Unit III: Message Authentication and Hash Functions

Contents:

3.1 Message Authentication

Encryption protects against passive attack (eavesdropping). A different requirement is to protect against active attack (falsification of data and transactions). Protection against such attacks is known as message or data authentication.

A message, file, document, or other collection of data is said to be authentic when it is genuine and came from its alleged source. Message or data authentication is a procedure that allows communicating parties to verify that received or stored messages are authentic. The two important aspects are to verify that the contents of the message have not been altered and that the source is authentic. We may also wish to verify a message's timeliness (it has not been artificially delayed and replayed) and sequence relative to other messages flowing between two parties. All of these concerns come under the category of data integrity.

Authentication Using Symmetric Encryption

It would seem possible to perform authentication simply by the use of symmetric encryption. If we assume that only the sender and receiver share a key (which is as it should be), then only the genuine sender would be able to encrypt a message successfully for the other participant, provided the receiver can recognize a valid message. Furthermore, if the message includes an error-detection code and a sequence number, the receiver is assured that no alterations have been made and that sequencing is proper. If the message also includes a timestamp, the receiver is assured that the message has not been delayed beyond that normally expected for network transit.

In fact, symmetric encryption alone is not a suitable tool for data authentication.

Message Authentication without Message Encryption

There are several approaches to message authentication that do not rely on message encryption. In all of these approaches, an authentication tag is generated and appended to each message for transmission. The message itself is not encrypted and can be read at the destination independent of the authentication function at the destination.

Because the approaches discussed in this section do not encrypt the message, message confidentiality is not provided. As was mentioned, message encryption by itself does not provide a secure form of authentication. However, it is possible to combine authentication and confidentiality in a single algorithm by encrypting a message plus its authentication tag. Typically, however, message authentication is provided as a separate function from message encryption. [DAVI89] suggests three situations in which message authentication without confidentiality is preferable:

1. There are a number of applications in which the same message is broadcast to a number of destinations. Two examples are notification to users that the network is now unavailable, and an alarm signal in a control center. It is cheaper and more reliable to have only one destination responsible for monitoring authenticity. Thus, the message must be broadcast in plaintext with

an associated message authentication tag. The responsible system performs authentication. If a violation occurs, the other destination systems are alerted by a general alarm.

2. Another possible scenario is an exchange in which one side has a heavy load and cannot afford the time to decrypt all incoming messages. Authentication is carried out on a selective basis, with messages being chosen at random for checking.
3. Authentication of a computer program in plaintext is an attractive service. The computer program can be executed without having to decrypt it every time, which would be wasteful of processor resources. However, if a message authentication tag were attached to the program, it could be checked whenever assurance is required of the integrity of the program. Thus, there is a place for both authentication and encryption in meeting security requirements.

Message Authentication Code

One authentication technique involves the use of a secret key to generate a small block of data, known as a message authentication code that is appended to the message. This technique assumes that two communicating parties, say A and B, share a common secret key KAB. When A has a message to send to B, it calculates the message authentication code as a complex function of the message and the key: MACM = F(KAB,M). The message plus code are transmitted to the intended recipient. The recipient performs the same calculation on the received message, using the same secret key, to generate a new message authentication code. The received code is compared to the calculated code (see Figure 2.3). If we assume that only the receiver and the sender know the identity of the secret key, and if the received code matches the calculated code, then:

Because messages may be any size and the message authentication code is a small fixed size, there must theoretically be many messages that result in the same MAC. However, it should be infeasible in practice to find pairs of such messages with the same MAC. This is known as collision resistance.

