Digital Signature Generation and Verification

Figure 3.15 is a generic model of the process of making and using digital signatures. All of the digital signature schemes in FIPS 186-4 have this structure. Suppose that Bob wants to send a message to Alice. Although it is not important that the message be kept as a secret, he wants Alice to be certain that the message is indeed from him. For this purpose, Bob uses a secure hash function, such as SHA-512, to generate a hash value for the message. That hash value, together with Bob's private key, serve as input to a digital signature generation algorithm that produces a short block that functions as a digital signature. Bob sends the message with the signature attached. When Alice receives the message plus signature, she (1) calculates

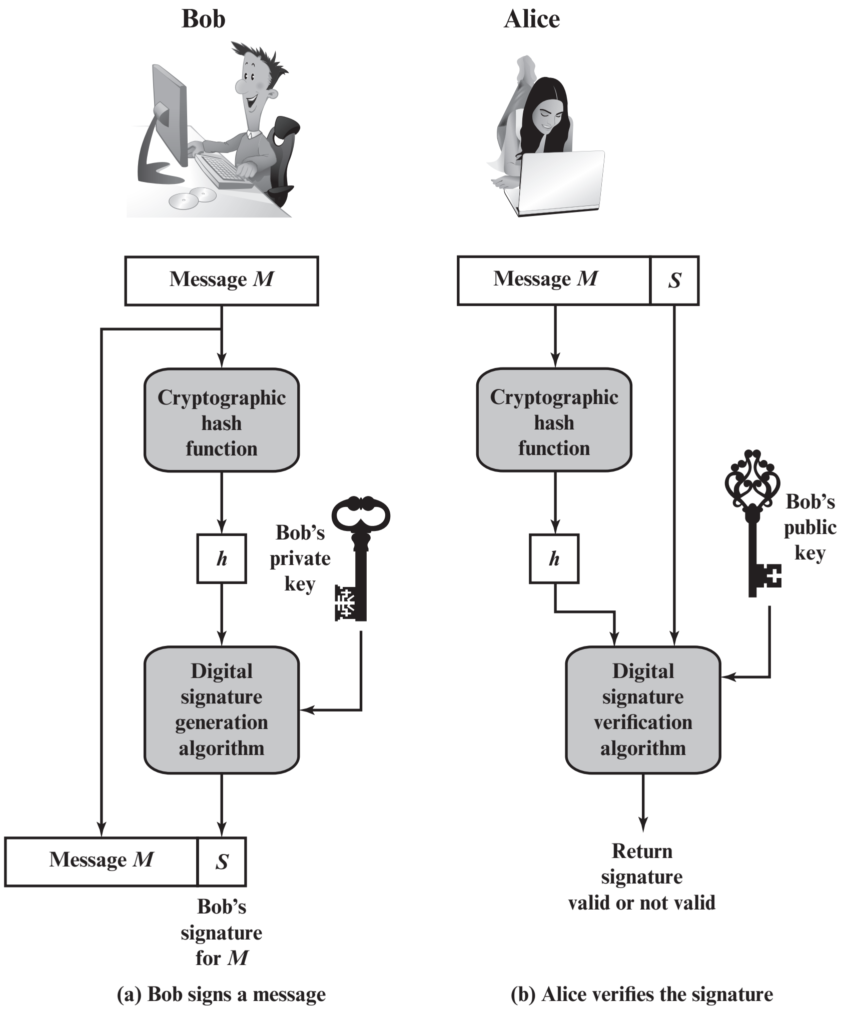
Bob
Alice
Message M
Message M
S
Cryptographic hash function
Cryptographic hash function
h
h
Bob's private key
Bob's public key
Digital signature generation algorithm
Digital signature verification algorithm
Message M
S
Bob's signature for M
Return signature valid or not valid
(a) Bob signs a message
(b) Alice verifies the signature

Figure 3.15 Simplified Depiction of Essential Elements of Digital Signature Process

a hash value for the message and (2) provides the hash value and Bob's public key as inputs to a digital signature verification algorithm. If the algorithm returns the result that the signature is valid, Alice is assured that the message must have been signed by Bob. No one else has Bob's private key and therefore no one else could have created a signature that could be verified for this message with Bob's public key. In addition, it is impossible to alter the message without access to Bob's private key, so the message is authenticated both in terms of source and in terms of data integrity.

It is important to emphasize that the encryption process just described does not provide confidentiality. That is, the message being sent is safe from alteration, but not safe from eavesdropping. This is obvious in the case of a signature based on a portion of the message, because the rest of the message is transmitted in the clear. Even in the case of complete encryption, there is no protection of confidentiality because any observer can decrypt the message by using the sender's public key.

