Problem 4. «Precomputed keys»

Ephemeral keys are often used in authenticated key establishment (AKE) protocols. However, their implementation on low-resource devices requires the generation of these keys, which is not always convenient. Therefore, it is possible to precompute these keys and store their private (public) parts in the (un)protected memory. So, let the adversary have the capability to replace the values written in the unprotected memory.

The problem is to make attacks on the AKE protocol described below that implement the following threats:

1) key compromise impersonation of the initiator;
2) key compromise impersonation of the responder;
3) forward secrecy breaking;

using only the capability to replace the values written in the unprotected memory and the capability to break the interaction (and the capability to know the key corresponding to the threats).

Key compromise impersonation. An adversary is said to impersonate a party B to party A if B is honest and A accepts the session (and establish the key) with B but party B has not completed (or even started) the protocol, and an adversary knows the established key. In a successful key compromise impersonation (KCI) attack, an adversary with the knowledge of the long-term private key of a party A can impersonate B to A.

Forward secrecy breaking. An adversary with the knowledge of the long-term private key of a party A (or of a party B) is said to break forward secrecy if he/she knows the session keys established before he knew the long-term key.

The AKE protocol. Calculations are described for the case of an additive group G of a prime order q. The generator of this group are denoted by P. We denote by $\mathbb{Z}_q$ a finite field of characteristic q and assume the canonic representation of the elements in $\mathbb{Z}_q$ as integers from the set $\{0, 1, \ldots, q-1\}$.

We use the following notations for cryptographic keys and shared non-secret values: (x, X) is a key pair consisting of a scalar x and corresponding multiple element of the group $X = x \cdot P$; (sk^s, pk^s) is a key pair of a signature scheme; K^a is a key of a message authentication scheme; K is a secret key established as a result of the protocol.

We use the following notations for the used cryptographic mechanisms primitives: Sig_{sk^s} and MAC_{K^a} are signing and generating the message authentication code algorithms, verification of signatures and message authentication codes is denoted by «Verify σ, τ»; $\mathsf{KDF}(\cdot)$ and $\mathsf{H}(\cdot)$ are key derivation and hash function. We assume that hash values are elements from $\mathbb{Z}_q$. Read operations from (un)protected memory are designated as $a \leftarrow PM$ ($a \leftarrow UPM$).

A : $(x_A, X_A), (sk_A^s, pk_A^s)$		**B** : $(x_B, X_B), , (sk_B^s, pk_B^s)$
$E_A \leftarrow UPM$		
$\sigma_A \leftarrow \mathsf{Sig}_{sk_A^s}(E_A, A)$		
	$\xrightarrow{A, E_A, \sigma_A}$	
		Verify σ_A
		$e_B \leftarrow PM$
		$E_B \leftarrow UPM$
		$u_1 \leftarrow \mathsf{H}(A, B, X_A)$
		$u_2 \leftarrow \mathsf{H}(A, B, X_B)$
		$Q \leftarrow u_1 \cdot e_B \cdot X_A + u_2 \cdot x_B \cdot E_A + e_B \cdot E_A$
		$K, K^a \leftarrow \mathsf{KDF}(Q)$
		$\sigma_B \leftarrow \mathsf{Sig}_{sk_B^s}(E_B, B)$
		$\tau_B \leftarrow \mathsf{MAC}_{K^a}(B, A, E_B, E_A)$
	$\xleftarrow{B, E_B, \tau_B}$	
$e_A \leftarrow PM$		
$u_1 \leftarrow \mathsf{H}(A, B, X_A)$		
$u_2 \leftarrow \mathsf{H}(A, B, X_B)$		
$Q \leftarrow u_1 \cdot x_A \cdot E_B + u_2 \cdot e_A \cdot X_B + e_A \cdot E_B$		
$K, K^a \leftarrow \mathsf{KDF}(Q)$		
Verify τ_B		
return K		**return** K