84

Message Authentication

James S. Tiller

For centuries, various forms of encryption have provided confidentiality of information and have become integral components of computer communication technology. Early encryption techniques were based on shared knowledge between the communication participants. Confidentiality and basic authentication were established by the fact that each participant must know a common secret to encrypt or decrypt the communication, or as with very early encryption technology, the diameter of a stick.

The complexity of communication technology has increased the sophistication of attacks and has intensified the vulnerabilities confronting data. The enhancement of communication technology inherently provides tools for attacking other communications. Therefore, mechanisms are employed to reduce the new vulnerabilities that are introduced by new communication technology. The mechanisms utilized to ensure confidentiality, authentication, and integrity are built on the understanding that encryption alone, or simply applied to the data, will not suffice any longer. The need to ensure that the information is from the purported sender, that it was not changed or viewed in transit, and to provide a process to validate these concerns is, in part, the responsibility of message authentication.

AU7495—Chapter84—25/1/2007—20:56—PARTHIBAN—14744—XML MODEL CRC12a – pp. 1079–1094.

This chapter describes the technology of message authentication, its application in various communication environments, and the security considerations of those types of implementations.

84.1 History of Message Authentication

An encrypted message could typically be trusted for several reasons. First and foremost, the validity of the message content was established by the knowledge that the sender had the appropriate shared information to produce the encrypted message. An extension of this type of assumed assurance was also recognized by the possession of the encrypting device. An example is the World War II German Enigma, a diabolically complex encryption machine that used three or four wheels to produce ciphertext as an operator typed in a message. The Enigma was closely guarded; if it fell into the enemy's possession, the process of deciphering any captured encrypted messages would become much less complex. The example of the Enigma demonstrates that possession of a device in combination with the secret code for a specific message provided insurance that the message contents received were genuine and authenticated.

As the growth of communication technology embraced computers, the process of encryption moved away from complex and rare mechanical devices to programs that provided algorithms for encryption. The mechanical algorithm of wheels and electrical conduits was replaced by software that could be loaded onto computers, which are readily available, to provide encryption. As algorithms were developed, many became open to the public for inspection and verification for use as a standard. Once the algorithm was exposed, the power of protection was in the key that was combined with the clear message and fed into the algorithm to produce ciphertext.

84.2 Why Authenticate a Message?

The ability of a recipient to trust the content of a message is placed squarely on the trust of the communication medium and the expectation that it came from the correct source. As one would imagine, this example of open communication is not suitable for information exchange and is unacceptable for confidential or any form of valuable data.

There are several types of attacks on communications that range from imposters posing as valid participants replaying or redelivering outdated information, to data modification in transit.

Communication technology has eliminated the basic level of interaction between individuals. For two people talking in a room, it can be assured—to a degree—that the information from one individual has not been altered prior to meeting the listener's ears. It can be also assumed that the person that is seen talking is the originator of the voice that is being heard. This example is basic, assumed, and never questioned—it is trusted. However, the same type of communication over an alternate medium must be closely scrutinized due to the massive numbers of vulnerabilities to which the session is exposed.

Computers have added several layers of complexity to the trusting process and the Internet has introduced some very interesting vulnerabilities. With a theoretically unlimited number of people on a single network, the options of attacks are similarly unlimited. As soon as a message takes advantage of the Internet as a communication medium, all bets are off without layers of protection.

How are senders sure that what they send will be the same when it reaches the intended recipient? How can senders be sure that the recipients are who they claim to be? The same questions hold true for the recipients and the question of initiator identity.

84.3 Technology Overview

It is virtually impossible to describe message authentication without discussing encryption. Message authentication is nothing more than a form of cryptography and, in certain implementations, takes advantage of encryption algorithms.

84.3.1 Hash Function

Hash functions are computational functions that take a variable-length input of data and produce a fixed-length result that can be used as a fingerprint to represent the original data. Therefore, if the hashes of two messages are identical, it can be reasonably assumed that the messages are identical as well. However, there are caveats to this assumption, which are discussed later.

Hashing information to produce a fingerprint will allow the integrity of the transmitted data to be verified. To illustrate the process, Alice creates the message "Mary loves basketball," and hashes it to produce a smaller, fixed-length message digest, "a012f7." Alice transmits the original message and the hash to Bob. Bob hashes the message from Alice and compares his result with the hash received with the original message from Alice. If the two hashes match, it can be assumed that the message was not altered in transit. If the message was changed after Alice sent it and before Bob received it, Bob's hash will not match, resulting in discovering the loss of message integrity. This example is further detailed in Exhibit 84.1.

In the example, a message from Alice in cleartext is used as input for a hash function. The result is a message digest that is a much smaller, fixed-length value unique to the original cleartext message. The message digest is attached to the original cleartext message and sent to the recipient, Bob. At this point, the message and the hash value are in the clear and vulnerable to attack. When Bob receives the message, he separates the message from the digest and hashes the message using the same hash function Alice used. Once the hash process is complete, Bob compares his message digest result with the one included with the original message from Alice. If the two match, the message was not modified in transit.

The caveat to the example illustrated is that an attacker using the same hashing algorithm could simply intercept the message and digest, create a new message and corresponding message digest, and forward it on to the original recipient. The type of attack, known as the "man in the middle," described here is the driving reason why message authentication is used as a component in overall message protection techniques.

