Covers All Exam Objectives for IINS 640-553

Includes Real-World Scenarios, Hands-On and Written Labs, and Leading-Edge Exam Prep Software Featuring:

- Custom Test Engine
- Hundreds of Sample Questions
- Electronic Flashcards
- Entire Book in PDF

CCNA®

Security

STUDY GUIDE

IINS Exam 640-553

Tim Boyles

SYBEX | SERIOUS SKILLS.

Table of Contents

Chapter 10

Using Digital Signatures

THE FOLLOWING CCNA SECURITY EXAM TOPICS ARE COVERED IN THIS CHAPTER:

- ✓ **Hashing overview**
- ✓ **Explain hash function and values**
- ✓ **Explain hashing algorithms**
- ✓ **Explain digital signatures**

In Chapter 10 we are going to discuss hashing functions. The hashing function will lay the groundwork for hashing values. To generate the hashing values, we will look at two primary hashing algorithms, MD5 and SHA-1.

Once you have an understanding of hashing and its components, we will discuss digital signatures. We will look at the features that make up digital signatures and the algorithms that generate them.

Hashing Overview

What is hashing? Simply put, hashing is taking some type of input data and generating some sort of value. This value is typically a fixed-length integer. The process of taking input data and generating the value is called a *hash function* . The output of the hash function is called the *hash value* .

There are a couple of other terms for hash value. One is *fingerprint* , because the value is supposed to be unique for any data, the same way a fingerprint is unique to a person. The other is *message digest* , or simply *digest* . The name of one popular hashing algorithm, Message Digest 5 (MD5), is based on this term.

We'll discuss the terminology and hashing process in greater detail throughout this chapter.

Hashing is used for many things within technology. It is used in file integrity, database indexing, and security features, to name a few. Hashing starts with data. The data can be as small as a one-page document or as large as a single program that is a gigabyte in size. A hashing program takes the data and runs it through a mathematical algorithm. The output of the mathematical algorithm is the hash value. In summary, hashing takes variable-length data and generates a fixed-length value.

To help explain hashing, we will look at a few types of hashing not specific to security. Have you ever downloaded a file from the Internet? Have you seen an SHA-1 or MD5 value next to the download? These SHA-1 and/or MD5 values are examples of *file integrity hashing* . The website hosting the download has taken the file you wish to download and run

it through a hashing program. The output is the hash value. It is listed on the website so that once you have completed your download, you can run it through your own hashing program and compare the hash values. If the value from your hashing program is the same as the hash value listed on the website, you know the program has not been altered or corrupted. Another thing you may have noticed is that the length of the value is always the same. This is called a *fixed-length output* .

HashCalc (`www.slavasoft.com/hashcalc/`) is a freeware program that allows you to calculate the hash values of a file.

Hashing is also used to improve search efficiency in databases. Table 10.1 is a list of names and addresses of people. Such a list inside a database is called a *data array* . Immediately you will notice that the names and addresses are not the same length. When data fields are different lengths, a search has to look at every field. This takes a long time in a large database.

TABLE 10.1 Data Array without Indexing

Name	Address
Teresa Von	123 Best Ave
Daniel Smith	4567 Sunset Lane
Robert Ramsalbottom	89101 Annapolis Circle

Now if we used a hashing algorithm against each line of data, it would generate a unique identifier. This process would be called database indexing via hashing.

- Teresa Von + 123 Best Ave = 7463
- Daniel Smith + 4567 Sunset Lane = 8374
- Robert Ramsalbottom + 89101 Annapolis Circle = 1923

Table 10.2 shows the result of hashing. To search, you take the search data and run it through the hashing program. This generates a unique identifier like the four-digit numbers shown in the Index column of the table. (Most hashing algorithms will produce a large identifier, but for this example and simplicity we are using a four-digit number.) Once the program knows the unique identifier, it can search a single field in the database—the index or search field—for a matching four-digit number. This is much more efficient than searching multiple fields of variable length.

