Lab No.5: Hashing

Objectives

1. To implement user defined hashing function.
2. To demonstrate the application of hash function.

Introduction

A hash function H takes an input data block M of variable length and generates a fixed-size hash value h=H(M). An effective hash function ensures that applying it to a large set of inputs yields outputs that are evenly distributed and seemingly random. The primary goal of a hash function is to maintain data integrity, such that any change in the input M, even by a single bit, will likely result in a different hash value.

For security purposes, a special type of hash function, known as a cryptographic hash function, is used. This algorithm has two key properties:

- The one-way property, which makes it computationally infeasible to find a data object that matches a pre-specified hash result.
- The collision-free property, which makes it computationally infeasible to find two distinct data objects that produce the same hash result.

Due to these properties, cryptographic hash functions are commonly employed to verify whether data has been altered.

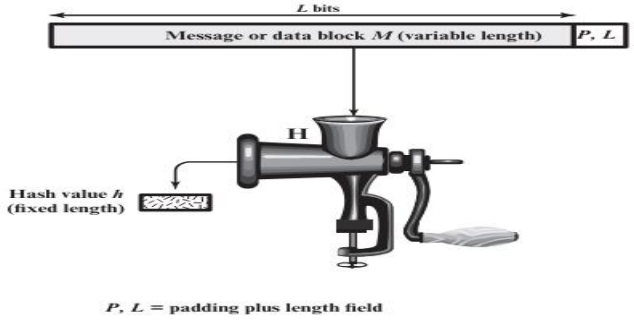


Figure 5.1 Cryptographic Hash Function

Figure 5.1 illustrates how a cryptographic hash function operates. Generally, the input data is padded to make its length an integer multiple of a specific fixed size (e.g., 1024 bits). This padding includes a field that represents the length of the original message in bits. The length field acts as a security measure, making it more challenging for an attacker to create a different message with the same hash value. [Ref: Cryptography and Network Security: Principles and Practice 7th Global Edition]

Lab Exercises

1. Implement the hash function in Python. Your function should start with an initial hash value of 5381 and for each character in the input string, multiply the current hash value by 33, add the ASCII value of the character, and use bitwise operations to ensure thorough mixing of the bits. Finally, ensure the hash value is kept within a 32-bit range by applying an appropriate mask.
2. Using socket programming in Python, demonstrate the application of hash functions for ensuring data integrity during transmission over a network. Write server and client scripts where the server computes the hash of received data and sends it back to the client, which then verifies the integrity of the data by comparing the received hash with the locally computed hash. Show how the hash verification detects data corruption or tampering during transmission.
3. Design a Python-based experiment to analyze the performance of MD5, SHA-1, and SHA-256 hashing techniques in terms of computation time and collision resistance. Generate a dataset of random strings ranging from 50 to 100 strings, compute the hash values using each hashing technique, and measure the time taken for hash computation. Implement collision detection algorithms to identify any collisions within the hashed dataset.

Additional Exercise

1. Write server and client scripts where the client sends a message in multiple parts to the server, the server reassembles the message, computes the hash of the reassembled message, and sends this hash back to the client. The client then verifies the integrity of the message by comparing the received hash with the locally computed hash of the original message.

Lab No. 6: Digital Signature

Objectives

- Demonstrate the use of digital signatures
- Demonstrate the verification of a digital signature

Background

A digital signature is a mathematical technique used to validate the authenticity and integrity of a digital message. A digital signature is the equivalent of a handwritten signature. Digital signatures can actually be far more secure. The purpose of a digital signature is to prevent the tampering and impersonation in digital communications.