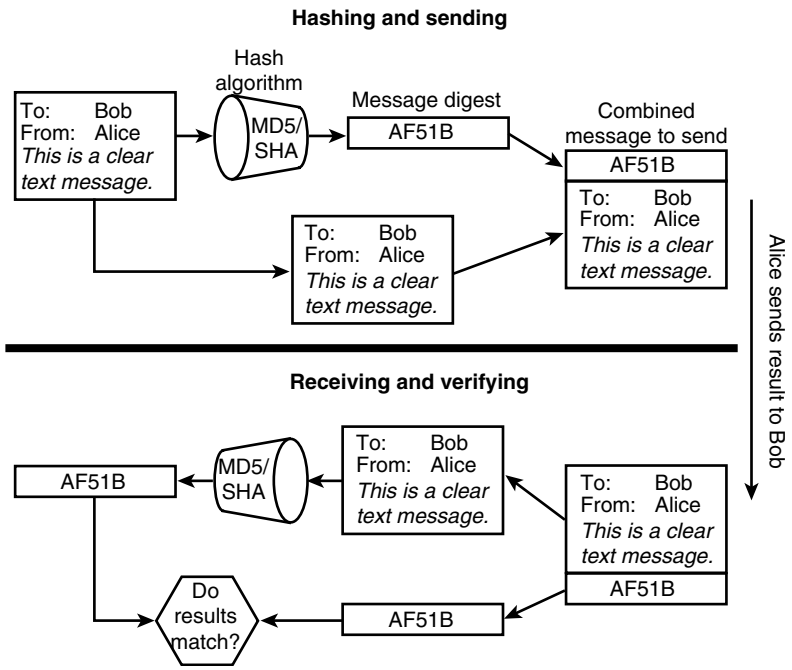

EXHIBIT 84.1 Hash function.

84.3.2 Encryption

Encryption, simply stated, is the conversion of plaintext into unintelligible ciphertext. Typically, this is achieved with the use of a key and an algorithm. The key is combined with the plaintext and computed with a specific algorithm.

There are two primary types of encryption keys: symmetrical and asymmetrical.

84.3.2.1 Symmetrical

Symmetrical keys, as shown in Exhibit 84.2, are used for both encryption and decryption of the same data. It is necessary for all the communication participants to have the same key to perform the encryption and decryption. This is also referred to as a shared secret.

In the example, Alice creates a message that is input into an encryption algorithm that uses a unique key to convert the clear message into unintelligible ciphertext. The encrypted result is sent to Bob, who has obtained the same key through a mechanism called "out-of-band" messaging. Bob can now decrypt the ciphertext by providing the key and the encrypted data as input for the encryption algorithm. The result is the original plaintext message from Alice.

84.3.2.2 Asymmetrical

To further accentuate authentication by means of encryption, the technology of public key cryptography, or asymmetrical keys, can be leveraged to provide message authentication and confidentiality.

Alice and Bob each maintain a private and public key pair that is mathematically related. The private key is well protected and is typically passphrase protected. The public key of the pair is provided to anyone who wants it and wishes to send an encrypted message to the owner of the key pair.

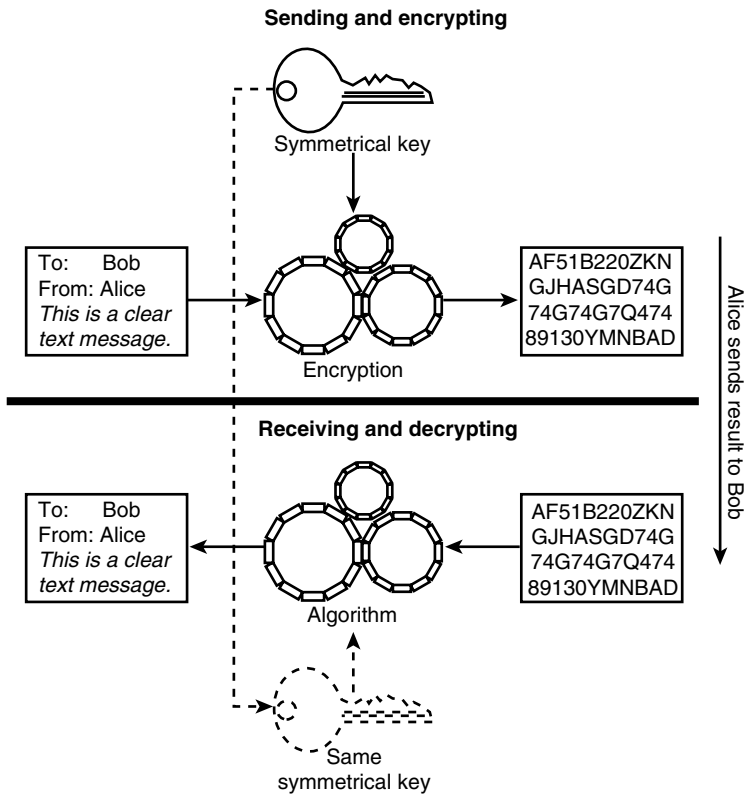

EXHIBIT 84.2 Symmetrical key encryption.

An example of public key cryptography, as shown in Exhibit 84.3, is that Alice could encrypt a message with Bob's public key and send the ciphertext to Bob. Because Bob is the only one with the matching private key, he would be the only recipient who could decrypt the message. However, this interaction only provides confidentiality and not authentication because anyone could use Bob's public key to encrypt a message and claim to be Alice.

As illustrated in Exhibit 84.3, the encryption process is very similar to normal symmetrical encryption. A message is combined with a key and processed by an algorithm to construct ciphertext. However, the key being used in the encryption cannot be used for decryption. As detailed in the example, Alice encrypts the data with the public key and sends the result to Bob. Bob uses the corresponding private key to decrypt the information.

