

Cryptographic Hash Functions

Prepared by Dr. Ayman Khallel

IT373

Information Security

Lecture 7



Course content

1. Overview of security concepts and definitions
2. Symmetric cryptography; modern encryption algorithms (DES, AES)
3. Modern data encryption standards
4. Block cipher cryptanalysis and usage
5. Modern stream ciphers
6. Asymmetric cryptography; public key cryptography,
7. **Cryptographic Hash Functions**
8. Message authentication
9. Digital signature and authentication protocols
10. Electronic mail security, IP security, web security.

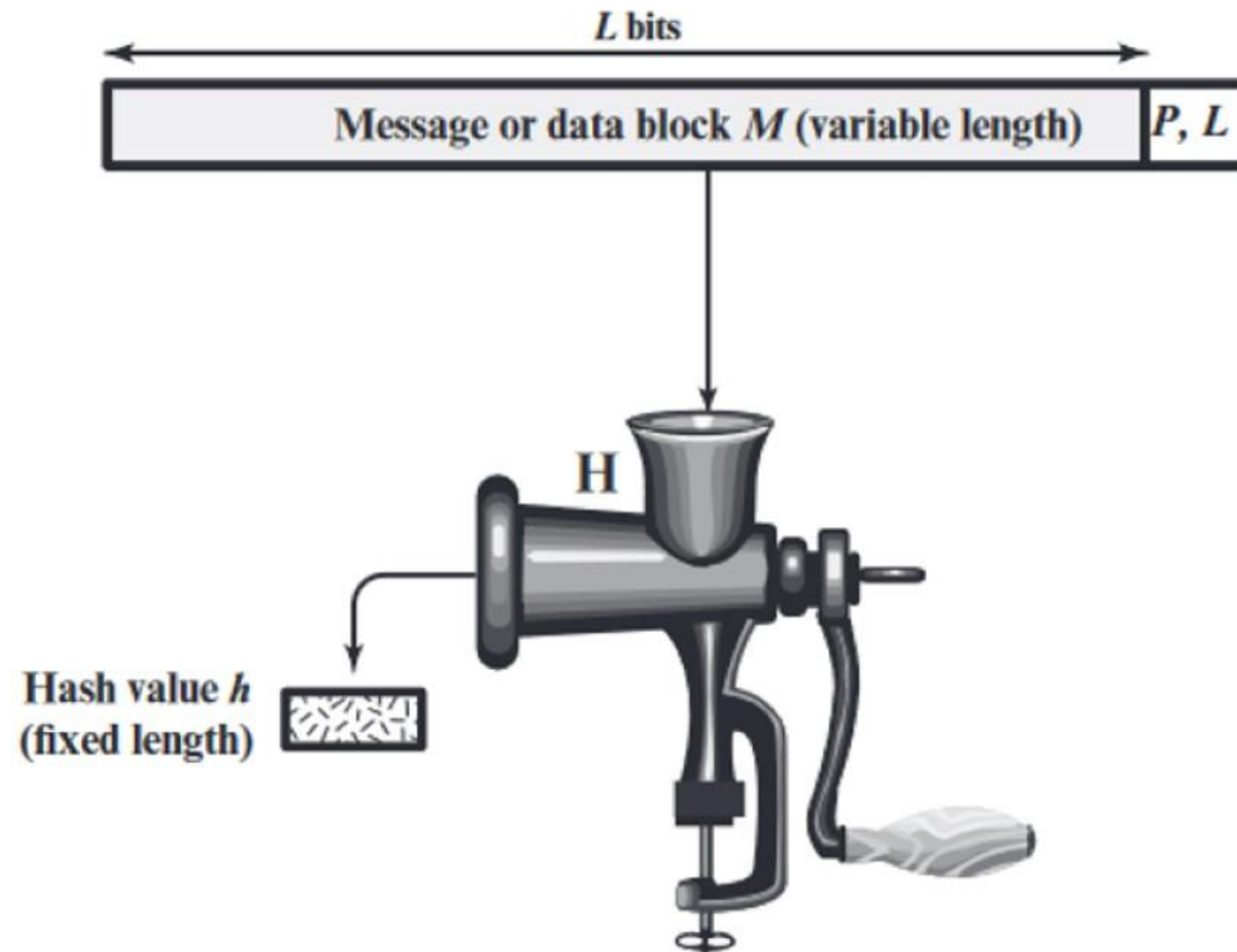
Agenda

- Introduction to Hash Functions
- Cryptographic Hash Function Properties
- Hash Function Application
 - Message Authentication Code (MAC)
 - Digital Signatures
 - Other application
- Requirements And Security
- Secure Hash Algorithms (SHA)
- SHA-3

Hash Functions

- A hash function H accepts a variable-length block of data M as input and produces a fixed-size hash value
 - $h = H(M)$
 - Principal object is data integrity
- Cryptographic hash function
 - An algorithm for which it is computationally infeasible to find either:
 - (a) a data object that maps to a pre-specified hash result (the one-way property)
 - (b) two data objects that map to the same hash result (the collision-free property)





