COMPUTER AND NETWORK SECURITY

Jonathan Katz

Modified by: Dr. Ramzi Saifan

"SECURITY"

- Most of computer science is concerned with *achieving desired behavior*
- Security is concerned with preventing undesired behavior
  - Different way of thinking!
  - An enemy/opponent/hacker/adversary who is actively and maliciously trying to circumvent any protective measures you put in place

ONE ILLUSTRATION OF THE DIFFERENCE

- Software testing determines whether a given program implements a desired functionality
  - Test I/O characteristics
  - Q/A
- How do you test whether a program does *not* allow for *undesired* functionality?
  - Penetration testing helps, but only up to a point

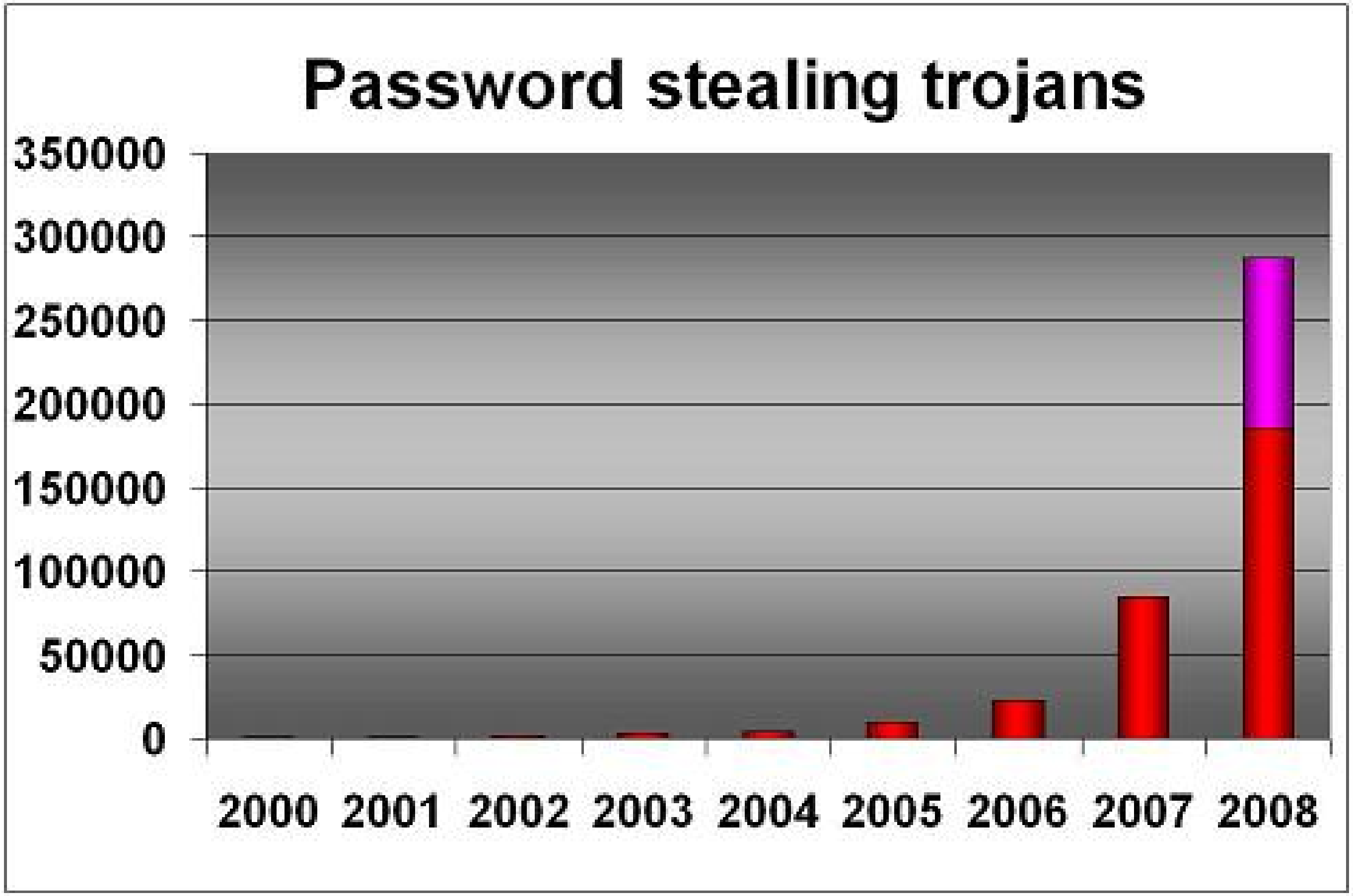
SECURITY IS INTERDISCIPLINARY

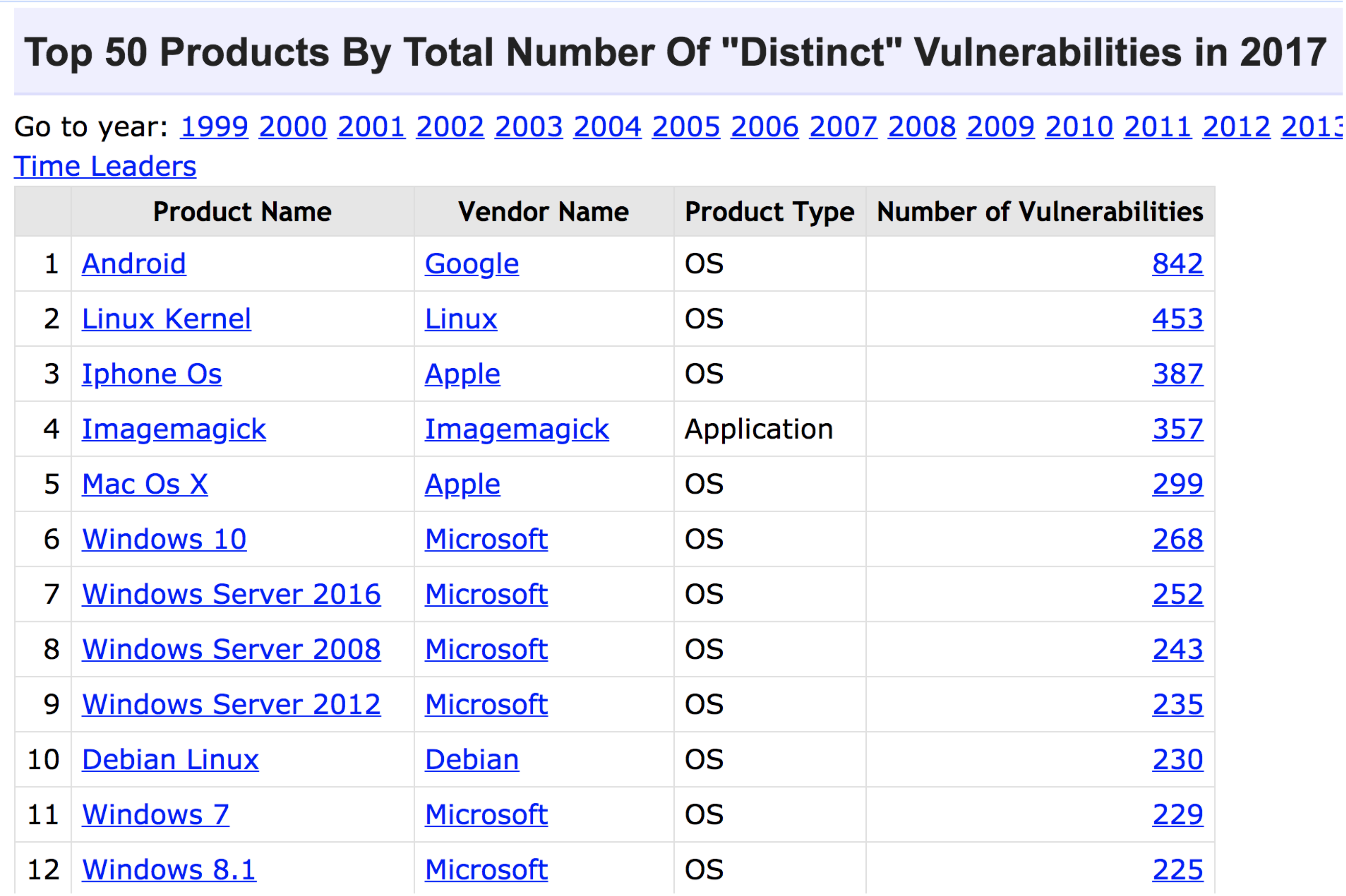
- Draws on <u>all</u> areas of CS
  - Theory (especially *cryptography*)
  - Networking
  - Operating systems
  - Databases
  - AI/learning theory
  - Computer architecture/hardware
  - Programming languages/compilers
  - HCI, psychology

FORTUNATELY, WE ARE WINNING THE SECURITY BATTLE

- Strong cryptography
- Firewalls, intrusion detection, virus scanners
- Buffer overflow detection/prevention
- User education

REALLY??!