To provide authentication, Alice can use her private key to encrypt a message digest generated from the original message, then use Bob's public key to encrypt the original cleartext message, and send it with the encrypted message digest. When Bob receives the message, he can use his private key to decrypt the message. The output can then be verified using Alice's public key to decrypt the message authentication that Alice encrypted with her private key. The process of encrypting information with a private key to allow the recipient to authenticate the sender is called digital signature. An example of this process is detailed in Exhibit 84.4.

The illustration conveys a typical application of digital signature. There are several techniques of creating digital signatures; however, the method detailed in the exhibit represents the use of a hash algorithm. Alice generates a message for Bob and creates a message digest with a hash function. Alice then encrypts the message digest with her private key. By encrypting the digest with her private key, Alice reduces the system load created by the processor-intensive encryption algorithm and provides an authenticator. The encrypted message digest is attached to the original cleartext message and encrypted

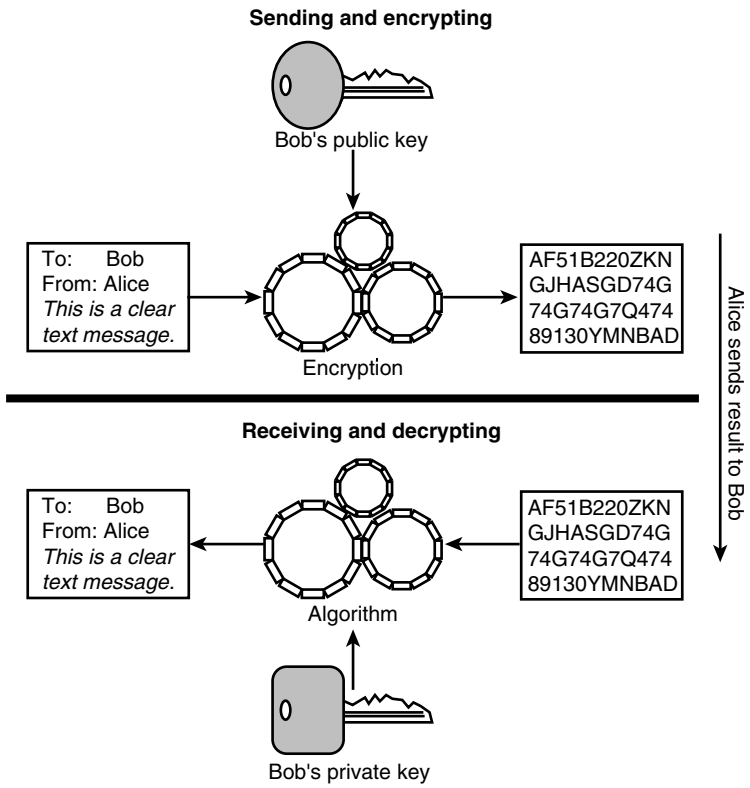

EXHIBIT 84.3 Asymmetrical key encryption.

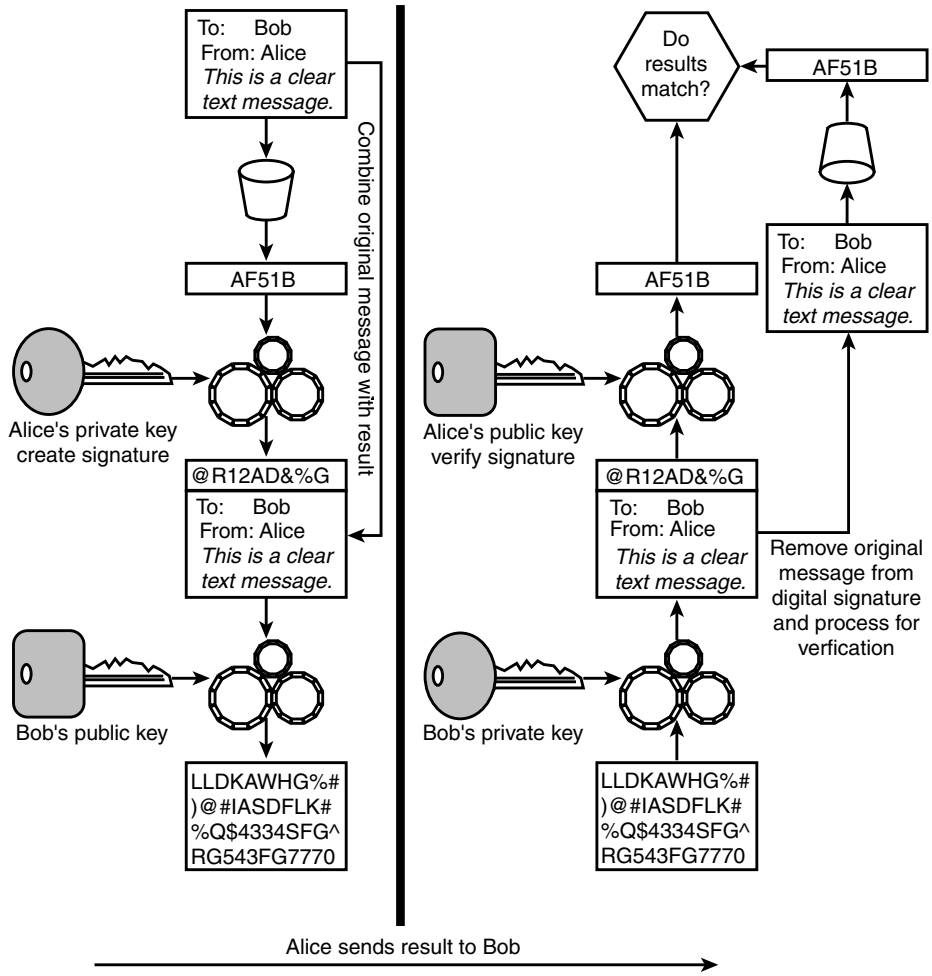

EXHIBIT 84.4 Digital signature with the use of hash functions.

