Week 14

SYMMETRIC ENCRYPTION AND MESSAGE CONFIDENTIALITY

PUBLIC-KEY CRYPTOGRAPHY AND MESSAGE AUTHENTICATION

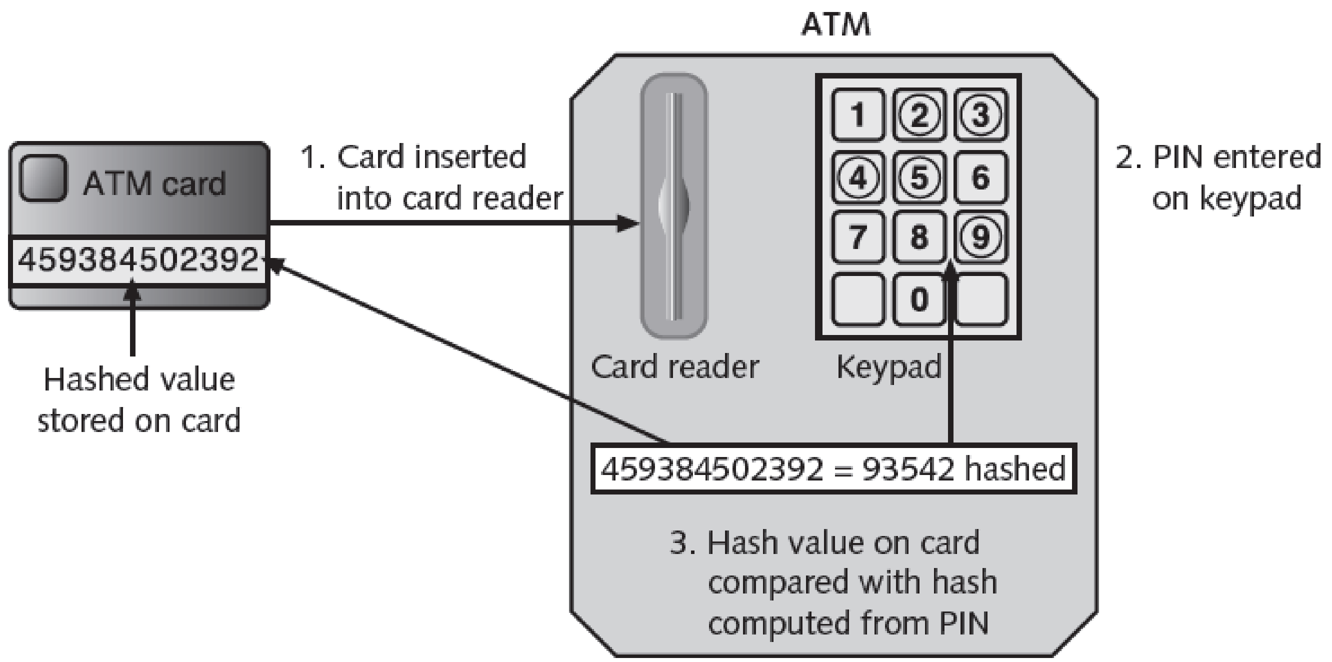
Review

Cryptographic Algorithms

Three categories of cryptographic algorithms

- Hash algorithms
- Symmetric encryption algorithms
- Asymmetric encryption algorithms

Hash Algorithms


ATM
ATM card
459384502392
1. Card inserted into card reader
Hashed value stored on card
Card reader
Keypad
1
2
3
4
5
6
7
8
9
0
2. PIN entered on keypad
459384502392 = 93542 hashed
3. Hash value on card compared with hash computed from PIN

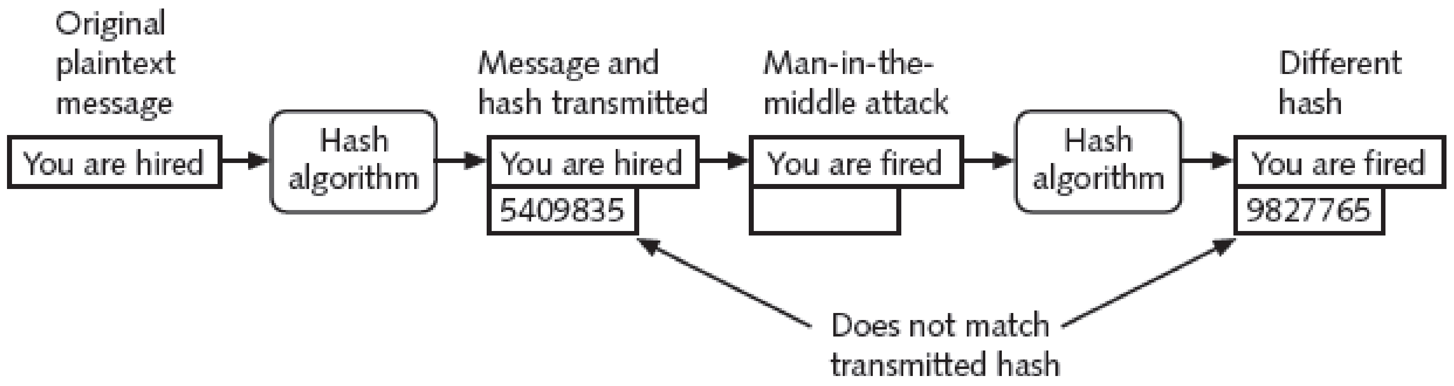
Example of hashing (ATMs)

- Bank customer has PIN of 93542
- Number is hashed and result stored on card's magnetic stripe
- User inserts card in ATM and enters PIN
- ATM hashes the pin using the same algorithm that was used to store PIN on the card
- If two values match, user may access ATM

Hashing Algorithms

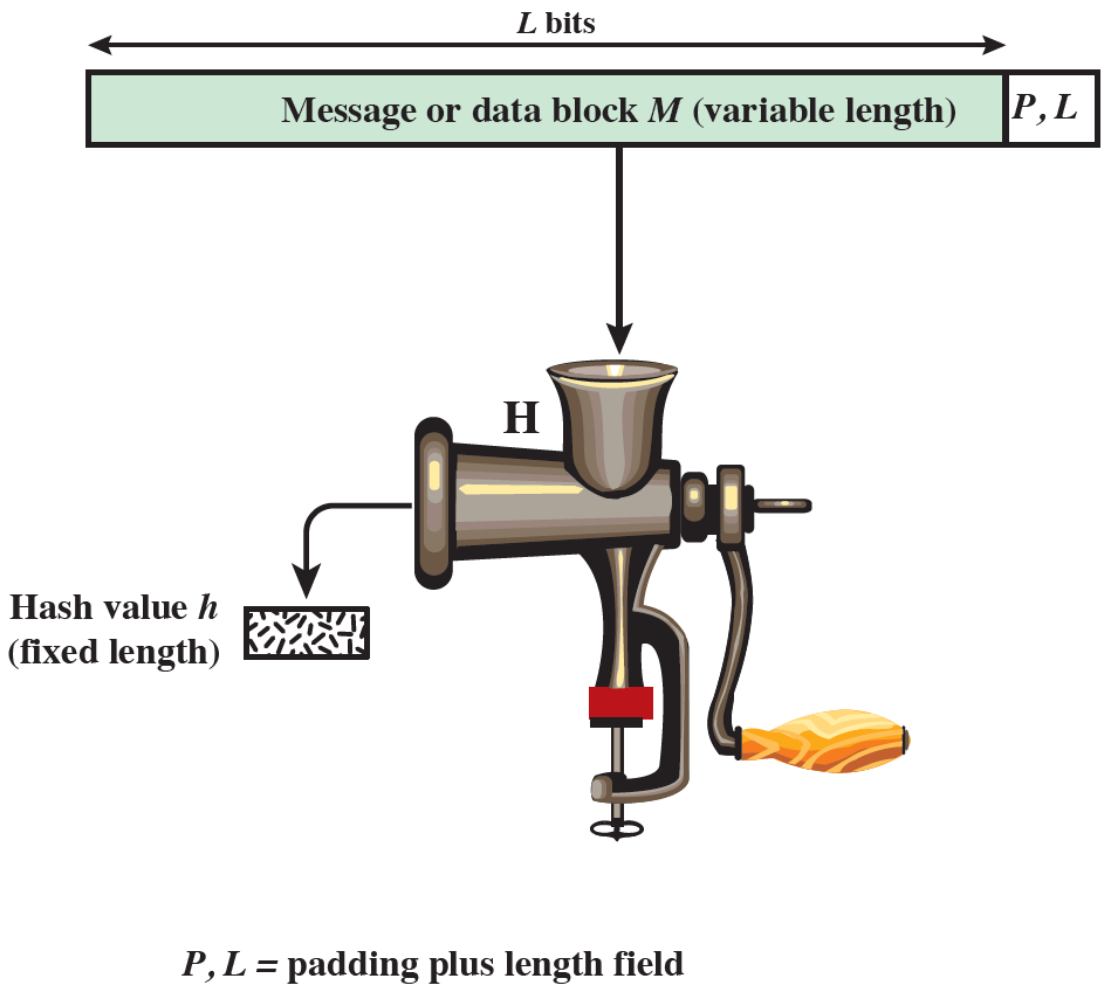
Hashing used to determine message integrity