Password stealing trojans
350000
300000
250000
200000
150000
100000
50000
0
2000
2001
2002
2003
2004
2005
2006
2007
2008



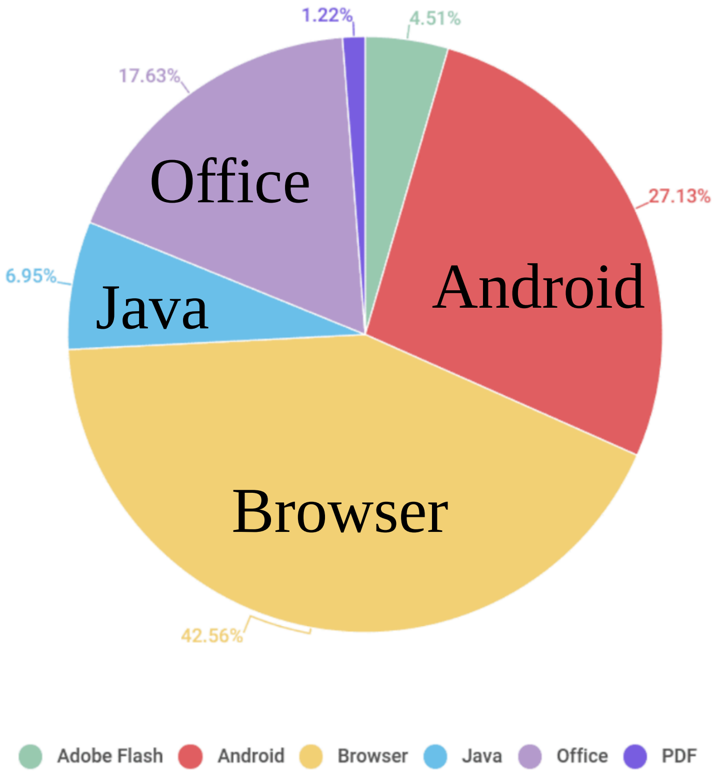
Top 50 Products By Total Number Of "Distinct" Vulnerabilities in 2017

Go to year: 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013
Time Leaders

	Product Name	Vendor Name	Product Type	Number of Vulnerabilities
1	Android	Google	OS	842
2	Linux Kernel	Linux	OS	453
3	Iphone Os	Apple	OS	387
4	Imagemagick	Imagemagick	Application	357
5	Mac Os X	Apple	OS	299
6	Windows 10	Microsoft	OS	268
7	Windows Server 2016	Microsoft	OS	252
8	Windows Server 2008	Microsoft	OS	243
9	Windows Server 2012	Microsoft	OS	235
10	Debian Linux	Debian	OS	230
11	Windows 7	Microsoft	OS	229
12	Windows 8.1	Microsoft	OS	225

source: https://www.cvedetails.com/top-50-products.php?year=2017

VULNERABLE APPLICATIONS BEING EXPLOITED



Source: Kaspersky Security Bulletin 2017

PHILOSOPHY OF THIS COURSE

- We are not going to be able to cover everything
  - We are not going to be able to even mention everything
- Main goa
  - A samp
  - The sec
  - Become familiar with basic acronyms (RSA, SSL, PGP, etc.), and "buzzwords" (phishing, …)
  - Becom
  - Try to k ,
    project