P, L = padding plus length field

Figure 11.1 Cryptographic Hash Function; $h = H(M)$



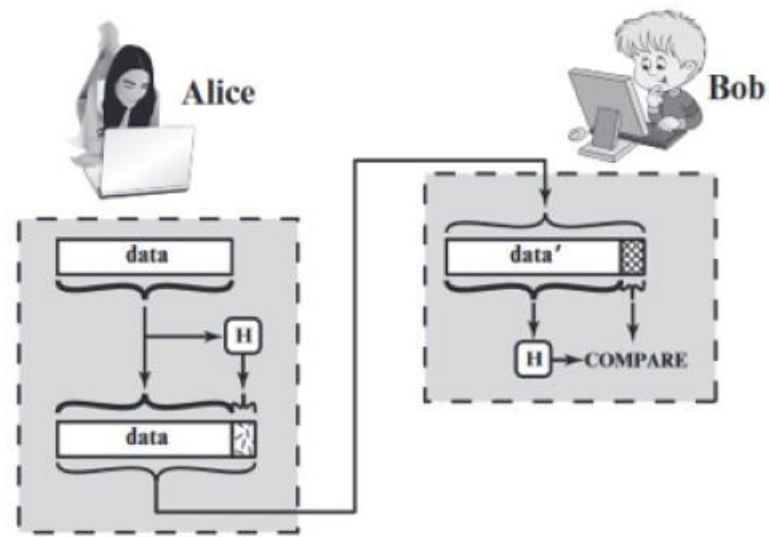
APPLICATIONS OF HASH FUNCTIONS

Message
Authentication

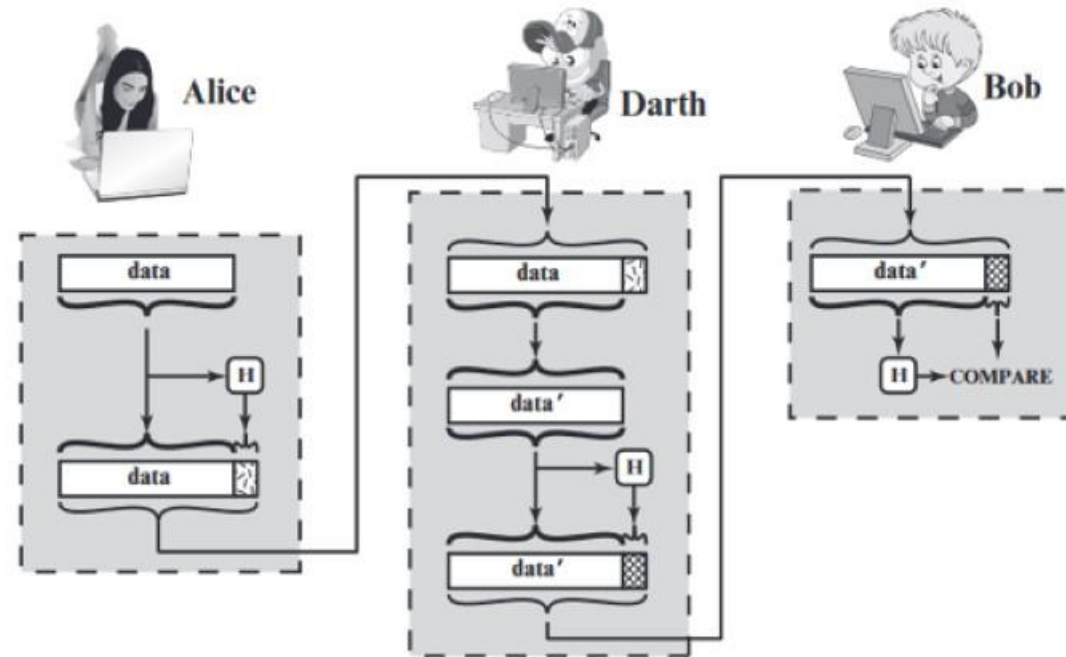
Digital
Signatures

Other
Applications





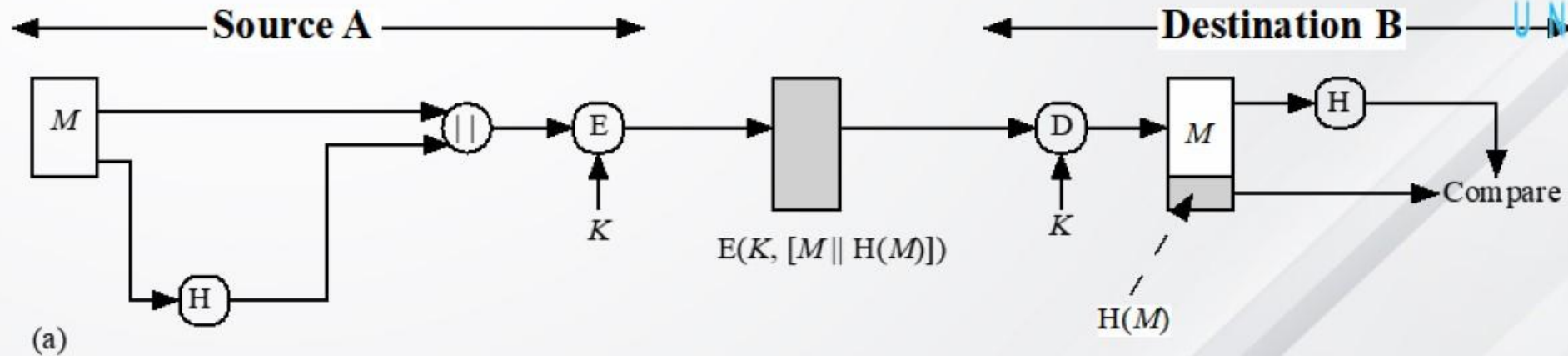
(a) Use of hash function to check data integrity



(b) Man-in-the-middle attack



Message Authentication

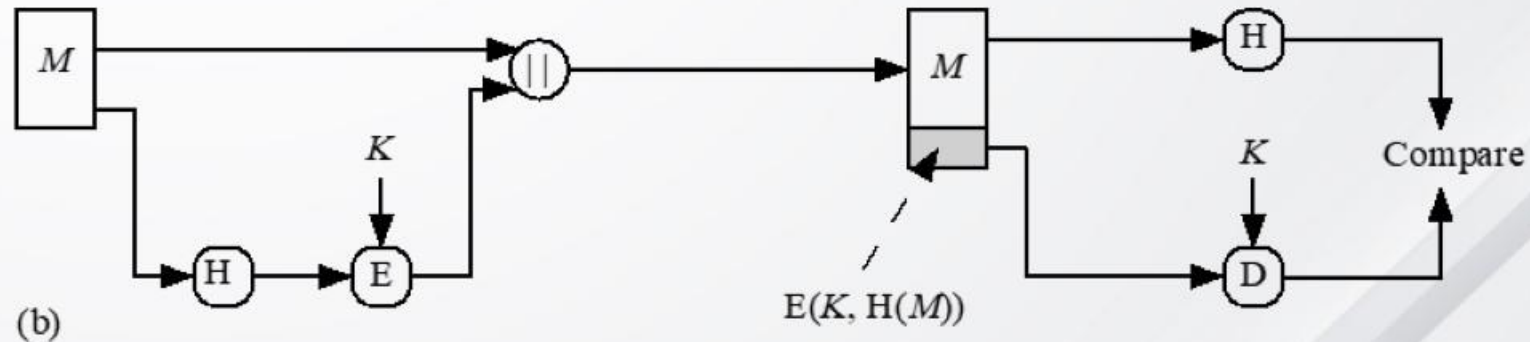


Authentication and Confidentiality

Process:

- **Encryption of Message + Hash Code:**
 - Message is hashed
 - Message and Hash code are concatenated.
 - Message + Hash code is Encrypted with a shared secret key.
- **Authentication:**
 - Only **A** and **B** know the secret key, so the message is verified to be from **A**.
 - The **hash code** provides the structure needed to ensure the message hasn't been altered.
- **Confidentiality:**
 - Encryption of the message and hash code protects the contents from unauthorized access.



