88

Hash Algorithms: From Message Digests to Signatures

88.1 Why Hash Algorithms Are Needed and the Problems They Solve 1141
88.2 What Are Hash Algorithms? 1142
Functions • Secure Hash Algorithms • Secure Hash Algorithm • How SHA-1 Works • MD5 • Keyed Hash (HMAC)
88.3 How Hash Algorithms Are Used in Modern Cryptographic Systems 1146
Transport Layer Security (TLS) • IPSec • Digital Signatures • Other Applications
88.4 Problems with Hash Algorithms 1148
Birthday Attack
88.5 Looking to the Future 1148
SHA-256, -384, and -512
88.6 Summary 1148

Keith Pasley

There are many information-sharing applications that are in use on modern networks today. Concurrently, there are a growing number of users sharing data of increasing value to both sender and recipient. As the value of data increases among users of information-sharing systems, the risks of unauthorized data modification, user identity theft, fraud, unauthorized access to data, data corruption, and a host of other business-related problems mainly dealing with data integrity and user authentication, are introduced. The issues of integrity and authentication play an important part in the economic systems of human society. Few would do business with companies and organizations that do not prove trustworthy or competent.

For example, the sentence "I owe Alice US$500" has a hash result of "gCWXVcL3fPV8VrJNajm8J-KA==," while the sentence "I owe Alice US$5000" has a hash of "DSAyXRTza2bHLH46IPMrSq==." As can be seen, there is a big difference in hash results between the two sentences. If an attacker were trying to misappropriate the $4500 difference, hashing would allow detection.

88.1 Why Hash Algorithms Are Needed and the Problems They Solve

- Is the e-mail you received really from who it says it is?
- Can you ensure the credit card details you submit are going to the site you expected?

AU7495—Chapter88—25/1/2007—21:03—PARTHIBAN—14748—XML MODEL CRC12a – pp. 1141–1149.

- Can you be sure the latest anti-virus, firewall, or operating system software upgrade you install is really from the vendor?
- Do you know if the Web link you click on is genuine?
- Does the program hash the password when performing authentication or just passing it in the clear?
- Is there a way to know who you are really dealing with when disclosing your personal details over the Internet?
- Are you really you?
- Has someone modified a Web page or file without authorization?
- Can you verify that your routers are forwarding data only to authorized peer routers?
- Has any of the data been modified in route to its destination?
- Can hash algorithms help answer these questions?

88.2 What Are Hash Algorithms?

A hash algorithm is a one-way mathematical function that is used to compress a large block of data into a smaller, fixed-size representation of that data.

To understand the concept of hash functions, it is helpful to review some underlying mathematical structures. One such structure is called a function. When hash functions were first introduced in the 1950s, the goal was to map a message into a smaller message called a message digest. This smaller message was used as a sort of shorthand of the original message. The digest was used originally for detection of random and unintended errors in processing and transmission by data processing equipment

88.2.1 Functions

A function is a mathematical structure that takes one or more variables and outputs a variable. To illustrate how scientists think about functions, one can think of a function in terms of a machine (see Exhibit 88.1). The machine in this illustration has two openings. In this case the input opening is labeled *x* and the output opening is labeled *y*. These are considered traditional names for input and output. The following are the basic processing steps of mathematical functions:

1. A number goes in.
2. Something is done to it.
3. The resulting number is the output.

The same thing is done to every number input into the function machine. Step 2 above describes the actual mathematical transformation done to the input value, or hashed value, which yields the

EXHIBIT 88.1 The Hash Function

4×3	12
Drop the first digit (1) leaves	2
2×next number (3)	6
6×next number (7)	42
Drop the first digit (4) leaves	2
2×next number (3)	6
6×next number (8)	48
Drop the first digit (4)	8

resulting output, or hash result. In this illustration, Step 2 can be described as a mathematical rule as follows: $x+3=y$. In the language of mathematics, if x is equal to 1, then y equals 4. Similarly, if x is equal to 2, then y equals 5. In this illustration the function, or mathematical structure, called an algorithm, is: for every number x, add 3 to the number. The result, y, is dependent on what is input, x.

As another example, suppose that, to indicate an internal company product shipment, the number 43738 is exchanged. The hash function, or algorithm, is described as: multiply each number from left to right, and the first digit of any multiplied product above 9 is dropped. The hash function could be illustrated in mathematical notation as: $x\times$the number to the right $=y$ (see Exhibit 88.1).