RSA Digital Signature Algorithm

The essence of the RSA digital signature algorithm is to encrypt the hash of the message to be signed using RSA. However, as with the use of RSA for encryption of keys or short messages, the RSA digital signature algorithm first modifies the hash value to enhance security. There are several approaches to this, one of which is the RSA Probabilistic Signature Scheme (RSA-PSS). RSA-PSS is the latest of the RSA schemes and the one that RSA Laboratories recommends as the most secure of the RSA digital signature schemes. We provide a brief overview here; for more detail see [STAL16].

Figure 3.16 illustrates the RSS-PSS signature generation process. The steps are as follows:

1. Generate a hash value, or message digest, mHash from the message M to be signed.
2. Pad mHash with a constant value padding1 and pseudorandom value salt to form M'
3. Generate hash value H from M'.
4. Generate a block DB consisting of a constant value padding 2 and salt.
5. Use the mask generating function MGF, which produces a randomized output from input H of the same length as DB.
6. Create the encoded message (EM) block by padding H with the hexadecimal constant BC and the XOR of H and DB.
7. Encrypt EM with RSA using the signer's private key.

The objective with this algorithm is to make it more difficult for an adversary to find another message that maps to the same message digest as a given message or to find two messages that map to the same message digest. Because the salt changes with every use, signing the same message twice using the same private key will yield two different signatures. This is an added measure of security.

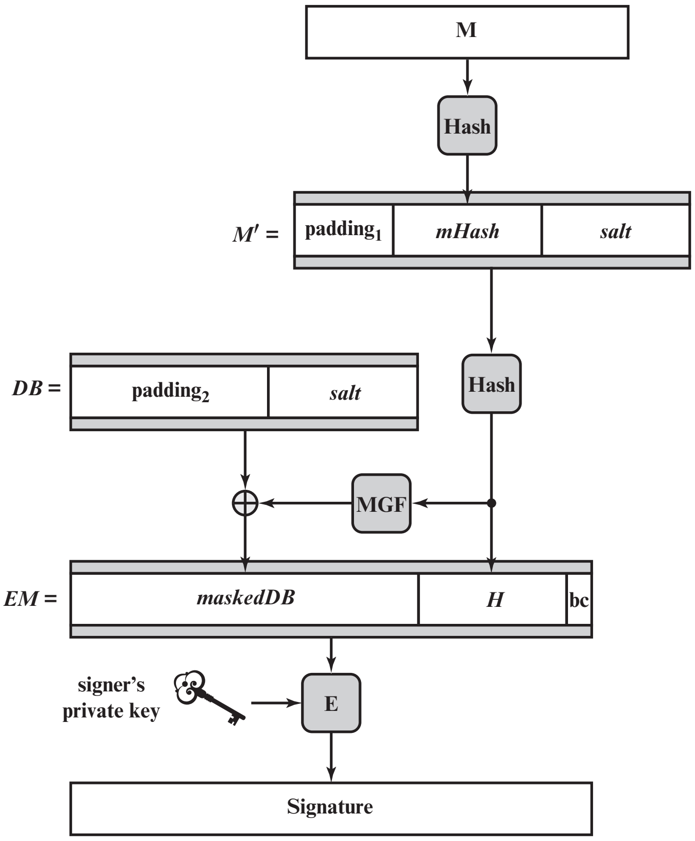

Figure 3.16 RSA-PSS Encoding and Signature Generation

3.7 KEY TERMS, REVIEW QUESTIONS, AND PROBLEMS

Key Terms

authenticated encryption Diffie–Hellman key exchange digital signature Digital Signature Standard (DSS) elliptic-curve cryptography (ECC) HMAC	key exchange MD5 message authentication message authentication code (MAC) message digest one-way hash function private key public key	public-key certificate public-key encryption RSA secret key secure hash function SHA-1 strong collision resistant weak collision resistant

Review Questions

3.1 List three approaches to message authentication.

3.2 What is a message authentication code?

3.3 What is a one-way hash function? How is it different from message authentication code?
3.4 List the properties of a strong hash function.
3.5 Compare SHA-1 and SHA-2 with respect to SHA parameters.
3.6 List the design objectives for HMAC.
3.7 Differentiate between public-key cryptosystem and symmetric encryption algorithm.
3.8 List the public-key cryptography algorithm used in digital signatures, encryption/decryption, and key exchange.
3.9 In what way is the Diffie–Hellman key exchange algorithm insecure against a man-in-the-middle attack?