using Bob's public key. The example includes the encrypted digest with the original message for the final encryption, but this is not necessary. The final result is sent to Bob. The entire package is decrypted with Bob's private key—ensuring recipient authentication. The result is the cleartext message and an encrypted digest. Bob decrypts the digest with Alice's public key, which authenticates the sender. The result is the original hash created by Alice that is compared to the hash Bob created using the cleartext message. If the two match, the message content has been authenticated along with the communication participants.

Digital signatures are based on the management of public and private keys and their use in the communication. The process of key management and digital signatures has evolved into certificates. Certificates, simply stated, are public keys digitally signed by a trusted Certificate Authority. This provides comfort in the knowledge that the public key being used to establish encrypted communications is owned by the proper person or organization.

84.3.3 Message Authentication Code

Message authentication code (MAC) with DES is the combination of encryption and hashing. As illustrated in Exhibit 84.5, as data is fed into a hashing algorithm, a key is introduced into the process.

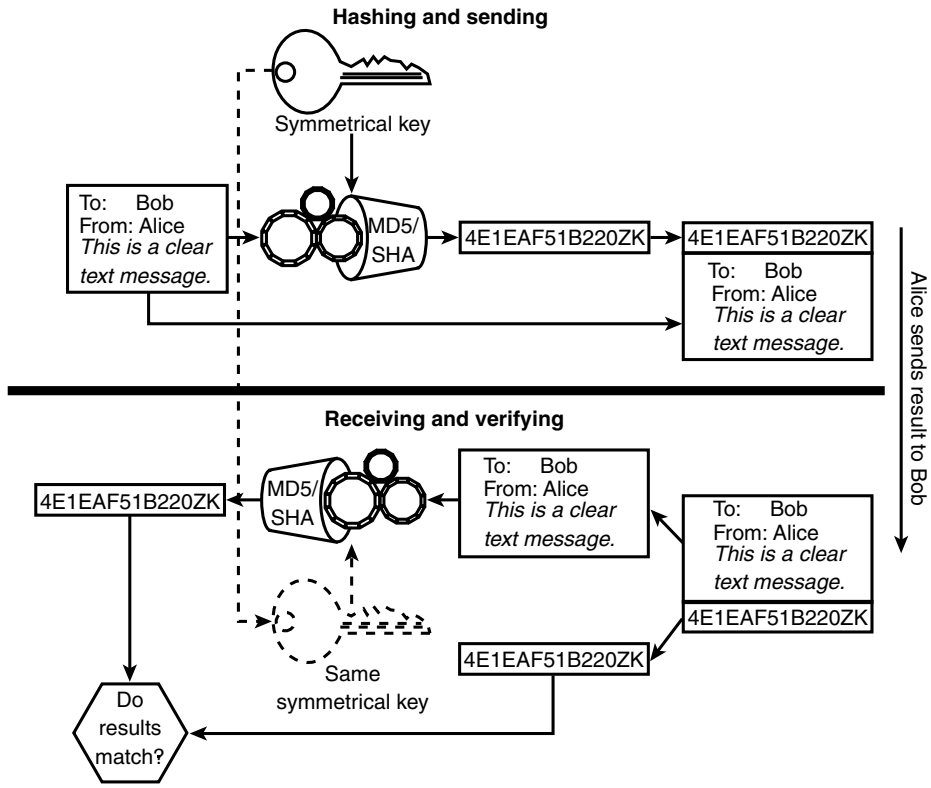

EXHIBIT 84.5 Message authentication code.

MAC is very similar to encryption but the MAC is designed to be irreversible, like a standard hash function. Because of the computational properties of the MAC process, and the inability to reverse the encryption designed into the process, MACs are much less vulnerable to attacks than encryption with the same key length. However, this does not prevent an attacker from forging a new message and MAC.

MAC ensures data integrity like a message digest but adds limited layers of authentication because the recipient would have to have the shared secret to produce the same MAC to validate the message.

The illustration of a message authentication code function appears very similar to symmetrical encryption; however, the process is based on compressing the data into a smaller fixed length that is not designed for decryption. A message is passed into the algorithm, such as DES–CBC (Cipher Block Chaining), and a symmetrical key is introduced. The result is much like that of a standard message digest, but the key is required to reproduce the digest for verification.

84.4 The Need for Authentication

As data is shared across networks—networks that are trusted or not—the opportunities for undesirables to interact with the session are numerous. Of the attacks that communications are vulnerable to, message authentication, in general application, addresses only a portion of the attacks. Message authentication is used as a tool to combine various communication-specific data that can be verified by the valid parties for each message received. Message authentication alone is not an appropriate countermeasure; but when combined with unique session values, it can protect against four basic categories of attacks:

1. Masquerading
2. Content modification

3. Sequence manipulation
4. Submission modification

To thwart these vulnerabilities inherent in communications, hash functions can be used to create message digests that contain information for origination authentication and timing of the communications. Typically, time-sensitive random information, or a nonce, is provided during the initialization of the session. The nonce can be input with the data in the hashing process or used as key material to further identify the peer during communications. Also, sequence numbers and time stamps can be generated and hashed for communications that require consistent session interaction—not like that of nontime-sensitive data such as e-mail. The process of authentication, verification through the use of a nonce, and the creation of a key for MAC computations provides an authenticated constant throughout the communication.