TABLE 10.2 Data Array with Indexing

Index (Search Field)	Name	Address
7463	Teresa Von	123 Best Ave
8374	Daniel Smith	4567 Sunset Lane
1923	Robert Ramsalbottom	89101 Annapolis Circle

Now let us expand on the two examples. We saw in the first example how hashing was used for *integrity* and in the second example how it was used for*efficiency* . In security we can use both of these methods. Say we want to transmit data between two parties over the Internet. We want to guarantee the data has not been altered (integrity), and we want to transfer the data as efficiently as possible. A hashing algorithm is what we need. The sender will hash the data that needs to be sent. The sender will then send the data and in a separate transmission send the hash value. The receiver will receive the data and hash value. It will then run the data through its own hash algorithm and generate a hash value. Then it will compare the generated hash value with the hash value that was sent by the sender. If the values are equal, the receiver knows the data was not altered. This process is very efficient because only the original data and a small hash value are sent to verify the integrity.

Real World Scenario

Verifying the Integrity of Downloaded Software

The remote sales office for the XYZ Corporation is hosting a web server for software downloads. Customers access the web server from the Internet and download the software for installation on their computer. Customers have been complaining that the software is not installing correctly. It was determined that the software file on the web server was corrupted. Your boss has tasked you with providing a solution to guarantee that the software being downloaded is not corrupt.

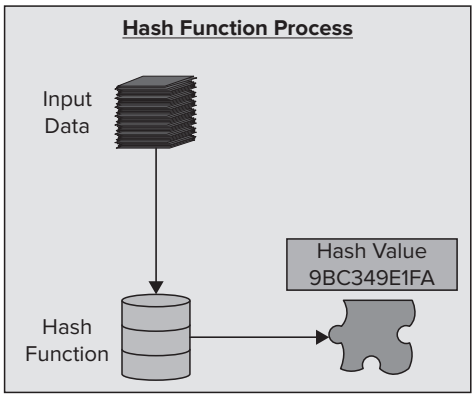
You decide to provide hashing values on the website for all software downloads. Customers will be able to use these hashing values to verify that the downloaded software is not corrupt.

Before any software is provided on the web server, it will be run through a hashing program to generate a hash value. The software will be listed on the website with the hash value displayed in a hexadecimal format next to the download link. Once the customers download the software, they can run the software file through a hashing program to generate the hash value. If the hash values the customers generate match the hash values listed on the website, the customers know the software is not corrupt.

Features of Hash Functions and Values

A hash function is a mathematical program that takes variable-length data and produces a fixed-length value. The process that takes place to generate the value from the data is called the *hash function* . Figure 10.1 shows the hash function process.

FIGURE 10.1 Hash function process



We start with the data. We need some type of data in order to hash. Once we have data to hash, the data then moves into the hash function. Each hashing algorithm computes the data differently.

Explaining the actual hashing algorithm computations is outside the scope of this book. The RFCs for the hashing algorithms would be a great place to view the mathematical computations. The RFCs for the common hashing algorithms are discussed in "Hashing Algorithms," later in this chapter.

Once the hash function has completed, it produces a hash value. The hash value is used to verify the integrity of the data after transfer.

We looked at the hash function from a logical view. Now we will look at the hash function from a simple equation perspective. Let's put some variables around each component in the hash function.

- *d*= Data to be hashed
- *F*= Hash function
- *h*= Hash value

As always, we will start with the data*d* . We put the data in the hash function*F(d)* . The output of *F(d)* will provide*h* . In simple equation terms*F(d) = h* .

In security applications, one of the essential features of a hash function is that the equation works only one way. So $h = F(d)$ cannot be true. You cannot take the h and determine the d. This is called a *one-way hash*.

Also, the hash function must not take a different d and generate the same h. If a different d generates an identical h, this is called a collision.

Now that you know how a hash function works, we need to look at what makes up a hash function. A hash function has five main features:

- Easily compute the hash value for any message
- Must never create the same hash value from two different sets of data
- Cannot modify the message without altering the hash value
- Cannot determine the message from the hash value
- Take variable-length data and produce a fixed-length value

Let us look at each of these features in greater detail. Table 10.3 will take the hash function features and turn them into simple terms.

TABLE 10.3 Hash Function Terms

Hash Function Features	Simple Terms
Easily compute the hash value for any message	Fast and efficient
Must never create the same hash value from two different sets of data	Collision resistant
Cannot modify the message without altering the hash value	Manipulation resistant
Cannot determine the message from the hash value	One-way hash
Take variable-length data and produce a fixed-length value	Fixed-length hash value

Fast and Efficient

A hash function must be fast and efficient, especially if it is used in cryptography. For example, a sender wants to start a communication with a receiver. If that communication is going to be hashed when sent and also when received, the hash function has to work as fast as possible so as not to delay the communication. This is especially important for digital signatures (see "Digital Signatures," later in this chapter, for more information).