- Can protect against man-in-the-middle attacks



Hash values often posted on download sites

- To verify file integrity after download
- Checksum is a kind of hash



L bits
Message or data block M (variable length)
P, L
H
Hash value h (fixed length)
P, L = padding plus length field

Figure 2.4 Cryptographic Hash Function; $h = \mathrm{H}(M)$

Hash Function

Symmetric Encryption

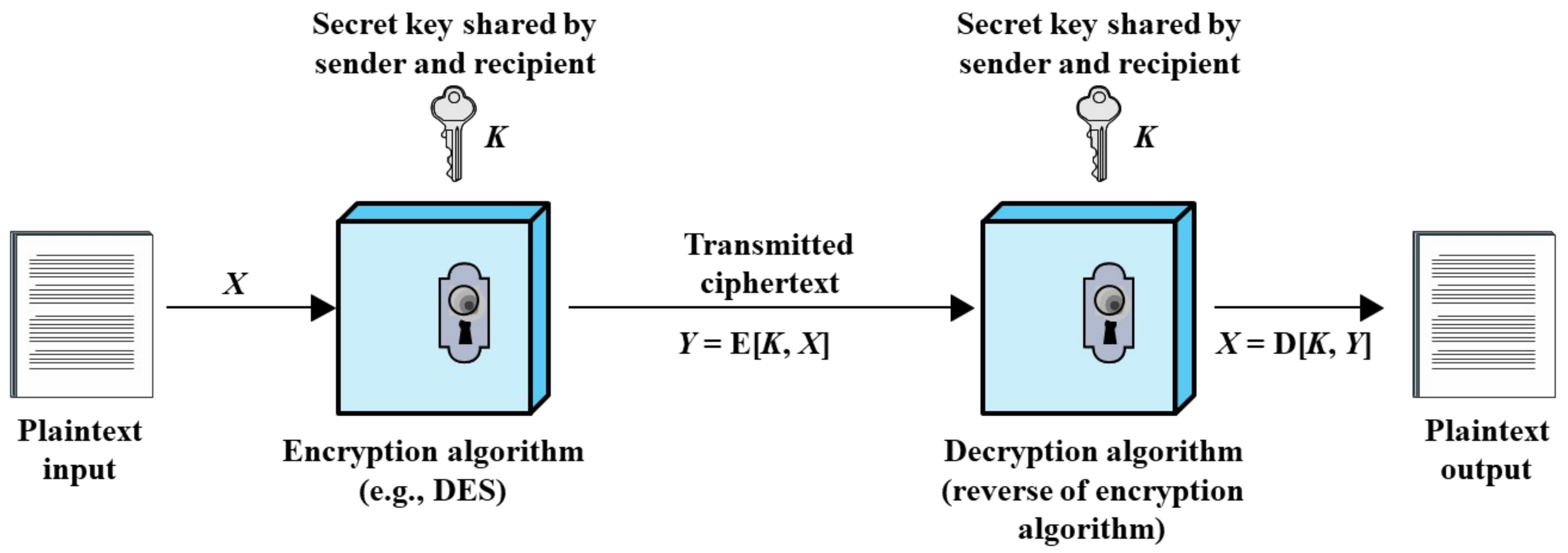
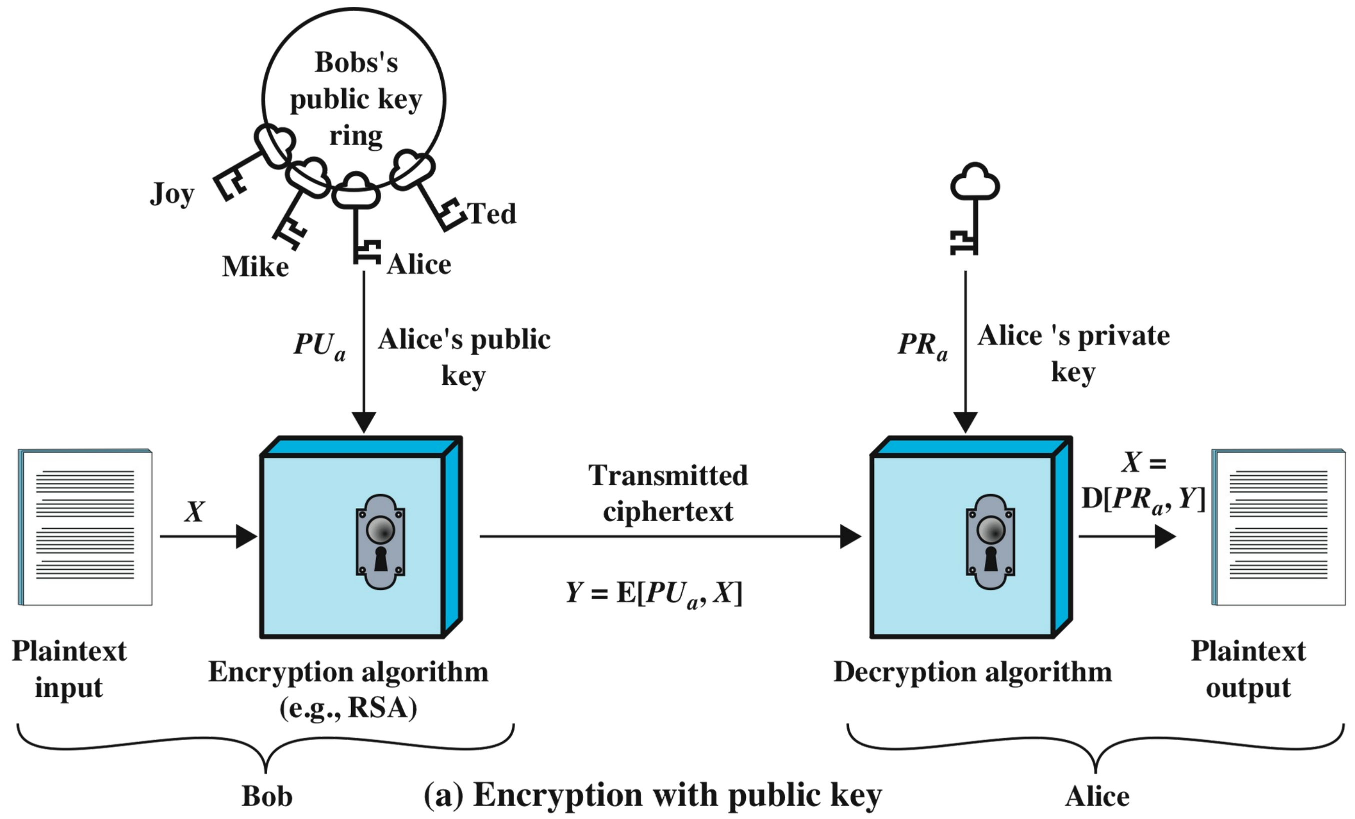

Secret key shared by sender and recipient
K
Secret key shared by sender and recipient
K
X
Transmitted ciphertext
$Y = \mathrm{E}[K, X]$
$X = \mathrm{D}[K, Y]$
Plaintext input
Encryption algorithm (e.g., DES)
Decryption algorithm (reverse of encryption algorithm)
Plaintext output

Figure 2.1 Simplified Model of Symmetric Encryption

Asymmetric Encryption



(a) Encryption with public key

- **Plaintext**
  - Readable message or data that is fed into the algorithm as input
- **Encryption algorithm**
  - Performs transformations on the plaintext
- **Public and private key**
  - Pair of keys, one for encryption, one for decryption
- **Ciphertext**
  - Scrambled message produced as output
- **Decryption key**
  - Produces the original plaintext

