IK2206 – Internet Security and Privacy

Chapter 16 – Real-time Communication Security

16.1 What layer? (READ)

What could be the advantage/disadvantage of realizing security at the network layer (as IPSec does) or above the transport layer (as SSL/TLS does)?

IPSec (layer 3)

Pro:	Cons
- All applications to be protected without the applications having to be modified - Each packet is cryptographically protected -> better protected against denial-of-service attack	- IPSec cannot tell the application anything more than which IP address is on the other end (most apps need to distinguish between users)

SSL, TLS or SSH (layer 4)

Pro:	Cons
- Not necessary to modify the operating system - Modifying an application to work on top of SSL requires minimal changes	- Denial-of-service attack possible (see in the comment for the next question)

Why is SSL/TLS sensitive to an attack where Trudy inserts packets into the TCP stream?

There is a problem operating above TCP. Since TCP will not be participating in the cryptography, it will have no way of noticing if malicious data is inserted into the packet stream, as long as the bogus data passes the (non-cryptographic!) TCP checksum. TCP will accept such data and forward it up to SSL. SSL will discard it because the integrity check will indicate the data is bogus. It is not possible to resend the packet, because TCP will discard it, if it has the same sequence number.

16.2 Session Key Establishment (READ)

What is "session hi-jacking"? In what situations could such attacks be launched?
Session hi-jacking is the action of taking over Alice’s session to Bob by forging Alice’s IP address as the source address on packets sent to Bob, and using TCP sequence numbers lager than what Alice would be using.

This could happen if the conversation is not protected after the initial handshake.

To avoid replay attacks a sequence numbers should be added to each packet. The sequence number should be unique for each session key. What happens if the sequence number wraps around during a session? How would you design your protocol to address this situation?
A new session key should be established for each new session, and during a session, if the sequence number might be reused (i.e. wrap around)

What may happen if the random number generator used to generate session keys have flaws? (Read about the SSL example.)
Someone will be able to impersonate one side or other on behalf of a replay attack.

Why should both parties "contribute" to the session key?

It is then less likely that the protocol will have flaws in which someone will be able to impersonate one side or the other in a replay attack.

Furthermore, there is a higher chance to generate a sufficiently random session key, if either side has a good random number generator.

16.3 Perfect Forward Secrecy (READ)

What is Perfect Forward Secrecy?
PFS is preserved, if it is impossible for an eavesdropper to decrypt a conversation between Alice and Bob even if Sam records the entire encrypted session, and steals their long-term secrets.

Perfect Forward Secrecy (PFS) refers to the notion that compromise of a single key will permit access to only data protected by a single key. For PFS to exist the key used to protect transmission of data MUST NOT be used to derive any additional keys, and if the key used to protect transmission of data was derived from some other keying material, that material MUST NOT be used to derive any more keys. (http://en.wikipedia.org/wiki/Perfect_forward_secrecy)

Can you give an example how to accomplish PFS? (see fig. 16-2)
In the first two messages, each side identifies itself, and supplies a Diffie-Hellman value signed by its private key. In the next two messages, each side proves knowledge of the agreed-upon Diffie-Hellman value by sending a hash of it, with each side sending a different hash. If each side forgets its private Diffie-Hellman number after the session, there is no way for anyone to reconstruct their conversation. It is even not possible if the attack obtains the long-term private keys and the entire recorded conversation.

Can you give any examples of protocols not providing PFS (The book gives three examples; can you explain why they don't provide PFS?)

- Kerberos (since the session key is inside the ticket to Bob and encrypted with Bob’s long term key)
- Alice chooses the session key, and sends it to Bob, encrypted with Bob’s public key.

16.4 PFS-Foilage (READ BRIEFLY)

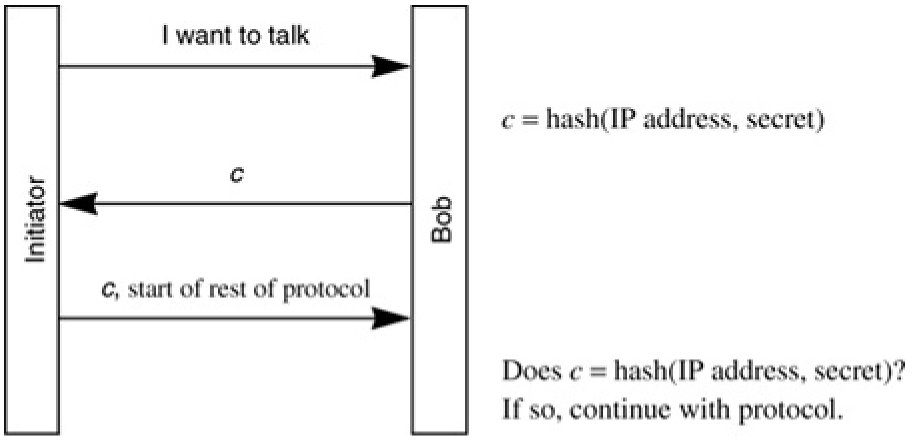
16.5 Denial-of-Service/Clogging Protection (READ)

How can "cookies" (as proposed in Photuris) help defeating DoS attacks based on "IP spoofing"? Can you explain the protocol in figure 16-3?
We can assure if the initiator can receive packets sent to the IP address from which it claims to be transmitting. If an initiator starts to talk with Bob, then Bob will send him a cookie (a number) in clear. The initiator only sends back the received cookie. Bob has to remember what cookie he sent for each request until he receives the answer.