You will *not* be a security expert after this class (after this class, you should realize why it would be dangerous to think you are)

You *should* have a better appreciation of security issues after this class

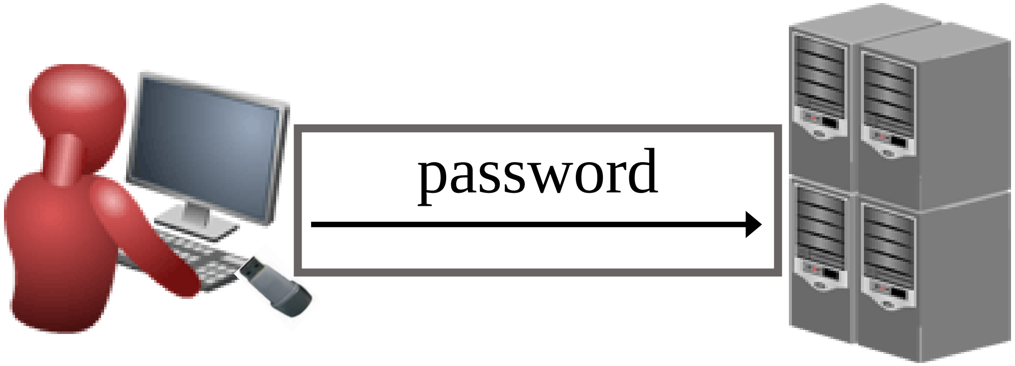
Course Organization

A NAÏVE VIEW

- Computer security is about CIA:
    - Confidentiality, integrity, and availability

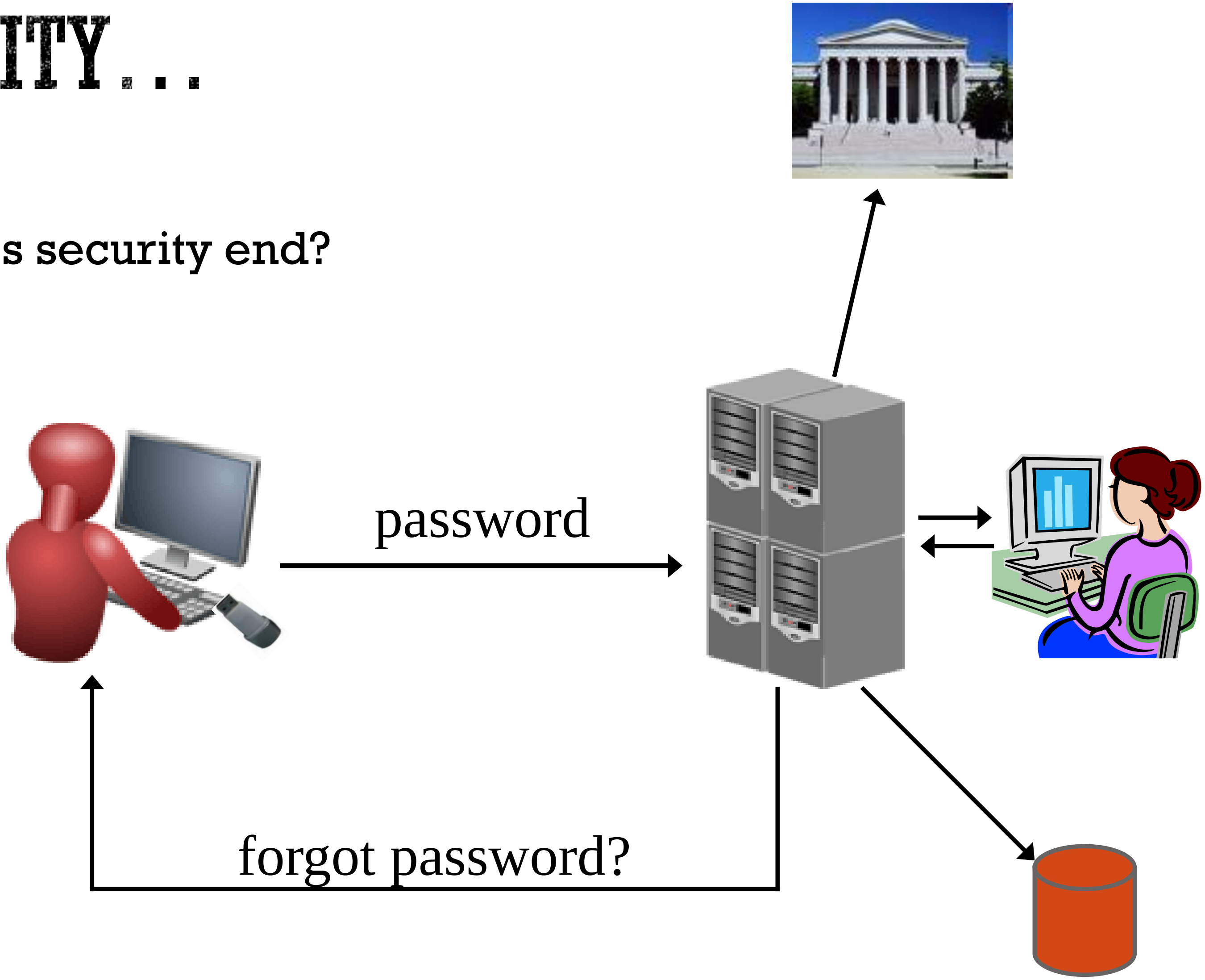
- These are important, but security is about much more…

A NAÏVE VIEW



IN REALITY…

- Where does security end?



password

forgot password?

