Fundamentals of Cryptography

Homework 7

Dr. Mohammad Dakhilalian

Fall 2024

Theory Part

Thoroughly review **Chapters 12 & 13** of the book *Understanding Cryptography* to confidently address the questions.

Question 1

For hash functions, it is crucial to have a sufficiently large number of output bits, with, e.g., 160 bits, in order to thwart attacks based on the birthday paradox. Why are much shorter output lengths of, e.g., 80 bits, sufficient for MACs?

For your answer, assume a message x that is sent in clear together with its MAC over the channel: $(x, \text{MAC}_k(x))$. Clearly clarify what Oscar needs to do to attack this system.

Question 2

We study two methods for integrity protection with encryption.

1. Assume we apply a technique for combined encryption and integrity protection in which a ciphertext c is computed as

   $$c = e_k(x||h(x))$$

   where $h()$ is a hash function. This technique is not suited for encryption with stream ciphers if the attacker knows the whole plaintext x. Explain *exactly* how an active attacker can now *replace* x by an arbitrary x' of his/her choosing and compute c' such that the receiver will verify the message correctly. Assume that x and x' are of equal length. Will this attack work too if the encryption is done with a one-time pad?

2. Is the attack still applicable if the checksum is computed using a keyed hash function such as a MAC:

   $$c = e_k(x||MAC_{k_2}(x))$$

   Assume that $e()$ is a stream cipher as above.

Question 3

We consider the Diffie–Hellman key exchange scheme with certificates. We have a system with three users Alice, Bob, and Charley. The Diffie–Hellman algorithm uses $p = 61$ and $\alpha = 18$. The three secret keys are $a = 11$, $b = 22$, and $c = 33$. The three IDs are $\text{ID}(A) = 1$, $\text{ID}(B) = 2$, and $\text{ID}(C) = 3$.

The Elgamal signature scheme is used to generate signatures. We apply the system parameters $p' = 467$, $d' = 127$, $\alpha' = 2$, and β. The CA uses the ephemeral keys $k_E = 213$, 215, and 217 for Alice's, Bob's, and Charley's signatures, respectively. (In practice, the CA should use a better pseudorandom generator to obtain the k_E values.)

To obtain the certificates, the CA computes $x_i = 4 \times b_i + \text{ID}(i)$ and uses this value as input for the signature algorithm. (Given x_i, $\text{ID}(i)$ follows then from $\text{ID}(i) \equiv x_i \mod 4$.)

1. Compute three certificates Cert_A, Cert_B, and Cert_C.
2. Verify all three certificates.
3. Compute the three session keys k_{AB}, k_{AC}, and k_{BC}.

Question 4

In this question, we want to analyze some variants of key derivation. In practice, one master key k_{MK} is exchanged securely (e.g. certificate-based DHKE) between the involved parties. Afterwards, the session keys are regularly updated by use of key derivation. For this purpose, three different methods are at our disposal:

(a) $k_0 = k_{MK}$; $k_{i+1} = k_i + 1$

(b) $k_0 = h(k_{MK})$; $k_{i+1} = h(k_i)$

(c) $k_0 = h(k_{MK})$; $k_{i+1} = h(k_{MK}||i||k_i)$

where $h()$ marks a (secure) hash function, and k_i is the ith session key.

1. What are the main differences between these three methods?
2. Which method provides Perfect Forward Secrecy?
3. Assume Oscar obtains the nth session key (e.g., via brute-force). Which sessions can he now decrypt (depending on the chosen method)?
4. Which method remains secure if the masterkey k_{MK} is compromised? Give a rationale!

Question 5

Another problem in certificate systems is the authenticated distribution of the CA's public key, which is needed for certificate verification. Assume Oscar has full control over all of Bob's communications, that is, he can alter all messages to and from Bob. Oscar now replaces the CA's public key with his own (note that Bob has no means to authenticate the key that he receives, so he thinks that he received the CA public key).

1. **Certificate issuing:** Bob requests a certificate by sending a request containing:
   (a) Bob's ID ID(B), and
   (b) Bob's public key B from the CA.

   Describe exactly what Oscar has to do so that Bob doesn't find out that he has the wrong public CA key.
2. **Protocol execution:** Describe what Oscar has to do to establish a session key with Bob using the authenticated Diffie–Hellman key exchange, such that Bob thinks he is executing the protocol with Alice.

Programming Part

Question 6

Using the SHA-1 algorithm, perform an attack against Secret Prefix MACs, as demonstrated in the 12.2 section. Implement the following components in your program:

- A **victim function** that generates a message and computes its MAC using a secret key concatenated with the message. This function then sends the message and its corresponding MAC.
- An **attacker function** that:
  1. Receives the original message and its MAC from the victim.
  2. Appends additional text to the original message.
  3. Computes a new MAC for the modified message without knowledge of the secret key.
- A **verification function** that verifies whether the MAC of the modified message (computed by the attacker) matches the expected MAC. This demonstrates that the attacker successfully forged a valid MAC for the modified message without knowing the key.

You may either use an already implemented SHA-1 algorithm or write your own custom SHA-1 implementation. Ensure that your program clearly demonstrates the attack and its success.

Question 7

Bob wants to send Alice a message securely over a secure channel like a socket. Design and implement a program in your favorite programming language that facilitates secure communication between Bob and Alice under the following requirements:

- First, Bob initiates a connection with Alice to establish a shared secret key. To do this, use the ElGamal encryption scheme that you have implemented before. Briefly, Bob encrypts a randomly generated secret value using Alice's public key, and Alice decrypts it using her private key to retrieve the shared secret.
- After establishing the shared secret, Bob generates and sends two random nonces to Alice. Both parties use these nonces, along with the shared secret, to derive two keys using an HMAC function:
  1. One key is used for AES encryption in CBC mode to encrypt the message.
  2. The other key is used to compute the HMAC of the ciphertext.
- When Bob wants to send a message to Alice:
  1. Bob first encrypts the message using AES in CBC mode you have implemented before and the first derived key.
  2. Then, Bob computes the HMAC of the ciphertext using the second derived key.
  3. Finally, Bob concatenates the ciphertext and its HMAC, and sends the combined data to Alice over the channel.
- Alice responds to Bob with a message in the same structure (encrypted message and its HMAC), following the same procedure. After the exchange, both Bob and Alice close the connection.
- Note that you can use AES in CBC mode and the ElGamal implementation you have already written in previous coursework. For HMAC, you are allowed to use a built-in library function.

Assume the attacker can eavesdrop on the connection and modify the transmitted packets but cannot impersonate Bob or Alice. Design your program to ensure the confidentiality and integrity of the exchanged messages.