Collision Resistant

Collisions are usually undesirable, and in the world of hashing this holds true as well. A collision occurs when the hashing algorithm produces the same hash value for two different sets of data. Because hashing is used for integrity, it is very important that the

hashing algorithm does not produce duplicate hashes for different data. Newer hashing algorithms are being created and deployed because of the collision issues seen in some of the most prominent algorithms like MD5 and SHA-1. Collisions may happen in everyday use of hashing but may go unseen by the average administrator.

Just to clarify, the collision is not physical in nature—two hashes are not bouncing around on the wire and hitting each other. The term means the algorithm produced the same hash value on two different types of data.

Collisions do not go unnoticed by hackers, however. If a hacker can make a hashing algorithm cause a collision, they can start to reverse engineer the algorithm. This is particularly important to hackers when they try to crack passwords. If a hacker knows how to manipulate the hash, they can create a table of known hashes. These tables are called *rainbow tables* . With a list of rainbow tables, a hacker can reverse engineer the hash and gain access to the data, in this case the password.

Manipulation Resistant

Just as two different data sets need to produce different hashes, it is important that changes to a data set produce a different hash. If there is any change in the data, no matter how minimal, the hash must change. If a hacker could modify the data and the hash stayed the same, the hashing function would be useless.

To demonstrate this feature, we will put a simple phrase through two different hashing functions, MD5 and SHA-1 (discussed in more detail in "Hashing Algorithms," later in this chapter). Then we will alter the phrase by one character and see how the hashing values change.

- "I will pass the Cisco CCNA Security Test!"
  - MD5 = f3fe82c8a8059137a9d65669f76bffe6
  - SHA-1 = b81fc23a64c6478d3738c58ff3d2f9e669da80c5
- "I will pass the Cisco CCNA Security Pest!"
  - MD5 = 89fc1190eeaa0ca0bbff176fc9f218ea
  - SHA-1 = 90fff0bff553ddd534e41f1f71a4b2f59930d296

All we did was change "Test" to "Pest." A simple letter change completely modified the hash value. This is manipulation resistance at work.

One-Way Hashing

A one-way hash takes data and creates a hash that cannot be used to re-create the data (although in some cases a one-way hash can be reverse engineered by a hacker). Two-way hashes also exist but are less common. With a two-way hash, you can re-create the data from the hash. In this chapter we are concerned only with one-way hashing.

Fixed-Length Hashing Values

Most hash values are of fixed length—that is, the hash value generated by the hashing algorithm is always the same length, no matter how much data is computed.

The two most common hashing algorithms are the MD5 and SHA-1 algorithms. The MD5 algorithm produces a 128-bit or 32-character hexadecimal hash value. The SHA-1 algorithm produces a 160-bit or 40-character hexadecimal hash value. Here are sample hash values of the same input data.

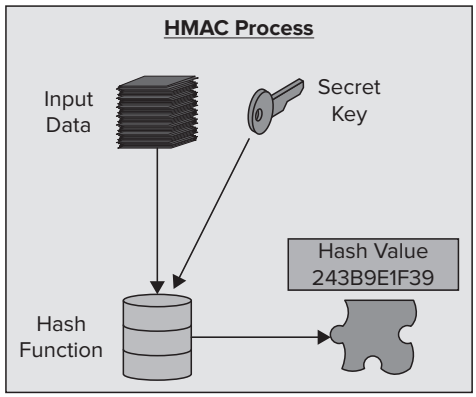
- MD5 = d41d8cd98f00b204e9800998ecf8427e
- SHA-1 = a9993e364706816aba3e25717850c26c9cd0d89d

Hash Message Authentication Code

Hash Message Authentication Code (HMAC) is a way to further secure a hash. HMAC is not a hash function requirement but has its place when we talk about securing the hash function. Because some popular hash algorithms have been shown not to be completely collision resistant, it is important to add newer techniques to validate the integrity of a hash. HMAC accomplishes this by adding another layer of data into the hashing mix. This layer is called a *secret key* . The secret key is known only by the sender and receiver, and it provides authentication to HMAC.

In the HMAC process, the input data is taken and a secret key is added. Both the input data and secret key are put through the hashing algorithm. This produces an *HMAC hash* . The size of the HMAC hash is the same as that of the corresponding hashing algorithm. (The two main types of HMAC hashes are HMAC-MD5, which produces a 128-bit hash, and HMAC-SHA-1, which produces a 160-bit hash.) Figure 10.2 shows the HMAC process.