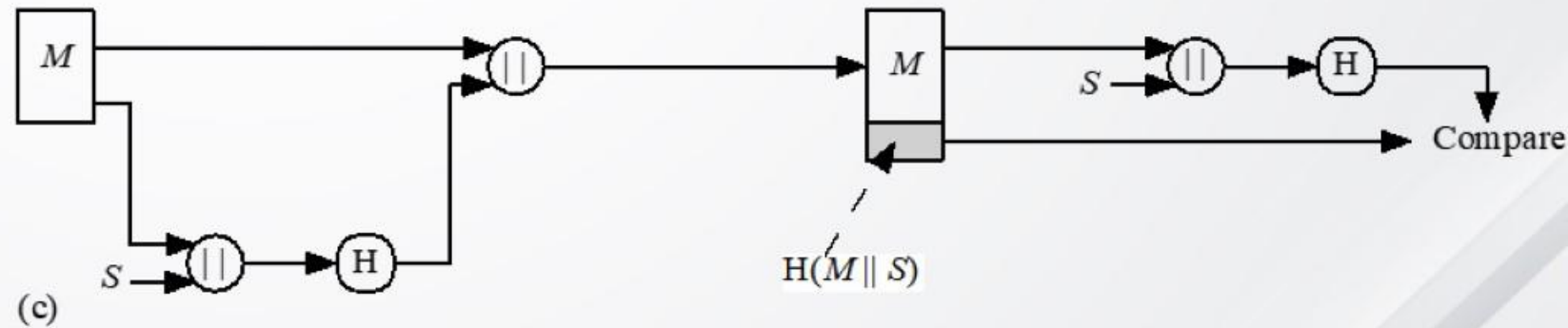


Selective Encryption for Hash Codes

- **Objective:** Encrypt only the hash code instead of the entire message
- **Benefits:**
 - **Efficiency:** Reduces processing burden by focusing on integrity, not confidentiality
 - **Use Case:** Suitable for applications where confidentiality is unnecessary, and only message integrity is required



Message Authentication



Message Authentication without Encryption

• **Technique:** Uses a **shared secret** S and a **hash function** for message integrity

• **Process:**

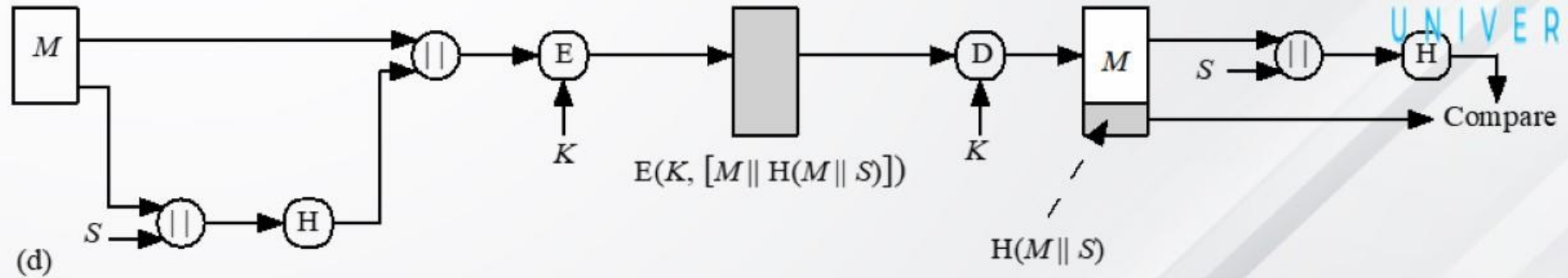
- **Sender (A):** Computes the hash of $M+S$ and appends it to the message M
- **Receiver (B):** Recomputes the hash using $M+S$ to verify authenticity

• **Security Advantage:**

- **No Secret Transmission:** The shared secret S is never sent, preventing tampering and forgery
- **Integrity without Confidentiality:** Ideal for systems needing validation rather than encryption



Message Authentication



Enhanced Confidentiality for Message Authentication

• **Approach:** Add confidentiality by encrypting **Message (M) + Hash Code**

• **Process:**

- **Sender:**
 - Computes the hash of M+S
 - Appends the hash to M
 - Encrypts the entire result with a symmetric key
- **Receiver:**
 - Decrypts the message
 - Recomputes and verifies the hash using M+S

• **Benefits:**

- **Integrity & Confidentiality:** Provides both message validation and protection against interception



Reasons for Interest in Avoiding Encryption

- **Encryption Software is Slow:**
Even small amounts of data per message can result in a steady stream of messages, slowing down system performance.
- **High Cost of Encryption Hardware:**
While low-cost chip implementations of DES exist, equipping all network nodes adds significant costs.
- **Optimization for Large Data Sizes:**
Encryption hardware is designed for large data volumes. For small data blocks, a significant amount of time is spent on initialization and overhead.
- **Patent and Licensing Costs:**
Some encryption algorithms are patented, requiring costly licenses to use.



Message Authentication Code (MAC)

- Also known as a *keyed hash function*
- Typically used between two parties that share a secret key to authenticate information exchanged between those parties

Takes as input a secret key and a data block and produces a hash value (MAC) which is associated with the protected message

- If the integrity of the message needs to be checked, the MAC function can be applied to the message and the result compared with the associated MAC value
- An attacker who alters the message will be unable to alter the associated MAC value without knowledge of the secret key

Digital Signature

- Operation is similar to that of the MAC
- The hash value of a message is encrypted with a user's private key
- Anyone who knows the user's public key can verify the integrity of the message
- An attacker who wishes to alter the message would need to know the user's private key
- Implications of digital signatures go beyond just message authentication