As by RFC-2522, the responder (usually the server; in our case "Bob") has only to remember a maximum of 254 concurrent cookies of a particular sender. (http://tools.ietf.org/html/rfc2522). This measure helps preventing DoS attacks.

If the attacker is using a legitimate IP address (i.e., no IP spoofing), would the use of "cookies" help?

What's a "stateless cookie"?
The responder, Bob, does not have to remember (store) the cookies he sent out. The cookie is a function of the IP address and a secret known to the responder.


Initiator
I want to talk
c = hash(IP address, secret)
c
Bob
c, start of rest of protocol
Does c = hash(IP address, secret)?
If so, continue with protocol.

Adding a "cookie"-scheme to a protocol implies an additional network round-trip, which may have a negative impact on performance. What suggestion does the book provide so that cookies are only used when necessary?

Use cookies only when a node is getting swamped.

16.6 Endpoint Identifier Hiding (READ)

Can you think of a situation where it's valuable to hide the identities of the endpoint(s)?
Hidden identities give less information for active attackers about their target. **(???)**

It is generally difficult to hide both identities (initiator and responder) from an active attacker. If one has to choose, what arguments would be in favour of hiding the initiator's and responder's identity respectivly?

One argument says that it is better to hide the initiator's identity because the responder's identity is already known.

But a different argument says that it is better to hide Bob's identity. If Bob divulges his identity first, then anyone can imitate a connection to Bob and get him to divulge his identity.

16.7 Live Partner Reassurance (READ)

What mechanism(s) could be used to defeat replay attacks?

Trudy tries to resend actions from Alice in direction of Bob.

Bob should choose a nonce for each connection attempt, and have the session key be a function of the nonce as well as the Diffie-Hellman key.

16.8 Arranging for Parallell Computation (READ)

* This section gives an example of how the performance of a protocol can be improved. How?

(To share a Diffie-Hellman key take a long time. How we can speed up?)

Sending an extra message in order to allow the computation-intensive calculations can be done in parallel.

16.9 Session Resumption (READ)

* What is the benefit of supporting "session resumption" in a protocol (e.g., SSL/TLS) ? What kind of cryptographic operations can (commonly) be skipped when "session resumption" is possible?
The initial expensive public key authentication can be skipped if the server has recently authenticated the client and established a shared secret session key.

Why is it desirable to design "session resumption" to be stateless?
Whenever a server has to keep a state, it becomes subject to DoS attacks.

16.10 Plausible Deniability (READ BRIEFLY)

16.11 Data Stream Protection (READ BRIEFLY)

16.12 Negotiating Crypto Parameters (READ)

When negotiating crypto parameters it is important to avoid "downgrade attacks". What is a "downgrade attack"? (The attack is discussed in this section of the book, but it's not called "downgrade attack".)
It is fashionable today for security protocols to negotiate cryptographic algorithms, rather than simply having the algorithms be part of the specification of the protocol.

Alice and Bob would like to agree on the strongest possible cryptography that they both support. Trudy might be able to trick Alice and Bob into using weaker cryptography by removing from Alice's suggestion the ones that Trudy can't break.

When negotiating crypto parameters, IKE(v1) and TLS/SSL made different design choices. Compare these approaches when you read about it in later sections. Which approach do you prefer?

IKE separately negotiates algorithms for encryption, integrity, authentication etc.

IKE phase 1's purpose is to establish a secure authenticated communication channel by using the Diffie–Hellman key exchange algorithm to generate a shared secret key to encrypt further IKE communications. This negotiation results in one single bi-directional ISAKMP Security Association (SA). The authentication can be performed using either pre-shared key (shared secret), signatures, or public key encryption. Phase 1 operates in either Main Mode or Aggressive Mode. Main Mode protects the identity of the peers; Aggressive Mode does not.

During IKE phase 2, the IKE peers use the secure channel established in Phase 1 to negotiate Security Associations on behalf of other services like IPsec. The negotiation results in a minimum of two unidirectional security associations (one inbound and one outbound). Phase 2 operates only in Quick Mode.

SSL/TLS' negotiation is much simpler. Every possible combination of cryptographic algorithms is a predefined suite. This is less flexible, but in reality there is no reason to support more than a few suites.