IPsec and Virtual Private Networks (VPNs) for Network-Layer Security: Part 3

Gaurav S. Kasbekar

Dept. of Electrical Engineering

IIT Bombay

References

- J. Kurose, K. Ross, "*Computer Networking: A Top Down Approach*", Sixth Edition, Pearson Education, 2013
- A. Tanenbaum, D. Wetherall, "*Computer Networks*", Fifth Edition, Pearson Education, 2012.
- L. Peterson, B. Davie, "*Computer Networks: A Systems Approach*", Fifth Edition, Morgan Kaufmann, 2012.
- B.L. Menezes, R. Kumar, "*Cryptography, Network Security, and Cyber Laws*", Cengage Learning India Pvt. Ltd., 2018
- W. Stallings, "*Cryptography and Network Security: Principles and Practice*", Pearson Education, 7th edition, 2016

IPsec SA Establishment Using the Internet Key Exchange (IKE) Protocol

IPsec SA Establishment using the Internet Key Exchange (IKE) Protocol

- Recall: to establish an IPSec SA from router R1 to router R2, they need to:
  - ❑authenticate each other,
  - ❑agree on the encryption and message integrity algorithms and keys and Security Parameter Index (SPI)
- Done using IKE protocol
- Has two phases

Internet Key Exchange (IKE) Protocol: Overview

- In phase 1:
  - the two sides use Diffie-Hellman to create a bi-directional **IKE SA** between the routers (different from the IPsec SAs discussed earlier)
  - the IKE SA provides encryption and message integrity between the two routers
  - then both sides reveal their identity to each other. However, the identities are not revealed to a passive sniffer, since the messages are sent over the secured IKE SA channel
- In phase 2:
  - the two sides create an IPsec SA in each direction by negotiating the IPsec encryption and message integrity algorithms to be employed by the IPsec SAs and establishing the encryption and message integrity session keys for the two SAs
- The two sides can then use the IPsec SAs to securely send packets, as discussed earlier

IKE Protocol (contd.)

- Recall: when IKE protocol is used, the identities of the two parties are not revealed to a passive sniffer
- One form of identification is the IP address
  - IP addresses of the two parties are known to intruder since they are included in IP headers in unencrypted form
- The two parties may (and usually do) use an alternative form of identification such as an email address
- IKE protects the confidentiality of such alternative forms of identification using encryption
- Why are two phases used in IKE protocol?
- It is good cryptographic practice to periodically change cryptographic keys used by the two communicating parties
- Hence, multiple IPsec SA instances are generated over time (which use different cryptographic keys)
- Reason for using two phases in IKE protocol:
  - to save on computational cost
  - since phase 2 does not involve any public key cryptography, it is much faster than phase 1
  - in phase 1, long-term keys are derived; later, phase 2 is executed from time to time to generate new IPsec SAs with new short term keys; these short term keys are functions of the long term keys computed in phase 1 and nonces exchanged in phase 2

Clogging Attack

- Recall: Diffie-Hellman algorithm:
  - Alice and Bob select secret numbers, say S_A and S_B, respectively
  - Alice computes $T_A = g^{S_A} \bmod p$; Bob computes $T_B = g^{S_B} \bmod p$
  - Alice sends T_A to Bob and Bob sends T_B to Alice
  - Alice computes ${T_B}^{S_A} \bmod p$ and Bob computes ${T_A}^{S_B} \bmod p$
  - Thus, both Alice and Bob agree on the same number $g^{S_A S_B} \bmod p$ (which is the shared key)
- In the clogging attack, an intruder forges the source address of a legitimate user, say B, and sends a public Diffie-Hellman key, say T, to a victim, say A
- Victim A then performs modular exponentiation, i.e., finds $T^{S_A} \bmod p$ to compute the secret key
  - this is a computationally expensive operation
- Repeated messages of this type can clog the victim's system with useless work
- To defend against this attack:
  - **cookies** are used in IKE protocol