84.4.1 Masquerading

The process of masquerading as a valid participant in a network communication is a type of attack. This attack includes the creation of messages from a fraudulent source that appears to come from an authorized origin. Masquerading can also represent the acknowledgment of a message by an attacker in place of the original recipient. False acknowledgment or denial of receipt could complicate nonrepudiation issues. The nonce that may have been used in the hash or the creation of a symmetrical key assists in the identification of the remote system or user during the communication. However, to accommodate origin authentication, there must be an agreement on a key prior to communication. This is commonly achieved by a preshared secret or certificate that can be used to authenticate the initial messages and create specific data for protecting the remainder of the communication.

84.4.2 Content Modification

Content modification is when the attacker intercepts a message, changes the content, and then forwards it to the original recipient. This type of attack is quite severe in that it can manifest itself in many ways, depending on the environment.

84.4.3 Sequence Manipulation

Sequence manipulation is the process of inserting, deleting, or reordering datagrams. This type of attack can have several types of effects on the communication process, depending on the type of data and communication standard. The primary result is denial of service. Destruction of data or confusion of the communication can also result.

84.4.4 Submission Modification

Timing modification appears in the form of delay or replay. Both of these attacks can be quite damaging. An example is session establishment. In the event that the protocol is vulnerable to replay, an attacker could use the existence of a valid session establishment to gain unauthorized access.

Message authentication is a procedure to verify that the message received is from the intended source and has not been modified or made susceptible to the previously outlined attacks.

84.5 Authentication Foundation

To authenticate a message, an authenticator must be produced that can be used later by the recipient to authenticate the message. An authenticator is a primitive reduction or representation of the primary message to be authenticated. There are three general concepts in producing an authenticator.

84.5.1 Encryption

With encryption, the ciphertext becomes the authenticator. This is related to the trust relationship discussed earlier by assuming the partner has the appropriate secret and has protected it accordingly.

Consider typical encrypted communications: a message sent from Alice to Bob encrypted with a shared secret. If the secret's integrity is maintained, confidentiality is assured by the fact that no unauthorized entities have the shared secret.

Bob can be assured that the message is valid because the key is secret and an attacker without the key would be unable to modify the ciphertext in a manner to make the desired modifications to the original plaintext message.

84.5.2 Message Digest

As briefly described above, hashing is a function that produces a unique fixed-length value that serves as the authenticator for the communication. Hash functions are one-way, in that the creation of the hash is quite simple, but the reverse is infeasible. A well-constructed hash function should be collision resistant. A collision is when two different messages produce the same result or digest. For a function to take a variable length of data and produce a much smaller fixed-length result, it is mathematically feasible to experience collisions. However, a well-defined algorithm with a large result should have a high resistance to collisions.

Hash functions are used to provide message integrity. It can be argued that encryption can provide much of the same integrity. An example is an attacker could not change an encrypted message to modify the resulting cleartext. However, hash functions are much faster than encryption processes and can be utilized to enhance performance while maintaining integrity. Additionally, the message digest can be made public without revealing the original message.

84.5.3 Message Authentication Code

Message authentication code with DES is a function that uses a secret key to produce a unique fixed-length value that serves as the authenticator. This is much like a hash algorithm but provides the added protection by use of a key. The resulting MAC is appended to the original message prior to sending the data. MAC is similar to encryption but cannot be reversed and does not directly provide any authentication process because both parties share the same secret key.

84.6 Hash Process

As mentioned, a hash function is a one-way computation that accepts a variable-length input and produces a fixed-length result. The hash function calculates each bit in a message; therefore, if any portion of the original message changes, the resulting hash will be completely different.

84.6.1 Function Overview

A hash function must meet several requirements to be used for message authentication. The function must:

- Be able to accept any size data input
- Produce a fixed-length output
- Be relatively easy to execute, using limited resources
- Make it computationally impractical to derive a message from the digest (one-way property)
- Make it computationally impractical to create a message digest that is equal to a message digest created from different information (collision resistance)

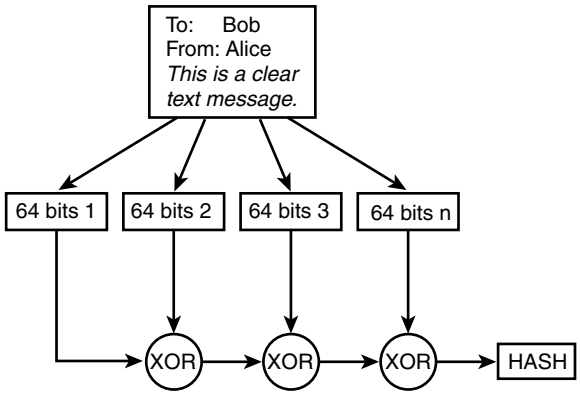

EXHIBIT 84.6 Simple hash function example.

Hash functions accommodate these requirements by a set of basic principles. A message is processed in a sequence of blocks, as shown in Exhibit 84.6. The size of the blocks is determined by the hash function. The function addresses each block one at a time and produces parity for each bit. Addressing each bit provides the message digest with the unique property that dramatic changes will occur if a single bit is modified in the original message.

As detailed in Exhibit 84.6, the message is separated into specific portions. Each portion is XOR with the next portion, resulting in a value the same size of the original portions, not their combined value. As each result is processed, it is combined with the next portion until the entire message has been sent through the function. The final result is a value the size of the original portions that were created and a fixed-length value is obtained.

