Fundamentals of Cryptography

Homework 6

Dr. Mohammad Dakhilalian

Fall 2023

Theory Part

Question 1

As we have seen, MACs can be used to authenticate messages. With this problem, we want to show the difference between two protocols—one with a MAC, and one with a digital signature. In the two protocols, the sending party performs the following operation:

1. **Protocol A:**
$$y = e_{k_1}[x||h(k_2||x)]$$
where x is the message, $h()$ is a hash function such as SHA-1, e is a private-key encryption algorithm, $||$ denotes simple concatenation, and k_1, k_2 are secret keys which are only known to the sender and the receiver.

2. **Protocol B:**
$$y = e_k[x||\mathrm{sig}_{k_{\mathrm{pr}}}(h(x))]$$
where x is the message, $h()$ is a hash function, e is a private-key encryption algorithm, $||$ denotes simple concatenation, and $\mathrm{sig}_{k_{\mathrm{pr}}}$ is a digital signature using the private key k_{pr}.

Provide a step-by-step description of what the receiver does upon receipt of y.

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

MACs are, in principle, also vulnerable to collision attacks. We discuss the issue in the following.

1. Assume Oscar found a collision between two messages, i.e.,

$$MAC_k(x_1) = MAC_k(x_2)$$

   Show a simple protocol with an attack that is based on a collision.

2. Even though the birthday paradox can still be used for constructing collisions, why is it in a practical setting much harder to construct them for MACs than for hash functions? Since this is the case: what security is provided by a MAC with 80-bit output compared to a hash function with 80-bit output?

Question 4

In this exercise, 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

People at your new job are deeply impressed that you worked through this book. As the first job assignment, you are asked to design a digital pay-TV system that uses encryption to prevent service theft through wire tapping. As a key exchange protocol, a strong Diffie–Hellman with, e.g., 2048-bit modulus is being used. However, since your company wants to use cheap legacy hardware, only DES is available for data encryption algorithm. You decide to use the following key derivation approach:

$$K^{(i)} = f(K_{AB}||i)$$

where f is an irreversible function.

1. First we have to determine whether the attacker can store an entire movie with reasonable effort (in particular, cost). Assume the data rate for the TV link is 1 Mbit/s, and that the longest movies we want to protect are 2 hours long. How many Gbytes of data must be stored for a 2-hour film (don't mix up bit and byte here)? Is this realistic?
2. We assume that an attacker will be able to find a DES key in 10 minutes using a brute-force attack. Note that this is a somewhat optimistic assumption from an attacker's point of view, but we want to provide some medium-term security by assuming increasingly faster key searches in the future. How frequently must a key be derived if the goal is to prevent an offline decryption of a 2-hour movie in less than 30 days?

Question 6

We reconsider the Diffie–Hellman key exchange protocol. Assume now that Oscar runs an active man-in-the-middle attack against the key exchange (as explained in Sect. 13.3.1). For the Diffie–Hellman key exchange, use the parameters $p = 467$, $\alpha = 2$, and $a = 228$, $b = 57$ for Alice and Bob, respectively. Oscar uses the value $o = 16$. Compute the key pairs k_{AO} and k_{BO}

1. the way Oscar computes them, and
2. the way Alice and Bob compute them.

Question 7

Given is a user domain in which users share the Diffie–Hellman parameters α and p. Each user's public Diffie–Hellman key is certified by a CA. Users communicate securely by performing a Diffie–Hellman key exchange and then encrypting/decrypting messages with a symmetric algorithm such as AES.

Assume Oscar gets hold of the CA's signature algorithm (and especially its private key), which was used to generate certificates. Can he now decrypt old ciphertexts which were exchanged between two users before the CA signature algorithm was compromised, and which Oscar had stored? Explain your answer.

Optional Part

Question 8

Write a client-server program in your favorite programming language, where the client and the server can send arbitrary messages with at most 4096-bit length. Use cryptographic mechanisms to guarantee the integrity and confidentiality of exchanged messages over the insecure communication channel. Take the following points into account:

- The adversary can read the source codes of the client and server. Additionally, he can access and read files on the system. So, you shall not hardcode or save any of your secrets within your codes or in any of the files on your system.
- The attacker can eavesdrop on the channel. Therefore, to ensure the confidentiality of messages, you might want to use a mechanism with which the sender converts a message into an alternative form unreadable by the attacker using a shared secret, which is only known to authorized parties, and the receiver reverts the message to its original form using the same secret. Take care about the strength of the mechanism you choose and pay attention to weaknesses such as determinism and malleability.
- As the channel is not initially secure and neither can you hardcode the shared secret, you shall use a mechanism through which one of the parties chooses the secret and sends it to the other party securely. Here, to guarantee the confidentiality of the exchanged secret, you should opt for a cryptographic mechanism with which the sending party converts the secret, itself, to a form, again, obscure to the adversary. But, this time, the only person who can convert the secret to its original form is the receiving party who has a private secret, which helps it return the shared secret to its initial form.
- Finally, besides eavesdropping, the attacker can actively modify the packets exchanged between parties, even if he doesn't have the slightest idea about their contents. To prevent this from happening and guarantee the integrity of the messages, you should attach to the message block, a unique value derived from the message, itself. This value should be the output of a cryptographic function that receives the message as input. It should, further, have this property that if the adversary changes any of the original message's bits the value associated with the new message should completely differ from the attached value. So that, this way, the other side can recognize the message has been modified on the way. Moreover, it should be computationally infeasible for the attacker to guess the content of the message from this attached value, or find a message different from the original message that, if given to the same cryptographic function as input, results in the same value.

For simplicity, assume that the capabilities of the attacker are limited to those mentioned above, and he cannot do anything further, such as impersonating communicating parties, and so on.

Question 9

Do the following exercises surrounding the Davies–Meyer algorithm;

1. Implement this compression function in your favorite programming language. You can use the ipython file provided in your question folder. You should only complete the requested parts if you are going to use this file. Your code should receive a string of arbitrary length and compute its Davies–Meyer hash.
2. Using the code you've written in the previous part and a random key, calculate the secret prefix Message Authentication Code of an arbitrary input string.
3. Conduct an attack against the previously calculated secret prefix MAC. Note that your original message's length in addition to the key length should be a multiple of the compression function's block size (the number of letters in the message should be a multiple of 16).
4. Check whether your attack in the previous part was successful by appending the original message string to the additional string in the previous part, computing the hash of the resulting string, and comparing the resultant with the hash in the previous part.