Using Digital Signatures

In this part, you will use a website to verify a document signature between Alice and Bob. Alice and Bob share a pair of private and public RSA keys. Each of them uses their private key to sign a legal document. They then send the documents to each other. Both Alice and Bob can verify each other's signature with the public key. They must also agree on a shared public exponent for calculation

Table 1 – RSA Public and Private Keys

Public RSA Key	d94d889e88853dd89769a18015a0a2e6bf82bf356fe14f251fb4f5e2df0d9f9a94a68a30c428b39e3362fb3779a497eceaea37100f264d7fb9fb1a97fbf621133de55fdcb9b1ad0d7a31b379216d79252f5c527b9bc63d83d4ecf4d1d45cbf843e8474babc655e9bb6799cba77a47eafa838296474afc24beb9c825b73ebf549
Private RSA Key	47b9cfde843176b88741d68cf096952e950813151058ce46f2b048791a26e507a1095793c12bae1e09d82213ad9326928cf7c2350acb19c98f19d32d577d666cd7bb8b2b5ba629d25ccf72a5ceb8a8da038906c84dcdb1fe677dffb2c029fd8926318eede1b58272af22bda5c5232be066839398e42f5352df58848adad11a1
Public Exponent	10001

Step 1: Sign the Document.

Alice signs a legal document and send it to Bob using the RSA public and private keys shown in the table above. Now Bob will have to verify Alice's digital signature in order to trust the authenticity of the electronic document.

Step 2: Verify Digital Signature.

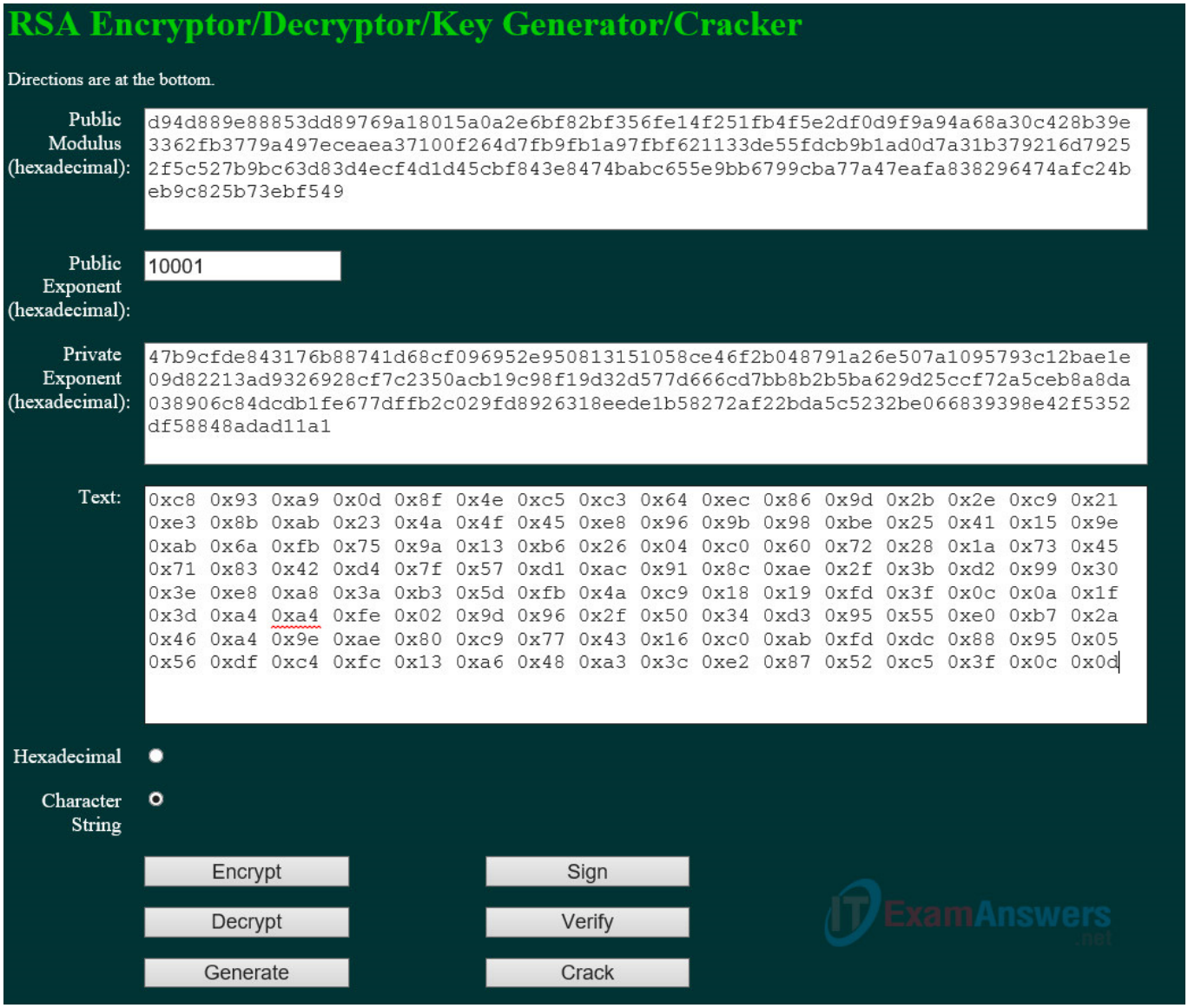
Bob receives the document with a digital signature shown in the table below.

