C. Olivier-Anclin

Session 5

Exercise 1 (TLS not unique)
A certain network protocol authenticates every packet of 384 bits using a MAC (that always maps a given pair (k, m) to the same tag t) that has tags of bitlength 96. For every session of the protocol (what a session is is not important here, but in a typical day one expects much more than 2^{40} sessions to be created worldwide), an identifier that is expected to uniquely identify the session among all possible sessions (past and future) is taken to be the 96-bit tag of a designated packet that is part of the session.

1. Identify a problem in the above process.
2. Propose a simple solution to fix it.

Exercise 2 (Bad authenticated encryption)
We consider a symmetric encryption scheme Enc and a deterministic MAC.

1. Show that $\mathsf{Enc} \,\|\, \mathsf{MAC} : (k_0, k, m) \mapsto \mathsf{Enc}(k_0, m) \,\|\, \mathsf{MAC}(k, m)$ has weak security w.r.t. the IND-CPA definition, regardless of the IND-CPA security of Enc.
2. Propose an alternative way of combining Enc with a MAC in order to get an "authenticated" encryption scheme, and informally justify its IND-CPA security and provides authenticity.

Exercise 3 (Understanding the Cryptography Behind the Signal App)
Signal is a secure messaging protocol that protects conversations even if attackers intercept messages or temporarily compromise a device. This exercise introduces the key cryptographic ideas behind the protocol.

1. **Basic Confidentiality.** A user Alice wants to send a secret message m to Bob using a symmetric key k (previously generated based on a Diffie–Hellman key exchange). Explain why encrypting m as $c = \mathsf{Enc}(k, m)$ does *not* protect Bob's past or future messages if the key k is ever stolen.
2. **Key Exchange and Fresh Keys.** Suppose Alice and Bob can agree on a fresh secret key sDH each time they start a *new session* (e.g., using a Diffie–Hellman exchange). Explain intuitively why using a *new* key for each session already improves security.

Post-Compromise Security (PCS). PCS means that even if an attacker learns some keys at time t, *future messages will eventually become secure again* (thanks to key updates).

For session i, Alice and Bob compute a fresh Diffie–Hellman shared secret sDH_i. This value is then fed into a key-derivation function $\mathsf{KDF}(\mathsf{sDH}_i) \longrightarrow \mathsf{RK}_i$, which behaves like a random function and is hard to invert (you may think of it as similar in spirit to a MAC or a hash). The newly derived key RK_i is then used, while previous secrets are deleted.

3. Does this process provide post-compromise security if we use the keys RK_i to encrypt messages ?
4. Give one practical reason why PCS matters for messaging applications.

In Signal, whenever the sending direction changes, the sender generates a fresh DH key pair and includes the new public key in the message. The receiver then uses this new DH value to update the root key via a KDF, after which both parties delete their old DH keys and continue with freshly derived keys.

Forward Secrecy (FS). Forward secrecy means that even if an attacker learns the current keys, *past messages remain safe.*

4. **FS.** Consider that Alice and Bob derive a new key k_1 from k_0, then k_2 from k_1, and so on for each new message in a session (for example based on a hash function):

$$k_0 \xrightarrow{\mathsf{KDF}(\cdot)} k_1 \xrightarrow{\mathsf{KDF}(\cdot)} k_2 \xrightarrow{\mathsf{KDF}(\cdot)} k_3 \xrightarrow{\mathsf{KDF}(\cdot)} k_4 \cdots$$

   Suppose an attacker learns k_3. Can the attacker recover k_4? And k_1? Explain why this property gives forward secrecy and the implication on the message secrecy ?

Assembling the Signal protocol. Signal additionally rely on a messaging server. From the keys ideas exposed above you will now derive a signal-like messageing protocol.

5. **Authenticated Diffie-Hellman.** Assume Alice and Bob each possess a signature public/private key pair $(\mathsf{pk}, \mathsf{sk})$ certified by the signal server via its keys $(\mathsf{pk}_\mathsf{S}, \mathsf{sk}_\mathsf{S})$. How can they securly execue a Diffie–Hellman key exchange to obtain sDH ?

6. Using the mechanisms introduced above, describe how to construct a messaging protocol that provides PCS and FS.

Exercise 4 (AND/OR composition of Schnorr zero-knowledge proofs)
Let $G = \mathbb{Z}_q$ be a cyclic group of prime order q with generator g. A Schnorr proof of knowledge shows knowledge of x such that $X = g^x$ without revealing x. We use the following standard Sigma-style Schnorr protocol (commit–challenge–response):
Prover: $r \xleftarrow{\$} \mathbb{Z}_q, R = g^r$, send R
Verifier: $c \in \mathbb{Z}_q$
Prover: $a = r - c \cdot x$, send a
Verifier: checks $R \stackrel{?}{=} g^a \cdot X^c$.

Simulation of a Schnorr proof (useful for OR-proofs). A simulator that does not know x can produce a transcript (R, c, a) distributed identically to a real execution by sampling $c, a \xleftarrow{\$} \mathbb{Z}_q$ uniformly and computing $R = g^a \cdot X^c$. This triple passes the verifier check by construction.

AND composition. Alice wants to prove knowledge of both x_1 and x_2 for $X_1 = g^{x_1}$ and $X_2 = g^{x_2}$.

1. Describe the AND-composition protocol using Schnorr proofs and explain why it is zero-knowledge and sound.

OR composition. Bob wants to prove that he knows either x_1 or x_2 (for $X_1 = g^{x_1}$ or $X_2 = g^{x_2}$) without revealing which one.

2. Using the Schnorr simulator above, show how Bob can simulate a proof for the branch whose secret he does not know.

3. Give the complete OR-proof which the verification is the : how to form commitments, how to compute challenges that add up to the verifier's challenge, and how to produce responses. **Hint:** consider a single value c send by the challenger that you can split $c = c_1 + c_2$.

4. Briefly justify why the OR-proof is zero-knowledge and sound.