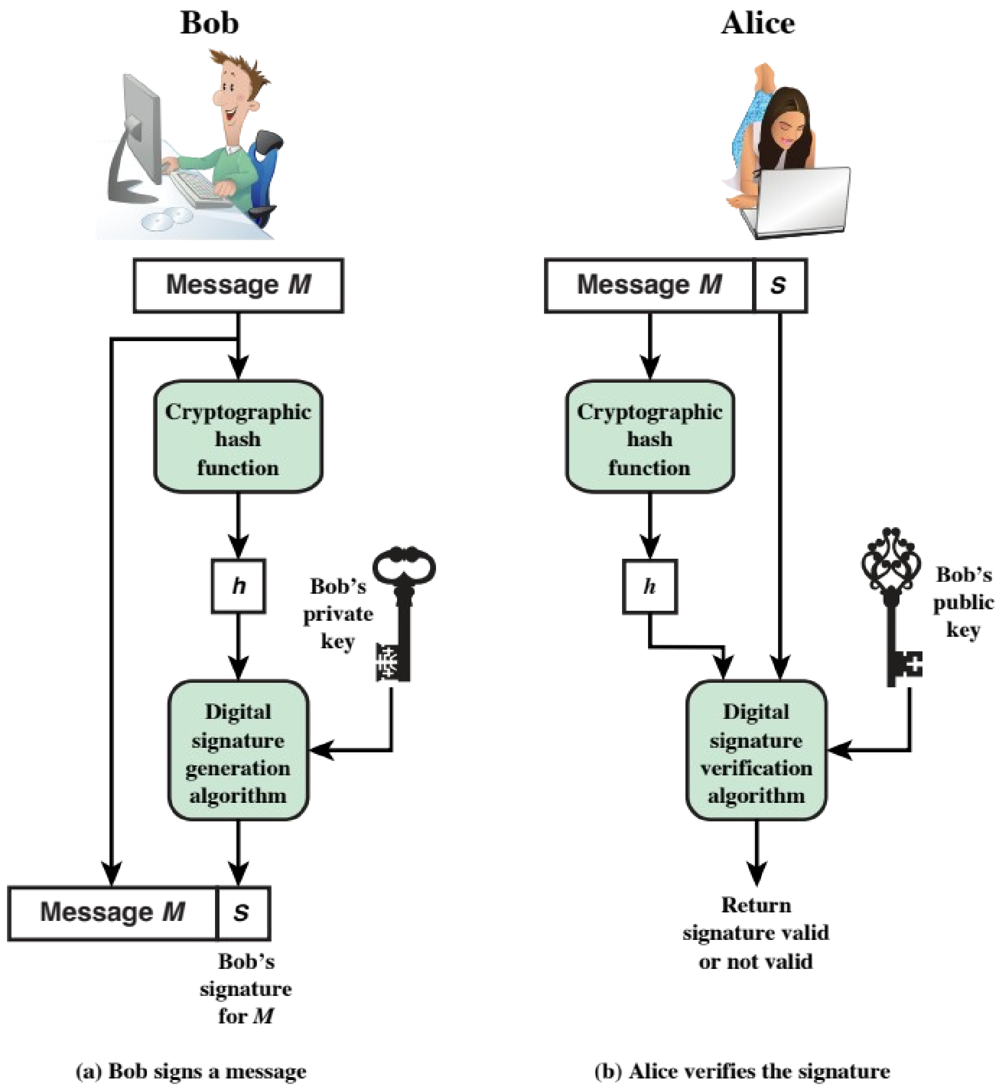

Bob
Alice
Message M
Message M
S
Cryptographic hash function
Cryptographic hash function
h
h
Bob's private key
Bob's public key
Digital signature generation algorithm
Digital signature verification algorithm
Message M
S
Bob's signature for M
Return signature valid or not valid
(a) Bob signs a message
(b) Alice verifies the signature

Figure 2.7 Simplified Depiction of Essential Elements of Digital Signature Process

Digital Signatures

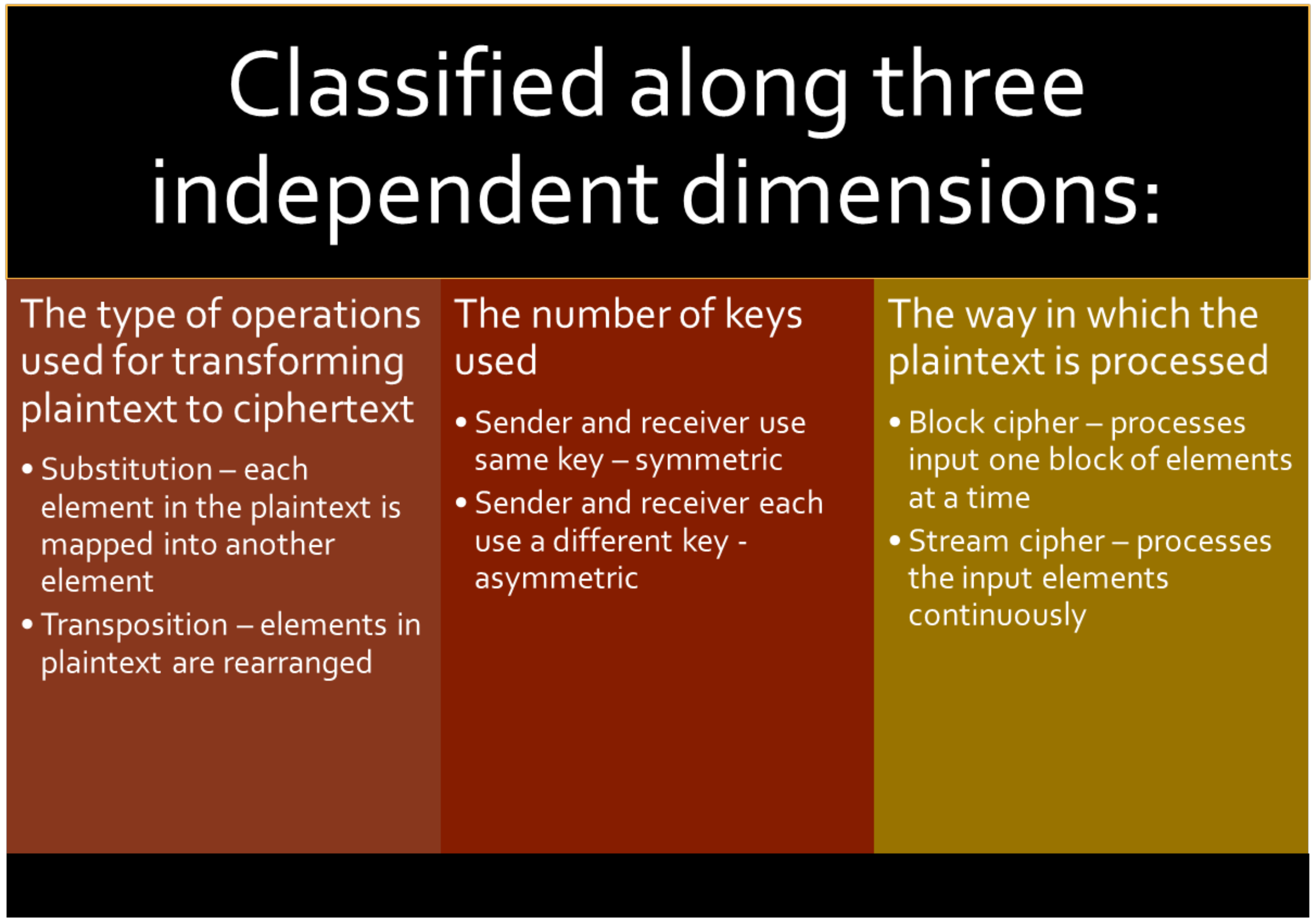
Chapter 20

Symmetric Encryption and Message Confidentiality

Symmetric Encryption

Also referred to as:

- Conventional encryption
- Secret-key or single-key encryption
- Only alternative before public-key encryption in 1970’s
- Still most widely used alternative
- Has five ingredients:
  - Plaintext
  - Encryption algorithm
  - Secret key
  - Ciphertext
  - Decryption algorithm



Classified along three independent dimensions:

The type of operations used for transforming plaintext to ciphertext	The number of keys used	The way in which the plaintext is processed
• Substitution – each element in the plaintext is mapped into another element • Transposition – elements in plaintext are rearranged	• Sender and receiver use same key – symmetric • Sender and receiver each use a different key - asymmetric	• Block cipher – processes input one block of elements at a time • Stream cipher – processes the input elements continuously

Cryptography

Cryptography

Attacks on Encrypted Messages

Table 20.1 Types of Attacks on Encrypted Messages

Type of Attack	Known to Cryptanalyst
Ciphertext only	•Encryption algorithm •Ciphertext to be decoded
Known plaintext	•Encryption algorithm •Ciphertext to be decoded •One or more plaintext-ciphertext pairs formed with the secret key
Chosen plaintext	•Encryption algorithm •Ciphertext to be decoded •Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key

Computationally Secure Encryption Schemes

Encryption is computationally secure if:

- Cost of breaking cipher exceeds value of information
- Time required to break cipher exceeds the useful lifetime of the information