Use of Cookie for Defending Against Clogging Attack

- Suppose A sends the first message of the IKE protocol to B
- To defend against clogging attack, B computes and includes a cookie, say C_B, in its response to A
- Cookie, C_B, computed by B is:
  - a 64-bit integer
  - a hash function of several variables including the IP address of A, a secret known only to B and the current time at B
- Value of cookie C_B is different for different IP addresses of the initiator A
- A is required to send the cookie C_B to B in all subsequent messages from A to B (including the message containing A's Diffie-Hellman public key)
- On receipt of a message from A, B checks whether the cookie corresponds to A's IP address
  - if the check fails, B aborts session establishment
  - hence, avoids performing the expensive modular exponentiation step of Diffie-Hellman
- Attacker can only obtain the cookie C_B if:
  - it sniffs the channel between B and A, which is relatively hard to do
- Thus, the use of cookie defends against the clogging attack

IKE Phase 1

- The following cases may arise:

1) A and B share a secret key, say s (possibly derived from a password)
2) A and B have a public-private key pair and certificates

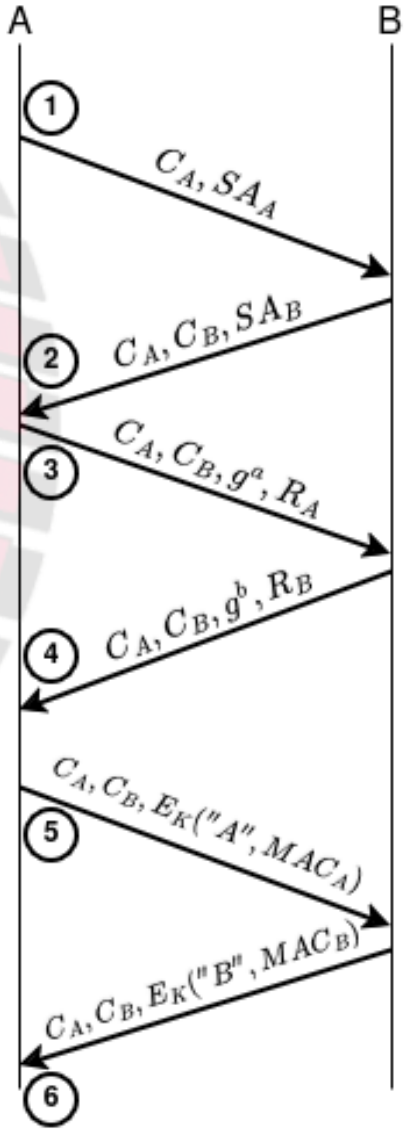
- We will study Case 1) in detail
- Case 2) is similar
- Assume that A initiates the IKE protocol and B is the responder

IKE Phase 1 When A and B Share a Secret Key s

1) A sends the cryptographic algorithms proposed by A for use in the IKE SA, say SA_A, and its cookie, C_A
   - ❑ the pair (C_A, C_B) serves the purpose of an IKE connection identifier (similar to SPI in IPsec SAs)
   - ❑ included in every message starting from second message
2) B responds with the cookie C_B, and the cryptographic algorithms, say SA_B, selected by B
3) A sends a nonce, say R_A, and its public DH key, say $g^a \bmod p$
4) B sends a nonce, say R_B, and its public DH key, say $g^b \bmod p$

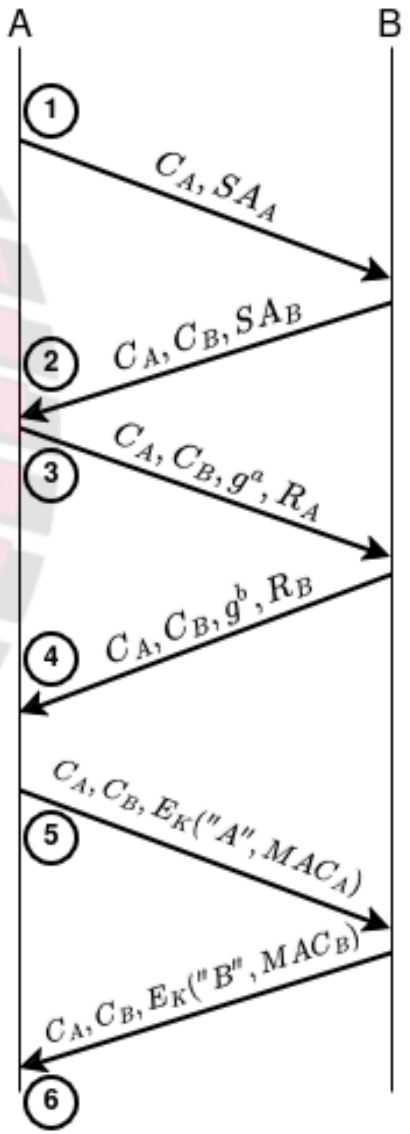
- Then both A and B independently compute separate keys for encryption and MAC computation; these keys are functions of C_A, C_B, R_A, R_B, s and $g^{ab} \bmod p$