FIGURE 10.2 HMAC process



The receiver verifies the data by taking the received data and running it through its own hashing algorithm and adding the secret key. It then compares the hash values. If the hash values are equal, then the message has integrity and simple authentication. The authentication is gained by the secret key shared between the sender and receiver. As long as no one else gains access to the secret key, the HMAC hash is secure. As with any secret key or password, you must safeguard it to prevent unauthorized usage.

To expand on HMAC we will use our equation from the hash function section. We will add a variable *s* for the secret key.

- *d* = Data to be hashed
- *F*= Hash function
- *h*= Hash value
- *s*= Secret key

We will start with the data *d*. We put the data and secret key in the hash function *F(d + s)*. The output of *F(d + s)* will provide *h*. In simple equation terms *F(d + s) = h*. This too is a one-way hash, so *h = F(d + s)* is not true (you cannot generate the original data from the hash value). The receiver takes the data, adds a secret key, and computes the hash. As long as the data and secret keys are the same, the hash values should be equal.

Hashing Algorithms

In this section we will talk about the mathematical computations used to create the hashing algorithms. The two specific hashing algorithms we will discuss are Message Digest 5 (MD5) and Secure Hash Algorithm (SHA-1).

MD5 Algorithm

In this section we will discuss the MD5 hashing algorithm. We will start with the creation and evolution of the algorithm. We will move into the features and finally the advantages and disadvantages of using MD5.

The MD5 algorithm was invented in 1991 by Ronald Rivest of MIT.

Ronald Rivest is the same Rivest who helped create the RSA algorithm. He is the *R* in RSA.

MD5 is based on an older hashing algorithm called MD4. The best-known use of MD4 is with Microsoft Windows NT-Hash, which is used in NT, XP, and Vista operating systems. The MD4 hashing algorithm was cracked in the late nineties. MD5 fixed the weakness in MD4 and was ratified in 1992 in RFC 1321. MD5 was designed to run efficiently on 32-bit processors.

The MD5 algorithm produces a fixed-length hash value, 128 bits in length. For binary people that is four times the length of an IPv4 address, or the same size as an IPv6 address. For the non-binary people it is a 32-character hexadecimal number. You may see this number when you download a file from the Internet.

MD5 is very collision resistant. The algorithm was designed to generate unique hash values for each unique input. However, lately there have been rumblings in the security community about the weaknesses in MD5. Many government agencies will be required to move to a stronger algorithm in a few years.

The following list describes the advantages and disadvantages of MD5.

- Advantages of MD5
  - Utilizes a fast computation algorithm
  - Provides collision resistance
  - Is in widespread use
  - Provides a one-way hash
- Disadvantages of MD5
  - Has known security flaws and vulnerabilities
  - Is less secure than the SHA-1 algorithm

SHA-1 Algorithm

In this section we will discuss the Secure Hash Algorithm version 1 (SHA-1). We will look at the birth of SHA-1 and then move to the features and functions. We will finish this section with the advantages and disadvantages of SHA-1.

SHA was created in 1993 by the National Institute of Standards and Technology (NIST). Soon after its creation a flaw was uncovered. The original version with the flaw was named SHA-0 and withdrawn from use. The revised version, SHA-1, came out in 1995. SHA-1 is the most widely used version. For further details on SHA-1 refer to RFC 3174.

There are two newer versions of SHA. SHA-2 is available for use and has options for the digest lengths. SHA-3 is in development to fix known issues in SHA-1 and SHA-2. SHA-2 and SHA-3 are outside the realm of this chapter.

SHA-1 produces a fixed-length hash value of 160 bits, or 40 hexadecimal characters. SHA-1 is able to process only input data with a maximum length of $(2^{64} - 1)$ bits. SHA-1 is based on the MD4 and MD5 hashing algorithms. SHA-1 produces a larger hash value (160 bits) than MD5 (128 bits), thus making it harder to crack with brute force or reverse engineering. However, the computation time for this longer hash value makes SHA-1 slower than MD5.

The following list describes the advantages and disadvantages of SHA-1.