Usually very difficult to estimate the amount of effort required to break

Can estimate time/cost of a brute-force attack

AES Encryption

https://www.youtube.com/watch?v=gP4PqVGudtg

AES Encryption

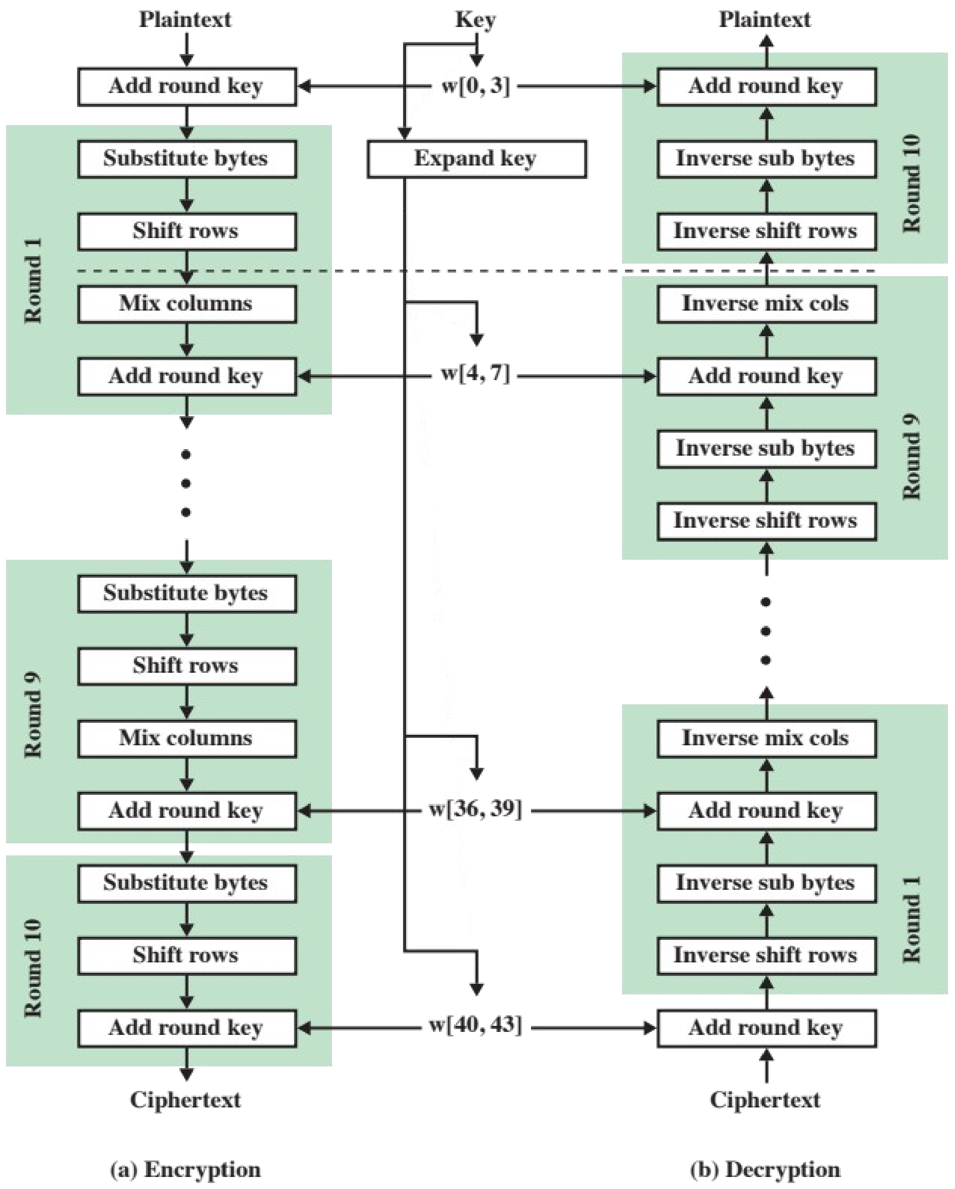


Figure 20.3 AES Encryption and Decryption

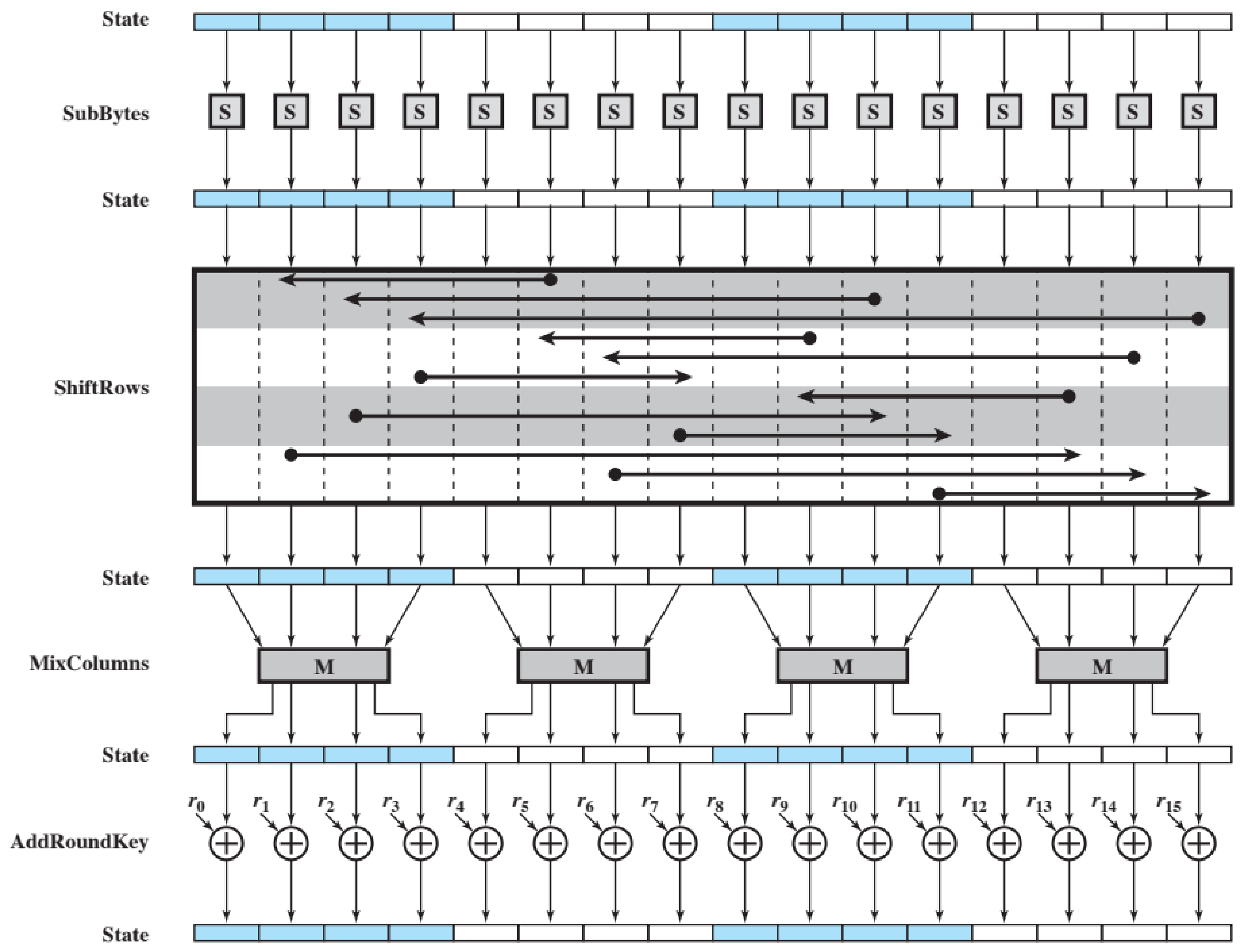


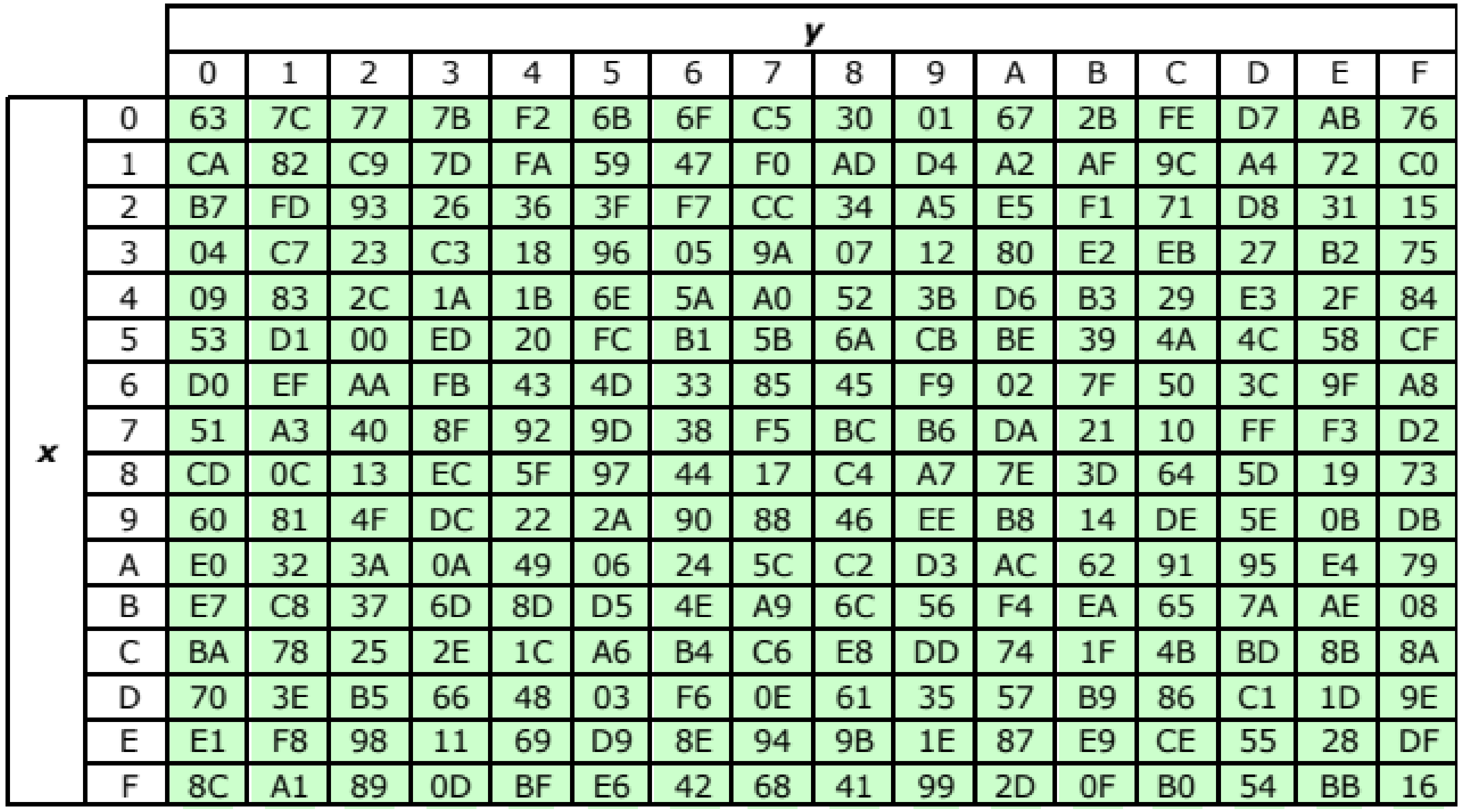
Figure 20.4 AES Encryption Round

AES Encryption Round