Table 2 – Alice's Digital Signature

Alice's Digital Signature
0xc8 0x93 0xa9 0x0d 0x8f 0x4e 0xc5 0xc3 0x64 0xec 0x86 0x9d 0x2b 0x2e 0xc9 0x21 0xe3 0x8b 0xab 0x23 0x4a 0x4f 0x45 0xe8 0x96 0x9b 0x98 0xbe 0x25 0x41 0x15 0x9e 0xab 0x6a 0xfb 0x75 0x9a 0x13 0xb6 0x26 0x04 0xc0 0x60 0x72 0x28 0x1a 0x73 0x45 0x71 0x83 0x42 0xd4 0x7f 0x57 0xd1 0xac 0x91 0x8c 0xae 0x2f 0x3b 0xd2 0x99 0x30 0x3e 0xe8 0xa8 0x3a 0xb3 0x5d 0xfb 0x4a 0xc9 0x18 0x19 0xfd 0x3f 0x0c 0x0a 0x1f 0x3d 0xa4 0xa4 0xfe 0x02 0x9d 0x96 0x2f 0x50 0x34 0xd3 0x95 0x55 0xe0 0xb7 0x2a 0x46 0xa4 0x9e 0xae 0x80 0xc9 0x77 0x43 0x16 0xc0 0xab 0xfd 0xdc 0x88 0x95 0x05 0x56 0xdf 0xc4 0xfc 0x13 0xa6 0x48 0xa3 0x3c 0xe2 0x87 0x52 0xc5 0x3f 0x0c 0x0d

Click here to use the online RSA tool to verify the authenticity of Alice's digital signature.

Table 3 – Online Digital Signature Tool



a. Copy and paste the **public** and **private** keys from Table 1 above into the **Public Modulus** and **Private Exponent** boxes on the website as shown in the picture above.

b. Make sure the Public Exponent is 10001.

c. Paste Alice's digital signature from Table 2 in the box labeled text on the website as shown above.

d. Now BOB can verify the digital signature by clicking the **Verify** button near the bottom center of the website. Whose signature is identified?

Alice's name should be displayed.

Step 3: Generate a Response Signature.

Bob receives and verifies Alice's electronic document and digital signature. Now Bob creates an electronic document and generates his own digital signature using the private RSA Key in Table 1 (Note: Bob's name is in all capital letters).