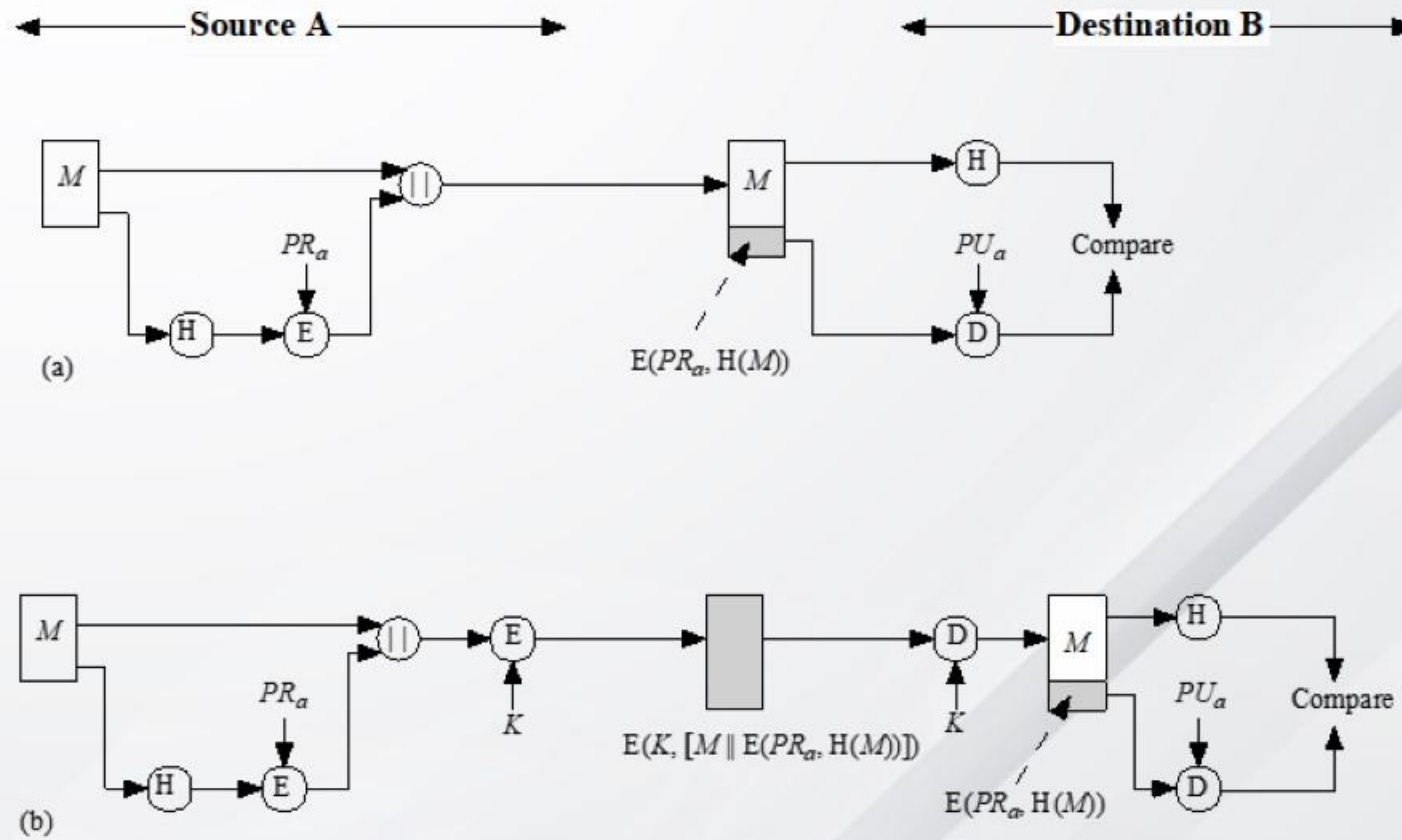


Figure 11.4 Simplified Examples of Digital Signatures

Other Hash Function Uses

Commonly used to create a one-way password file

When a user enters a password, the hash of that password is compared to the stored hash value for verification

This approach to password protection is used by most operating systems

Can be used for intrusion and virus detection

Store $H(F)$ for each file on a system and secure the hash values

One can later determine if a file has been modified by recomputing $H(F)$

An intruder would need to change F without changing $H(F)$

Can be used to construct a pseudorandom function (PRF) or a pseudorandom number generator (PRNG)

A common application for a hash-based PRF is for the generation of symmetric keys



First Simple Hash Functions

- Bit-by-bit exclusive-OR (XOR) of every block
 - $C_i = b_{i1} \oplus b_{i2} \oplus \dots \oplus b_{im}$
 - C_i = ith bit of the hash code, $1 \dots i \dots n$
 - m = number of n-bit blocks in the input
 - b_{ij} = ith bit in jth block
 - $\oplus$ = XOR operation
- Produces a simple parity for each bit position and is known as a longitudinal redundancy check
- Reasonably effective for random data as a data integrity check
- **Effectiveness with Predictable Data:**
 - LRC is less effective with predictable, structured data (e.g., text files).
 - Example: For normal text files, the high-order bit of each octet is often zero.
 - In this case, a 128-bit hash function has an **effectiveness of 2^{112}** instead of 2^{128} .

First Simple Hash Functions

- Bit-by-bit exclusive-OR (XOR) of every block
 - $C_i = b_{i1} \oplus b_{i2} \oplus \dots \oplus b_{im}$
 - C_i = i th bit of the hash code, $1 \dots i \dots n$
 - m = number of n -bit blocks in the input
 - b_{ij} = i th bit in j th block
 - $\oplus$ = XOR operation
- Produces a simple parity for each bit position and is known as a longitudinal redundancy check
- Reasonably effective for random data as a data integrity check
- **Effectiveness with Predictable Data:**
 - LRC is less effective with predictable, structured data (e.g., text files).
 - Example: For normal text files, the high-order bit of each octet is often zero.
 - In this case, a 128-bit hash function has an **effectiveness of 2^{112}** instead of 2^{128} .

Second Simple Hash Functions

- Initially set the n-bit hash value to zero.
- Process each successive n-bit block of data as follows:
 - Rotate the current hash value to the left by one bit.
 - XOR the block into the hash value.
- Vulnerabilities: It's easy to craft a new message that matches the hash code:
 - Prepare the alternate message and append an n-bit block to match the hash.