AES S-Boxes

Table 20.2 AES S-Boxes

(a) S-box



		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Performance Comparisons

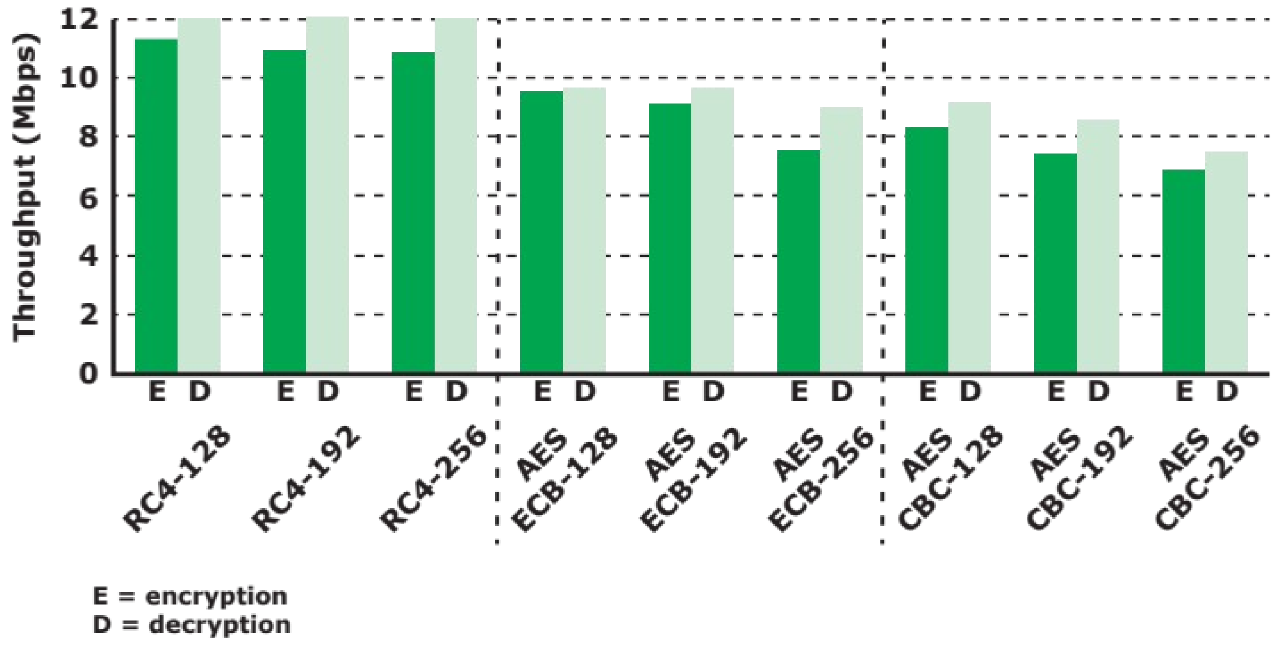


Figure 20.5 Performance Comparison of Symmetric Ciphers on a 3-GHz Processor

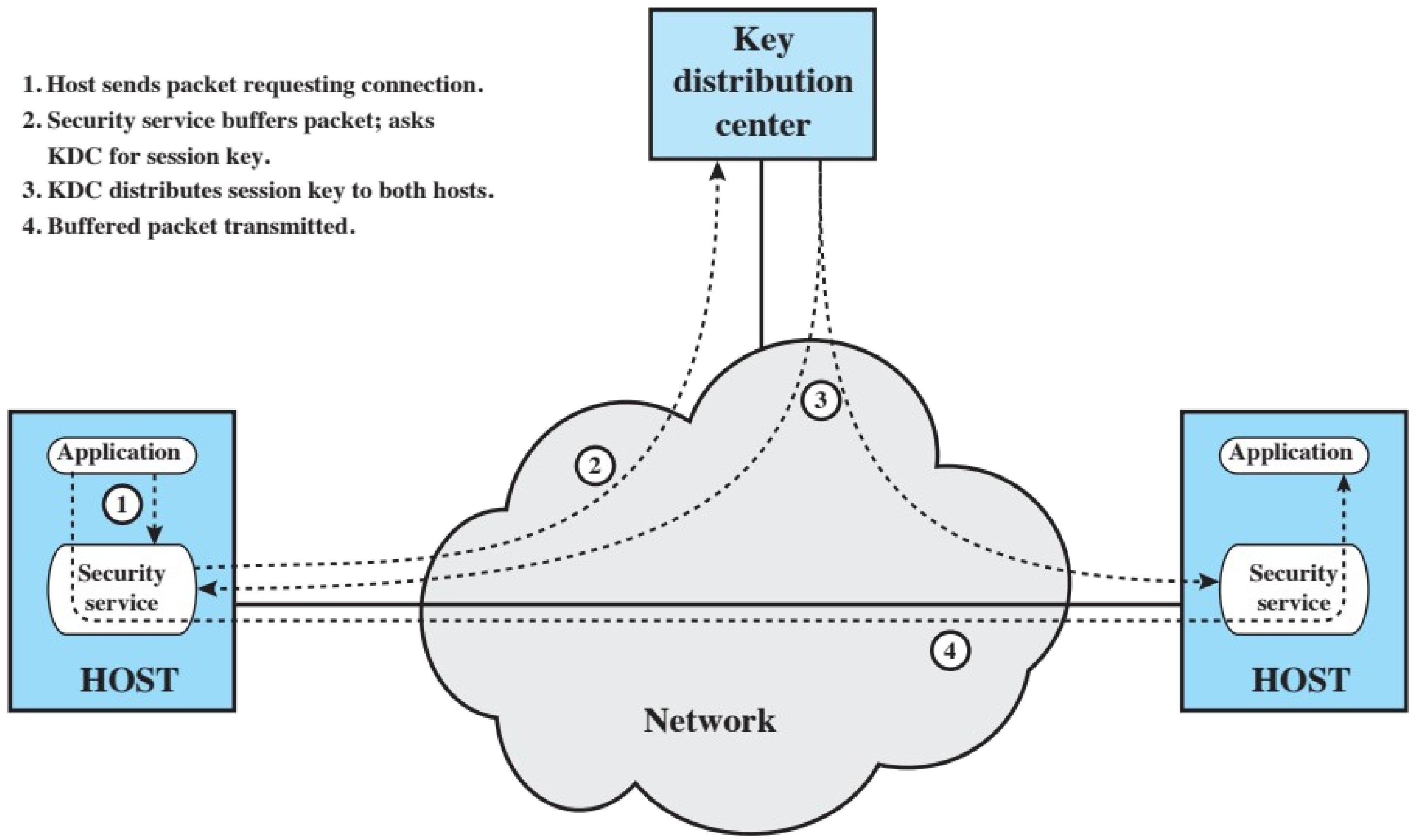
Key Distribution

The means of delivering a key to two parties that wish to exchange data without allowing others to see the key