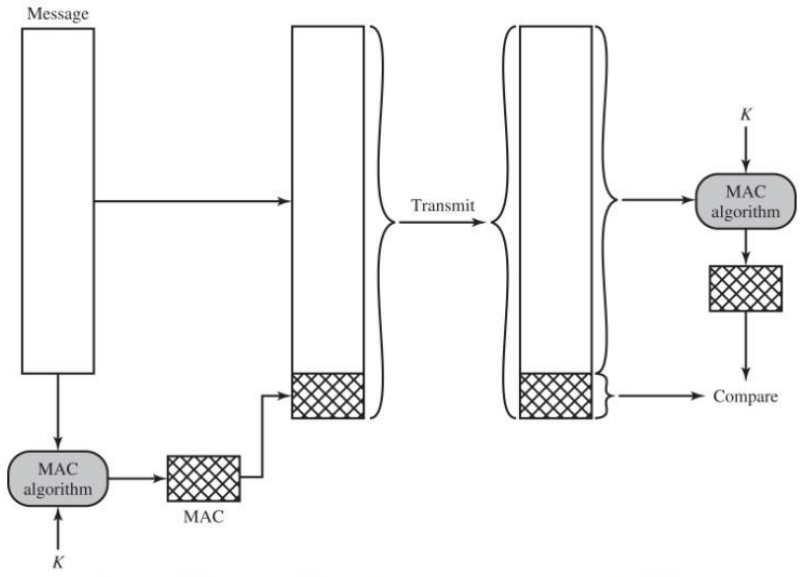


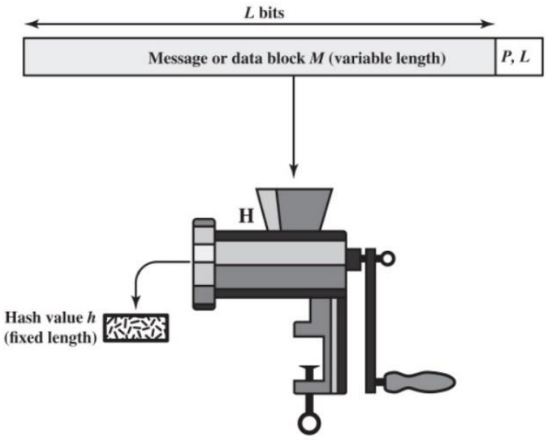
Figure 2.3 Message Authentication Using a Message Authentication Code (MAC)

1. The receiver is assured that the message has not been altered. If an attacker alters the message but does not alter the code, then the receiver's calculation of the code will differ from the received code. Because the attacker is assumed not to know the secret key, the attacker cannot alter the code to correspond to the alterations in the message.
2. The receiver is assured that the message is from the alleged sender. Because no one else knows the secret key, no one else could prepare a message with a proper code.

3. If the message includes a sequence number (such as is used with X.25, HDLC, and TCP), then the receiver can be assured of the proper sequence, because an attacker cannot successfully alter the sequence number.

3.2 Hash Functions

A hash function accepts a variable-size message M as input and produces a fixed-size message digest H(M) as output (see Figure 2.4). Typically, the message is padded out to an integer multiple of some fixed length (e.g., 1024 bits) and the padding includes the value of the length of the original message in bits. The length field is a security measure to increase the difficulty for an attacker to produce an alternative message with the same hash value.



P, *L* = padding plus length field

Figure 2.4 Cryptographic Hash Function; h=H(M)

Unlike the MAC, a hash function does not take a secret key as input. Figure 2.5 illustrates three ways in which the message can be authenticated using a hash function. The message digest can be encrypted using symmetric encryption (see Figure 2.5a); if it is assumed that only the sender and receiver share the encryption key, then authenticity is assured. The message digest can also be encrypted using public-key encryption (see Figure 2.5b); this is explained in Section 2.3. The public-key approach has two advantages: It provides a digital signature as well as message authentication, and it does not require the distribution of keys to communicating parties.

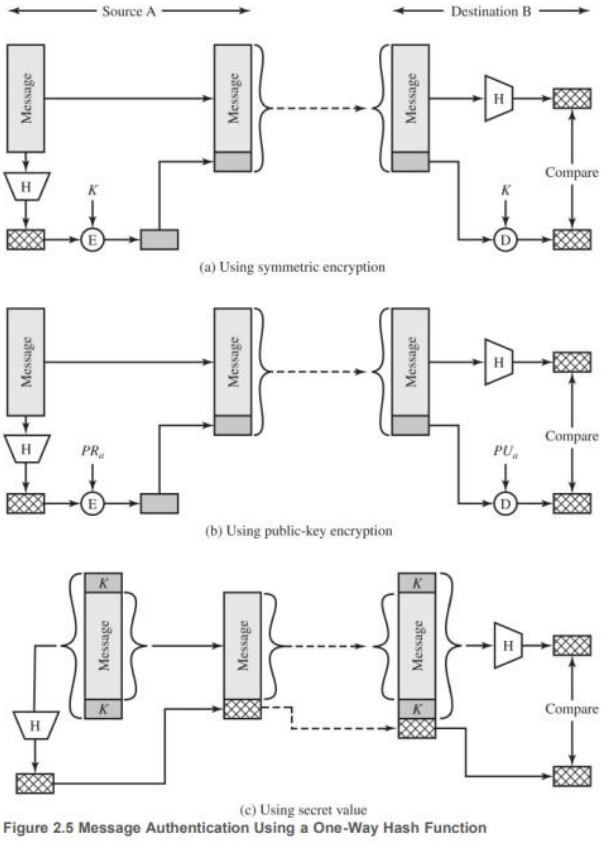


Figure 2.5 Message Authentication Using a One-Way Hash Function

Figure 2.5c shows a technique that uses a hash function but no encryption for message authentication. This technique, known as a keyed hash MAC, assumes that two communicating parties, say A and B, share a common secret key K. This secret key is incorporated into the process of generating a hash code. In the approach illustrated in Figure 2.5c, when A has a message to send to B, it

calculates the hash function over the concatenation of the secret key and the message: It then sends to B. Because B possesses K, it can recompute and verify, because the secret key itself is not sent, it should not be possible for an attacker to modify an intercepted message. As long as the secret key remains secret, it should not be possible for an attacker to generate a false message.