- Advantages of SHA-1
  - Produces a longer hash value than MD5
  - Is collision resistant
  - Is in widespread use
  - Provides a one-way hash
- Disadvantages of SHA-1
  - Is a slower computational algorithm than the MD5 algorithm
  - Has known security vulnerabilities

MD5 and SHA-1 Comparison Chart

Table 10.4 provides a high-level comparison between the MD5 and SHA-1 hashing algorithms.

TABLE 10.4 Comparing MD5 to SHA-1

	MD5	SHA-1
Hashing Type	One-way	One-way
Input Data Size	Unlimited	$(2^{64} - 1)$
Hash Value	128-bit	160-bit
Computation Speed	Faster	Slower
Attack Protection	Weaker	Stronger
Hashing Collisions	Resistant	Resistant
RFC	1321	3174

Most of the time it is best practice to use SHA-1 over MD5. MD5 is a little faster than SHA-1, but the weaker security in MD5 should be a driving force to use SHA-1.

Digital Signatures

In the following sections we will discuss digital signatures. We will start with an overview, which includes a description and the features of digital signatures. We will move into the digital signature process by looking at the two main algorithms, DSA and RSA. Finally, we will compare the DSA and RSA algorithms.

Digital Signatures Overview

A digital signature is an electronic means to validate the authenticity and integrity of a message, software, or document. Most digital signatures use asymmetric cryptography to accomplish the authenticity.

You know that your signature on paper proves you validated and received a document. Likewise, a digital signature can validate an electronic document, message, or even a software program. It is probably easier to forge your personal signature than it is to forge a digital signature. We will get into the reasons why throughout this section.

Digital signatures provide three main features.

- Integrity
- Authentication
- Non-repudiation

We will look at each of these features in more detail.

Integrity

In the first part of this chapter we discussed hashing and how hashing provides integrity. Hashing also provides integrity to digital signatures. To create a digital signature the data is hashed. We will get into the actual process in the next section.

Authentication

We first talked about authentication when we discussed HMAC. The drawback for HMAC is that both parties have to know the secret key. This does not work well when the parties do not know each other. Digital signatures provide a different type of authentication feature, while still ensuring as efficiently as possible that the party one is dealing with is who they say they are.

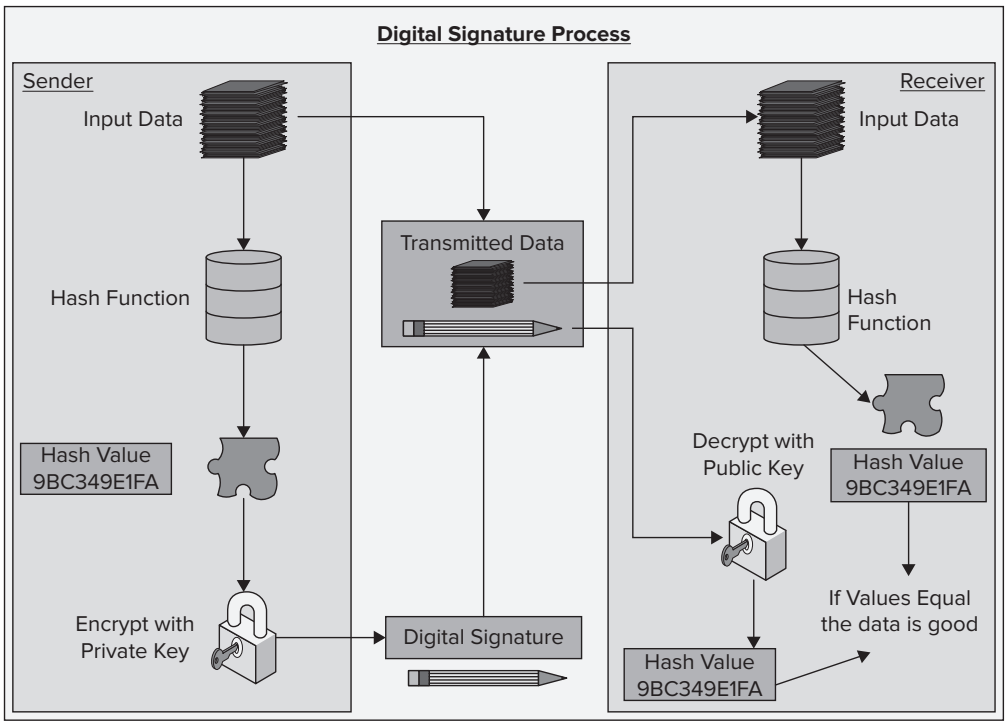
Non-repudiation

What is non-repudiation? Let us first look at repudiation. Repudiation is the ability to refute something. It is simply being able to say "I did not say that" with no proof that you did. Non-repudiation is the ability to prove that someone *did* say something. If we can prove something came from someone or someone said something, they cannot deny the fact. Just like a real signature, a digital signature may be used to prove that you communicated something or that you saw a document.