Two Simple Hash Functions

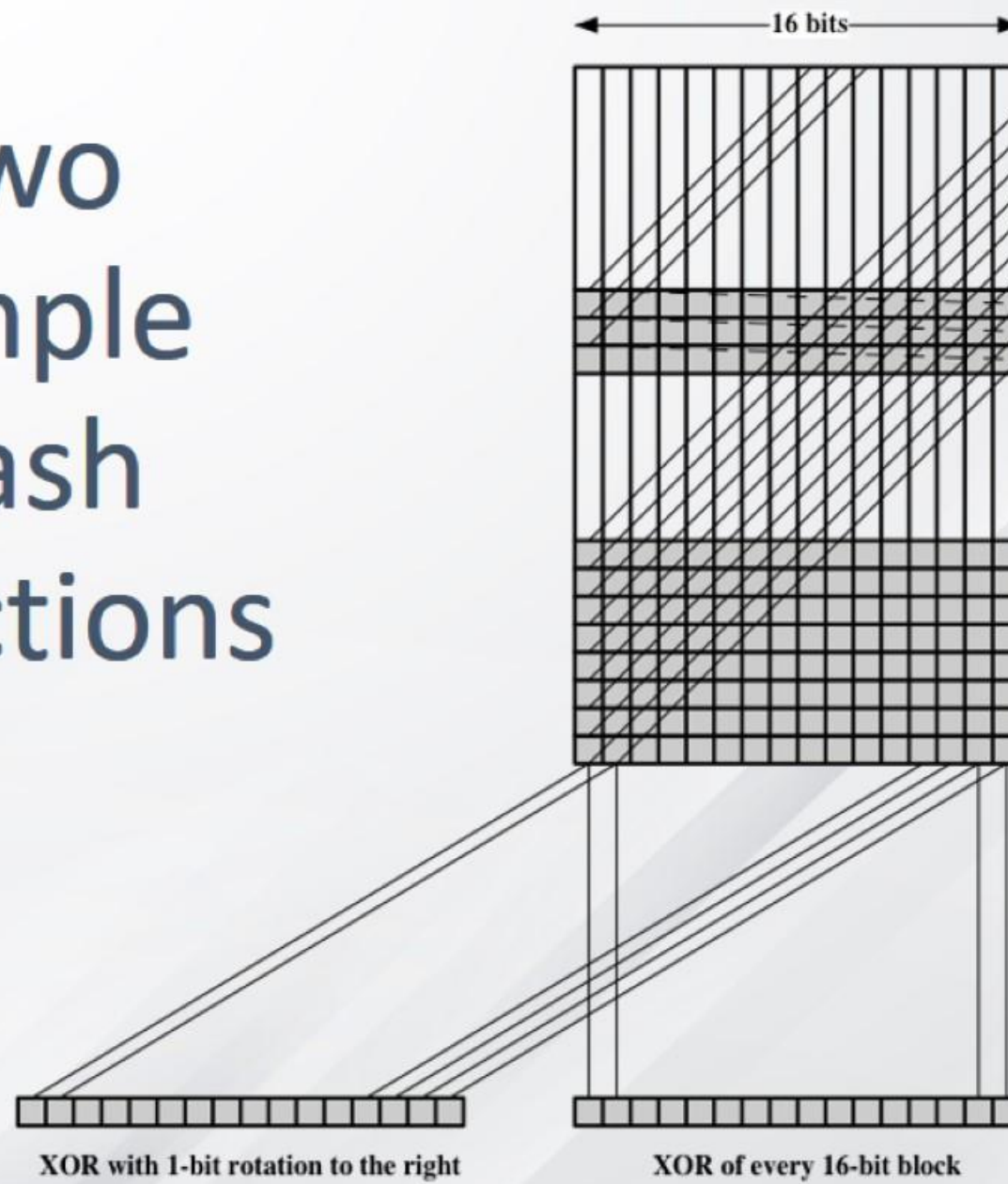


Figure 11.5 Two Simple Hash Functions

Dr. Ayman Khallil Ibrahim

Requirements and Security

Preimage

- x is the preimage of h for a hash value $h = H(x)$
- Is a data block whose hash function, using the function H , is h
- Because H is a many-to-one mapping, for any given hash value h , there will in general be multiple preimages

Collision

- Occurs if we have $x \neq y$ and $H(x) = H(y)$
- Because we are using hash functions for data integrity, collisions are clearly undesirable



Table 11.1

Requirements for a Cryptographic Hash Function H

		Call	Requirement	Description
strong hash function	second preimage resistant	practical application	Variable input size	H can be applied to a block of data of any size.
			Fixed output size	H produces a fixed-length output.
			Efficiency	$H(x)$ is relatively easy to compute for any given x , making both hardware and software implementations practical.
		preimage resistant	Preimage resistant	For any given hash value h , it is computationally infeasible to find y such that $H(y) = h$.
			Second preimage resistant (weak collision resistant)	For any given block x , it is computationally infeasible to find $y \neq x$ with $H(y) = H(x)$.
			Collision resistant	It is computationally infeasible to find any pair (x, y) with $x \neq y$, such that $H(x) = H(y)$.
			Pseudorandomness	Output of H meets standard tests for pseudorandomness.



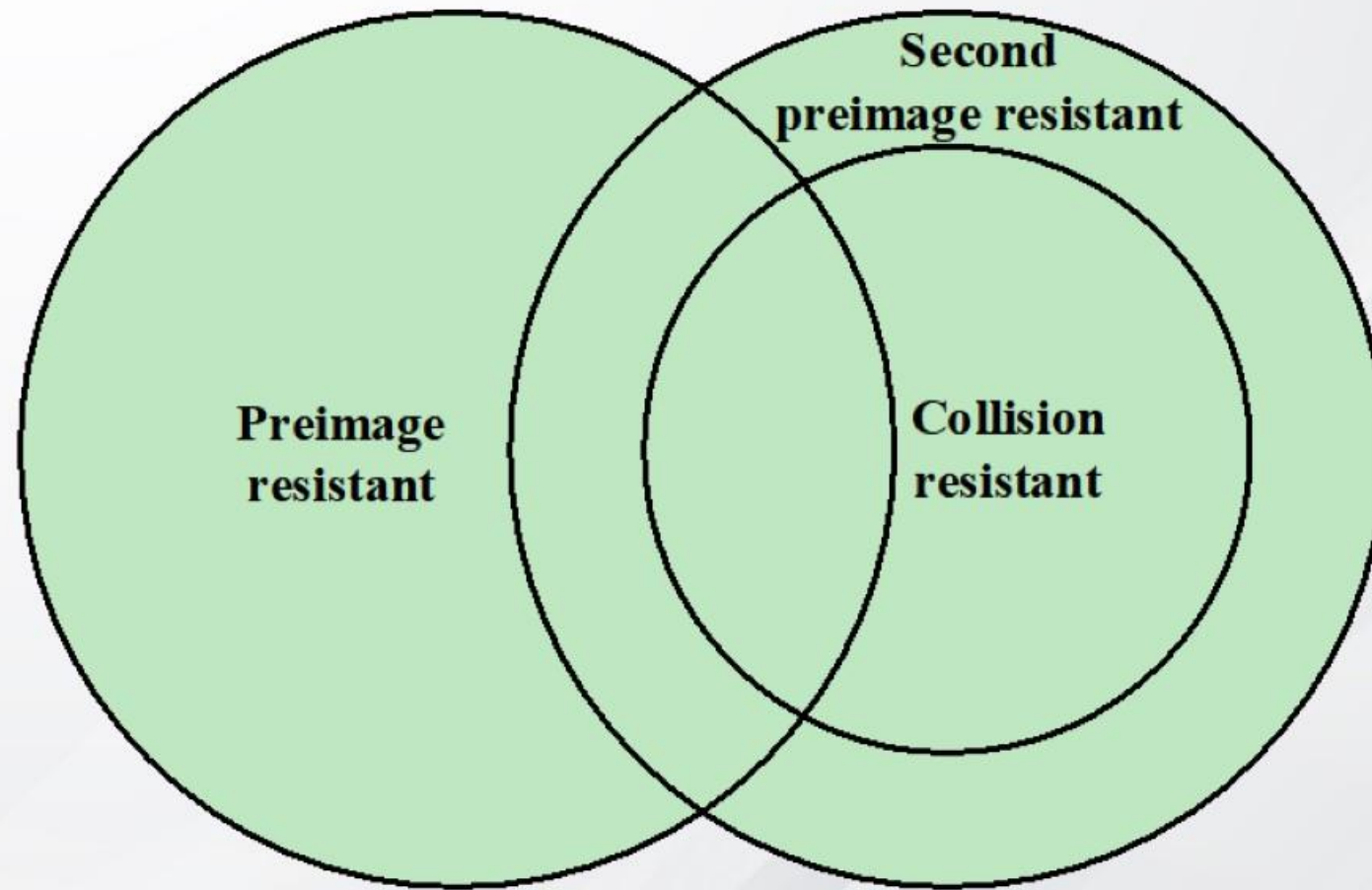


Figure 11.6 Relationship Among Hash Function Properties

Table 11.2

Hash Function Resistance Properties Required for Various Data Integrity Applications