The input into a hash algorithm can be of variable length, but the output is usually of fixed length and somewhat shorter in length than the original message. The output of a hash function is called a message digest. In the case of the above, the hash input was of arbitrary (and variable) length; but the hash result, or message digest, was of a fixed length of 1 digit, 8. As can be seen, a hash function provides a shorthand representation of the original message. This is also the concept behind error checking (checksums) done on data transmitted across communications links. Checksums provide a nonsecure method to check for message accuracy or message integrity. It is easy to see how the relatively weak mathematical functions described above could be manipulated by an intruder to change the hash output. Such weak algorithms could result in the successful alteration of message content leading to inaccurate messages. If you can understand the concept of what a function is and does, you are on your way to understanding the basic concepts embodied in hash functions. Providing data integrity and authentication for such applications requires reliable, secure hash algorithms.

88.2.2 Secure Hash Algorithms

A hash algorithm was defined earlier as a one-way mathematical function that is used to compress a large block of data into a smaller, fixed size representation of that data. An early application for hashing was in detecting unintentional errors in data processing. However, due to the critical nature of their use in the high-security environments of today, hash algorithms must now also be resilient to deliberate and malicious attempts to break secure applications by highly motivated human attackers—more so than by erroneous data processing. The one-way nature of hash algorithms is one of the reasons they are used in public key cryptography. A one-way hash function processes a bit stream in a manner that makes it highly unlikely that the original message can be deduced by the output value. This property of a secure hash algorithm has significance in situations where there is zero tolerance for unauthorized data modification or if the identity of an object needs to be validated with a high assurance of accuracy. Applications such as user authentication and financial transactions are made more trustworthy by the use of hash algorithms.

Hash algorithms are called secure if they have the following properties:

- The hash result should not be predictable. It should be computationally impractical to recover the original message from the message digest (one-way property).
- No two different messages, over which a hash algorithm is applied, will result in the same digest (collision-free property).

Secure hash algorithms are designed so that any change to a message will have a high probability of resulting in a different message digest. As such, the message alteration can be detected by comparing hash results before and after hashing. The receiver can tell that a message has suspect validity by the fact that the message digest computed by the sender does not match the message digest computed by the receiver, assuming both parties are using the same hash algorithm. The most common hash algorithms as of this writing are based on Secure Hash Algorithm-1 (SHA-1) and Message Digest 5 (MD5).

EXHIBIT 88.2 Output Bit Lengths

Hash Algorithm	Output Bit Length
SHA-1	160
SHA-256	256
SHA-384	384
SHA-512	512

88.2.3 Secure Hash Algorithm

SHA-1, part of the Secure Hash Standard (SHS), was one of the earliest hash algorithms specified for use by the U.S. federal government (see Exhibit 88.2). SHA-1 was developed by NIST and the NSA. SHA-1 was published as a federal government standard in 1995. SHA-1 was an update to the SHA, which was published in 1993.

88.2.4 How SHA-1 Works

Think of SHA-1 as a hash machine that has two openings, input and output. The input value is called the hashed value, and the output is called the hash result. The hashed values are the bit streams that represent an electronic message or other data object. The SHA-1 hash function, or algorithm, transforms the hashed value by performing a mathematical operation on the input data. The length of the message is the same as the number of bits in the message. The SHA-1 algorithm processes blocks of 512 bits in sequence when computing the message digest. SHA-1 produces a 160-bit message digest. SHA-1 has a limitation on input message size of less than 18 quintillion (that is, 2^{64} or 18,446,744,073,709,551,616) bits in length.

SHA-1 has five steps to produce a message digest:

1. Append padding to make message length 64 bits less than a multiple of 512.
2. Append a 64-bit block representing the length of the message before padding out.
3. Initialize message digest buffer with five hexadecimal numbers. These numbers are specified in the FIPS 180-1 publication.
4. The message is processed in 512-bit blocks. This process consists of 80 steps of processing (four rounds of 20 operations), reusing four different hexadecimal constants, and some shifting and adding functions.
5. Output blocks are processed into a 160-bit message digest.

