

## EXPERIMENT NUMBER 6

Name of Student: Sangeet Agrawal	PRN: 21070122140	Section: CS-B3
----------------------------------	------------------	----------------

**Title:** Hashing - one-way randomness in hash-function.

**Aim:** Study, understand and demonstrate hash function and its tool

**Objective:** To make students understand and demonstrate hash function and security benefit of hashing in encryption-decryption

### **Theory:**

1. **Hash function:** A function that maps a message of any length into a fixed-length hash value, which serves as the authenticator
2. A variation on the message authentication code is the one-way hash function.
3. As with the message authentication code, a hash function accepts a variable-size message  $M$  as input and produces a fixed size output, referred to as a **hash code**  $H(M)$ .
4. Unlike a MAC, a hash code does not use a key but is a function only of the input message.
5. The hash code is also referred to as a **message digest** or **hash value**.
6. The hash code is a function of all the bits of the message and provides an error-detection capability: A change to any bit or bits in the message results in a change to the hash code.

### **Simple Hash Function – Type I**

1. All hash functions operate using the following general principles.
2. The input (message, file, etc.) is viewed as a sequence of  $n$ -bit blocks. The input is processed one block at a time in an iterative fashion to produce an  $n$ -bit hash function.
3. One of the simplest hash functions is every block's bit-by-bit exclusive-OR (XOR).
4. This operation produces a simple parity for each bit position, known as a longitudinal redundancy check.

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

where

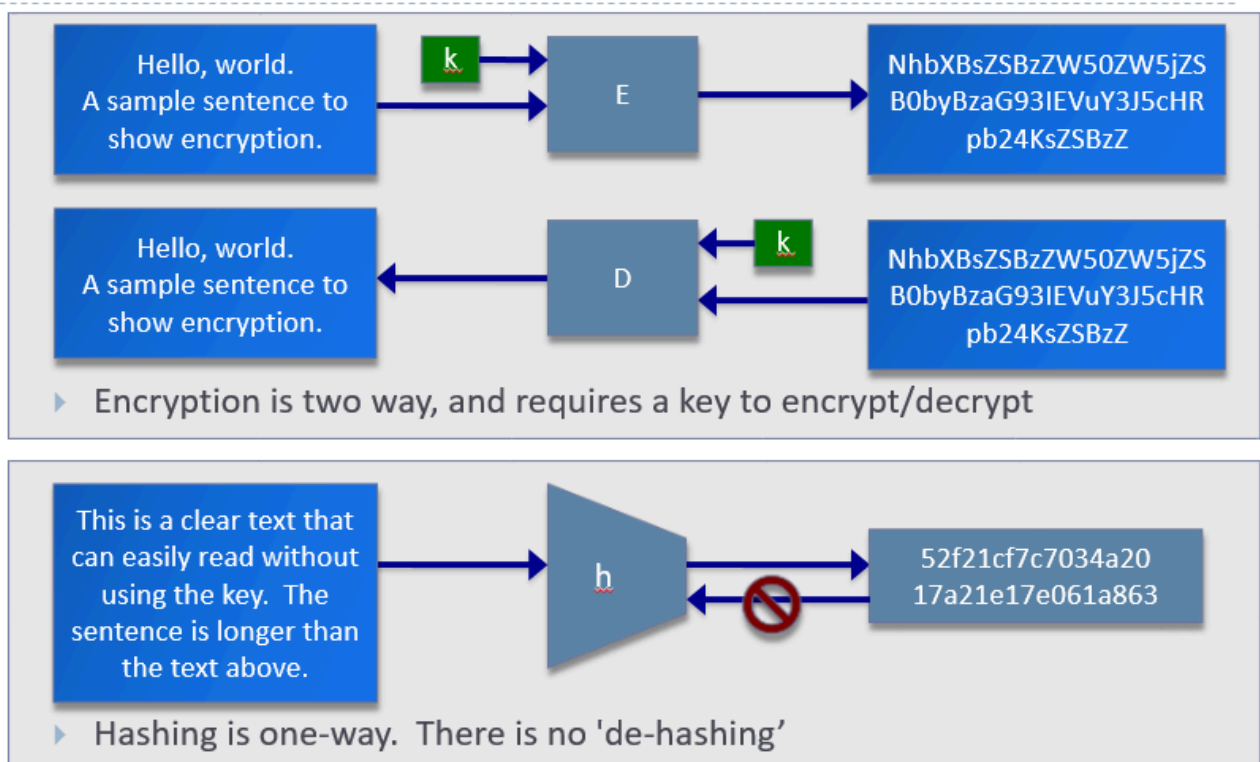
$C_i$  =  $i$ th bit of the hash code,  $1 \leq i \leq n$