Problems

3.1 Consider a 32-bit hash function defined as the concatenation of two 16-bit functions: XOR and RXOR, which are defined in Section 3.2 as "two simple hash functions."
  a. Will this checksum detect all errors caused by an odd number of error bits? Explain.
  b. Will this checksum detect all errors caused by an even number of error bits? If not, characterize the error patterns that will cause the checksum to fail.
  c. Comment on the effectiveness of this function for use as a hash function for authentication.

3.2 Suppose H(m) is a collision-resistant hash function that maps a message of arbitrary bit length into an n-bit hash value. Is it true that, for all messages x, x' with $x \not\equiv x'$, we have $\text{H}(x) \neq \text{H}(x')$? Explain your answer.

3.3 State the value of the padding field in SHA-512 if the length of the message is
  a. 4987 bits
  b. 4199 bits
  c. 1227 bits

3.4 State the value of the length field in SHA-512 if the length of the message is
  a. 3967 bits
  b. 3968 bits
  c. 3969 bits

3.5 a. Consider the following hash function. Messages are in the form of a sequence of decimal numbers, $M = (a_1, a_2, \ldots, a_t)$. The hash value h is calculated as $\left(\sum_{i=1}^{t} a_i\right) \bmod n$, for some predefined value n. Does this hash function satisfy any of the requirements for a hash function listed in Section 3.2? Explain your answer.
  b. Repeat part (a) for the hash function $h = \left(\sum_{i=1}^{t} (a_i)^2\right) \bmod n$.
  c. Calculate the hash function of part (b) for $M = (237, 632, 913, 423, 349)$ and $n = 757$.

3.6 This problem introduces a hash function similar in spirit to SHA that operates on letters instead of binary data. It is called the *toy tetragraph hash* (tth).[6] Given a message consisting of a sequence of letters, tth produces a hash value consisting of four letters. First, tth divides the message into blocks of 16 letters, ignoring spaces, punctuation, and capitalization. If the message length is not divisible by 16, it is padded out with nulls. A four-number running total is maintained that starts out with the value (0, 0, 0, 0); this is input to the compression function for processing the first block. The compression function consists of two rounds. **Round 1:** Get the next block of text and arrange it as a row-wise 4 block of text and covert it to numbers (A = 0, B = 1, etc.). For example, for the block ABCDEFGHIJKLMNOP, we have

[6] I thank William K. Mason of the magazine staff of *The Cryptogram* for providing this example.

A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Then, add each column mod 26 and add the result to the running total, mod 26. In this example, the running total is (24, 2, 6, 10). **Round 2:** Using the matrix from round 1, rotate the first row left by 1, second row left by 2, third row left by 3, and reverse the order of the fourth row. In our example:

B	C	D	A
G	H	E	F
L	I	J	K
P	O	N	M

1	2	3	0
6	7	4	5
11	8	9	10
15	14	13	12

Now, add each column mod 26 and add the result to the running total. The new running total is (5, 7, 9, 11). This running total is now the input into the first round of the compression function for the next block of text. After the final block is processed, convert the final running total to letters. For example, if the message is ABCDE FGHIJKLMNOP, then the hash is FHJL.

a. Draw figures comparable to Figures 3.4 and 3.5 to depict the overall tth logic and the compression function logic.
b. Calculate the hash function for the 48-letter message "I leave twenty million dollars to my friendly cousin Bill."
c. To demonstrate the weakness of tth, find a 48-letter block that produces the same hash as that just derived. *Hint:* Use lots of A's.

3.7 It is possible to use a hash function to construct a block cipher with a structure similar to DES. Because a hash function is one way and a block cipher must be reversible (to decrypt), how is it possible?

3.8 Now consider the opposite problem: Use an encryption algorithm to construct a one-way hash function. Consider using RSA with a known key. Then process a message consisting of a sequence of blocks as follows: Encrypt the first block, XOR the result with the second block and encrypt again, and so on. Show that this scheme is not secure by solving the following problem. Given a two-block message B1, B2, and its hash, we have

$$\text{RSAH}(B1, B2) = \text{RSA}(\text{RSA}(B1) \oplus B2)$$

Given an arbitrary block C1, choose C2 so that RSAH(C1, C2) = RSAH(B1, B2). Thus, the hash function does not satisfy weak collision resistance.

3.9 One of the most widely used MACs, referred to as the Data Authentication Algorithm, is based on DES. The algorithm is both a FIPS publication (FIPS PUB 113) and an ANSI standard (X9.17). The algorithm can be defined as using the cipher block chaining (CBC) mode of operation of DES with an initialization vector of zero (Figure 2.9). The data (e.g., message, record, file, or program) to be authenticated is grouped into contiguous 64-bit blocks: $P_1, P_2, \ldots, P_N$. If necessary, the final block is padded on the right with 0s to form a full 64-bit block. The MAC consists of either the entire ciphertext block C_N or the leftmost M bits of the block with $16 \le M \le 64$. Show that the same result can be produced using the cipher feedback mode.