ONE GOOD ATTACK

- Use public records to figure out someone's password
  - Or, e.g., their SSN, so can answer security question…
- The problem is *not* (necessarily) that SSNs are public
- The problem *is* that we "overload" SSNs, and use them for more than they were intended

A NAÏVE VIEW

- Achieve “absolute” security

IN REALITY…

- Absolute security is easy to achieve!
  - How…?

- Absolute security is impossible to achieve!
  - Why…?

- Good security is about *risk management*

SECURITY AS A TRADE-OFF

- The goal is not (usually) "to make the system as secure as possible"…
- …but instead, "to make the system as secure as possible *within certain constraints*" (cost, usability, convenience)
- Must understand the existing constraints
  - E.g., passwords…

COST-BENEFIT ANALYSIS

- Important to evaluate what level of security is necessary/appropriate
  - Cost of mounting a particular attack vs. value of attack to an adversary
  - Cost of damages from an attack vs. cost of defending against the attack
  - Likelihood of a particular attack

- Sometimes the best security is to make sure you are not the easiest target for an attacker…

"MORE" SECURITY NOT ALWAYS BETTER

- "No point in putting a higher post in the ground when the enemy can go around it"
- Need to identify the *weakest link*
  - Security of a system is only as good as the security at its weakest point…
- Security is not a "magic bullet"
- Security is a process, not a product

COMPUTER SECURITY IS NOT JUST ABOUT SECURITY

- Detection, response, audit
  - How do you know when you are being attacked?
  - How quickly can you stop the attack?
  - Can you identify the attacker(s)?
  - Can you prevent the attack from recurring?
- Recovery
  - Can be much more important than prevention
- Economics, insurance, risk management…
- Offensive techniques

COMPUTER SECURITY IS NOT JUST ABOUT COMPUTERS

- What is “the system”?
- Physical security
- Social engineering
  - Bribes for passwords
  - Phishing
- “External” means of getting information
  - Legal records
  - Trash cans

SECURITY MINDSET

- Learn to think with a "security mindset" in general
  - What is "the system"?
  - How could this system be attacked?
    - What is the weakest point of attack?
  - How could this system be defended?
    - What threats am I trying to address?
    - How effective will a given countermeasure be?
    - What is the trade-off between security, cost, and usability?

SUMMARY

- "The system" is not just a computer or a network
- Prevention is not the only goal
  - Cost-benefit analysis
  - Detection, response, recovery

- Nevertheless…in this course, we will focus on *computer* security, and primarily on *prevention*
  - If you want to be a security expert, you need to keep the rest in mind

COMPUTERS ARE EVERYWHERE…

- …and can always be attacked
- Electronic banking, social networks, e-voting
- iPods, iPhones, PDAs, RFID transponders
- Automobiles
- Appliances, TVs
- (Implantable) medical devices
- Cameras, picture frames(!)
  - See http://www.securityfocus.com/news/11499

"TRUSTING TRUST"

- Consider a compiler that embeds a trapdoor into anything it compiles
- How to catch?
  - Read source code? (What if replaced?)
  - Re-compile compiler?
- What if the compiler embeds the trojan code whenever it compiles a compiler?
  - (That's nasty…)

“TRUSTING TRUST”

- Whom do you trust?
- Does one really need to be this paranoid??
  - Probably not
  - Sometimes, yes
- Shows that security is complex…and essentially impossible
- Comes back to risk/benefit trade-off

Next time:

begin cryptography

COMPUTER AND NETWORK SECURITY

LECTURE 2

Jonathan Katz

Modified By: Dr. Ramzi Saifan

A high-level survey of cryptography

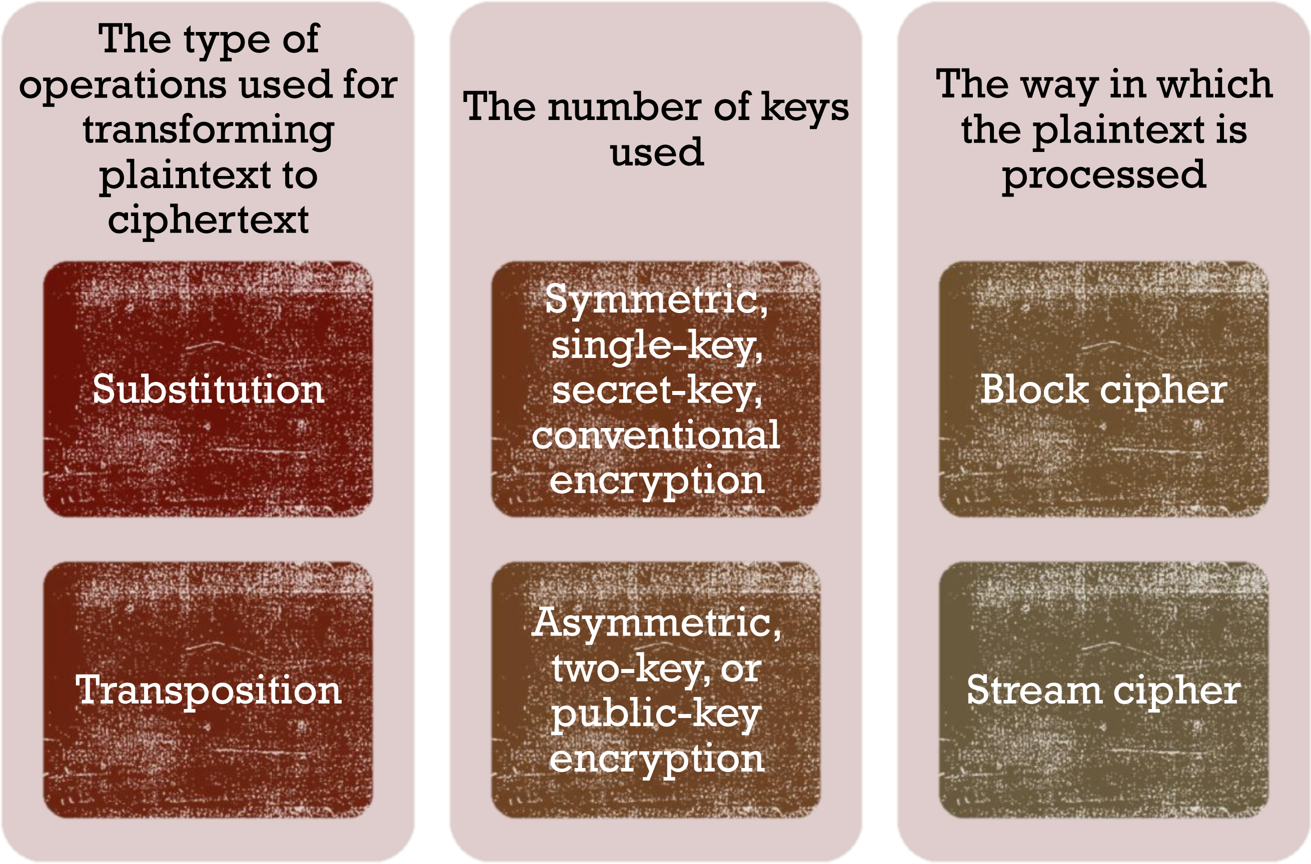
GOALS OF CRYPTOGRAPHY

- Crypto deals primarily with three goals:
  - Confidentiality
  - Integrity (of data)
  - Authentication (of resources, people, systems)

- Other goals also considered
  - E.g., non-repudiation
  - Accountability
  - Anonymity
  - …

CRYPTOGRAPHIC SYSTEMS

- Characterized along three independent dimensions:



The type of operations used for transforming plaintext to ciphertext

Substitution

Transposition

The number of keys used

Symmetric, single-key, secret-key, conventional encryption

Asymmetric, two-key, or public-key encryption

The way in which the plaintext is processed

Block cipher

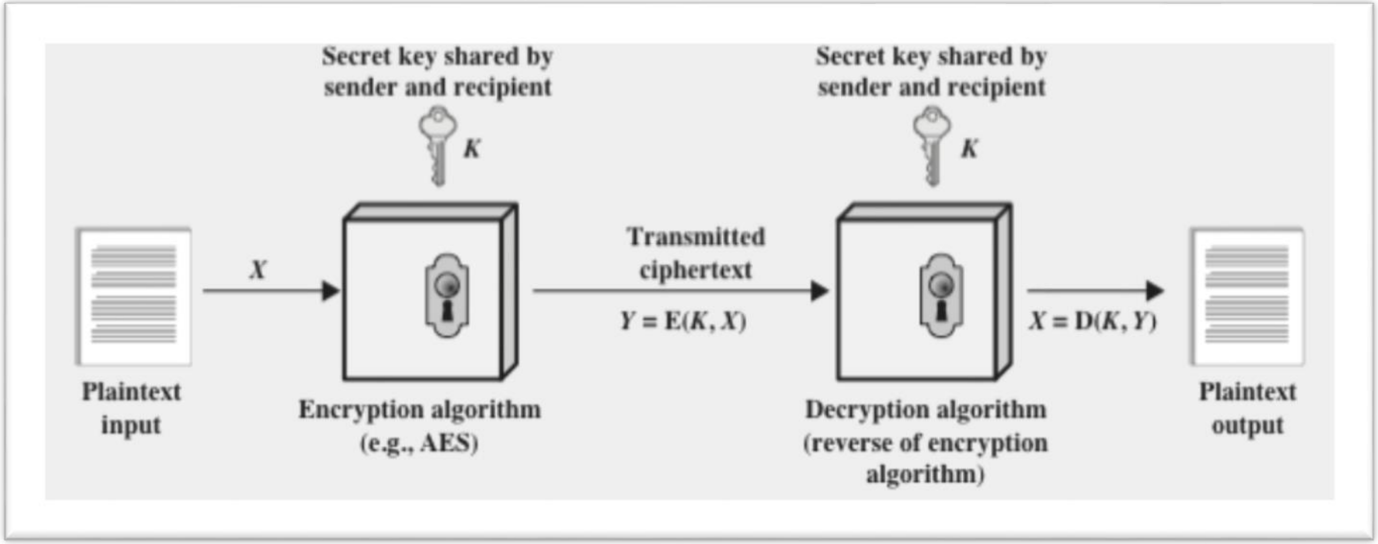
Stream cipher

PRIVATE- VS. PUBLIC-KEY SETTINGS

- For the basic goals, there are two settings:
  - Private-key / shared-key / symmetric-key / secret-key
  - Public-key
- The private-key setting is the "classical" one (thousands of years old)
- The public-key setting dates to the 1970s

PRIVATE-KEY CRYPTOGRAPHY

- The communicating parties share some information that is **random** and **secret**
  - This shared information is called a **key**
  - Key is not known to an attacker
  - This key must be shared (somehow) in advance of their communication


Secret key shared by sender and recipient
K
Secret key shared by sender and recipient
K
X
Transmitted ciphertext
$Y = E(K, X)$
$X = D(K, Y)$
Plaintext input
Encryption algorithm (e.g., AES)
Decryption algorithm (reverse of encryption algorithm)
Plaintext output

TO EMPHASIZE

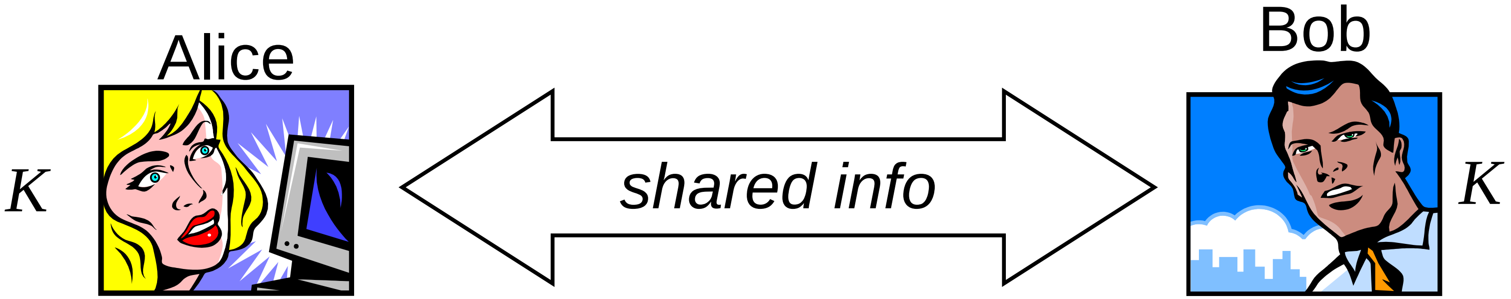
- Alice and Bob share a key K
  - Must be shared securely
  - Must be completely random
  - Must be kept completely secret from attacker