$m$  = number of  $n$ -bit blocks in the input

$b_{ij}$  =  $i$ th bit in  $j$ th block

$\oplus$  = XOR operation

## Hashing V.S. Encryption



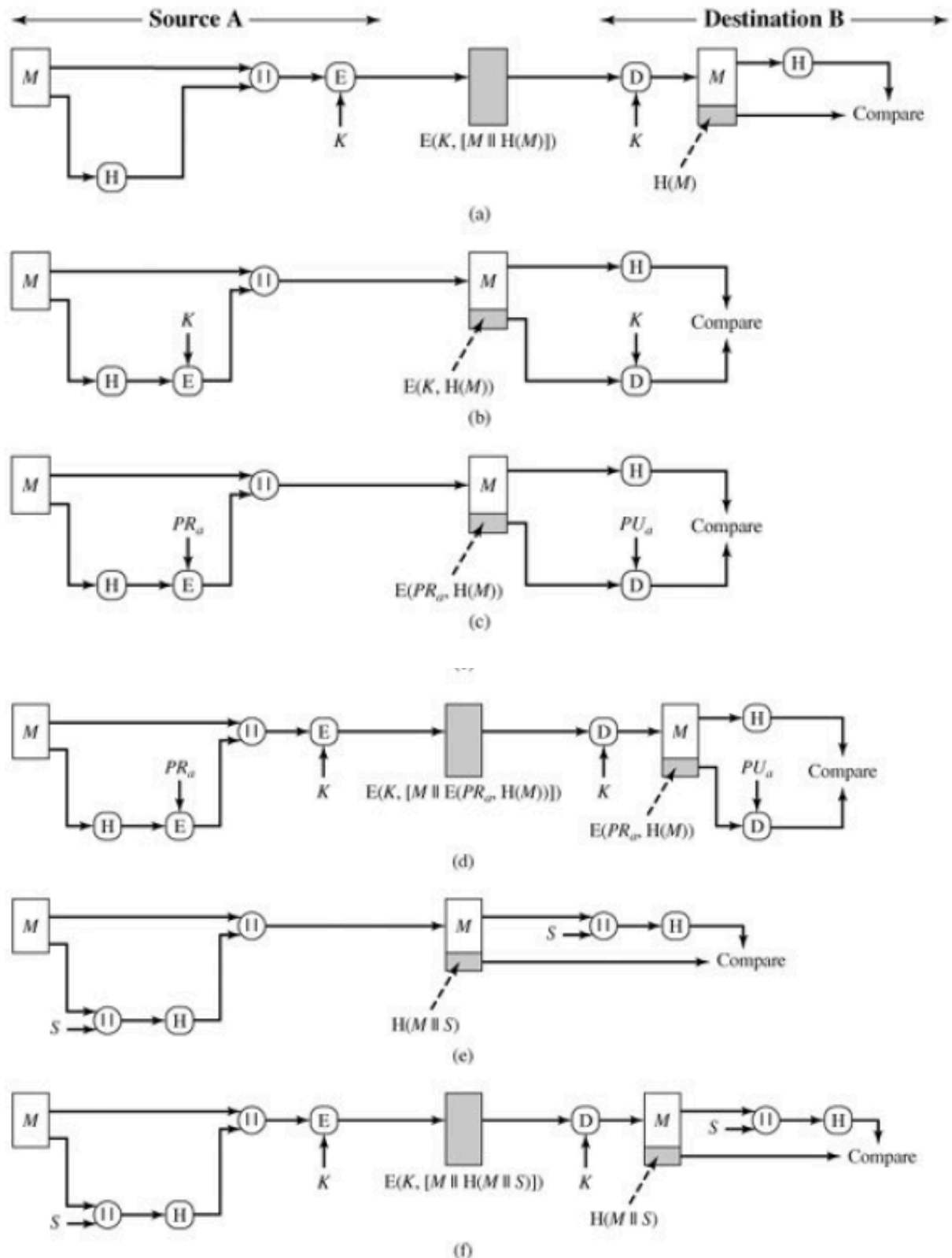


Figure: Basic Uses of Hash Function

[Source: Cryptography and Network Security: Principles and Practice by William Stallings]

**Tools to be practiced:**

- 1) Assignment from Cryptography Virtual Lab (website link 1)
- 2) HashCalc
- 3) MD6 Hash Generator
- 4) All Hash Generator
- 5) HashMyFiles
- 6) Some mobile hash calculation tools