88.2.5 MD5

SHA was derived from the secure hash algorithms MD4 and MD5, developed by Professor Ronald L. Rivest of MIT in the early 1990s. As can be expected, SHA and MD5 work in a similar fashion. While SHA-1 yields a 160-bit message digest, MD5 yields a 128-bit message digest. SHA-1, with its longer message digest, is considered more secure than MD5 by modern cryptography experts, due in part to the longer output bit length and resulting increased collision resistance. However, MD5 is still in common use as of this writing.

88.2.6 Keyed Hash (HMAC)

Modern cryptographers have found the hash algorithms discussed above to be insufficient for extensive use in commercial cryptographic systems or in private electronic communications, digital signatures, electronic mail, electronic funds transfer, software distribution, data storage, and other applications that require data integrity assurance, data origin authentication, and the like. The use of asymmetric

cryptography and, in some cases, symmetric cryptography, has extended the usefulness of hashing by associating identity with a hash result. The structure used to convey the property of identity (data origin) with a data object's integrity is hashed message authentication code (HMAC), or keyed hash.

For example, how does one know if the message and the message digest have not been tampered with? One way to provide a higher degree of assurance of identity and integrity is by incorporating a cryptographic key into the hash operation. This is the basis of the keyed hash or hashed message authentication code (HMAC). The purpose of a message authentication code (MAC) is to provide verification of the source of a message and integrity of the message without using additional mechanisms. Other goals of HMAC are as follows:

- To use available cryptographic hash functions without modification
- To preserve the original performance of the selected hash without significant degradation
- To use and handle keys in a simple way
- To have a well-understood cryptographic analysis of the strength of the mechanism based on reasonable assumptions about the underlying hash function
- To enable easy replacement of the hash function in case a faster or stronger hash is found or required

To create an HMAC, an asymmetric (public/private) or a symmetric cryptographic key can be appended to a message and then processed through a hash function to derive the HMAC. In mathematical terms, if $x = (\text{key} + \text{message})$ and $f = \text{SHA-1}$, then $f(x) = \text{HMAC}$. Any hash function can be used, depending on the protocol defined, to compute the type of message digest called an HMAC. The two most common hash functions are based on MD5 and SHA. The message data and HMAC (message digest of a secret key and message) are sent to the receiver. The receiver processes the message and the HMAC using the shared key and the same hash function as that used by the originator. The receiver compares the results with the HMAC included with the message. If the two results match, then the receiver is assured that the message is authentic and came from a member of the community that shares the key.

Other examples of HMAC usage include challenge-response authentication protocols such as Challenge Handshake Authentication Protocol (CHAP, RFC 1994). CHAP is defined as a peer entity authentication method for Point-to-Point Protocol (PPP), using a randomly generated challenge and requiring a matching response that depends on a cryptographic hash of the challenge and a secret key. Challenge-Response Authentication Mechanism (CRAM, RFC 2195), which specifies an HMAC using MD5, is a mechanism for authenticating Internet Mail Access Protocol (IMAP4) users. Digital signatures, used to authenticate data origin and integrity, employ HMAC functions as part of the "signing" process. A digital signature is created as follows:

1. A message (or some other data object) is input into a hash function (i.e., SHA-1, MD5, etc.).
2. The hash result is encrypted by the private key of the sender.

The result of these two steps yields what is called a *digital signature* of the message or data object. The properties of a cryptographic hash ensure that, if the data object is changed, the digital signature will no longer match it. There is a difference between a digital signature and an HMAC. An HMAC uses a shared secret key (symmetric cryptography) to "sign" the data object, whereas a digital signature is created by

EXHIBIT 88.3 Other Hash Algorithms

Hash Algorithm	Output Bit Length	Country
RIPEMD (160,256,320)	160, 256, 320	Germany, Belgium
HAS-160	160	Korea
Tiger	128,160,192	United Kingdom

using a private key from a private/public key pair (asymmetric cryptography) to sign the data object. The strengths of digital signatures lend themselves to use in high-value applications that require protection against forgery and fraud.

See Exhibit 88.3 for other hash algorithms.

88.3 How Hash Algorithms Are Used in Modern Cryptographic Systems

In the past, hash algorithms were used for rudimentary data integrity and user authentication; today hash algorithms are incorporated into other protocols—digital signatures, virtual private network (VPN) protocols, software distribution and license control, Web page file modification detection, database file system integrity, and software update integrity verification are just a few. Hash algorithms used in hybrid cryptosystems discussed next.

