10. 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 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 authentication is a procedure that allows communicating parties to verify that received message is 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 been artificially delayed and replayed) and sequence relative to other messages flowing between two parties.

10.1. Authentication Using Conventional Encryption

It is possible to perform authentication simply by the use of conventional 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. 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.

10.2. Message Authentication without Message Encryption

We examine several approaches to message authentication that do not rely on 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.

10. 2. 1 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 $K_{AB.}$ When A has a message to send to B, it calculates the message authentication code as a function of the message and the key: $MAC_M = F(K_{AB,}\ M)$. 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. If we assume that only the receiver and the sender know the identity of the key, and if the received code matches the calculate code, then

1. The receiver is assured that the message has not been altered.
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, then the receiver can be assured of the proper sequence, because an attacker cannot successfully alter the sequence number.

A number of algorithms could be used to generate the code. The national Bureau of Standards, in its publication DES Modes of Operation, recommends the use of Data Encryption Algorithm (DEA).

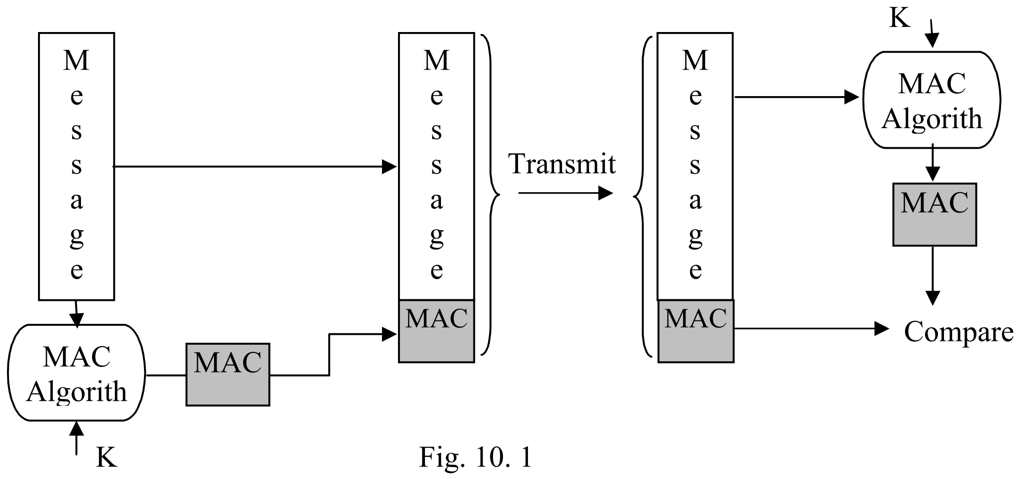

Fig. 10. 1

10. 3. Hash Functions

In general, **a hash function** is an efficiently evaluated function that takes an input string (usually binary) of arbitrary length and produces an output string of some fixed length, called **a hash value** or **message digest.** According to the dictionary, the word hash used as a verb can mean " to chop into pieces; to mince" or" to make a mess of; to mangle. "Both of these capture some of what hash functions are supposed to do. Chop up the input data and make " mess" of it so that the original data would be difficult or impossible to deduce from the mangled remains. Value provides a way of checking whether the message has been manipulated or corrupted in transit or storage. It is a sort of "digital fingerprint". Moreover, the message digest can be encrypted using either conventional or public-key cryptography to produce a **digital signature,** which is used to help the recipient feel confident that the received message is not forget.

The hash function H must be satisfied:

- It should be **one-way:** For a given hash value $v=H(x)$ it should be infeasible for an opponent to find a message x such that $x=H^{-1}(v)$.
- It should at least be **weakly collision resistant:** Given a hash value $v=H(x)$ and the message x from which it was computed, it should be computationally infeasible for an opponent to find another message y different from x such that $v =H(y)$.
- It might be **strongly collision resistant:** It is computationally infeasible for an opponent to find a pair of distinct messages x *and* y such that $H(x)=H(y)$.

Sender, recipient, and opponent all know the hash function. A recipient of a hashed message can check for manipulation by "hashing" the message (that is, applying the hash function to the received message) and comparing the result with the hash value that was pretended to the received message. If the two hash values agree, the probability is high that there was no manipulation; if the two disagree, it is certain that some manipulation or corruption in the data took place. Thus hash functions generate a sort of error detecting code.