3.10 In this problem, we will compare the security services that are provided by digital signatures (DS) and message authentication codes (MAC). We assume that Oscar is able to observe all messages sent from Alice to Bob and vice versa. Oscar has no knowledge of any keys but the public one in case of DS. State whether and how (i) DS and (ii) MAC protect against each attack. The value `auth(x)` is computed with a DS or a MAC algorithm, respectively.

a. (Message integrity) Alice sends a message `x = "Transfer $1000 to Mark"` in the clear and also sends `auth(x)` to Bob. Oscar intercepts the message and replaces "Mark" with "Oscar." Will Bob detect this?
b. (Replay) Alice sends a message `x = "Transfer $1000 to Oscar"` in the clear and also sends `auth(x)` to Bob. Oscar observes the message and signature and sends them 100 times to Bob. Will Bob detect this?
c. (Sender Authentication with cheating third party) Oscar claims that he sent some message `x` with a valid `auth(x)` to Bob, but Alice claims the same. Can Bob clear the question in either case?
d. (Authentication with Bob cheating) Bob claims that he received a message `x` with a valid signature `auth(x)` from Alice (e.g., "Transfer $1000 from Alice to Bob") but Alice claims she has never sent it. Can Alice clear this question in either case?

3.11 Figure 3.17 shows an alternative means of implementing HMAC.
a. Describe the operation of this implementation.
b. What potential benefit does this implementation have over that shown in Figure 3.6?

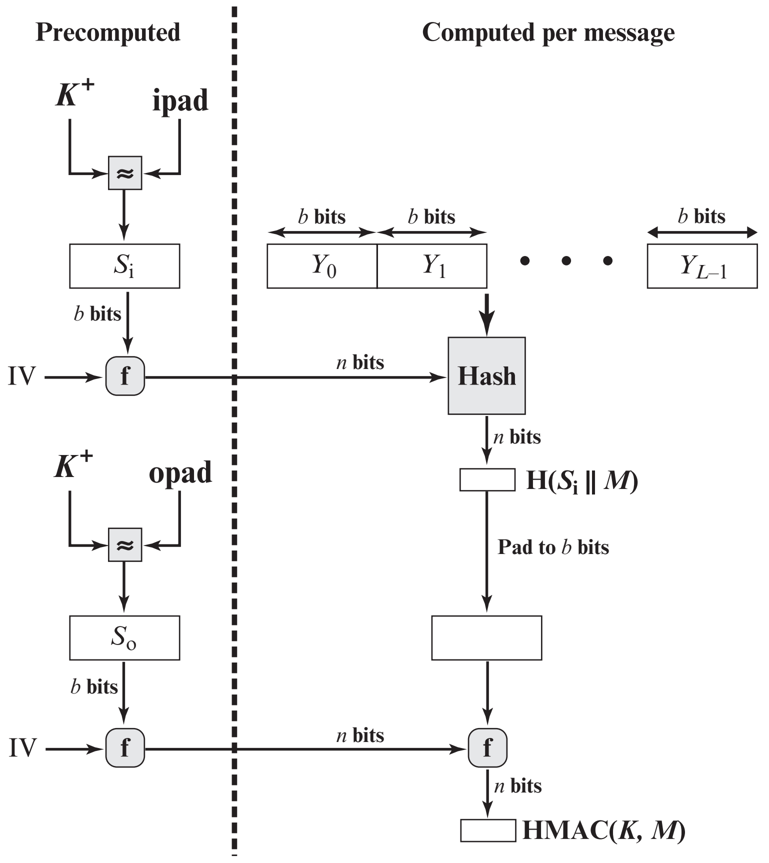

Figure 3.17 Efficient Implementation of HMAC

3.12 In this problem, we demonstrate that for CMAC, a variant that XORs the second key after applying the final encryption doesn't work. Let us consider this for the case of the message being an integer multiple of the block size. Then the variant can be expressed as $\text{VMAC}(K, M) = \text{CBC}(K, M) \oplus K_1$. Now suppose an adversary is able to ask for the MACs of three messages: the message $\mathbf{0} = 0^n$, where n is the cipher block size; the message $\mathbf{1} = 1^n$; and the message $\mathbf{1} \| \mathbf{0}$. As a result of these three queries, the adversary gets $T_0 = \text{CBC}(K, \mathbf{0}) \oplus K_1$; $T_1 = \text{CBC}(K, \mathbf{1}) \oplus K_1$ and $T_2 = \text{CBC}(K, [\text{CBC}(K, \mathbf{1})]) \oplus K_1$. Show that the adversary can compute the correct MAC for the (unqueried) message $\mathbf{0} \| (T_0 \oplus T_1)$.