	Preimage Resistant	Second Preimage Resistant	Collision Resistant
Hash + digital signature	yes	yes	yes*
Intrusion detection and virus detection		yes	
Hash + symmetric encryption			
One-way password file	yes		
MAC	yes	yes	yes*

* Resistance required if attacker is able to mount a chosen message attack

Attacks on Hash Functions

Brute-Force Attacks

- Does not depend on the specific algorithm, only depends on bit length
- In the case of a hash function, attack depends only on the bit length of the hash value
- Method is to pick values at random and try each one until a collision occurs

Cryptanalysis

- An attack based on weaknesses in a particular cryptographic algorithm
- Seek to exploit some property of the algorithm to perform some attack other than an exhaustive search





Collision Resistant Attacks

- For a collision resistant attack, an adversary wishes to find two messages or data blocks that yield the same hash function
 - The effort required is explained by a mathematical result referred to as the *birthday paradox*
 - https://www.youtube.com/watch?v=ofTb57aZHZs&t=423s&ab_channel=Vsauce2
- Yuval proposed the following strategy to exploit the birthday paradox in a collision resistant attack:
 - The source (A) is prepared to sign a legitimate message x by appending the appropriate m -bit hash code and encrypting that hash code with A's private key
 - Opponent generates $2^{m/2}$ variations x' of x , all with essentially the same meaning, and stores the messages and their hash values
 - Opponent prepares a fraudulent message y for which A's signature is desired
 - Opponent generates minor variations y' of y , all of which convey essentially the same meaning. For each y' , the opponent computes $H(y')$, checks for matches with any of the $H(x')$ values, and continues until a match is found. That is, the process continues until a y' is generated with a hash value equal to the hash value of one of the x' values
 - The opponent offers the valid variation to A for signature which can then be attached to the fraudulent variation for transmission to the intended recipient
 - Because the two variations have the same hash code, they will produce the same signature and the opponent is assured of success even though the encryption key is not known

Collision Resistant Attacks



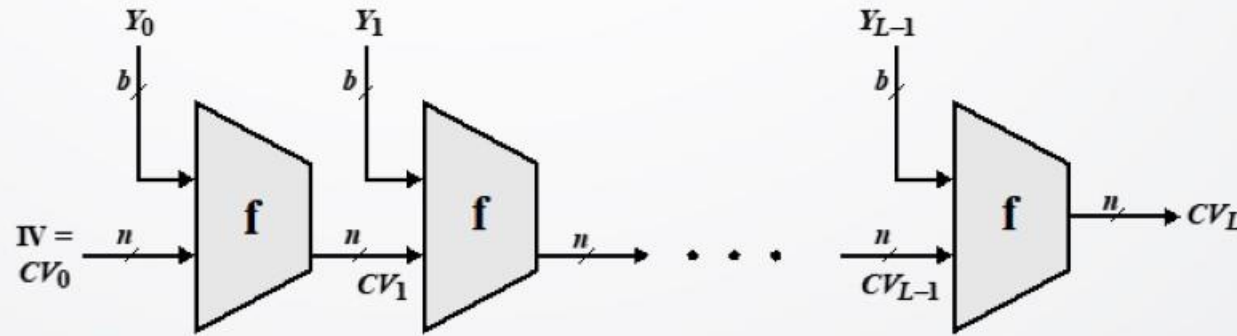
A Letter in 2³⁸ Variation

(Letter is located on page 330 in textbook)

- In **1994**, Van Oorschot and Wiener proposed a machine to find hash collisions for **MD5** (128-bit hash).
- Cost **\$10 million** collision search machine.
- **Collision time**: 24 days for a 128-bit MD5 hash.

As { the } Dean of Blakewell College, I have { had the pleasure of knowing } Cherise
 -- } known
 Rosetti for the { last } four years. She { has been } { a tremendous } { asset to }
 past } was } an outstanding } role model in }
 { our } school. I { would like to take this opportunity to } recommend Cherise for your
 the } wholeheartedly }
 { school's } graduate program. I { am } { confident } { that } { she } will
 -- } feel } certain } -- } Cherise }
 { continue to } succeed in her studies. { She } is a dedicated student and
 -- } Cherise }
 { thus far her grades } { have been } { exemplary } . In class, { she }
 her grades thus far } are } excellent } Cherise }
 { has proven to be } a take-charge { person } { who is } able to successfully develop
 has been } individual } -- } plans and implement them.
 { She } has also assisted { us } in our admissions office. { She } has
 Cherise } -- }
 { successfully } demonstrated leadership ability by counseling new and prospective students.
 -- }
 { Her } advice has been { a great } help to these students, many of whom have
 Cherise's } of considerable }
 { taken time to share } their comments with me regarding her pleasant and { encouraging }
 shared } reassuring }
 attitude. { For these reasons } I { highly recommend } Cherise
 { It is for these reasons that } offer high recommendations for }
 { without reservation } . Her { ambition } and { abilities } will { truly } be an
 unreservedly } drive } potential } surely }
 { asset to } your { establishment }
 plus for } school }

Cryptanalysis of Hash Functions



IV = Initial value
 CV_i = chaining variable
 Y_i = i th input block
 f = compression algorithm

L = number of input blocks
 n = length of hash code
 b = length of input block

- **Focus:** Analyzing the **internal structure** of the **compression function** in hash algorithms.
- **Goal:** Develop **efficient techniques** to produce **collisions** in a single execution of the compression function.
- **Challenges:**
 - Once a collision is found, the attack must consider the **fixed IV (Initialization Vector)**.
- **Attack Process:**
 - Relies on **exploiting** the compression function's internal structure.
 - The attack analyzes the **pattern of bit changes** across Hash's rounds to identify weaknesses.

Figure 11.8 General Structure of Secure Hash Code

Secure Hash Algorithm (SHA)

- SHA was originally designed by the National Institute of Standards and Technology (NIST) and published as a federal information processing standard (FIPS 180) in 1993
- Was revised in 1995 as SHA-1
- Based on the hash function MD4 and its design closely models MD4
- Produces 160-bit hash values
- In 2002 NIST produced a revised version of the standard that defined three new versions of SHA with hash value lengths of 256, 384, and 512
 - Collectively known as SHA-2

Table 11.3

Comparison of SHA Parameters

Algorithm	Message Size	Block Size	Word Size	Message Digest Size
SHA-1	$< 2^{64}$	512	32	160
SHA-224	$< 2^{64}$	512	32	224
SHA-256	$< 2^{64}$	512	32	256
SHA-384	$< 2^{128}$	1024	64	384
SHA-512	$< 2^{128}$	1024	64	512
SHA-512/224	$< 2^{128}$	1024	64	224
SHA-512/256	$< 2^{128}$	1024	64	256

In 2005, NIST announced the intention to phase out approval of SHA-1 and move to a reliance on SHA-2 by 2010.