Figure shows a technique that uses a hash function. This technique assumes that two communicating parties, say A and B, share a common secret value S_{AB}. When A has a message to send B, it calculates the hash function over the concatenation of the secret value and the message: $MD_M = H(S_{AB} \| M)$. It then sends $[M \| MD_M]$ to B. because B possesses S_{AB}, it can recomputed $H(S_{AB} \| M)$ and verify MD_M. because the secret value itself is not sent, it is not possible for an attacker to modify an intercepted message. As long as the secret value remains secret, it is also not possible for an attacker to generate a false message.

A variation on the third technique, called HMAC, is the one adopted for IP security.

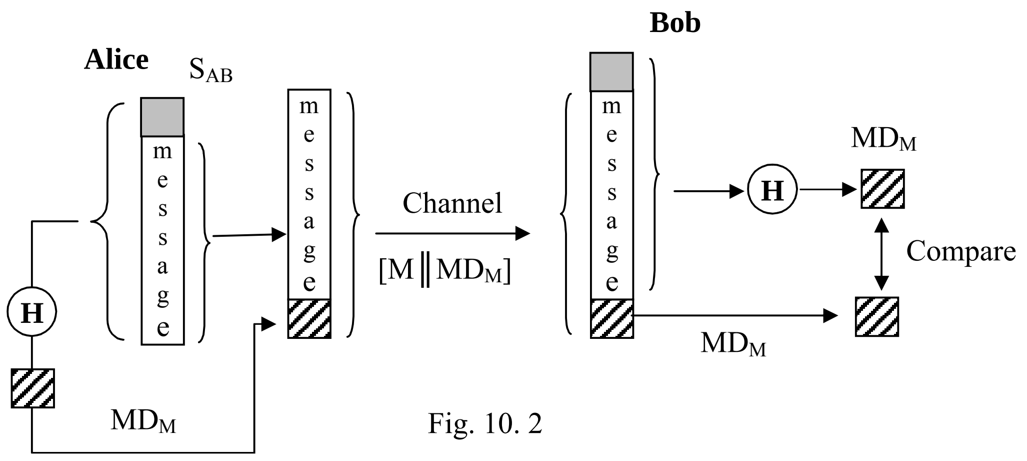

Fig. 10. 2

Example 1. Grace and Alan are concerned that their e-mails maybe intercepted and modified in transit. They agree to compute a hash value H(x) of an e-mail message x as follows. Group the letters(ignoring spaces and punctuation) into five-letter blocks and pad if necessary at the end to make the message length exactly a multiple of 5. Then , treating the letters a representing numbers in the range 0 to 25, they sum letters 1,6,11...modulo 26 to obtain the first hash letter y_1, sum letters 2,7,12,... modulo 26 to obtain a second hash letter y_2, and so on. The five letters representing these five sums are the value of this y_2, and so on. The five letters representing these five sums are the value of this hash function:

$H(x) = y_1 y_2 y_3 y_4 y_5$.The hash value will be sent first in a preliminary message, and then the message itself will be sent.

For example, suppose Grace wants to send the message:

It is much easier to apologise than it is to get permission.

