CSE 3400/CSE 5850 - Introduction to Cryptography and Cybersecurity
/ Introduction to Cybersecurity

Lecture 9
Shared Key Protocols – Part II

Ghada Almashaqbeh
UConn

Adapted from the Textbook Slides

Outline

- Handshake protocol extensions.
- Key distribution centers.
- Improving resiliency to key exposure.

Handshake Protocol Extensions

Authenticated Request-Response Protocols

- Beside authenticating entities, these protocols authenticate the exchange of a request and a response between the entities.
- Required properties:
  - **Request authentication.**
    - The request was indeed sent by the sender.
  - **Response authentication**
    - The response was indeed sent by the receiver (to which the request was intended).
  - **No replay.**
    - Every request/response was received at most the number of times it was sent by the sender.

Authenticated Request-Response Protocols

- Five variants:
  - 2PP-RR
  - 2RT-2PP
  - Counter-based-RR
  - Time-based-RR
  - Key-exchange

2PP-RR

- A three-flow nonce-based protocol.
- Significant drawback:
  - The request is sent by the responder and the initiator sends the response.
  - So initiator must wait for a request rather than sending it!!

2RT-2PP

- A four-flow nonce-based protocol.
- Mainly fixes the drawback of 2PP-RR discussed in the previous slide.

Counter-Based Authenticated RR

- **Simple stateful (counter) solution, requiring only one round:**
  - Unidirectional (if bidirectional is needed, a separate instance of the protocol for each direction needs to be executed).
  - Parties maintain synchronized counter i of requests (and responses) to avoid replay attacks.
  - Recipient (e.g., Bob) validates counter received is $i + 1$
  - Both parties must remember counter

Alice (requester)
Init: $i_{A,sent}, i_{A,rec} \leftarrow 0$

Bob (responder)
Init: $i_B \leftarrow 0$

$i_{A,sent} \leftarrow i_{A,sent} + 1$

$req_{,A,sent}, MAC_k(1 \parallel \text{'}A \rightarrow B\text{'} \parallel i_A \parallel req)$ →

If $i_{A,sent} \neq i_B + 1$: ignore
Else; $i_B \leftarrow i_B + 1$

← $resp, i_B, MAC_k(2 \parallel \text{'}A \leftarrow B\text{'} \parallel i_B \parallel resp)$

If $i_{A,rec} \neq i_B - 1$: ignore
Else: accept $resp$ and set $i_{A,rec} \leftarrow i_{A,rec} + 1$

Time-Based Authenticated RR

- **Simple stateful (time) solution, requiring only one round:**
  - Use local clocks T_A, T_B instead of counters with two assumptions: bounded delays and bounded clock skews.
  - Responder (Bob):
    - Rejects request if: $T_B > T_A + \Delta$ OR if he received larger T_A already
      - Where $\Delta \equiv \Delta_{skew} + \Delta_{delay}$
    - Maintains last T_A received, until $T_A + \Delta$
  - Initiator (Alice) does not need **any** state, when can Bob discard his?

Remember T_A

2RT-2PP with Confidentiality

- **Secure connection: authentication, freshness, secrecy**
  - Independent keys: for encryption $k.e$, for authentication: $k.a$
  - How can we derive them both from a single key k ? The PRF idea from before:
    - $k.e = PRP_k(\text{“Encrypt”}),\ k.a = PRP_k(\text{“MAC”})$
  - Hmm… same key encrypts all messages, in all sessions ☹
- Can we improve security, by changing keys, e.g., between sessions ?

2PP Key Exchange Protocol

- Allows generating independent session keys, i.e., each session will have its own key denoted as k_i^S.
- These keys are derived from the shared key that we now call a long-term 'master key' denoted as k^M.
- Improves security:
  - Exposure of a session key does not expose the master key, and does not expose keys of other sessions.
  - Thus, limited amount of ciphertext exposed if a session key is exposed (only that session rather than all ciphertexts of all sessions).

$k_i^S = PRF_{k^M}(N_{A,i} \parallel N_{B,i})$ $\qquad$ $k_i^S = PRF_{k^M}(N_{A,i} \parallel N_{B,i})$

A PRF is used instead of a MAC, is that ok?

Key Distribution Centers (KDCs)

Establish a shared key between two or more entities, usually with the help of a trusted third party referred to as KDC

Key Distribution Center (KDC)

- Will focus on three party protocols; Alice, Bob, and KDC.
- KDC: shares long-term master keys with all parties (each party will have a key for encryption and another for MAC).
  - denoted as $k_A^M, k_B^M, \ldots$ for MAC and $k_A^E, k_B^E, \ldots$ for encryption.
- Goal: help parties (A, B) to <u>establish</u> a shared master key k_{AB}
  - Based on which the parties generate two keys for MAC and encryption, namely, k_{AB}^M and k_{AB}^E
- We will study two protocols; simplified versions of:
  - The Kerberos protocol (secure) widely used in computer networks.
  - The GSM protocol (insecure) used by cellular networks.