3.13 Prior to the discovery of any specific public-key schemes, such as RSA, an existence proof was developed whose purpose was to demonstrate that public-key encryption is possible in theory. Consider the functions $f_1(x_1) = z_1$; $f_2(x_2, y_2) = z_2$; $f_3(x_3, y_3) = z_3$, where all values are integers with $1 \le x_i, y_i, z_i \le N$. Function f_1 can be represented by a vector M1 of length N in which the kth entry is the value of $f_1(k)$. Similarly, f_2 and f_3 can be represented by $N \times N$ matrices M2 and M3. The intent is to represent the encryption/decryption process by table lookups for tables with very large values of N. Such tables would be impractically huge but in principle could be constructed. The scheme works as follows: Construct M1 with a random permutation of all integers between 1 and N; that is, each integer appears exactly once in M1. Construct M2 so that each row contains a random permutation of the first N integers. Finally, fill in M3 to satisfy the condition:

$$f_3(f_2(f_1(k), p), k) = p \quad \text{for all } k, p \text{ with } 1 \le k, p \le N$$

In words,

1. M1 takes an input k and produces an output x.
2. M2 takes inputs x and p giving output z.
3. M3 takes inputs z and k and produces p.

The three tables, once constructed, are made public.

a. It should be clear that it is possible to construct M3 to satisfy the preceding condition. As an example, fill in M3 for the following simple case:

M1 =

5
4
2
3
1

M2 =

5	2	3	4	1
4	2	5	1	3
1	3	2	4	5
3	1	4	2	5
2	5	3	4	1

M3 =

Convention: The ith element of M1 corresponds to $k = i$. The ith row of M2 corresponds to $x = i$; the jth column of M2 corresponds to $p = j$. The ith row of M3 corresponds to $z = i$; the jth column of M3 corresponds to $k = j$.

b. Describe the use of this set of tables to perform encryption and decryption between two users.

c. Argue that this is a secure scheme.

3.14 Perform encryption and decryption using the RSA algorithm (Figure 3.10) for the following:

a. $p = 3; q = 11, e = 7; M = 2$
b. $p = 5; q = 11, e = 3; M = 5$
c. $p = 7; q = 11, e = 17; M = 2$
d. $p = 11; q = 13, e = 11; M = 3$
e. $p = 17; q = 11, e = 7; M = 88$

Hint: Decryption is not as hard as you think; use some finesse.

3.15 In a public-key system using RSA, you intercept the ciphertext $C = 16$ sent to a user whose public key is $e = 6, n = 40$. What is the plaintext M?

3.16 In an RSA system, the public key of a given user is $e = 7, n = 137$. What is the private key of this user?

3.17 Suppose we have a set of blocks encoded with the RSA algorithm and we don't have the private key. Assume $n = pq$, e is the public key. Suppose also someone tells us they know one of the plaintext blocks has a common factor with n. Does this help us in any way?

3.18 Show how RSA can be represented by matrices M1, M2, and M3 of Problem 3.4.

3.19 Consider the following scheme.

1. Pick an odd number, E.
2. Pick two prime numbers, P and Q, where $(P - 1)(Q - 1)$ is relatively prime to E.
3. Multiply P and Q to get N.

5. Calculate $D = \dfrac{(P - 1)(Q - 1)(E + 1) + 1}{E}$

Is this scheme equivalent to RSA? Show why or why not.

3.20 Suppose Bob uses the RSA cryptosystem with a very large modulus n for which the factorization cannot be found in a reasonable amount of time. Suppose Alice sends a message to Bob by representing each alphabetic character as an integer between 0 and 25 ($A \rightarrow 0, \ldots, Z \rightarrow 25$), and then encrypting each number separately using RSA with large e and large n. Is this method secure? If not, describe the most efficient attack against this encryption method.

3.21 Consider a Diffie–Hellman scheme with a common prime $q = 353$ and a primitive root $\alpha = 3$.

a. If user A has public key $Y_A = 40$, what is A's private key X_A?

b. If user B has public key $Y_B = 248$, what is the shared secret key K?

This page intentionally left blank

PART TWO: NETWORK SECURITY APPLICATIONS

CHAPTER 4

KEY DISTRIBUTION AND USER AUTHENTICATION