84.7 Message Authentication Codes and Processes

Message authentication code with DES is applying an authentication process with a key. MACs are created using a symmetrical key so the intended recipient or the bearer of the key can only verify the MAC. A plain hash function can be intercepted and replaced or brute-force attacked to determine collisions that can be of use to the attacker. With MACs, the addition of a key complicates the attack due to the secret key used in its computation.

There are four modes of DES that can be utilized:

1. Block cipher-based
2. Hash function-based
3. Stream cipher-based
4. Unconditionally secure

84.7.1 Block Cipher-Based Mode

Block cipher-based message authentication can be derived from block cipher algorithms. A commonly used version is DES–CBC–MAC, which, simply put, is DES encryption based on the CBC mode of block cipher to create a MAC. A very common form of MAC is Data Authentication Algorithm (DAA), which is based on DES. The process uses the CBC mode of operation of DES with a zero initialization vector. As illustrated in Exhibit 84.7, the message is grouped into contiguous blocks of 64 bits; the last group is padded on the right with zeros to attain the 64-bit requirement. Each block is fed into the DES algorithm with a key to produce a 64-bit Data Authentication Code (DAC). The resulting DAC is XOR and the next

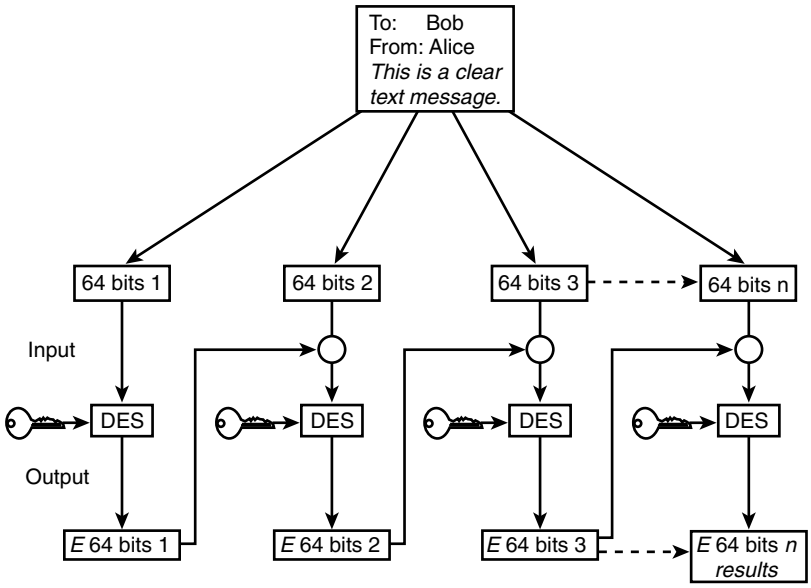

EXHIBIT 84.7 MAC based on DES–CBC.

64-bits of data is then fed again into the DES algorithm. This process continues until the last block, and returns the final MAC.

A block cipher is a type of symmetric key encryption algorithm that accepts a fixed block of plaintext to produce ciphertext of the same length—a linear relationship. There are four primary modes of operation on which the block ciphers can be based:

1. *Electronic Code Book (ECB)*. Electronic Code Book mode accepts each block of plaintext and encrypts it independently of previous block cipher results. The weakness in ECB is that identical input blocks will produce identical cipher results of the same length. Interestingly, this is a fundamental encryption flaw that affected the Enigma. For each input, there was a corresponding output of the same length. The "step" of the last wheel in an Enigma could be derived from determinations in ciphertext patterns.
2. *Cipher Block Chaining (CBC)*. With CBC mode, each block result of ciphertext is exclusively OR'ed (XOR) with the previous calculated block, and then encrypted. Any patterns in plaintext will not be transferred to the cipher due to the XOR process with the previous block.
3. *Cipher Feedback (CFB)*. Similar to CBC, CFB executes an XOR between the plaintext and the previous calculated block of data. However, prior to being XORed with the plaintext, the previous block is encrypted. The amount of the previous block to be used (the feedback) can be reduced and not utilized as the entire feedback value. If the full feedback value is used and two cipher blocks are identical, the output of the following operation will be identical. Therefore, any patterns in the message will be revealed.
4. *Output Feedback (OFB)*. Output Feedback is similar to CFB in that the result is encrypted and XORed with the plaintext. However, the creation of the feedback is generated independently of the ciphertext and plaintext processes. A sequence of blocks is encrypted with the previous block, the result is then XORed with the plaintext.

84.7.2 Hash Function-Based Mode

Hash function-based message authentication code (HMAC) uses a key in combination with hash functions to produce a checksum of the message. RFC 2104 defines that HMAC can be used with any

iterative cryptographic hash function (e.g., MD5, SHA-1) in combination with a secret shared key. The cryptographic strength of HMAC depends on the properties of the underlying hash function.

The definition of HMAC requires a cryptographic hash function and a secret key. The hash function is where data is hashed by iterating a basic compression function on blocks of data, typically 64 bytes in each block. The symmetrical key to be used can be any length up to the block size of the hash function. If the key is longer than the hash block size, the key is hashed and the result is used as the key for the HMAC function.

This process is very similar to the DES–CBC–MAC discussed above; however, the use of the DES algorithm is significantly slower than most hashing functions, such as MD5 and SHA-1.

HMAC is a process of combining existing cryptographic functions and a keyed process. The modularity of the standard toward the type of cryptographic function that can be used in the process has become the point of acceptance and popularity. The standards treat the hash function as a variable that can consist of any hash algorithm. The benefits are that legacy or existing hash implementations can be used in the process and the hash function can be easily replaced without affecting the process. The latter example represents an enormous security advantage. In the event the hash algorithm is compromised, a new one can be immediately implemented.