The Kerberos KDC Protocol

- KDC shares keys k_A^E (enc.), k_A^M (MAC) with Alice and k_B^E, k_B^M with Bob
- Goal: Alice and Bob share k_{AB}^M, then derive: k_{AB}^E, k_{AB}^M
- KDC performs access control as well; controlling whom Alice can contact.

The GSM Handshake Protocol

- Mobile client
  - Identified by i (IMSI: International Mobile Subscriber Identifier)
- Visited network (aka Base station); not fully trusted
- Home network; trusted, shares key k_i with client i

A38: derive secret, random K_c, s , from K_i and r.

GSM spec: OWF, but really should be a PRF!

Example – Sending two messages

(… and so on for more messages)

K_c is the session key
s is a secret authenticator

A5: provide 'pad' for encryption

Several variants:
A5/1 - `regular'
A5/2 - `weak'
A5/3 – more secure

Really should be a PRF!

ECC: error correcting code. Used to allow recovery from errors.

Attacks on GSM Handshake Protocol

- We will explore two attacks:
  - Visited network impersonation replay attack.
    - We will study this one in detail.
  - Ciphersuite downgrade attack.
    - Only high level description.

Visited-network Impersonation Attack

Note: does NOT Impersonate **mobile,** only Visited network.

In the cryptanalysis phase, the attacker exposes Kc based on the cyphertexts it collected in the eavesdropping phase (recall A5/1 and A5/2 are not secure).

In the impersonate phase, the attacker will send the same r and s from before (replay attack), which will lead to the same Kc he obtained in the cryptanalysis phase.

GSM Ciphersuites Downgrade Attack

- A ciphersuite is the set of cryptographic schemes used in a protocol execution.
- Ciphersuite negotiation:
  - Mobile sends a list of cipher-suites it supports
  - Visited-network selects best one (the strongest/most secure) that it also supports
  - Goal of negotiation is to support interoperability between devices of different capabilities.
- GSM encryption algorithms E_k:
  - A5/0: none, A5/1: broken, A5/2: useless (break with only 1sec), A5/3: 'other'
- A MitM attacker may trick these parties to use a weak suite although the parties can support a stronger one.
  - It works due to key reuse in GSM (same key is used across various encryption schemes).
- For full details, see the textbook.

Improving Resiliency to Key Exposure

Forward Secrecy

- **So far:** session key $k_i^S \not\Rightarrow k_j^S$ (expose no other keys)
  - And master key was fixed for all sessions
- Idea: we can do better!
  - Change the master key each session: $k_1^M, k_2^M, \ldots$
- **Forward Secrecy (FS):** <u>master</u> key $k_i^M \not\Rightarrow k_j (j < i)$
  - I.e., k_i^M(and k_i^S) don't expose keys, communication of <u>previous</u> sessions $(j < i)$

Forward Secrecy 2PP Key Exchange

- This protocol generates a different master key for each session (or period) i denoted as k_i^M
- The initial master key shared between the parties is k_0^M

$$k_i^M = PRF_{k_{i-1}^M}(0)$$
$$k_i^S = PRF_{k_i^M}(N_{A,i} \parallel N_{B,i})$$

$$k_i^M = PRF_{k_{i-1}^M}(0)$$
$$k_i^S = PRF_{k_i^M}(N_{A,i} \parallel N_{B,i})$$

Forward Secrecy 2PP Key Exchange

- This protocol produces unidirectional master keys:

$$k_i^M \Rightarrow k_{i+1}^M \text{ but } k_{i+1}^M \not\Rightarrow k_i^M$$

- Exposing a session master key does not impact prior sessions.
  - But future sessions will be exposed!

Recover Security (RS)

- Can we **recover** security? So if k_i^M is exposed, yet future sessions can remain secure?
  - Idea: assume **no attack** during a 'recovery session'.
- Thus, recover security means that ***a single session*** without eavesdropping or other attacks suffices to recover security from previous key exposures.
- We can achieve that using a modified version of the 2PP protocol key exchange protocol (se next slide).
  - Thus, we get **BOTH** forward secrecy and recover security with this protocol.

2PP Key Exchange with RS and FS

- Run the 2PP key exchange protocol from before but generate the master session keys in a slightly different way.
  - And of course, you get recover security if there is a single session that is attack free.

So now generating a session master key not only requires prior session master key but also the random nonces that the parties exchanged in the session (as part of the 2PP protocol).

Covered Material From the Textbook

- Chapter 5
  - Section 5.3
  - Section 5.4
  - Section 5.5
    - Except Sections 5.5.4 and 5.5.5 (only what we covered in class about these sections)
  - Section 5.6
    - Except Section 5.6.3

Thank You!