88.3.1 Transport Layer Security (TLS)

TLS is a network security protocol that is designed to provide data privacy and data integrity between two communicating applications. TLS was derived from the earlier Secure Sockets Layer (SSL) protocol developed by Netscape in the early 1990s. TLS is defined in IETF RFC 2246. TLS and SSL do not interoperate due to differences between the protocols. However, TLS 1.0 does have the ability to drop down to the SSL protocol during initial session negotiations with an SSL client. Deference is given to TLS by developers of most modern security applications. The security features designed into the TLS protocol include hashing.

The TLS protocol is composed of two layers:

1. The Record Protocol provides in-transit data privacy by specifying that symmetric cryptography be used in TLS connections. Connection reliability is accomplished by the Record Protocol through the use of HMACs.
2. TLS Handshake Protocol (really a suite of three subprotocols). The Handshake Protocol is encapsulated within the Record Protocol. The TLS Handshake Protocol handles connection parameter establishment. The Handshake Protocol also provides for peer identity verification in TLS through the use of asymmetric (public/private) cryptography.

There are several uses of keyed hash algorithms (HMAC) within the TLS protocol.

TLS uses HMAC in a conservative fashion. The TLS specification calls for the use of both HMAC MD5 and HMAC SHA-1 during the Handshake Protocol negotiation. Throughout the protocol, two hash algorithms are used to increase the security of various parameters:

- Pseudorandom number function
- Protect record payload data
- Protect symmetric cryptographic keys (used for bulk data encrypt/decrypt)
- Part of the mandatory cipher suite of TLS

If any of the above parameters were not protected by security mechanisms such as HMACs, an attacker could thwart the electronic transaction between two or more parties. The TLS protocol is the basis for most Web-based in-transit security schemes. As can be seen by this example, hash algorithms provide an intrinsic security value to applications that require secure in-transit communication using the TLS protocol.

88.3.2 IPSec

The Internet Protocol Security (IPSec) Protocol was designed as the packet-level security layer included in IPv6. IPv6 is a replacement TCP/IP protocol suite for IPv4. IPSec itself is flexible and modular in design, which allows the protocol to be used in current IPv4 implementations. Unlike the session-level security of TLS, IPSec provides packet-level security. VPN applications such as intranet and remote access use IPSec for communications security.

Two protocols are used in IPSec operations, Authentication Header (AH) and Encapsulating Security Payload (ESP). Among other things, ESP is used to provide data origin authentication and connectionless integrity. Data origin authentication and connectionless integrity are joint services and are offered as an option in the implementation of the ESP. RFC 2406, which defines the ESP used in IPSec, states that either HMAC or one-way hash algorithms may be used in implementations. The authentication algorithms are used to create the integrity check value (ICV) used to authenticate an ESP packet of data. HMACs ensure the rapid detection and rejection of bogus or replayed packets. Also, because the authentication value is passed in the clear, HMACs are mandatory if the data authentication feature of ESP is used. If data authentication is used, the sender computes the integrity check value (ICV) over the ESP packet contents minus the authentication data. After receiving an IPSec data packet, the receiver computes and compares the ICV of the received datagrams. If they are the same, then the datagram is authentic; if not, then the data is not valid, it is discarded, and the event can be logged. MD5 and SHA-1 are the currently supported authentication algorithms.

The AH protocol provides data authentication for as much of the IP header as possible. Portions of the IP header are not authenticated due to changes to the fields that are made as a matter of routing the packet to its destination. The use of HMAC by the ESP has, according to IPSec VPN vendors, negated the need for AH.