Digital Signature Process

In today's world there are two main digital signature processes—Digital Signature Standard (DSS) and the RSA algorithm. We will review the two standards in the follow sections. Figure 10.3 shows the digital signature process.

FIGURE 10.3 Digital signature process



Digital Signature Standard

In 1994, the National Institute of Standards and Technology (NIST) issued a Federal Information Processing Standards (FIPS) publication for Digital Signature Standard (DSS). This standard is outlined in FIPS Publication 186.

You can view the actual standard on the NIST website at `www.itl.nist.gov/fipspubs/fip186.htm`.

The DSS outlines a specific algorithm to be used in creating the digital signature. This algorithm is the Digital Signature Algorithm (DSA). In the next section we will look at the DSA process.

DSA Digital Signature Process

DSA is considered to be *public key cryptography (PKC)*. Public key cryptography is different from secret key cryptography like that used in HMAC. Public key cryptography utilizes both private and public keys. The public key is assumed to be known by all of the public.

DSA utilizes two keys for generating the digital signature. The first key comes from the hashing function. The original hashing function used with DSA was SHA-1. The current version of DSA is capable of using newer versions of the SHA algorithm such as SHA-2. The second key is created from a private key. These two keys are used to generate the digital signature. It is good to note that DSA is used only for digital signatures. It is not used for encryption.

Once DSA has generated the message digest from hashing the data, it uses its private key to encrypt the message digest. The encrypted message digest is the digital signature. Because DSA utilizes a two-key system, it is considered to be more processor intensive than other algorithms such as RSA.

The main features of the DSA digital signature process are as follows:

- Utilizes multiple keys to generate the digital signature.
- Uses SHA1 as the hashing algorithm.
- Considered slower than RSA.
- The DSA software can legally be exported from the United States because it lacks encryption.

RSA Digital Signature Process

RSA is also publiekey cryptography. The RSA digital signature process is more streamlined, faster, and more flexible than the DSA process. But some would argue that with the speed and flexibility comes a decrease in security.

RSA encrypts only the hash value or message digest of the data. It does this by using the private key. The encrypted message digest is the digital signature. The digital signature is added to the original data. This is sent to the recipient.

The recipient then generates its own message digest from the data that was sent. The recipient decrypts the digital signature with the sender's public key. If the generated message digest and the decrypted message digest are equal, the data has integrity and is authenticated. Integrity is proven when the two hash values are the same. Authentication is proven because the public key will decrypt only messages created by the sender's private key.

The main features of the RSA digital signature process are as follows:

- Considered faster and more flexible than DSA.
- Encrypts only the message digest.
- Uses private key to encrypt and public key to decrypt.
- It typically cannot be exported from the United States because of the encryption laws.

The Department of Commerce controls the export law on encryption software. The law restricts the export of encryption software, components, and hardware. If you have ever downloaded software or bought hardware with encryption capabilities, you probably had to agree not to export the item.

DSA and RSA Comparison Chart

Table 10.5 provides a high-level comparison between the DSA and RSA digital signature algorithms.

TABLE 10.5 Comparing DSA to RSA

	DSA	RSA
Key Type	Public key	Public key
Encryption	No	Yes
Computation Speed	Slower	Faster
Security Strength	Strong	Strong

Summary

In this chapter you learned about the overall function of hashing. We addressed how hashing can be used for file integrity checking, database indexing, and ultimately security functionality. We showed you examples of these functions ranging from integrity checking a file to the efficiency of database indexing and brought it all together with security. With integrity checking and efficiency, we laid the groundwork for the rest of the chapter.

You learned about the features and functions of the hash function. The hash function is the main process of hashing. It takes in data and spits out a hash value. Important things to know about the hash function are the features needed to provide a good hash. The value section of this chapter outlined different types of values that can be generated from the hash function. We focused on the one-way hash and the fixed-length hash value.

We discussed hashing algorithms, which compute the data and create the hash value. The two types of hashing algorithms we reviewed are MD5 and SHA-1. These are the two primary hashing algorithms in existence today.

