TCP header: included in the inner IP header

TCP payload: included in the inner IP header

Note that the specific contents of the IPsec header (e.g. ESP or AH protocol, SPI) will depend on the configuration of the IPsec implementation being used.

d)
IPsec is not firewall friendly because of the encrypted payload and, as a result, cannot make decisions based on the source, destination, or content of the packet.

e)

When a TCP/IP segment/packet is sent from a host inside a private network through a NAT to a public web server, the NAT changes the following fields in the TCP header Destination port number.

f)
NAT is unable to modify the source IP address and source port number in an IPSec ESP packet because the packet is encrypted, and NAT cannot read its contents.

3)
The reason why SSL uses a sequence number with each record for HMAC computation, even though TCP delivers data in the correct order, is to protect against certain types of attacks, such as replay attacks. The sequence number provides an extra layer of protection to ensure that data is delivered in the correct order and is not tampered with or replayed.

4)
a)
Building security at the TCP layer has several advantages, such as utilizing TCP's existing handshake mechanism for key exchange and security parameter negotiation, allowing for flexible encryption on a per-connection basis, and providing end-to-end security. TCPsec is also backward compatible with regular TCP and interoperable with NATs. The paper explores two approaches for implementing TCPsec, one using application layer proxies and the other using a kernel sandboxing framework. The authors implemented TCPsec in the FreeBSD 4.7 kernel and found that it incurs only a modest overhead compared to TCP and performs competitively with SSL. The paper discusses the design, implementation, deployment problems, performance evaluation, formal verification, related work, and conclusion of TCPsec.

b)

TCPsec allows for flexible encryption of the TCP segment, where an application can choose to encrypt only a part of the TCP header or the entire TCP segment. However, it is mandatory that the port numbers of the TCP header are left unencrypted for identifying the TCP data structures and the destination application. By default, the header is left unencrypted, and only the payload (user data) is encrypted. Other details of the data handshake phase, including the choice of explicit versus explicit initialization vectors and the placement of the message authentication code, are described in the detailed version of the paper.

c)

TCPsec includes a new option called the NAT option in the TCPsec handshake to provide interoperability with Network Address Translation (NAT) devices. When a client sends the first SYN message of TCPsec, it includes the NAT option containing the port numbers and the source IP address used for the connection as part of the TCPsec options. If there is a NAT along the path to the server, the port numbers and IP address are modified, and the TCPsec server verifies this by comparing the received values with those in the option. When a NAT is detected, the server replies with the NAT option containing the source address and port number it uses. T

d)

TCPsec generates the keyed message digest over the encrypted data, which provides a secure solution compared to SSL's encryption after authentication. TCPsec uses explicit sequence numbers, which are secure as they can't be easily deduced by observing the data stream. Additionally, TCPsec uses explicit IVs, which are more secure than SSL's implicit IVs. While TCPsec has some vulnerabilities, such as the increased size of the syncache data structure, it is still secure in its design. In a measurement of handshake latency, TCPsec's handshake is significantly faster than SSL's handshake. In terms of data throughput, TCPsec performs slightly better than SSL in two networks and 30% better in one network