**Reference web links:**

- 1) <https://cse29-iiith.vlabs.ac.in/>
- 2) <https://www.nirsoft.net>
- 3) <https://convert-tool.com>
- 4) <https://www.slavasoft.com>
- 5) <https://onlinehashtools.com>
- 6) <https://www.devglan.com/cryptotools/cryptography-tools>
- 7) <https://www.browserling.com>

**Conclusion:****Implementation question:**

- 1) Demonstrate any other two free cryptography tools based on hash function
- 2) Perform 1 assignments from virtual lab based on hash function
- 3) Study and differentiate between MD5 and Hash.

**Note:** Students are suggested to use Linux OS based tools or free Windows OS based tools.

1) MD4 Hash: <http://www.practicalcryptography.com/hashes/md4-hash/>

<b>Input</b>	<input type="text" value="Sangeet"/>
<b>Calculate</b>	<input type="button" value="MD4"/>
<b>Result</b>	<input type="text" value="33e35db4ce1886620d0d213ceb48b09d"/>

MD5 Hash: <https://www.md5online.org/md5-encrypt.html>

Enter a word here to get its MD5 hash :

<input type="text" value="Sangeet"/>
<input type="button" value="Crypt"/>

*No credit required in this tool*

The MD5 hash for Sangeet is : **6a44aeed457c48cfe9b709365ce9edd2**

## 2) Simulation:

**HMAC Construction using a "Dummy" Hash Function**

**HMAC construction**

Plaintext:  Next Plaintext

length of Initialization Vector (IV), 1,

IV:  Next IV

Key, k:  Next Key

ipad: 0x5C (01011100)  
opad: 0x36 (00110110)

Put your text of size 21 to get the corresponding value of hash of size 1.

Your text:  get hash

Hashed value:

Final Output:  Check Answer!

**CORRECT!!!**

## Assignment:

**Cryptographic Hash Functions and Applications(HMAC)**

**Aim**

**Theory**

**Objective**

**Procedure**

**Simulation**

**Assignment**

**References**

**Feedback**

- Which criterion ensures that we can't find two messages that hash to the same digest?
  - One-wayness
  - Weak-collision-resistance
  - Strong-collision-resistance
  - None
- Which criterion Ensures that it must be extremely difficult or impossible to create the message if the message digest is given.
  - One-wayness
  - Weak-collision-resistance
  - Strong-collision resistance
  - None
- Consider the function  $h: [0,18] \rightarrow [0,14]$ . Suppose  $h(x) = x \bmod 5$ . The collision in  $h$  occurs for.
  - (1, 17)
  - (2, 16)
  - (1, 16)
  - (2, 17)
- The Merkle-Damgard Transform is mainly useful for
  - Converting any fixed-length collision resistant hash function to an arbitrary length collision resistant hash function
  - Converting arbitrary length hash function to a fixed length hash function
  - Constructing hash function from random function
  - None
- Understand HMAC scheme and find a break it using available source code
- Understand Merkel-Damgard transform and Explain,how we are using it for HMAC?
- Understand and explain analogy between SHA1 and our dummy HMAC function
- Explain why HMAC is secure and on what assumptions this security is based?

Answers:

1. (c) Strong-collision-resistance
2. (a) One-wayness
3. (c) (1, 16)
4. (a) Converting any fixed-length collision-resistant hash function to an arbitrary-length collision-resistant hash function
5. **Breaking HMAC:** HMAC relies on a secure hash function; breaking it is difficult unless the hash function is weak.
6. **Merkle-Damgård in HMAC:** Used to extend fixed-length hash functions to variable-length inputs, ensuring efficient hashing in HMAC.
7. **SHA-1 and Dummy HMAC Analogy:** Both involve hashing in multiple rounds with compression functions and padding.
8. **HMAC Security:** Secure due to the use of a secret key and a collision-resistant hash function. Security assumes the hash function is strong and the key is secret.

3) Differentiating between MD5 and Hash:

Features	MD5	General Hash Function
Output Size	128-bit	Varies (e.g., 256-bit for SHA-256)
Speed	Fast	Varies by algorithm
Security	Vulnerable to collision attacks	Depends on algorithm (e.g., SHA-256 is secure)
Use Cases	Checksums, file integrity (non-critical)	Cryptography, digital signatures, data integrity
Current Relevance	Considered insecure	Modern algorithms (like SHA-256) are secure and widely used