X=ITISM UCHEA SIERT OAPOL OGIZE THANI TISTO GETPE RMISS IONXX

Alan then computes the hash letters by

$y_1 \equiv I+U+S+O+O+T+T+G+R+I = N (mod\ 26)$

$y_2 \equiv T+C+I+A+G+H+I+E+M+O = C (mod\ 26)$

$y_3 \equiv I+H+E+P+I+A+S+T+I+N = W (mod\ 26)$

$y_4 \equiv S+E+R+O+Z+N+T+P+S+X = K (mod\ 26)$

$y_5 \equiv M+A+I+L+E+I+O+E+S+X = J (mod\ 26)$

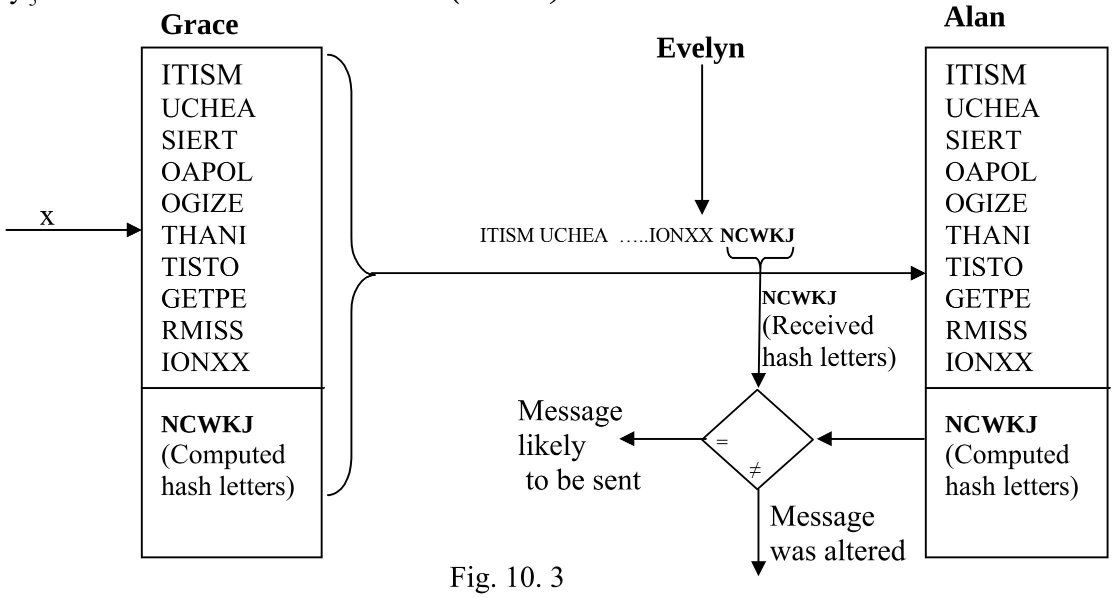

Fig. 10. 3

She first sends NCWKJ and then she sends the plaintext unencrypted.

On the other end, after receiving a hash value and a message, Alan sums the message letters in the same way to produce a hash word. It compares the message letters in the same way to produce a hash word. He compares this with the one that preceded the message. If the two are the same, he regards the message as likely to be the one Grace sent. If they are different, he is certain that either the message was altered, or the hash value was altered, or both. In this case, he rejects the received message.

Why does agreement between Alan's computed and received hash words make it likely that the message was unaltered? Suppose, for instance, that Evelyn, an opponent, knows the hashing algorithm, has learned the hash value Grace sent, and would like to alter or replace the message to Alan. If , for example, she changed the letters EASI to HARD. In the message, then she would change the message, but Alan would (you should verify) compute the hash word MXWNJ and compare it with NCWKJ to see that something was a miss. If Evelyn picked an intelligible English sentence at random, what would be the probability that it hashed to NCWKJ? An easier question is, If Evelyn picked a random string of letters, what would be the probability that it hashed to NCWKH? Assuming that the strings that do hash to NCWKJ are in some sense uniformly distributed through all strings, then this probability is $\frac{1}{26^5} = \frac{1}{11881376} \approx 8.4x10^{-8}(why?.)$. The sub collection of these strings hashing to NCWKJ that are also intelligible English sentences is only a tiny proportion, so the probability of Evelyn choosing an English text at random that hashes to NCWKJ is small indeed. Of course, knowing the details of the hashing algorithm, Evelyn might be able to improve her odds significantly. Hash functions are selected so that their values will be essentially uniformly distributed for the population of expected messages. If there are M possible messages, and z is a k-bit hash value, then there are about $M/2^k$ messages that hash to z, so the probability of an adversary guessing a message that hashes to z is about

$$\frac{M/2^k}{M} = \frac{1}{2^k}$$

10. 3. 1. Simple Hash Functions

All hash functions operate using the following general principles. The input (message, file, etc.) is viewed as a sequence of n-bit blocks. The input is processed one block at a time in an iterative fashion to produce an n-bit hash function.

One of the simplest hash functions is the bit-by-bit exclusive-OR (XOR) of every block. This can be expressed as follows:

$$C_i = b_{i1} \oplus b_{i2} \oplus \ldots \oplus b_{im}$$

Where

C_i = ith bit of the hash code, $1 \le i \le n$

m = number of n-bit blocks in the input

b_{ij} = ith bit in jth block

$\oplus$ = XOR operation

Figure illustrates this operation; it produces a simple parity for each bit position and is known as a longitudinal redundancy check. It is reasonably effective for random data as a data integrity check. Each n-bit hash value is equally likely. Thus, the probability that a data error will result in an unchanged hash value is 2^{-n}. with more predictably formatted data, the function is less effective. For example, in most normal text files, the high-order bit of each octet is always zero. So if a 128-bit hash value is used, instead of an effectiveness of 2^{-128}, the hash function on this type of data has an effectiveness of 2^{-112}.