Note: All sizes are measured in bits.

SHA-3

Sha 1 considered broken due to vulnerabilities to collision attacks.

Cryptographers demonstrated practical collision attacks on SHA-1.

Notable example: 2017 SHattered attack by Google and CWI Amsterdam, which created two distinct inputs with the same SHA-1 hash.

NIST announced in 2007 a competition for the SHA-3 next generation NIST hash function

- **Winning design was announced by NIST in October 2012**
- **SHA-3 is a cryptographic hash function that is intended to complement SHA-2 as the approved standard for a wide range of applications**

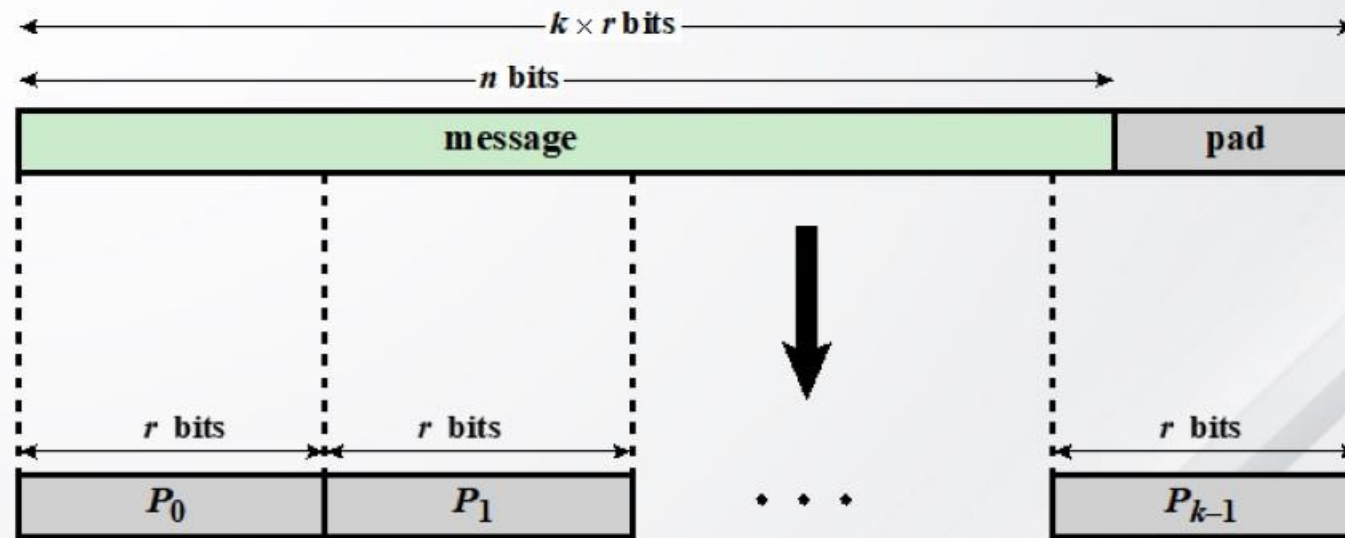
SHA-2 shares the same structure and mathematical operations as its predecessors so this is a cause for concern

- **Because it will take years to find a suitable replacement for SHA-2 should it become vulnerable, NIST decided to begin the process of developing a new hash standard**

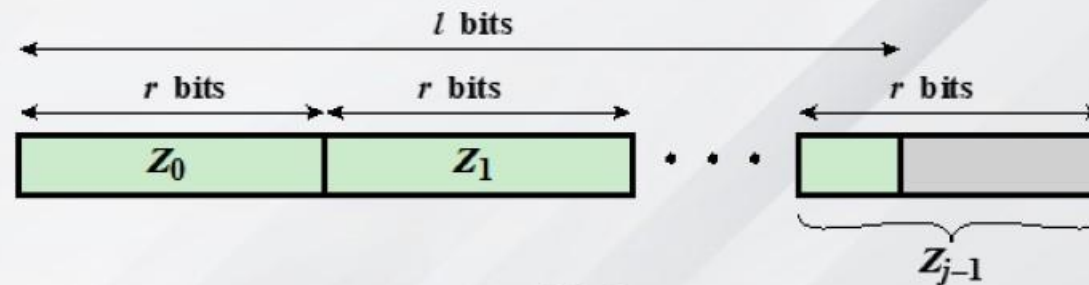
The Sponge Construction

- Underlying structure of SHA-3 is a scheme referred to by its designers as a *sponge construction*
- Takes an input message and partitions it into fixed-size blocks
- Each block is processed in turn with the output of each iteration fed into the next iteration, finally producing an output block
- The sponge function is defined by three parameters:
 - f = the internal function used to process each input block
 - r = the size in bits of the input blocks, called the *bitrate*
 - pad = the padding algorithm