88.3.3 Digital Signatures

Digital signatures serve a similar purpose as those of written signatures on paper—to prove the authenticity of a document. Unlike a pen-and-paper signature, a digital signature can also prove that a message has not been modified. HMACs play an important role in providing the property of integrity to electronic documents and transactions. Briefly, the process for creating a digital signature is very much like creating an HMAC. A message is created, and the message and the sender's private key (asymmetric cryptography) serve as inputs to a hash algorithm. The hash result is attached to the message. The sender creates a symmetric session encryption key to optionally encrypt the document. The sender then encrypts the session key with the sender's private key, reencrypts it with the receiver's public key to ensure that only the receiver can decrypt the session key, and attaches the signed session key to the document. The sender then sends the digital envelope (keyed hash value, encrypted session key, and the encrypted message) to the intended receiver. The receiver performs the entire process in reverse order. If the results match when the receiver decrypts the document and combines the sender's public key with the document through the specified hash algorithm, the receiver is assured that (1) the message came from the original sender and (2) the message has not been altered. The first case is due to use of the sender's private key as part of the hashed value. In asymmetric cryptography, a mathematical relationship exists between the public and private keys such that either can encrypt and decrypt; but the same key cannot both encrypt and decrypt the same item. The private key is known only to its owner. As such, only the owner of the private key could have used it to develop the HMAC.

88.3.4 Other Applications

HMACs are useful when there is a need to validate software that is downloaded from download sites. HMACs are used in logging onto various operating systems, including UNIX. When the user enters a password, the password is usually run through a hash algorithm; and the hashed result is compared to a user database or password file.

An interesting use of hash algorithms to prevent software piracy is in the Windows XP registration process. SHA-1 is used to develop the installation ID used to register the software with Microsoft.

During installation of Windows XP, the computer hardware is identified, reduced to binary representation, and hashed using MD5. The hardware hash is an eight-byte value that is created by running ten different pieces of information from the PC's hardware components through the MD5 algorithm. This means that the resultant hash value cannot be backward-calculated to determine the original values. Further, only a portion of the resulting hash value is used in the hardware hash to ensure complete anonymity.

Unauthorized file modification such as Web page defacement, system file modification, virus signature update, signing XML documents, and signing database keys are all applications for which various forms of hashing can increase security levels.

88.4 Problems with Hash Algorithms

Flaws have been discovered in various hash algorithms. One such basic flaw is called the birthday attack.

88.4.1 Birthday Attack

This attack's name comes from the world of probability theory out of any random group of 23 people, it is probable that at least two share a birthday. Finding two numbers that have the same hash result is known as the birthday attack. If hash function f maps into message digests of length 60 bits, then an attacker can find a collision using only 230 inputs ($2^{f/2}$). Differential cryptanalysis has proven to be effective against one round of MD5. (There are four rounds of transformation defined in the MD5 algorithm.) When choosing a hash algorithm, speed of operation is often a priority. For example, in asymmetric (public/private) cryptography, a message may be hashed into a message digest as a data integrity enhancement. However, if the message is large, it can take some time to compute a hash result. In consideration of this, a review of speed benchmarks would give a basis for choosing one algorithm over another. Of course, implementation in hardware is usually faster than in a software-based algorithm.

88.5 Looking to the Future

88.5.1 SHA-256, -384, and -512

In the summer of 2001, NIST published for public comment a proposed update to the Secure Hash Standard (SHS) used by the U.S. government. Although SHA-1 appears to be still part of SHS, the update includes the recommendation to use hash algorithms with longer hash results. Longer hash results increase the work factor needed to break cryptographic hashing. This update of the Secure Hash Standard coincides with another NIST update—selection of the Rijndael symmetric cryptography algorithm for U.S. government use for encrypting data. According to NIST, it is thought that the cryptographic strength of Rijndael requires the higher strength of the new SHS algorithms. The new SHS algorithms feature similar functions but different structures. Newer and more secure algorithms, such as SHA-256, -384, and -512, may be integrated into the IPSec specification in the future to complement the Advanced Encryption Standard (AES), Rijndael. In May 2002, NIST announced that the Rijndael algorithm had been selected as the AES standard, FIPS 197.

88.6 Summary

Hash algorithms have existed in many forms at least since the 1950s. As a result of the increased value of data interactions and the increased motivation of attackers seeking to exploit electronic communications, the requirements for hash algorithms have changed. At one time, hashing was used to detect inadvertent

errors generated by data processing equipment and poor communication lines. Now, secure hash algorithms are used to associate source of origin with data integrity, thus tightening the bonds of data and originator of data. So-called HMACs facilitate this bonding through the use of public/private cryptography. Protocols such as TLS and IPSec use HMACs extensively. Over time, weaknesses in algorithms have been discovered and hash algorithms have improved in reliability and speed. The present digital economy finds that hash algorithms are useful for creating message digests and digital signatures.

Further Reading

http://www.deja.com/group/sci.crypt.