**NATIONAL INSTITUTE OF TECHNOLOGY DELHI**

A blue and white logo

Description automatically generated

**Department of Computer Science & Engineering**

**CSB451 – Network Security & Cryptography**

***Assignment – 10***

**Submitted By:**

**Himanshu Verma**

**201210023**

**Submitted To: Dr. Karan Verma**

1. **You are given are two protocols…………and E is a public-key encryption algorithm.**
   1. **Provide a step-by-step description of what the receiver does upon receipt of y.**

**Sol:**

The receiver does the following in protocol A:

* Decrypt using k1 and obtain x ∥ H(k2 ∥ x). Extract x.
* Hash k2 ∥ x and compare to H(k2 ∥ x).

In case of protocol B (which is completely flawed):

* Decrypt using SK and obtain H(x). Extract x.
* Hash x and compare to H(x).
  1. **State whether confidentiality and integrity is achieved for each of the two protocols given in the previous problem. Briefly justify each answer.**

Sol: In the case of protocol A:

* Confidentiality: Can be achieved through encryption.
* Integrity: Can be achieved through hashing.

In case of protocol B:

* Confidentiality: It doesn’t provide confidentiality because it doesn’t have any encryption.
* Integrity: It doesn’t provide data integrity as anybody can replace the message and compute the hash.

1. **Alice wishes to provide Chosen Ciphertext Attack (CCA)… by choosing a particular Mac.**

**Sol:**

We can show that Alice's scheme is not CCA-secure by constructing a specific attack scenario that exploits a weakness in how the message (m) is used. Here's the attack strategy using a specific insecure Mac:

Insecure Mac: Define a trivial Mac function denoted by Mac'(k, m) that always outputs a constant value "0" regardless of the key (k) or message (m).

Attacker's Goal: The attacker wants to learn the difference between two messages m0 and m1 chosen by the attacker.

Attack Steps:

Phase 1 - Learning ciphertext of m0:

The attacker submits message m0 to the encryption oracle. It receives the ciphertext c0 = Enc(k1, m0) || Mac'(k2, m0) = Enc(k1, m0) || "0".

Phase 2 - Constructing a malleable ciphertext:

The attacker prepares a new message m' = (m0, 1) where "(m0, 1)" represents concatenation. This message appends the value "1" to the original message m0.

Since the attacker doesn't know k1, they cannot directly encrypt m'. However, they can leverage the insecure Mac.

The attacker submits the modified message m' to the decryption oracle. It receives c' = Enc(k1, m') || Mac'(k2, m') = Enc(k1, (m0, 1)) || "0".

Analysis:

The attacker observes that the first part of both ciphertexts (c0 and c') will be identical because they share the same prefix (m0) encrypted with the unknown key k1.

The attacker focuses on the second part (the Mac portion).

In c0, Mac'(k2, m0) always outputs "0" due to the insecure Mac definition.

In c', Mac'(k2, m') also outputs "0" because the insecure Mac ignores the message content.

Learning the difference:

If the decrypted message from the oracle corresponds to c0 (meaning successful decryption), the attacker knows the received message doesn't contain the appended "1" because the Mac outputs "0" consistently.

If the decrypted message corresponds to c' (meaning decryption failed due to the appended "1"), the attacker can infer that the original message contained "1".

By comparing the decryption responses for c0 and c', the attacker can indirectly distinguish between the original messages m0 and m1 because the insecure Mac leaks no information about the actual message content. This violates the CCA security principle, where the attacker shouldn't be able to learn anything about the messages just by observing the ciphertexts and manipulating them with a weak authentication tag.

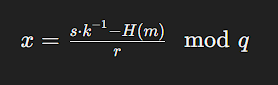
This attack demonstrates that relying solely on CPA-secure encryption and an insecure Mac is not sufficient for achieving CCA security. A secure Mac that binds the message content to the tag is crucial to prevent such manipulations.

1. **The Digital Signature Algorithm starts by selecting a……………..the consequences if an attacker could learn k in addition to the signature (r, s)?**

**Sol:**

Yes, it is crucial to protect the random value 𝑘 used in the Digital Signature Algorithm (DSA), even though it is not directly related to the signer's private key. If an attacker could learn the value of *k* in addition to the signature (*r*,*s*), it would have serious consequences for the security of the digital signature due to the following reasons:

1. **Recovery of Private Key:** If an attacker obtains both 𝑘*k* and the signature (𝑟,𝑠), they could potentially recover the signer's private key. The private key can be calculated using the equation:



where:

* 𝑥is the signer's private key.
* *H*(*m*) is the hash of the message being signed.
* *k*−1 is the modular multiplicative inverse of *k* modulo 𝑞.
* *r* and 𝑠are the components of the signature.
* 𝑞 is the order of the group.

1. Forgery of Signatures: With knowledge of k, an attacker could forge arbitrary signatures for any message without knowing the private key. Since k is part of the signature calculation, knowing it allows the attacker to generate valid signatures for arbitrary messages.
2. Compromised Non-repudiation: If an attacker can forge signatures using the recovered private key, they could impersonate the legitimate signer, leading to compromised non-repudiation. This means the signer could deny having signed a message when, in fact, the attacker generated the signature.

An attacker could potentially exploit the knowledge of a past k value to predict future k selections, especially if the random number generation process used for k is weak. By recreating the relationship between k, the message, and the signature components, they could forge new signatures for different messages.