Finally, we looked at digital signatures. Digital signatures employ the use of hashing to generate a unique signature for each document. We looked at the DSS and how it is processed. We also focused on two digital signature processes, DSA and RSA. We then compared the DSA and RSA processes.

Exam Essentials

Understand what a hashing function does. A hash function is a mathematical program that takes variable-length data and produces a fixed-length value. It is simply the process, function, and/or algorithm that takes the data and generates the hash.

Understand the features of a hashing function. Hashing functions need to be fast and efficient. A hashing function needs to be collision resistant to provide unique output for all data. Manipulation resistance ensures that changes in the data change the output of the hash value. A one-way hash ensures that the hash value cannot reproduce the original data. The hashing function produces a fixed-length hash value from variable-length inputs.

Know the two primary hashing algorithms. The two algorithms are MD5 and SHA-1. The MD5 algorithm generates a 128-bit hash value output. The SHA-1 algorithm generates a 160-bit hash value output. The MD5 algorithm computes faster than the SHA-1 algorithm, but the SHA-1 algorithm is more secure than the MD5 algorithm. You should use the SHA-1 algorithm instead of the MD5 algorithm when speed is not an issue.

Understand what a digital signature is and its features. A digital signature is a way to sign a document, message, or software to prove who it came from and that it has not been altered. A digital signature provides three main features: integrity, authentication, and non-repudiation.

Know the different digital signature processes and features. The two digital signature processes discussed in this chapter are DSA and RSA. DSA utilizes multiple keys to generate the digital signature. It uses SHA-1 as the hashing algorithm. DSA can be exported from the United States because it lacks encryption. RSA encrypts only the message digest, not all of the data (encrypting all of the data would double the amount of data that is transferred). RSA is considered faster and more flexible than DSA, but RSA cannot be exported from the United States because of the encryption laws.

Written Lab

Write the answers to the following questions:

1. What functionality does hashing provide: confidentiality, integrity, or availability?
2. HMAC adds what to the hashing equation?
3. Which hashing algorithm is considered faster?
4. Which hashing algorithm produces a 160-bit hash?

5. What is the term when a hashing function produces the same hash from different data?
6. What are other names for hash value?
7. Which digital signature process is considered faster?
8. What does DSS stand for?
9. What does the RSA encrypt in digital signatures?
10. Which key is used to generate the digital signature?

Handson Lab

Handson Lab 10.1: Generate a Hash Value from a File

In this hands-on lab you will need a text editor such as Notepad (any text editor will work) and a hashing program such as HashCalc. The hashing program will need to provide either an MD5 or SHA-1 output value to complete this lab.

1. Open your favorite text program, such as Notepad.
2. Generate simple text within a file.
3. Save the file.
4. Open a hashing program, such as HashCalc.
5. Choose the text file you just created. Select the MD5 and/or SHA-1 hash.
6. Generate and view the hash value for the file.
7. Open the text file and modify the text.
8. Run the file through the hash program again.
9. View the difference in the hash values.

Review Questions

1. MD5 generates what length of hash value?
   - **A.** 168 bit
   - **B.** 128 bit
   - **C.** Variable length
   - **D.** 160 bit
2. What are two common hashing algorithms? (Choose two.)
   - **A.** RSA
   - **B.** SHA1
   - **C.** Diffie-Hellman
   - **D.** MD5
3. Which hashing algorithm is considered faster?
   - **A.** SHA1
   - **B.** AES
   - **C.** MD5
   - **D.** All algorithms are the same speed.
4. Which hashing algorithm's maximum input data is $(2^{64} - 1)$?
   - **A.** SHA1
   - **B.** SHA1 and MD5
   - **C.** MD5
   - **D.** None of the above
5. Which of the following is not a main feature of a hashing function?
   - **A.** Unable to modify the message without altering the hash value
   - **B.** Take variable-length data and produce a fixed-length value
   - **C.** Determine the message from the hash value
   - **D.** Easily compute the hash value for any message
6. Which hashing algorithm is considered more secure?
   - **A.** MD5
   - **B.** RSA
   - **C.** SHA1
   - **D.** DSS