	Bit1	Bit 2	…	Bit n
Block 1	b_{11}	b_{21}		b_{n1}
Block 2	b_{12}	b_{22}		b_{n2}
. . .	• • •	• • •	• • •	• • •
Block *m*	b_{1m}	b_{2m}		b_{nm}
Hash code	C_1	C_2		C_n

A technique originally proposed by the National Bureau of Standards used the simple XOR applied to 64-bit blocks of the message and then an encryption of the entire message that used the cipher block chaining (CBC) mode. We can define the scheme as follows: given a message consisting of a sequence of 64-bit blocks X_1, X_2, … X_N, define the hash code C as the block-by-block XOR or all blocks and append the hash code as the final block:

$$C = X_{N+1} = X_1 \oplus X_2 \oplus ... \oplus X_N$$

Next, encrypt the entire message plus hash code, using CBC mode to produce the encrypted message Y_1, Y_1,… Y_{N+1}.

10. 3. 2. The SHA-1 Secure Hash Function

The secure hash algorithm (SHA) was developed by the National Institute of Standards and Technology (NIST) and published as a federal information processing standard 1993; a revised version was issued as FIPS PUB 180-1 in 1995and is generally referred to as SHA-1.

The algorithm takes as input a message with a maximum length of less than 2^{64} bits and produces as output a 160-bit message digest. The input is processed in 512-bit blocks.

The SHA-1 algorithm has the property that every bit of the hash code is a function of every bit of the input. The complex repetition of the basic function f produces results that are well mixed, that is, it is unlikely that two messages chosen at random, even if they exhibit similarly regularities, will have the same hash code. Unless there is some hidden weakness in SHA-1, which has not so far been published, the difficulty of coming up with two messages having the same message digest is on the order of 2^{80} operations, while the difficulty of finding a message with a given digest is on the order of 2^{160} operations.

In this section we look at two other secure hash functions that, in addition to SHA-1, have gained commercial acceptance. Some of the principal characteristics are compared in table.

So with a secure hash function that produces, say, a 160-bit hash value- as does the federal **Secure Hash Standard** [8] – the probability of random guesswork leading to a message for a given hash value is

$$\frac{1}{2^{160}} = \frac{1}{1461501637330902918203684832716283019655932542976} = 6.8422810^{-49}$$

Presumably, very sophisticated guessing might raise this number a few orders of magnitude, but the probability will still be infinitesimal.

10. 4 MD5 Message Digest Algorithm

The MD5 message-digest algorithm (RFC 1321) was developed by Ron Rivest. Until the last few years, MD5 was the most widely used secure hash algorithm. Table shows comparison of Secure Hash Functions.

	MD5	**SHA-1**	**RIPEMD-160**
Digest lenght	128 bits	160 bits	160 bits
Basic unit of processing	512 bits	512 bits	512 bits
Number of steps	64 (4 rounds of 16)	80 (4 rounds of 20)	160(5 rounds of 16)
Maximum message size	∞	2^{64}-1 bits	∞
Primitive logical func.	4	4	5
Additive constants used	64	4	9

The algorithm takes as input a message of arbitrary length and produces as output a 128-bit message digest. The input is processed in 512-bit blocks.

As processor speeds have increased, the security of a 128-bit hash code has become questionable. It can be shown that the difficulty of coming up with two messages having the same message digest is on the order of 2^{64} operations, whereas the difficulty of finding a message with a given digest is on the on the order 2^{128} operations. The former figure is too small for security. Further, a number of cryptanalytic attacks have been developed that suggest the vulnerability of MD5 to cryptanalysis.

10. 4. 1 RIPEMD-160

The RIPEMD-160 message-digest algorithm was developed under the European RACE Integrity Primitives Evaluation (RIPE) project, by a group of researchers that launched partially successful attacks on MD4 and MD5. The group originally developed a 128-bit version of RIPEM. After the end of the RIPE project, H. Dobbertin (who was not a part of the RIPE project) found attacks on two rounds of RIPEMD, and later on MD4 and MD5. Because of these attacks, some members of the RIPE consortium decided to upgrade RIPEMD. The design work was done by them and by Dobbertin.

RIPEMD-160 is quite similar in structure to SHA-1. The algorithm takes as input a message of arbitrary length and produces as output a 160-bit message digest. The input is processed in 512 –bit blocks.

10. 4. 2 HMAC

In recent years, there has been increased interest in developing a MAC derived from a cryptographic hash code, such as SHA-1.

A hash function such as SHA-1 was not designed for use as a MAC and can not be used directly for that purpose because it does not rely on a secret key. There have been a number of proposals for the incorporation of a secret key into an existing hash algorithm. The approach that has received the most support is HMAC. HMAC has been issued as RFC 2104, has been chosen as the mandatory-to-implement MAC for IP Security, and is used in other Internet protocols, such as transport layer security (TLS, soon to replace secure sockets layer) and secure electronic transaction (SET).