- We don't discuss (for now) how they do this
  - You can imagine they meet on a dark street corner and Alice hands a USB device (with a key on it) to Bob

CANONICAL APPLICATIONS

- Two (or more) distinct parties communicating over an insecure network
  - E.g., secure communication

- A single party who is communicating “with itself” over time
  - E.g., secure storage



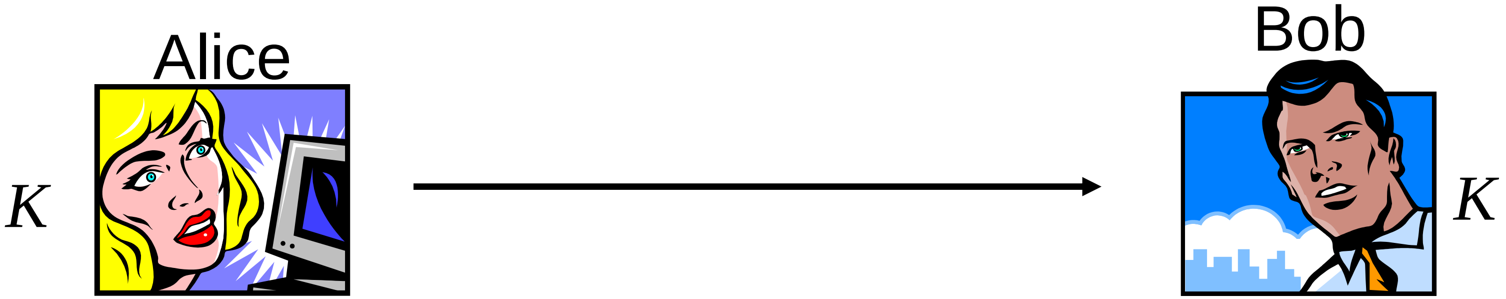
Alice

Bob

K

shared info

K

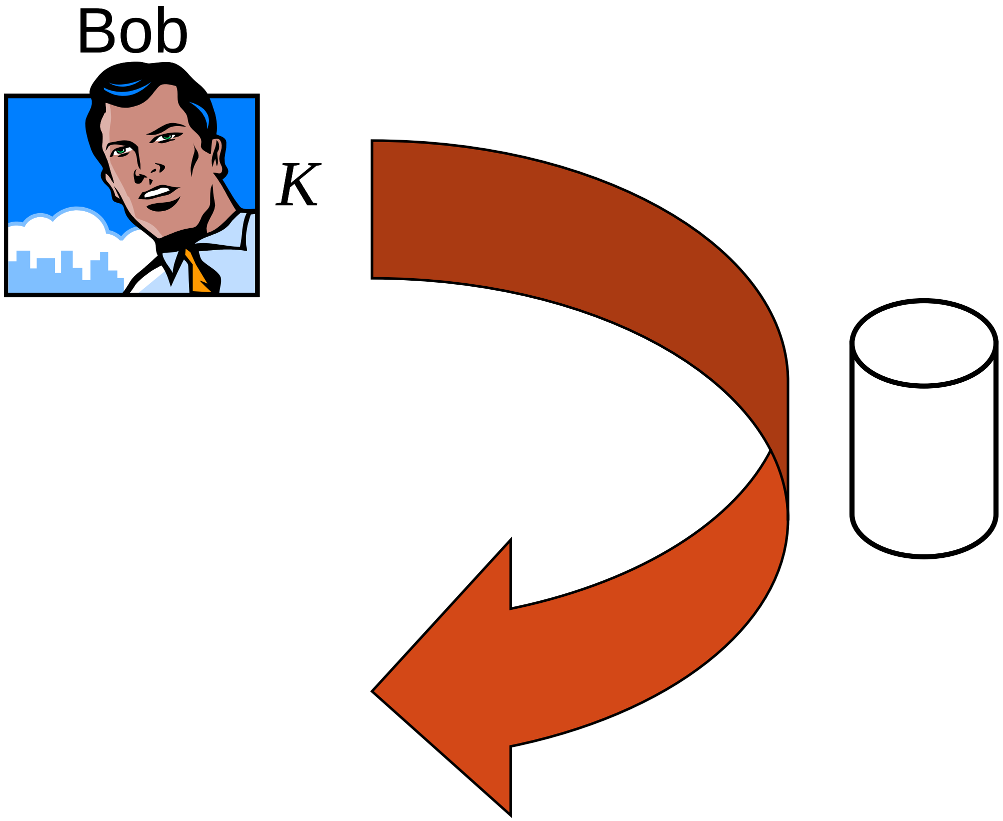


Alice

Bob

K

K



Bob

K

SECURITY THROUGH OBSCURITY?