(a) Input



(b) Output

Figure 11.14 Sponge Function Input and Output

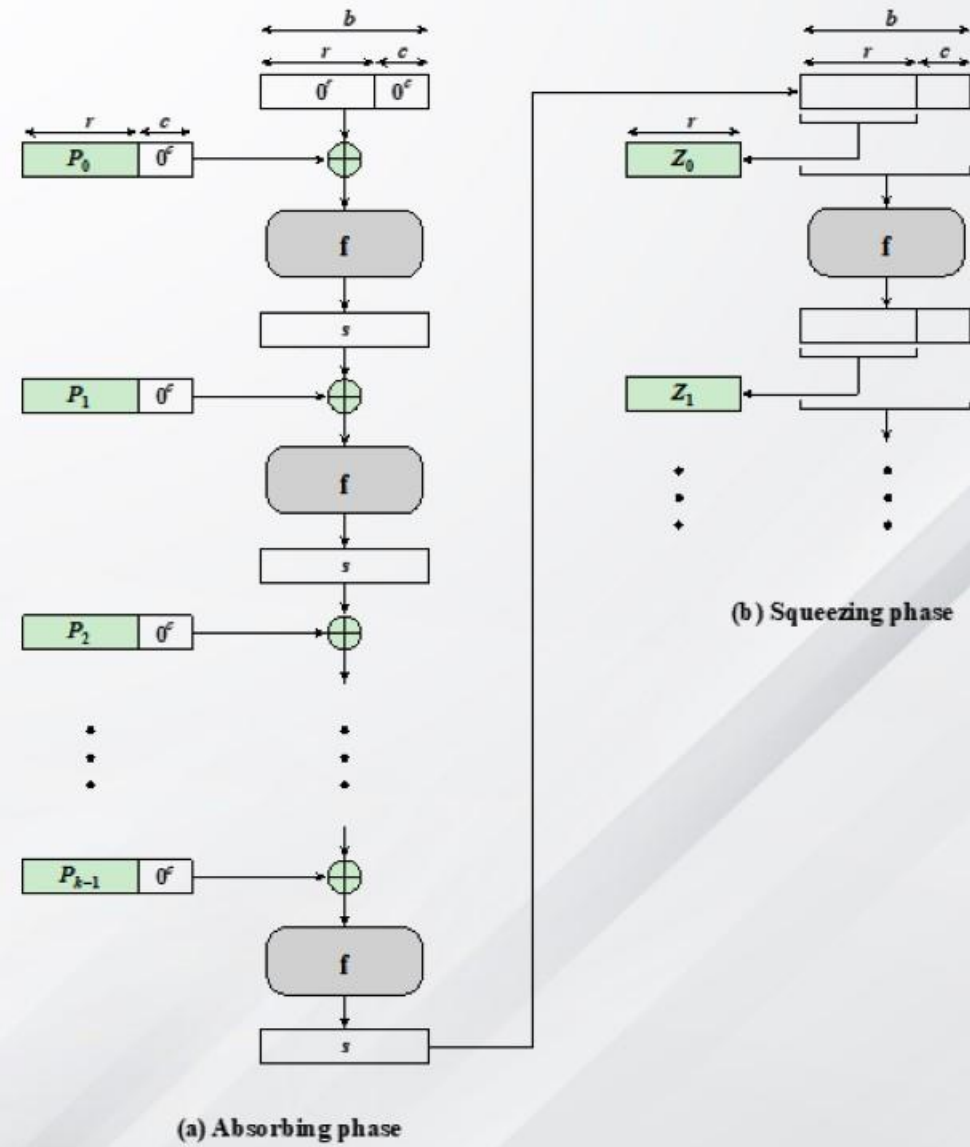


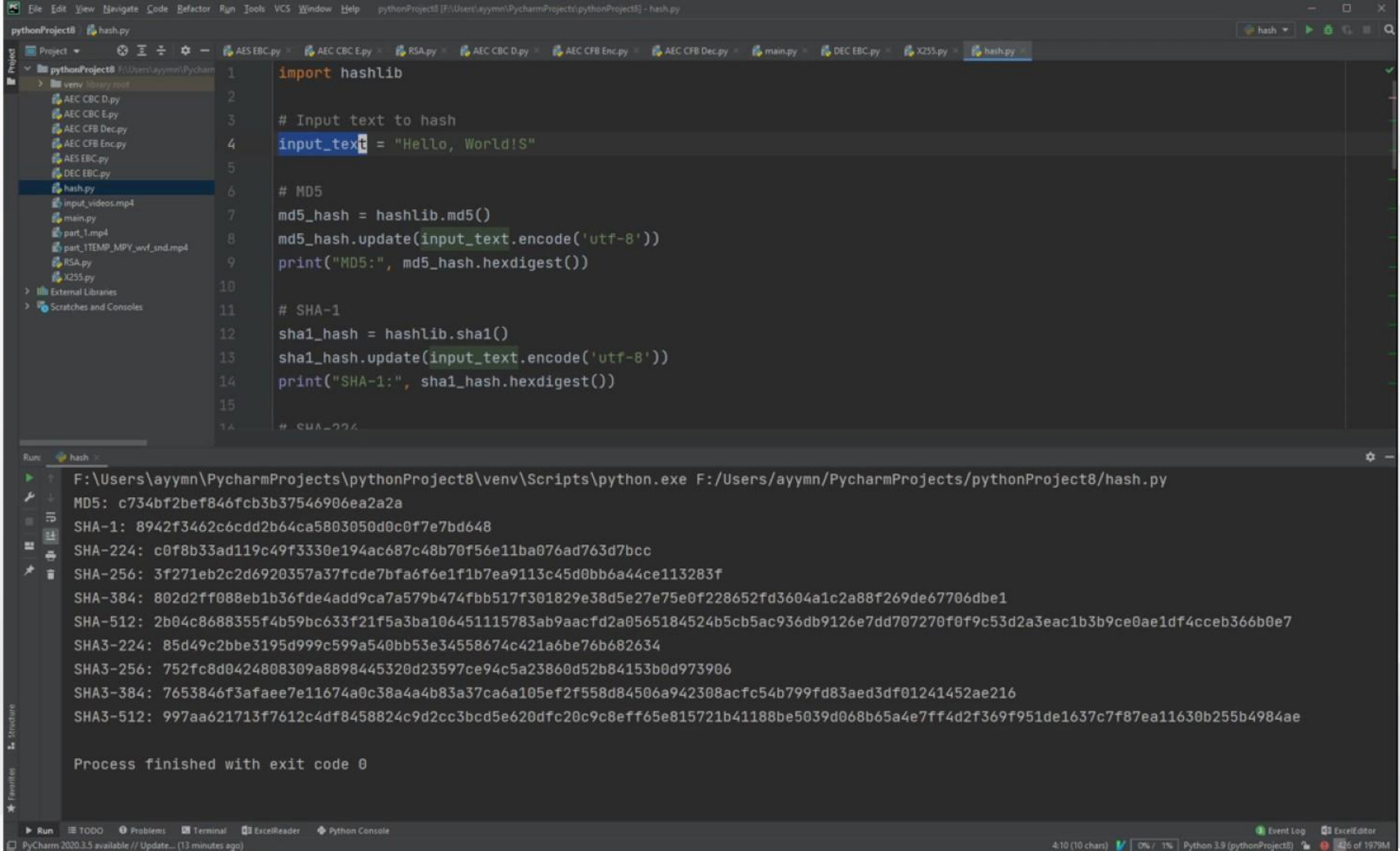
Figure 11.15 Sponge Construction

Table 11.5

SHA-3 Parameters

Message Digest Size	224	256	384	512
Message Size	no maximum	no maximum	no maximum	no maximum
Block Size (bitrate r)	1152	1088	832	576
Word Size	64	64	64	64
Number of Rounds	24	24	24	24
Capacity c	448	512	768	1024
Collision resistance	2^{112}	2^{128}	2^{192}	2^{256}
Second preimage resistance	2^{224}	2^{256}	2^{384}	2^{512}

Note: All sizes and security levels are measured in bits.



Summary

- **Hash Functions**

- Ensure **data integrity** by generating fixed-size outputs from variable inputs.
- **Cryptographic Hash:** Computationally infeasible to reverse (one-way) or find collisions.

- **Message Authentication Code (MAC)**

- Keyed hash function used to verify messages between trusted parties.

Digital Signatures

Provide message integrity and non-repudiation using private/public key encryption.

Hash Function Attacks

Brute-force (based on bit length) and **Cryptanalysis** (exploit algorithm flaws).

Collision-Resistant: Birthday paradox, Yuval's strategy to find matching hash values.

Secure Hash Algorithms (SHA)

SHA-1, SHA-2, and SHA-3 developments to strengthen data security.





Thank You