Two parties (A and B) can achieve this by:

1. A key could be selected by A and physically delivered to B
2. A third party could select the key and physically deliver it to A and B
3. If A and B have previously and recently used a key, one party could transmit the new key to the other, encrypted using the old key
4. If A and B each have an encrypted connection to a third party C, C could deliver a key on the encrypted links to A and B

Automatic Key Distribution



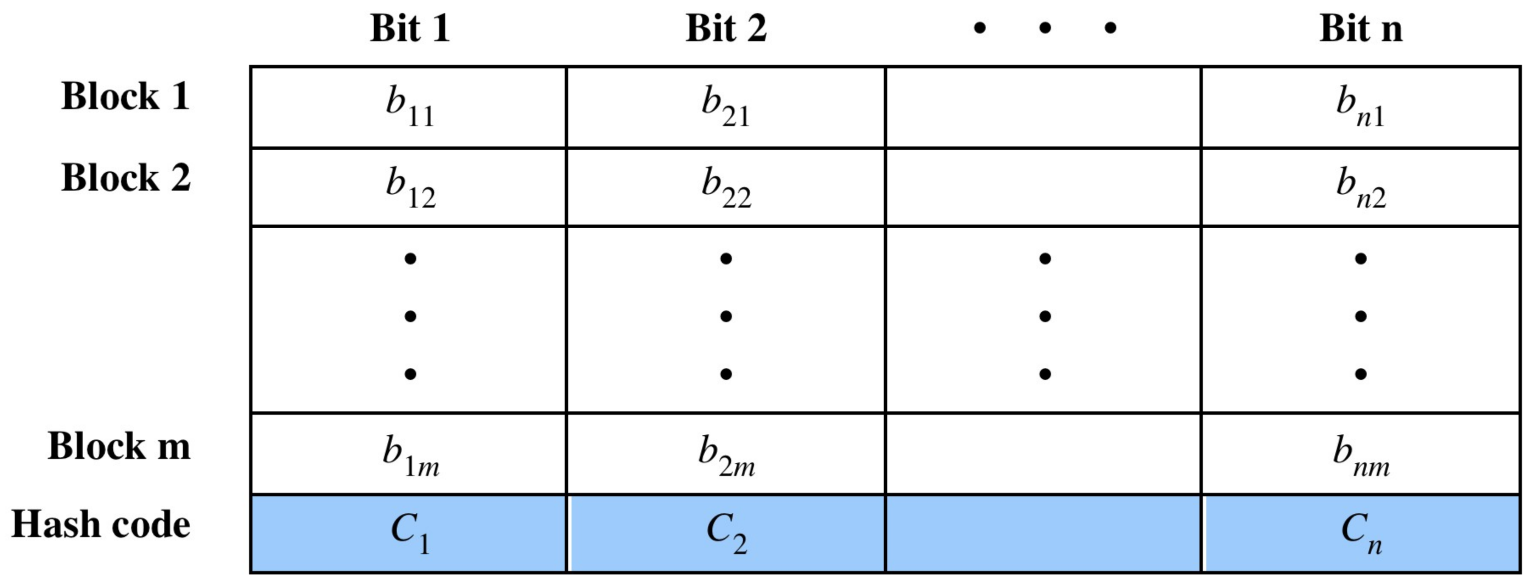
1. Host sends packet requesting connection.
2. Security service buffers packet; asks KDC for session key.
3. KDC distributes session key to both hosts.
4. Buffered packet transmitted.
Key distribution center
Application
Security service
HOST
1
2
3
4
Network
Application
Security service
HOST

Figure 20.10 Automatic Key Distribution for Connection-Oriented Protocol

Chapter 21

Public-Key Cryptography and Message Authentication

Simple Hash Function



	Bit 1	Bit 2	• • •	Bit n
Block 1	b_{11}	b_{21}		b_{n1}
Block 2	b_{12}	b_{22}		b_{n2}
	⋮	⋮	⋮	⋮
Block m	b_{1m}	b_{2m}		b_{nm}
Hash code	C_1	C_2		C_n

Figure 21.1 Simple Hash Function Using Bitwise XOR

Secure Hash Algorithm (SHA)

SHA was originally developed by NIST

Published as FIPS 180 in 1993

Was revised in 1995 as SHA-1

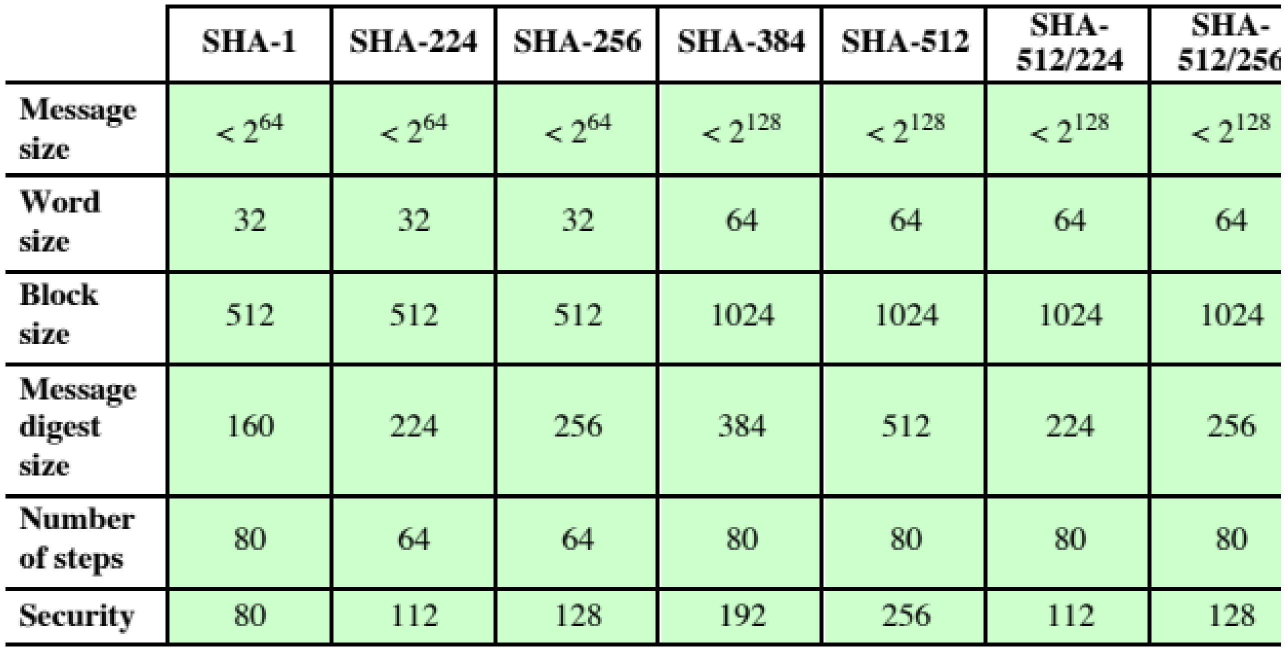
- Produces 160-bit hash values

NIST issued revised FIPS 180-2 in 2002

- Adds 3 additional versions of SHA
- SHA-256, SHA-384, SHA-512
- With 256/384/512-bit hash values
- Same basic structure as SHA-1 but greater security

The most recent version is FIPS 180-4 which added two variants of SHA-512 with 224-bit and 256-bit hash sizes

Comparison of SHA Parameters



	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512	SHA-512/224	SHA-512/256
Message size	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$	$< 2^{128}$	$< 2^{128}$
Word size	32	32	32	64	64	64	64
Block size	512	512	512	1024	1024	1024	1024
Message digest size	160	224	256	384	512	224	256
Number of steps	80	64	64	80	80	80	80
Security	80	112	128	192	256	112	128

Notes: 1. All sizes are measured in bits.

2. Security refers to the fact that a birthday attack on a message digest of size *n* produces a collision with a work factor of approximately $2^{n/2}$.