5) A sends its identity, "A", and a MAC, MAC_A, to B after encrypting it; MAC_A is function of $C_A, C_B, g^a \bmod p$, $g^b \bmod p, SA_A, SA_B, R_A, R_B$ and s
6) Similarly, B sends its identity, "B", and a MAC, MAC_B, to B after encrypting it



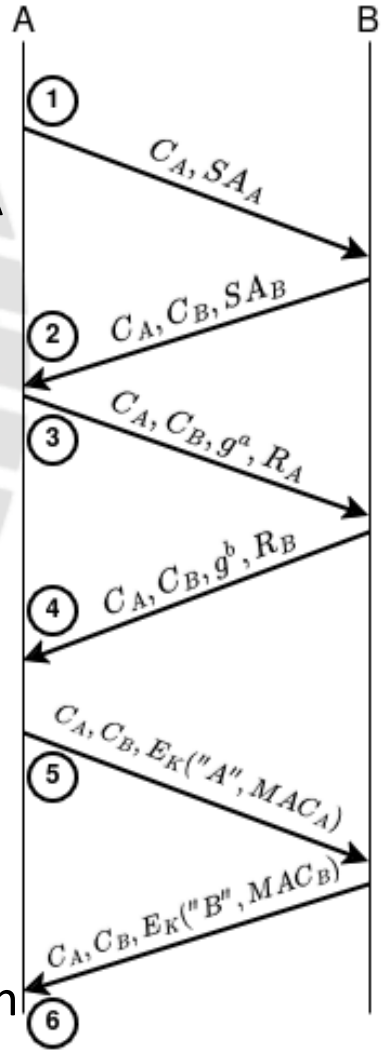
IKE Phase 1 When A and B Share a Secret Key s (contd.)

- Is the Diffie-Hellman algorithm used in above protocol vulnerable to man-in-the-middle attack?
  - ❑ No, since MACs sent in steps 5 and 6 are functions of $g^a \bmod p$, $g^b \bmod p$, and also s, which is unknown to intruder
- Why are nonces R_A and R_B used?
  - ❑ To protect A and B against connection replay attack
- Why are the MACs sent in steps 5 and 6 also functions of C_A, C_B, SA_A, SA_B?
  - ❑ To detect modification in these values by intruder on path between A and B
- Recall: encryption and MAC computation keys are functions of C_A, C_B, R_A, R_B, s and $g^{ab} \bmod p$. Advantage of this protocol over protocol that skips Diffie-Hellman and uses keys that are functions of only C_A, C_B, R_A, R_B, s:
  - ❑ former protocol provides *forward secrecy*



IKE Phase 1 When A and B Share a Secret Key s (contd.)

- There is a major drawback with this option in which A and B share a secret key s
- When B receives message 5, it must first decrypt the message
  - ❑ But for this, B needs to know the secret s that it shares with A
  - ❑ To know s, B needs to know identity of A, “A”
  - ❑ However, “A” is itself encrypted
- One option: use source IP address to identify sender
- Shortcoming:
  - ❑ if source IP address reveals sender’s identity, then there was no point in protecting sender’s identity, “A”, using encryption
- Better option:
  - ❑ B keeps track of all entities that may communicate from a given source address
  - ❑ then when message 5 arrives from a particular source address, B attempts to decrypt the message using secrets it shares with each entity at that IP address
  - ❑ successful decryption would happen if key used for decryption matched that shared by the entity whose ID appeared in message 5



IKE Phase 2

- Recall: after IKE phase 1, a channel that provides encryption and message integrity (IKE SA) has been created between A and B
- In phase 2, the two sides create an IPsec SA in each direction by negotiating the IPsec encryption and message integrity algorithms to be employed by the IPsec SAs and establishing the encryption and message integrity session keys for the two IPsec SAs
- The two sides can then use the IPsec SAs to securely send packets, as discussed earlier
- IKE phase 2 is executed from time to time to create new IPsec SAs
  - since good cryptographic practice to change keys from time to time