3.3 Message Digests: MD4 and MD5

A message digest (also known as a cryptographic checksum or cryptographic hashcode) is nothing more than a number - a special number that is effectively a hashcode produced by a function that is very difficult to reverse.

A message digest is also a hash function. It takes a variable length input - often an entire disk file - and reduces it to a small value (typically 128 to 512 bits). Give it the same input, and it always produces the same output. And, because the output is very much smaller than the potential input, for at least one of the output values there must be more than one input value that can produce it; we would expect that to be true for all possible output values for a good message digest algorithm.

There are two other important properties of good message digest algorithms:

1. The first is that the algorithm cannot be predicted or reversed. That is, given a particular output value, we cannot come up with an input to the algorithm that will produce that output, either by trying to find an inverse to the algorithm, or by somehow predicting the nature of the input required. With at least 128 bits of output, a brute force attack is pretty much out of the question, as there will be 1.7 x 1038 possible input values of the same length to try, on average, before finding one that generates the correct output. Compare this with some of the figures given in "Strength of RSA" earlier in this chapter, and you'll see that this task is beyond anything anyone would be able to try with current technology. With numbers as large as these, the idea that any two different documents produced at random during the course of human history would have the same 128-bit message digest is unlikely!
2. The second useful property of message digest algorithms is that a small change in the input results in a significant change in the output. Change a single input bit, and roughly half of the output bits should change. This is actually a consequence of the first property, because we don't want the output to be predictable based on the input. However, this aspect is a valuable property of the message digest all by itself.

The MD4/MD5 Message-Digest Algorithm is a hash-based cryptographic function. It takes a message of arbitrary length as its input and produces a 128-bit digest. Both MD4 and MD5 have a padding and appending process before digest the message of arbitrary length. The difference between MD4 and MD5 is the digest process. The MD4 have 3 round hash calculations while the MD5 have 4. For each round, both of them have intra loop-carried dependencies.

MD4 was developed by Rivest in 1990. The message is padded to ensure that its length in bits plus 448 is divisible by 512. A 64-bit binary representation of the original length of the message is then concatenated to the message. The message is processed in 512-bit blocks in the Damgård/Merkle iterative structure, and each block is processed in three distinct rounds. Attacks on versions of MD4 with either the first or the last rounds missing were developed very quickly by Den Boer, Bosselaers [DB92] and others. Dobbertin [Dob95] has shown how collisions for the full version of MD4 can be found in under a minute on a typical PC. In recent work, Dobbertin (Fast Software Encryption, 1998) has shown that a reduced version of MD4 in which the third round of the compression function is not executed but everything else remains the same, is not one-way. Clearly, MD4 should now be considered broken.

MD5 was developed by Rivest in 1991. It is basically MD4 with "safety-belts" and while it is slightly slower than MD4, it is more secure. The algorithm consists of four distinct rounds, which has a slightly different design from that of MD4. Message-digest size, as well as padding requirements, remain the same. Den Boer and Bosselaers [DB94] have found pseudo-collisions for MD5. More recent work by Dobbertin has extended the techniques used so effectively in the analysis of MD4 to find collisions for the compression function of MD5 [DB96b]. While stopping short of providing collisions for the hash function in its entirety this is clearly a significant step. For a comparison of these different techniques and their impact the reader is referred to [Rob96]

3.4 Secure Hash Algorithms: SHA-1

In recent years, the most widely used hash function has been the Secure Hash Algorithm (SHA). SHA was developed by the National Institute of Standards and Technology (NIST) and published as a federal information processing standard (FIPS 180) in 1993. When weaknesses were discovered in SHA, a revised version was issued as FIPS 180-1 in 1995 and is generally referred to as SHA-1.

HA-1 or Secure Hash Algorithm 1 is a cryptographic hash function which takes an input and produces a 160-bit (20-byte) hash value. This hash value is known as a message digest. This message digest is usually then rendered as a hexadecimal number which is 40 digits long. It is a U.S. Federal Information Processing Standard and was designed by the United States National Security Agency.

SHA-1 is now considered insecure since 2005. Major tech giants browsers like Microsoft, Google, Apple and Mozilla have stopped accepting SHA-1 SSL certificates by 2017.

In 2002, NIST produced a revised version of the standard, FIPS 180-2, that defined three new versions of SHA, with hash value lengths of 256, 384, and 512 bits, known as SHA-256, SHA-384, and SHA-512. These new versions, collectively known as SHA-2, have the same underlying structure and use the same types of modular arithmetic and logical binary operations as SHA-1. SHA-2, particularly the 512-bit version, would appear to provide unassailable security. However, because of the structural similarity of SHA-2 to SHA-1, NIST decided to standardize a new hash function that is very different from SHA-2 and SHA-1. This new hash function, known as SHA-3, was published in 2015 and is now available as an alternative to SHA-2.