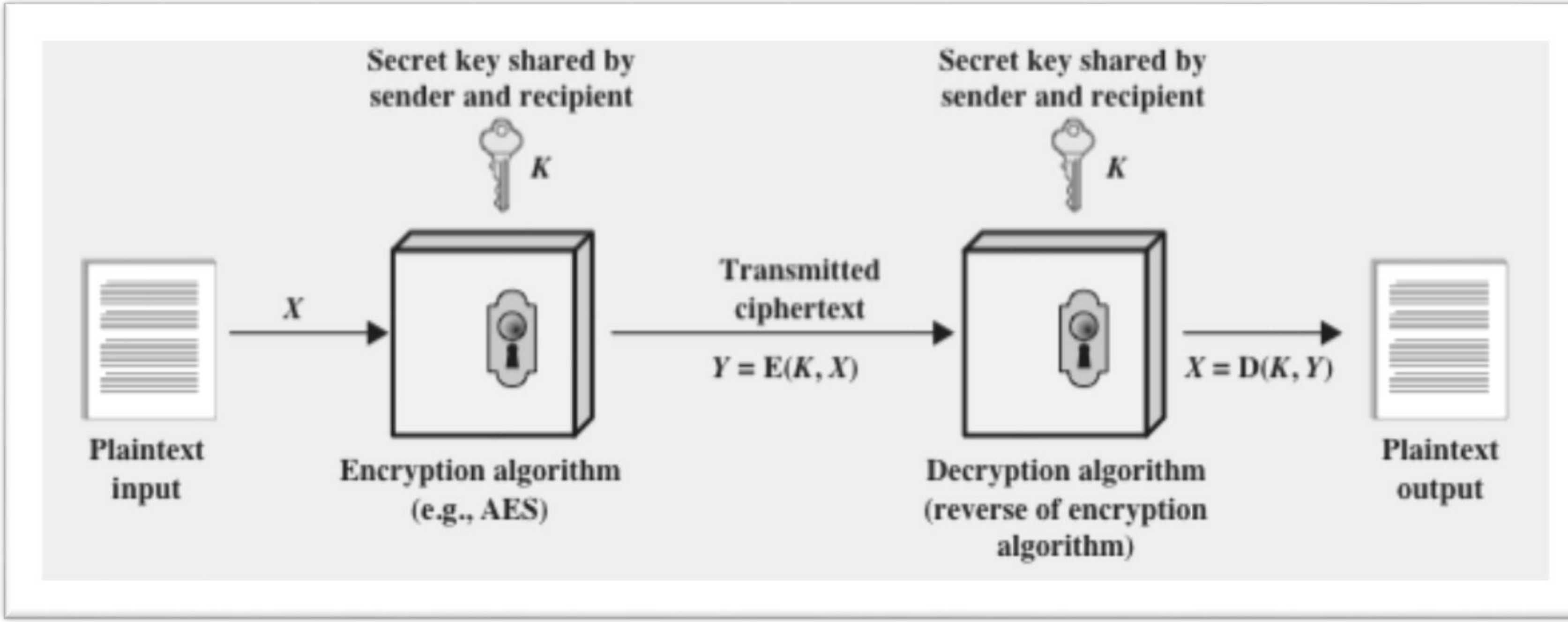
- Always assume that the full details of crypto protocols and algorithms are public
  - Known as **Kerckhoffs' principle**
  - The *only* secret information is a key
- "Security through obscurity" is a bad idea…
  - True in general; even more true in the case of cryptography
  - Home-brewed solutions are BAD!
  - Standardized, widely-accepted solutions are GOOD!

SECURITY THROUGH OBSCURITY?

- Why not?
- Easier to maintain secrecy of a *key* than an *algorithm*
  - Reverse engineering
  - Insider attacks
- Easier to change the key than the algorithm
- In general setting, much easier to share an algorithm than for everyone to use their own

Private-key encryption

Functional definition



Secret key shared by sender and recipient
K
Secret key shared by sender and recipient
K
X
Transmitted ciphertext
Y = E(K, X)
X = D(K, Y)
Plaintext input
Encryption algorithm (e.g., AES)
Decryption algorithm (reverse of encryption algorithm)
Plaintext output

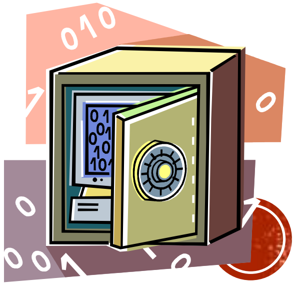
Encryption: c ← $E_K(m)$ possibly randomized!

Decryption algorithm: $m = D_K(c)$

Correctness: for all K, we have $D_K(E_K(m)) = m$

ENCRYPTION SCHEME SECURITY

- Unconditionally secure
  - No matter how much time an opponent has, it is impossible for him or her to decrypt the ciphertext simply because the required information is not there
- Computationally secure
  - The cost of breaking the cipher exceeds the value of the encrypted information
  - The time required to break the cipher exceeds the useful lifetime of the information