There are several steps to the production of an HMAC; these are graphically represented in Exhibit 84.8. The first step is to determine the key length requested and compare it to the block size of the hash being implemented. As described above, if the key is longer than the block size it is hashed, the result will match the block size defined by the hash. In the event the key is smaller, it is padded with zeros to accommodate the required block size.

Once the key is defined, it is XOR'ed with a string of predefined bits "A" to create a new key that is combined with the message. The new message is hashed according to the function defined (see Exhibit 84.6). The hash function result is combined with the result of XOR the key with another defined set of bits "B." The new combination of the second key instance and the hash results are hashed again to create the final result.

84.7.3 Stream Cipher-Based Mode

A stream cipher is a symmetric key algorithm that operates on small units of plaintext, typically bits. When data is encrypted with a stream cipher, the transformation of the plaintext into ciphertext is

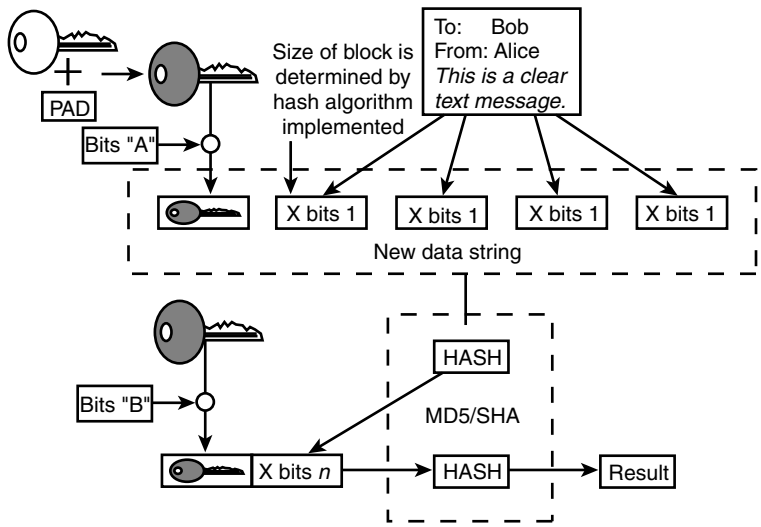

EXHIBIT 84.8 Simple HMAC example.

dependent on when the bits were merged during the encryption. The algorithm creates a keystream that is combined with the plaintext. The keystream can be independent of the plaintext and ciphertext (typically referred to as a synchronous cipher), or it can depend on the data and the encryption (typically referred to as self-synchronizing cipher).

84.7.4 Unconditionally Secure Mode

Unconditional stream cipher is based on the theoretical aspects of the properties of a one-time pad. A one-time pad uses a string of random bits to create the keystream that is the same length as the plaintext message. The keystream is combined with the plaintext to produce the ciphertext. This method of employing a random key is very desirable for communication security because it is considered unbreakable by brute force. Security at this level comes with an equally high price: key management. Each key is the same size and length as the message it was used to encrypt, and each message is encrypted with a new key.

84.8 Message Authentication over Encryption

Why use message authentication (e.g., hash functions and message authentication codes) when encryption seems to meet all the requirements provided by message authentication techniques? Following are brief examples and reasoning to support the use of message authentication over encryption.

84.8.1 Speed

Cryptographic hash functions, such as MD5 and SHA-1, execute much faster and use less system resources than typical encryption algorithms. In the event that a message only needs to be authenticated, the process of encrypting the entire message, such as a document or large file, is not entirely logical and consumes valuable system resources.

The reasoning of reducing load on a system holds true for digital signatures. If Alice needs to send a document to Bob that is not necessarily confidential but may contain important instructions, authentication is paramount. However, encrypting the entire document with Alice's private key is simply overkill. Hashing the document will produce a very small rendition of the original message, which then can be encrypted with her private key. The much smaller object encrypts quickly and provides ample authentication and abundant message integrity.

84.8.2 Limited Restrictions

No export restrictions on cryptographic functions are defined. Currently, the laws enforcing import and export restrictions in the international community are complicated and constantly changing. Basically, these laws are to control the level of technology and intellectual property of one country from another. Message authentication releases the communication participants from these restrictions.

84.8.3 Application Issues

There are applications where the same message is broadcast to several destinations. One system is elected as the communication monitor and verifies the message authentication on behalf of the other systems. If there is a violation, the monitoring system alerts the other systems.

Simple Network Management Protocol (SNMP) is an example where command messages can be forged or modified in transit. With the application of MAC, or HMAC, a password can be implemented to act as a key to allow a degree of authentication and message authentication. Each system in the community is configured with a password that can be combined with the data during the hash process and verified upon receipt. Because all the members are configured with the same password, the data can be hashed with the locally configured password and verified. It can also be forged at the destination.

84.8.4 System Operation

In the event that one of a communication pair is overburdened, the process of decryption would be overwhelming. Authentication can be executed in random intervals to ensure authentication with limited resources. Given the hashing process is much less intensive than encryption, periodical hashing and comparisons will consume fewer system cycles.