3.5 HMAC

HMAC algorithm stands for Hashed or Hash-based Message Authentication Code. It is a result of work done on developing a MAC derived from cryptographic hash functions. HMAC is a great resistance towards cryptanalysis attacks as it uses the Hashing concept twice. HMAC consists of twin benefits of Hashing and MAC and thus is more secure than any other authentication code. RFC 2104 has issued HMAC, and HMAC has been made compulsory to implement in IP security. The FIPS 198 NIST standard has also issued HMAC.

Objectives:

- As the Hash Function, HMAC is also aimed to be one way, i.e, easy to generate output from input but complex the other way round.
- It aims at being less affected by collisions than the hash functions.
- HMAC reuses the algorithms like MD5 and SHA-1 and checks to replace the embedded hash functions with more secure hash functions, in case found.
- HMAC tries to handle the Keys in a simpler manner.

The working of HMAC starts with taking a message M containing blocks of length b bits. An input signature is padded to the left of the message and the whole is given as input to a hash function which gives us a

temporary message-digest MD'. MD' again is appended to an output signature and the whole is applied a hash function again, the result is our final message digest MD.

3.6 Digital Signatures

Public-key encryption can be used for authentication with a technique known as the digital signature. NIST FIPS PUB 186-4 [Digital Signature Standard (DSS), July 2013] defines a digital signature as follows: The result of a cryptographic transformation of data that, when properly implemented, provides a mechanism for verifying origin authentication, data integrity and signatory non-repudiation.

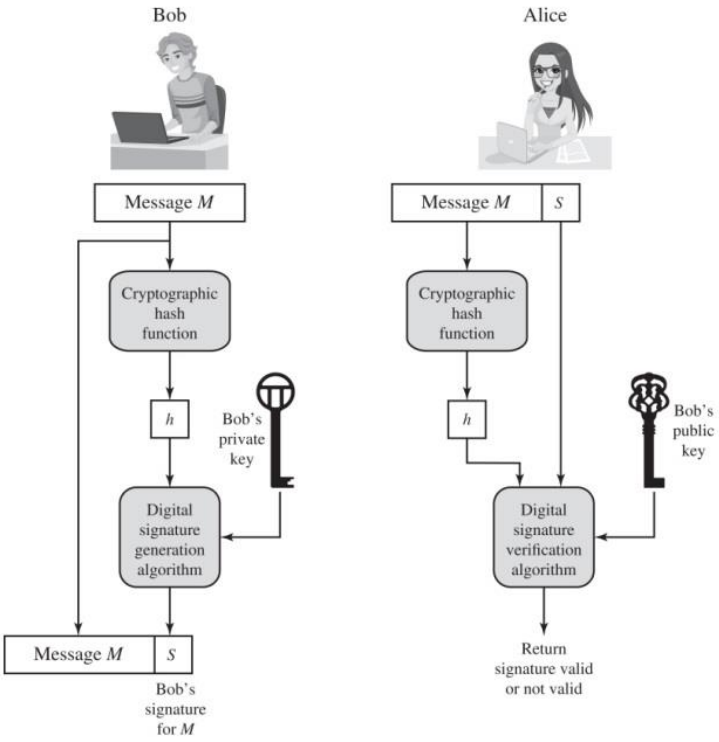
Thus, a digital signature is a data-dependent bit pattern, generated by an agent as a function of a file, message, or other form of data block. Another agent can access the data block and its associated signature and verify (1) the data block has been signed by the alleged signer, and (2) the data block has not been altered since the signing. Further, the signer cannot repudiate the signature.

FIPS 186-4 specifies the use of one of three digital signature algorithms:

- Digital Signature Algorithm (DSA): The original NIST-approved algorithm, which is based on the difficulty of computing discrete logarithms.
- RSA Digital Signature Algorithm: Based on the RSA public-key algorithm.
- Elliptic Curve Digital Signature Algorithm (ECDSA): Based on elliptic-curve cryptography.

Figure 2.7 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 Bob wants to send a message to Alice. Although it is not important that the message be kept 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 a hash value for the message; (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.

The digital signature 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.



(a) Bob signs a message (b) Alice verifies the signature

Figure 2.7 Simplified Depiction of Essential Elements of Digital Signature Process

Practice Questions

1. What is message authentication? Explain in detail.
2. Write short note on Hash functions.
3. Explain message digest along with important properties of good message digest algorithms.
4. What do you mean by SHA-1, SHA-2 and SHA-3?
5. Describe HMAC with its objectives.
6. What is a digital signature? Explain in detail.