7. Which hashing algorithm has a longer hash value?

A. DSA

B. ROIBCE

C. ABC

D. SHA1

8. HMAC uses what additional feature for authentication in hashing?

A. Public key

B. Secret key

C. Private key

D. Car key

9. Who invented the MD5 algorithm?

A. Ronald Rivest

B. Adi Shamir

C. Len Ableman

D. Michael Dedorf

10. When a hash value is written out, it is typically displayed in what format?

A. Binary

B. PHP

C. ASP

D. Hexadecimal

11. How much does the hash value typically change when a simple change is made to the data?

A. One character

B. Two characters

C. No characters

D. Many characters

12. What processor speed was MD5 designed for?

A. 128 bit

B. 32 bit

C. 8 bit

D. 64 bit

13. What was the original algorithm used in DSS?

A. RSA

B. DSA

C. DSP

D. MD5

14. Which of the following is *not* a feature of a digital signature?

A. Providing authentication

B. Providing integrity

C. Providing repudiation

D. Providing non-repudiation

15. Which hashing algorithm does DSA utilize?

A. MD4

B. SHA-1

C. DSS

D. RSA

16. Which digital signature algorithm is considered faster?

A. RSA

B. DSA

C. DSS

D. DSP

17. Which part of the message does RSA encrypt?

A. Entire message

B. Message header

C. Message digest

D. RSA does not encrypt.

18. Which algorithms are considered public key cryptography? (Choose all that apply.)

A. RSA

B. DSA

C. HMAC

D. CA

19. How does a digital signature provide authentication?

A. A message digest can be decrypted only with a public key generated by a private key.

B. A message digest can be decrypted only with a private key generated by a public key.

C. With the use of a secret key.

D. Only integrity is provided with digital signatures.

20. What part of the message does DSA encrypt?

A. Entire message

B. Message header

C. Message digest

D. None of it

Answers to Review Questions

1. B. MD5 generates a fixed-length 128-bit output value.

2. B, D. SHA-1 and MD5 are the two common hashing algorithms. RSA and Diffie-Hellman are public-key cryptography protocols.

3. C. MD5 is faster than SHA-1, but it is commonly thought of as less secure than SHA-1. AES is an encryption algorithm.

4. A. SHA-1 has a maximum input data limit of $(2^{64} - 1)$. MD5 does not have a limit.

5. C. A hashing function should generate a hash value but is unable to determine the message from the hash.

6. C. SHA-1 is considered more secure than MD5. RSA is a digital signature process and DSS is a standard.

7. D. SHA-1 is the only hashing algorithm listed, and it generates a 160-bit hash value compared to MD5's 128-bit, which is not listed.

8. B. A secret key is added in the hash for HMAC. Only the sender and receiver know the secret key. Public and private keys are used in digital signatures, not hashing.

9. A. Ronald Rivest invented MD5 and MD4. Adi Shamir and Len Ableman are the *S* and the *A* in RSA. Michael Dedorf is fictitious.

10. D. Most of the time you see a hash value in print it will be in hexadecimal. Binary would be very long if it was typed out. PHP and ASP are programming languages.

11. D. It is hard to say how many changes will be made to the hash value. The best answer is many characters. If you change one character in the data, it doesn't change just one character in the hash value.

12. B. MD5 was designed for 32-bit processors; 8-bit processors were out but much slower. In the early nineties 64-bit and 128-bit processors did not exist.

13. B. Digital Signature Algorithm (DSA) was the original algorithm used in DSS. RSA is available now. DSP stands for digital signal processor, and MD5 is a hashing algorithm.

14. C. Repudiation—the ability to deny that you did something—is not wanted with digital signatures. We use digital signatures to be able to prove who *did* do something.

15. B. SHA-1 is the correct answer. RSA is an alternative to DSA. MD4 is an older hashing algorithm. DSS is the standard outlining DSA.

16. A. RSA is considered faster, but some would argue that it is less secure than DSA. DSS and DSP are not digital signature algorithms.

17. C. RSA encrypts only the message digest. This limits the amount of data that needs to be transmitted.

18. A, B. The only two that apply are RSA and DSA. HMAC is secret key, and CA is part of Public Key Infrastructure (PKI), not PKC.

19. A. If the message digest is decrypted with the public key and it matches the hash sent by the sender, it is considered authenticated.

20. D. Technically DSA does not encrypt anything. It requires other protocols to accomplish this.

Answers to Written Lab

1. Integrity
2. Secret key
3. MD5
4. SHA1
5. Collision
6. Message digest, fingerprint
7. RSA
8. Digital Signature Standard
9. Message digest
10. Private key