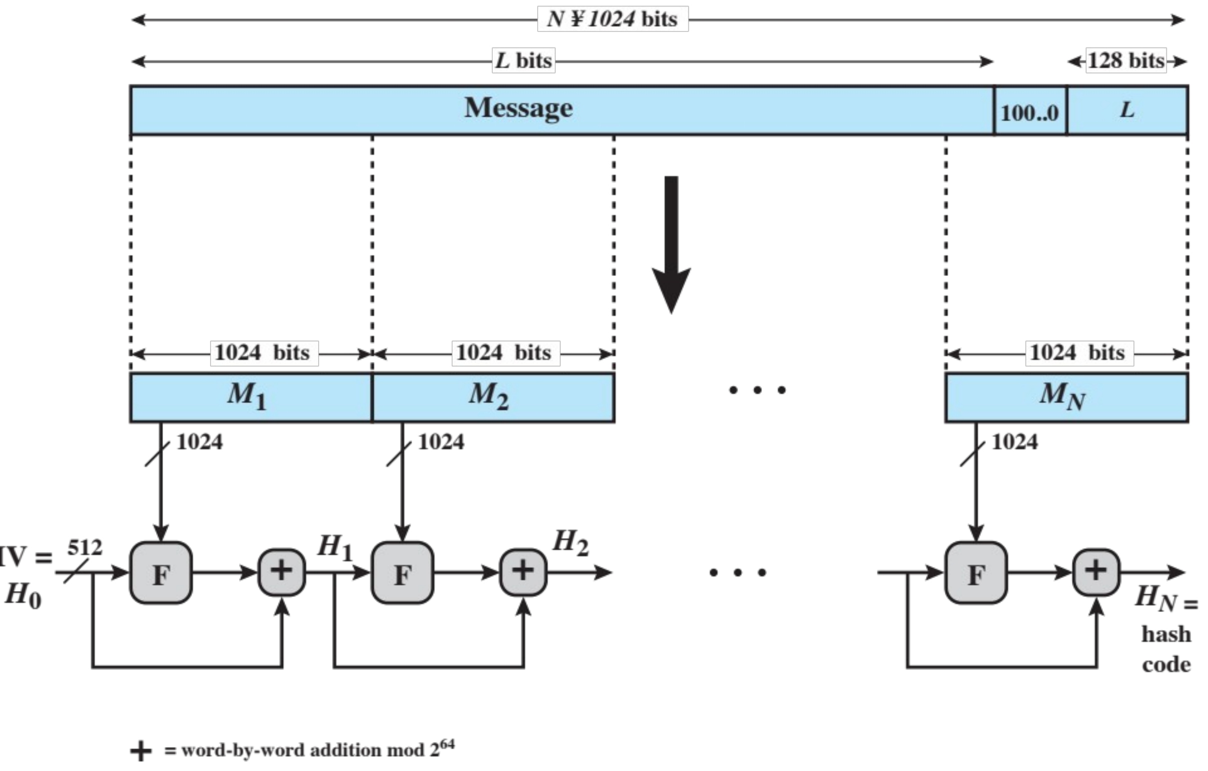


Figure 21.2 Message Digest Generation Using SHA-512

Message Digest Generation

● **Step 1: Append padding bits**: so that message length is congruent to 896 modulo 1024 [length ≡ 896 (mod 1024)]. The padding consists of a single 1-bit followed by the necessary number of 0-bits.

● **Step 2: Append length:** as a block of 128 bits being an unsigned 128-bit integer length of the original message (before padding).

● **Step 3: Initialize hash buffer:** A 512-bit buffer is used to hold intermediate and final results of the hash function. The buffer can be represented as eight 64-bit registers (a, b, c, d, e, f, g, h).

● **Step 4: Process the message in 1024-bit (128-word) blocks**, The heart of the algorithm is a module that consists of 80 rounds; this module is labeled F in Figure 21.2.

• **Step 5: Output.** After all N 1024-bit blocks have been processed, the output from the N th stage is the 512-bit message digest.

HMAC

Interest in developing a MAC derived from a cryptographic hash code

- Cryptographic hash functions generally execute faster
- Library code is widely available
- SHA-1 was not designed for use as a MAC because it does not rely on a secret key
- Issued as RFC2014
- Has been chosen as the mandatory-to-implement MAC for IP security
- Used in other Internet protocols such as Transport Layer Security (TLS) and Secure Electronic Transaction (SET)

HMAC Design Objectives

To use, without modifications, available hash functions

To allow for easy replaceability of the embedded hash function in case faster or more secure hash functions are found or required

To preserve the original performance of the hash function without incurring a significant degradation

To use and handle keys in a simple way

To have a well-understood cryptographic analysis of the strength of the authentication mechanism based on reasonable assumptions on the embedded hash function

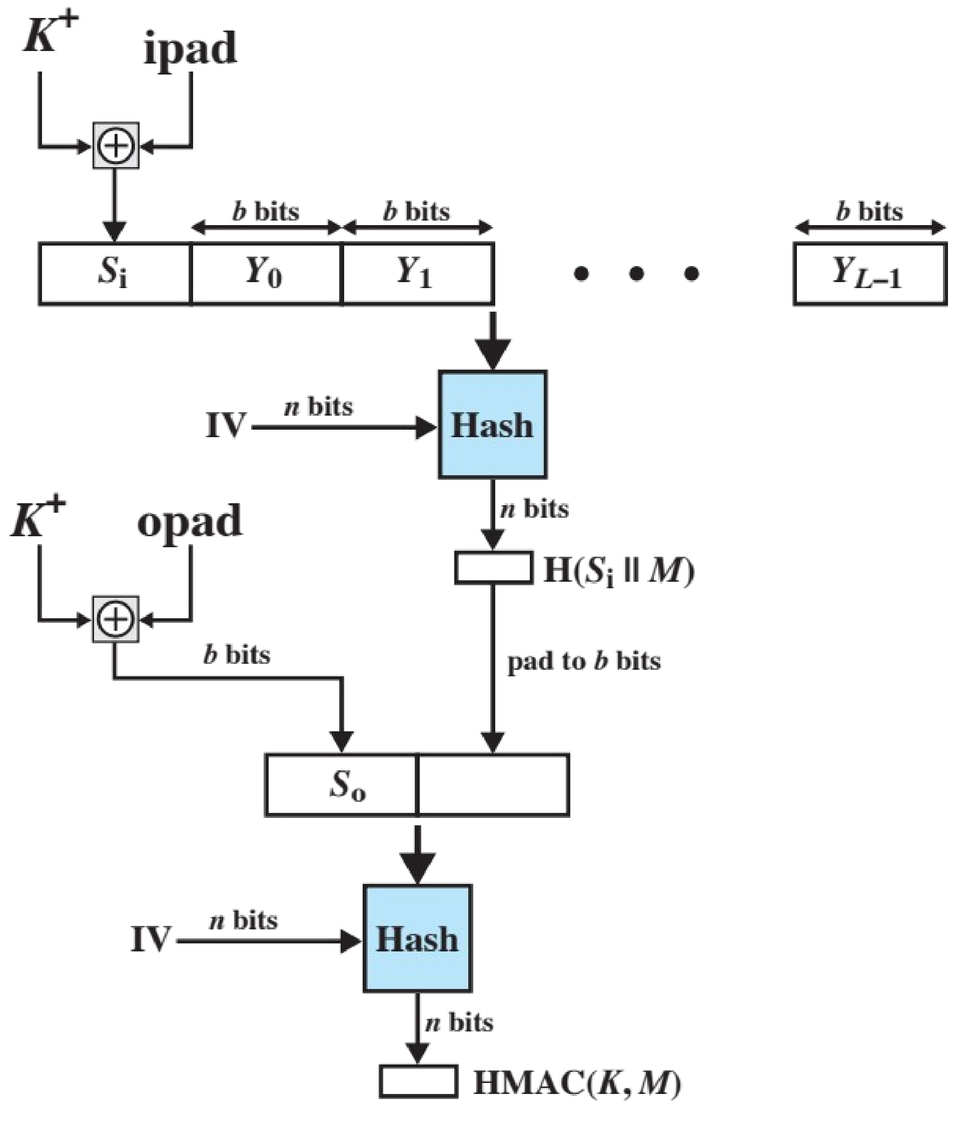


Figure 21.4 HMAC Structure

HMAC Structure

1. Append zeros to the left end of *K* to create a *b*-bit string K^+ (e.g., if *K* is of length 160 bits and *b* = 512, then *K* will be appended with 44 zero bytes 0x00).