84.8.5 Code Checksum

Application authentication is achieved by adding the checksum to the program. While the program itself may be open to modification, the checksum can be verified at runtime to ensure that the code is in the original format and should produce the expected results. Otherwise, an attacker could have constructed a malicious activity to surreptitiously operate while the original application was running. It can be argued that if an attacker can modify the code, the checksum should pose little resistance because it can also be simply regenerated. Given the typically small size of checksums, it is typically published on several Web pages or included in an e-mail. In other words, an attacker would have to modify every existence of the checksum to ensure that the recipient would inadvertently verify the modified application. If encryption was utilized, the program would have to decrypt at each runtime, consuming time and resources. This is very important for systems that provide security functions, such as firewalls, routers, and VPN access gateways.

An example of the need for code protection can be illustrated by the heavy reliance on the Internet for obtaining software, updates, or patches. In early computing, systems patches and software were mailed to the recipient as the result of a direct request, or as a registered system user. As communication technology advanced, Bulletin Board Systems (BBS) could be directly accessed with modems to obtain the necessary data. In both of these examples, a fair amount of trust in the validity of the downloaded code is assumed.

In comparison, the complexity of the Internet is hidden from the user by a simple browser that is used to access the required files. The data presented in a Web page can come from dozens of different sources residing on many different servers throughout the Internet. There are few methods to absolutely guarantee that the file being downloaded is from a trusted source. To add to the complexity, mirrors can be established to provide a wider range of data sources to the Internet community. However, the security of a mirrored site must be questioned. The primary site may have extensive security precautions, but a mirror site may not. An attacker could modify the code on an alternate download location. When the code is finally obtained, a checksum can be validated to ensure that the code obtained is the code the creator intended for receipt.

84.8.6 Utilization of Existing Resources

There is available installed technology designed for DES encryption processes. The use of DEC–CBC–MAC can take advantage of existing technology to increase performance and support the requirements of the communication. The DES encryption standard has been available for quite some time. There are many legacy systems that have hardware designed specifically for DES encryption. As more advanced encryption becomes available and new standards evolve, the older hardware solutions can be utilized to enhance the message authentication process.

84.9 Security Considerations

The strength of any message authentication function, such as a MAC or hash, is determined by two primary factors:

1. One-way property
2. Collision resistance

One-way property is the ability of the hash to produce a message digest that cannot be used to determine the original message. This is one of the most significant aspects of message authentication algorithms. If a message authentication algorithm is compromised and a weakness is discovered, the result could have a detrimental effect on various forms of communication.

MD4 is an example of a function's poor one-way property. Within MD4, the data is padded to obtain a length divisible by 512, plus 448. A 64-bit value that defines the original message's length is appended to the padded message. The result is separated into 512-bit blocks and hashed using three distinct rounds of computation. Weaknesses were quickly discovered if the first or last rounds were not processed. However, it was later discovered that without the last round, the original message could be derived. MD4 had several computation flaws that proved the function had limited one-way capabilities.

Collision resistance is the most considered security aspect of message authentication functions. A collision is typically defined as when two different messages have the same hash result. In the event that a hash function has a collision vulnerability, such as MD2, a new message can be generated and used to replace the original in a communication, and the hash will remain valid. The combination of the original hash and the known vulnerability will provide the attacker with enough information to produce an alternative message that will produce the same checksum. An example is the hash algorithm MD2. It was created for 8-bit computers in the late 1980s and uses 16-bit blocks of the message against which to execute the hash. MD2 produces a 16-bit checksum prior to passing through the hash function. If this checksum is omitted, the production of a collision would be trivial. MD4 was subject to weak collision resistance as well, and it was proven that collisions could be produced in less than a minute on a simple personal computer.

The concept of a collision is a fundamental issue concerning probabilities. Take, for example, a hash function that produces an n-bit digest. If one is looking for a result of x, it can be assumed that one would have to try 2^n input possibilities. This type of brute-force attack is based on a surprising outcome referred to as the "birthday paradox": What is the least number of people in a group that can provide the probability, greater than half, that at least two people will have the same birthday?

If there are 365 days per year, and if the number of people exceeds 365, there will be a successful collision. If the number of people in the group is less than 365, then the number of possibilities is 365^n, where n is the number of people in a group. For those still wondering, the number of people, assuming there is a collision, is 23. This is a very small number; but when calculated against the number of possibilities that any two people's birthdays match, one sees that there are 253 possibilities. This is simply calculated as $n(n-1)/2$, which results in the probability of $P(365, 23) = .5073$, or greater than one half.

The birthday paradox states that given a random integer with a constant value between 1 and n, what is the selection of the number of permutations (the number of people required to meet 0.5 probability) that will result in a collision?

Given a fixed-length output that can represent an infinite amount of variation, it is necessary to understand the importance of a robust algorithm. It is also necessary for the algorithm to produce a relatively large result that remains manageable.

However, as certificates and other public key cryptography is utilized, message authentication processes will not be exposed to direct attack. The use of a hash to accommodate a digital signature process is based on the ownership and trust of a private key; the hash, while important, is only a step in a much more complicated process.

84.10 Conclusion

Communication technology has provided several avenues for unauthorized interaction with communications requiring the need to address security in ways previously unanalyzed. Message authentication provides a means to thwart various forms of attack and can enhance other aspects of communication security. A message "fingerprint" can be created in several ways, ranging from simple bit parity functions (hash) to utilization of encryption algorithms (DES–CBC–MAC) to complicated hybrids (HMAC). This

fingerprint cannot only be used to ensure message integrity, but also given the inherent process of message reduction, it lends itself to authentication and signature processes.

Message authentication is a broad activity that employs several types of technology in various applications to achieve timely, secure communications. The combinations of the application of these technologies are virtually limitless and, as advancements in cryptography, cryptanalysis, and overall communication technology are realized, message authentication will most certainly remain an interesting process.