Table 4 – BOB Digital Signature

BOB's Digital Signature
0x6c 0x99 0xd6 0xa8 0x42 0x53 0xee 0xb5 0x2d 0x7f 0x0b 0x27 0x17 0xf1 0x1b 0x62 0x92 0x7f 0x92 0x6d 0x42 0xbd 0xc6 0xd5 0x3e 0x5c 0xe9 0xb5 0xd2 0x96 0xad 0x22 0x5d 0x18 0x64 0xf3 0x89 0x52 0x08 0x62 0xe2 0xa2 0x91 0x47 0x94 0xe8 0x75 0xce 0x02 0xf8 0xe9 0xf8 0x49 0x72 0x20 0x12 0xe2 0xac 0x99 0x25 0x9a 0x27 0xe0 0x99 0x38 0x54 0x54 0x93 0x06 0x97 0x71 0x69 0xb1 0xb6 0x24 0xed 0x1c 0x89 0x62 0x3d 0xd2 0xdf 0xda 0x7a 0x0b 0xd3 0x36 0x37 0xa3 0xcb 0x32 0xbb 0x1d 0x5e 0x13 0xbc 0xca 0x78 0x3e 0xe6 0xfc 0x5a 0x81 0x66 0x4e 0xa0 0x66 0xce 0xb3 0x1b 0x93 0x32 0x2c 0x91 0x4c 0x58 0xbf 0xff 0xd8 0x97 0x2f 0xa8 0x57 0xd7 0x49 0x93 0xb1 0x62

Bob sends the electronic document and digital signature to Alice.

Step 4: Verify Digital Signature.

a. Copy and paste the **public** and **private** keys from Table 1 above into the **Public Modulus** and **Private Exponent** boxes on the website as shown in the picture above.

b. Make sure the Public Exponent is 10001.

c. Paste Bob's digital signature from Table 4 in the box labeled text on the website as shown above.

d. Now Alice can verify the digital signature by clicking the **Verify** button near the bottom center of the website. Whose signature is identified?

Bob's name should be displayed.

Part 2: Create Your Own Digital Signature

Now that you see how digital signatures work, you can create your own digital signature.

Step 1: Generate a New Pair of RSA Keys.

Go to the website tool and generate a new set of RSA public and private keys.

a. Delete the contents of the boxes labeled **Public Modulus**, **Private Modulus** and **Text**. Just use your mouse to highlight the text and press the delete key on your keyboard.

b. Make sure the "Public Exponent" box has **10001**.

c. Generate a new set of RSA keys by clicking the **Generate** button near the bottom right of the website.

d. Copy the new keys in Table 5.

Table 5 – New RSA Keys

Public Key	A string of 256 hexadecimal characters will be displayed for both the public and private keys. The keys will be different from one another.

Private key	A string of 256 hexadecimal characters will be displayed for both the public and private keys. The keys will be different from one another.

e. Now type in your full name into the box labeled **Text** and click **Sign**.

Table 6 – Personal Digital Signature

Personal Digital Signature	A string of characters with this format will be displayed. 0x23 0x90 …….

Part 3: Exchange and Verify Digital Signatures

Now you can use this digital signature.

Step 1: Exchange your new public and private keys in Table-5 with your lab partner.

a. Record your lab partner's public and private RSA keys from their Table-5.

b. Record both keys in the table below.

Table 7- Lab Partners RSA Keys

Public key	A string of 256 hexadecimal characters will be displayed for both the public and private keys. The keys will be different from one another.
Private key	A string of 256 hexadecimal characters will be displayed for both the public and private keys. The keys will be different from one another.

c. Now exchange their digital signature from their Table-6. Record the digital signature in the table below.

Lab Partner's Digital Signature	A string of characters with this format will be displayed. 0x23 0x90 …….

Step 2: Verify Lab Partners Digital Signature

a. To verify your lab partner's digital signature, paste his or her public and private keys in the appropriate boxes labeled **Public and Private modulus** on the website.

b. Now paste the digital signature in the box labeled **Text**.

c. Now verify his or her digital signature by clicking the button labeled verify.

d. What shows up in the Text box?

Answers will vary.

Lab Exercises

1. Try using the Elgammal, Schnor asymmetric encryption standard and verify the above steps.
2. Try using the Diffie-Hellman asymmetric encryption standard and verify the above steps.
3. Try the same in a client server-based scenario and record your observation and analysis.

Additional Exercise

1. Explore the link https://www.nmichaels.org/rsa.py for better understanding.

   Demonstrate CIA traid using RSA encryption and digital signature along with SHA hashing