2. XOR (bitwise exclusive-OR) K^+ with ipad to produce the *b*-bit block S_i.

3. Append *M* to S_i.

4. Apply H to the stream generated in step 3.

5. XOR K^+ with opad to produce the *b*-bit block S_o.

6. Append the hash result from step 4 to S_o.

7. Apply H to the stream generated in step 6 and output the result.

Security of HMAC

Security depends on the cryptographic strength of the underlying hash function

The appeal of HMAC is that its designers have been able to prove an exact relationship between the strength of the embedded hash function and the strength of HMAC

For a given level of effort on messages generated by a legitimate user and seen by the attacker, the probability of successful attack on HMAC is equivalent to one of the following attacks on the embedded hash function:

- The attacker is able to compute an output of the compression function even with an IV that is random, secret, and unknown to the attacker
- The attacker finds collisions in the hash function even when the IV is random and secret

RSA Public-key Encryption

- By Rivest, Shamir & Adleman of MIT in 1977
- Best known and widely used public-key algorithm
- Uses exponentiation of integers modulo a prime
- Encrypt: C = Me mod n
- Decrypt: M = Cd mod n = (Me)d mod n = M
- Both sender and receiver know values of n and e
- Only receiver knows value of d
- Public-key encryption algorithm with public key PU = {e, n} and private key PR = {d, n}
- https://www.youtube.com/watch?v=wXB-V_Keiu8

Security of RSA

Brute force

- Involves trying all possible private keys

Mathematical attacks

- There are several approaches, all equivalent in effort to factoring the product of two primes

Timing attacks

- These depend on the running time of the decryption algorithm

Chosen ciphertext attacks

- This type of attack exploits properties of the RSA algorithm

Timing Attacks

Paul Kocher, a cryptographic consultant, demonstrated that a snooper can determine a private key by keeping track of how long a computer takes to decipher messages

Timing attacks are applicable not just to RSA, but also to other public-key cryptography systems

This attack is alarming for two reasons:

- It comes from a completely unexpected direction
- It is a ciphertext-only attack

Timing Attack Countermeasures

Constant exponentiation time

- Ensure that all exponentiations take the same amount of time before returning a result
- This is a simple fix but does degrade performance

Random delay

- Better performance could be achieved by adding a random delay to the exponentiation algorithm to confuse the timing attack
- If defenders do not add enough noise, attackers could still succeed by collecting additional measurements to compensate for the random delays

Blinding

- Multiply the ciphertext by a random number before performing exponentiation
- This process prevents the attacker from knowing what ciphertext bits are being processed inside the computer and therefore prevents the bit-by-bit analysis essential to the timing attack

Diffie-Hellman Key Exchange

First published public-key algorithm

By Diffie and Hellman in 1976 along with the exposition of public key concepts

Used in a number of commercial products

Practical method to exchange a secret key securely that can then be used for subsequent encryption of messages

Security relies on difficulty of computing discrete logarithms

Diffie-Hellman Depicted

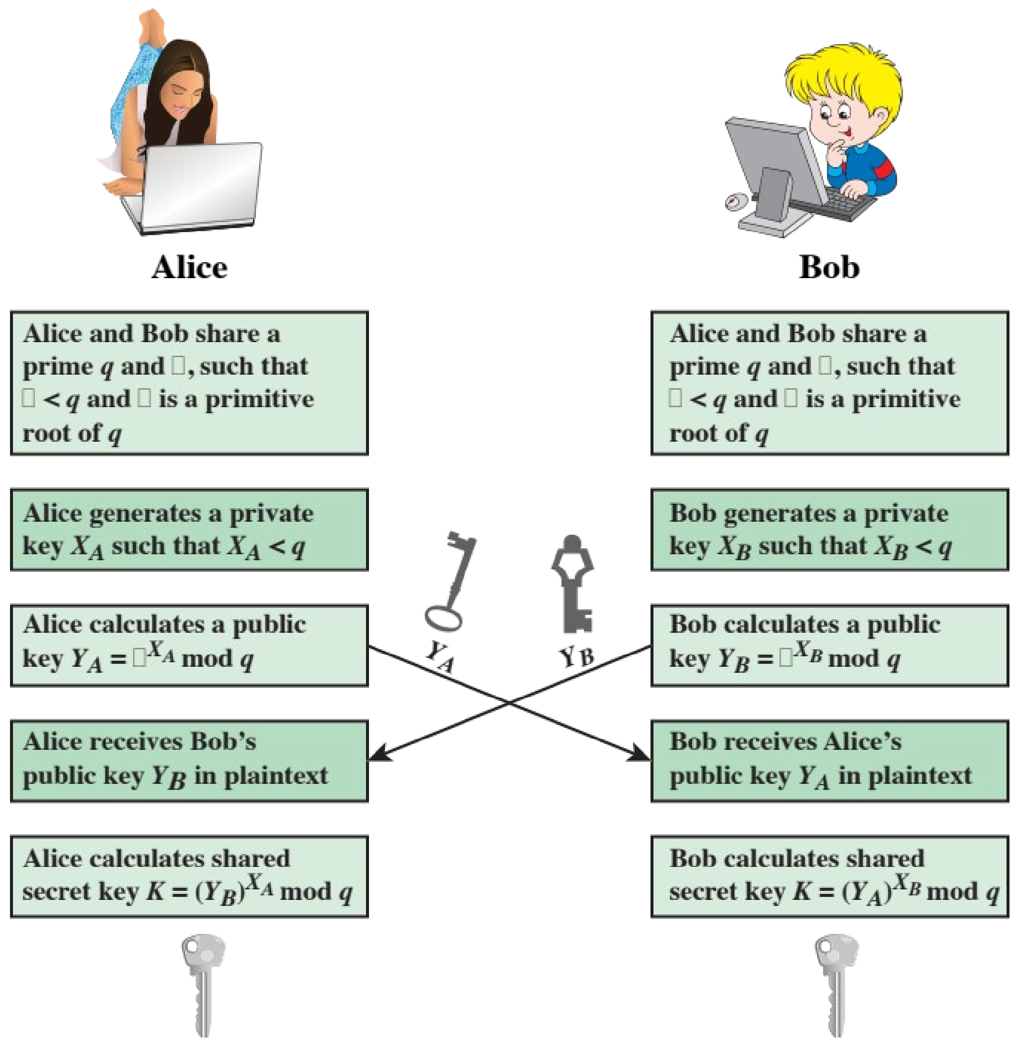

Alice
Bob
Alice and Bob share a prime q and □, such that □ < q and □ is a primitive root of q
Alice and Bob share a prime q and □, such that □ < q and □ is a primitive root of q
Alice generates a private key X_A such that $X_A < q$
Bob generates a private key X_B such that $X_B < q$
Alice calculates a public key $Y_A =$ □^{X_A} mod q
Bob calculates a public key $Y_B =$ □^{X_B} mod q
Y_A
Y_B
Alice receives Bob's public key Y_B in plaintext
Bob receives Alice's public key Y_A in plaintext
Alice calculates shared secret key $K = (Y_B)^{X_A} \bmod q$
Bob calculates shared secret key $K = (Y_A)^{X_B} \bmod q$

Figure 21.10 Diffie-Hellman Key Exchange

Man-in-the-middle Attack

The protocol depicted in Figure 21.10 is insecure against a man-in-the-middle attack. Suppose Alice and Bob wish to exchange keys, and Darth is attacks as follows:

1. Darth generates two private keys X_{D1} and X_{D2}, and public keys Y_{D1} & Y_{D2}.

2. Alice transmits Y_A to Bob.

3. Darth intercepts Y_A and transmits Y_{D1} to Bob. Darth also calculates K2

4. Bob receives Y_{D1} and calculates K1.

5. Bob transmits X_A to Alice.

6. Darth intercepts X_A and transmits Y_{D2} to Alice. Darth calculates K1.

7. Alice receives Y_{D2} and calculates .

At this point, Bob and Alice think that they share a secret key, but instead Bob and Darth share secret key *K*1 and Alice and Darth share secret key *K*2. All future communication between Bob and Alice is compromised in the following way:

1. Alice sends an encrypted message *M*: E(*K*2, *M*).

2. Darth intercepts the encrypted message and decrypts it, to recover *M*.

3. Darth sends Bob E(*K*1, *M*) or E(*K*1, *M*'), where *M*' is any message. In the first case, Darth simply wants to eavesdrop on the communication without altering it. In the second case, Darth wants to modify the message going to Bob.

Other Public-key Algorithms

DIGITAL SIGNATURE STANDARD (DSS)

FIPS PUB 186

Makes use of SHA-1 and the Digital Signature Algorithm (DSA)

Originally proposed in 1991, revised in 1993 due to security concerns, and another minor revision in 1996

Cannot be used for encryption or key exchange

Uses an algorithm that is designed to provide only the digital signature function

ELLIPTIC-CURVE CRYPTOGRAPHY (ECC)

Equal security for smaller bit size than RSA

Seen in standards such as IEEE P1363

Confidence level in ECC is not yet as high as that in RSA

Based on a mathematical construct known as the elliptic curve