C. Olivier-Anclin

Session 4

Exercise 1 (Attack on the Needham–Schroeder Symmetric-Key Protocol)
Alice and Bob want to establish a fresh session key with the help of a trusted server S. Alice shares a long-term symmetric key K_{AS} with the server, and Bob shares a long-term symmetric key K_{BS} with the server. The Needham–Schroeder symmetric-key protocol proceeds as follows:

i. $A \to S : A, B, N_A$

ii. $S \to A : \{N_A, K_{AB}, B, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$

iii. $A \to B : \{K_{AB}, A\}_{K_{BS}}$

iv. $B \to A : \{N_B\}_{K_{AB}}$

v. $A \to B : \{N_B - 1\}_{K_{AB}}$

1. What is the purpose of the nonce N_A in step (i)?
2. Why does Alice return $N_B - 1$ in step (v)?
3. Suppose Eve can eavesdrop on all messages and also possesses an *old* session key K_{AB}. Show how she can convince Bob that she is Alice.
4. What modification can be made to the protocol to prevent this attack? Why is it sufficient, and what does the corrected protocol look like?

Exercise 2 (PGP Certificates)
A PGP certificate always contains the public key of a *master key*, along with one or more additional public keys called *subkeys*.

1. Is the master key used for signatures or for encryption?
2. Is a subkey typically used for signatures or for encryption?
3. Why must every subkey in a PGP certificate be certified by the master key?
4. What trust model does PGP rely on?
5. How should we think about the lifespan of the master key?
6. Since it is always possible to add additional encryption subkeys to a PGP certificate, is there any benefit in having several encryption subkeys?
7. What considerations apply to the lifespan of an encryption subkey?
8. How should one react if an encryption subkey (or any subkey) is compromised?

Exercise 3 (Signature Based on the Idendity)
From 1984, Adi Shamir proposed a way to obtain signatures that are based on the identity. We describe this solution as follows, where H is a cryptographic hash function:

Key generation	i Choose two big prime numbers p and q, and compute $n = p \cdot q$. ii Choose e such that $\gcd(e, \varphi(n)) = 1$.
Public key	(n, e)
Private keys	Only known by the *Key Distributor Center*: (p, q). Generated by the KDC and only knows by the user of identity i: g_i such that $g_i^e = i \mod n$.
Signature	For a message m: • Pick randomly r; • Compute $t = r^e \mod n$; • Compute $s = g_i \cdot r^{H(t\|m)} \mod n$; • The signature of the message m is the pair (s, t).
Verification	Check if $s^e = i \cdot t^{H(t\|m)} \mod n$.

1. Show that the verification is correct.
2. How the KDC can generate g_i from i?
3. Why g_i must be keep secret?
4. On which hard problem this protocol is based?

Exercise 4 (Secure Shell)
On request of SSH connection by a client, the server replies by sending its public key K_p in plain text. The client saves in this memory (more specifically in the file `$HOME/.ssh/known_host`; if this key is already stored, the client compare it to the stored key and warn the user if there is any difference). The client select one session key k and sends it to the server in an encrypted way. The server decrypts the session key k. The client and the server can now communicate with the shared secret key k.

1. Why secure a session with symmetric encryption instead of asymmetric encryption?
2. Assuming that all messages in the above protocol are authenticated, explain why the next connections are confidential and authenticated.
3. If the first connection is not authenticated, how to explain that an active attacker can masquerade as a server.
4. Why the client must inform the user when the public key changed?

Exercise 5 (Introduction to the Birthday Paradox)
The birthday paradox explains why an n-bit hash function only provides about $2^{n/2}$ security against collision attacks. In this exercise, you will derive this bound.

1. **Step 1 - No-collision probability.** A hash function outputs uniformly in a space of size 2^n. Argue that the probability of getting no collision after q queries is: $P(\text{no collision}) = \prod_{i=0}^{q-1} \left(1 - \frac{i}{2^n}\right)$.
2. **Step 2 - Exponential approximation.** Using $1 - x \approx e^{-x}$ for small x, show that: $P(\text{no collision}) \approx \exp\left(-\frac{q(q-1)}{2^{n+1}}\right)$.
3. **Step 3 - Collision probability.** Deduce an approximate expression for $P(\text{collision}) = 1 - P(\text{no collision})$.
4. **Step 4 - Birthday bound.** Solve for q when $P(\text{collision}) = 1/2$ and verify that: $q \approx 1.177 \cdot 2^{n/2}$.
5. **Step 5 - Security implication.** Using the formula above, explain briefly:
   - why 64-bit hash outputs are insecure,
   - why 256-bit outputs (e.g., SHA-256) provide $\approx 2^{128}$ collision resistance.