MODEL OF SYMMETRIC CRYPTOSYSTEM

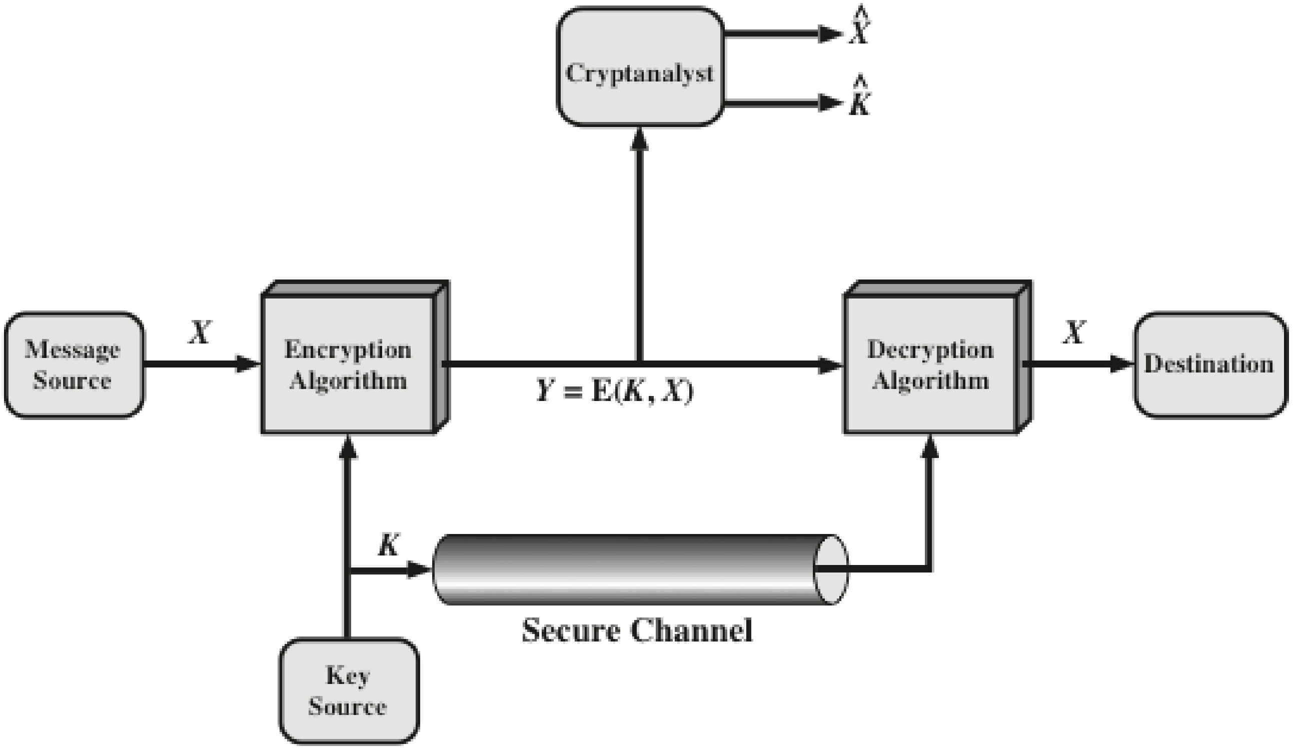


Figure 2.2 Model of Symmetric Cryptosystem

CRYPTANALYSIS AND BRUTE-FORCE ATTACK

Cryptanalysis

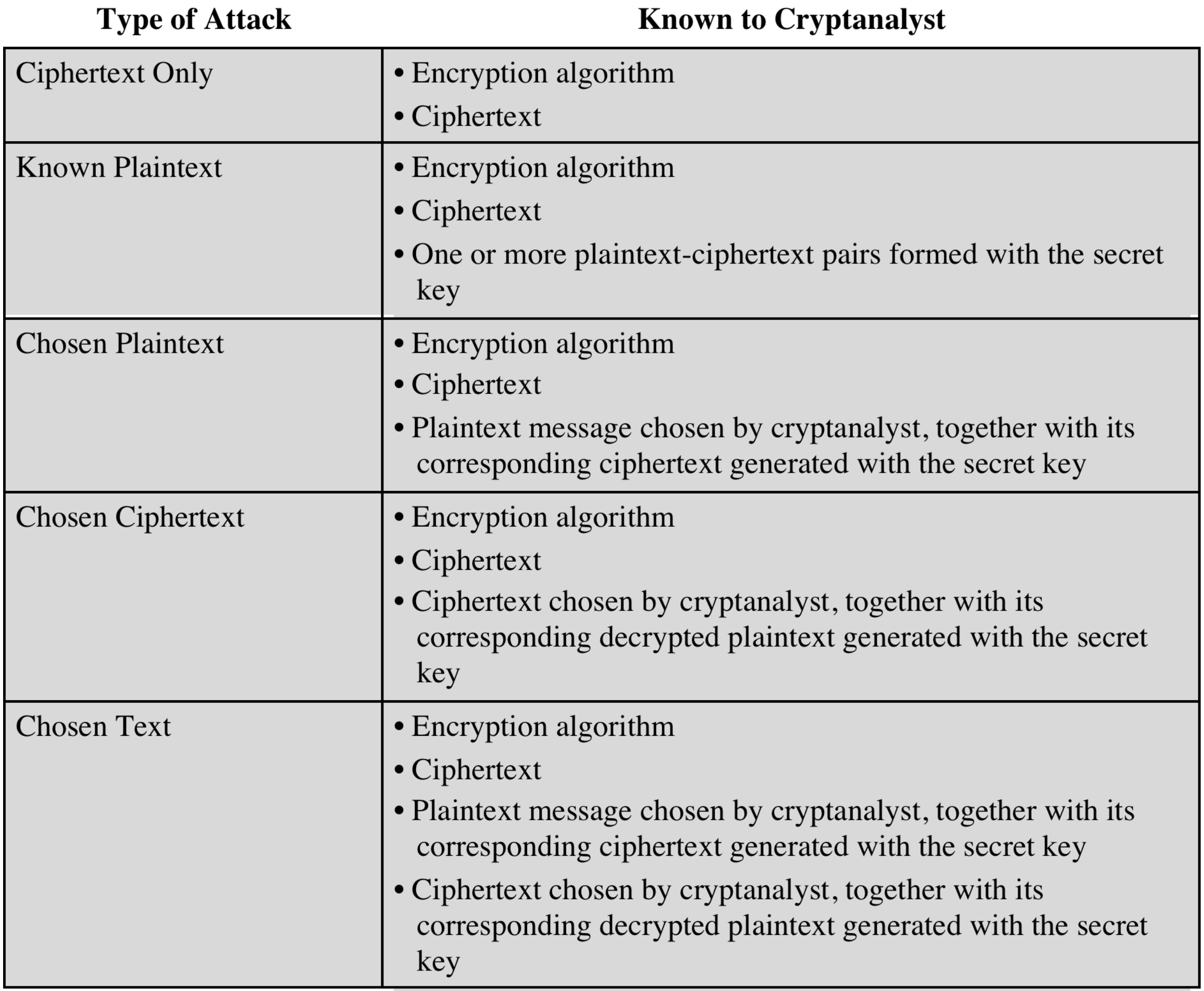
- Attack relies on the nature of the algorithm plus some knowledge of the general characteristics of the plaintext

- to attempt to deduce a specific plaintext or to deduce the key being used

Brute-force attack

- Attacker tries every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained
- On average, half of all possible keys must be tried to achieve success.
- To supplement the brute-force approach, some degree of knowledge about the expected plaintext is needed,

TABLE 2.1 TYPES OF ATTACKS ON ENCRYPTED MESSAGES



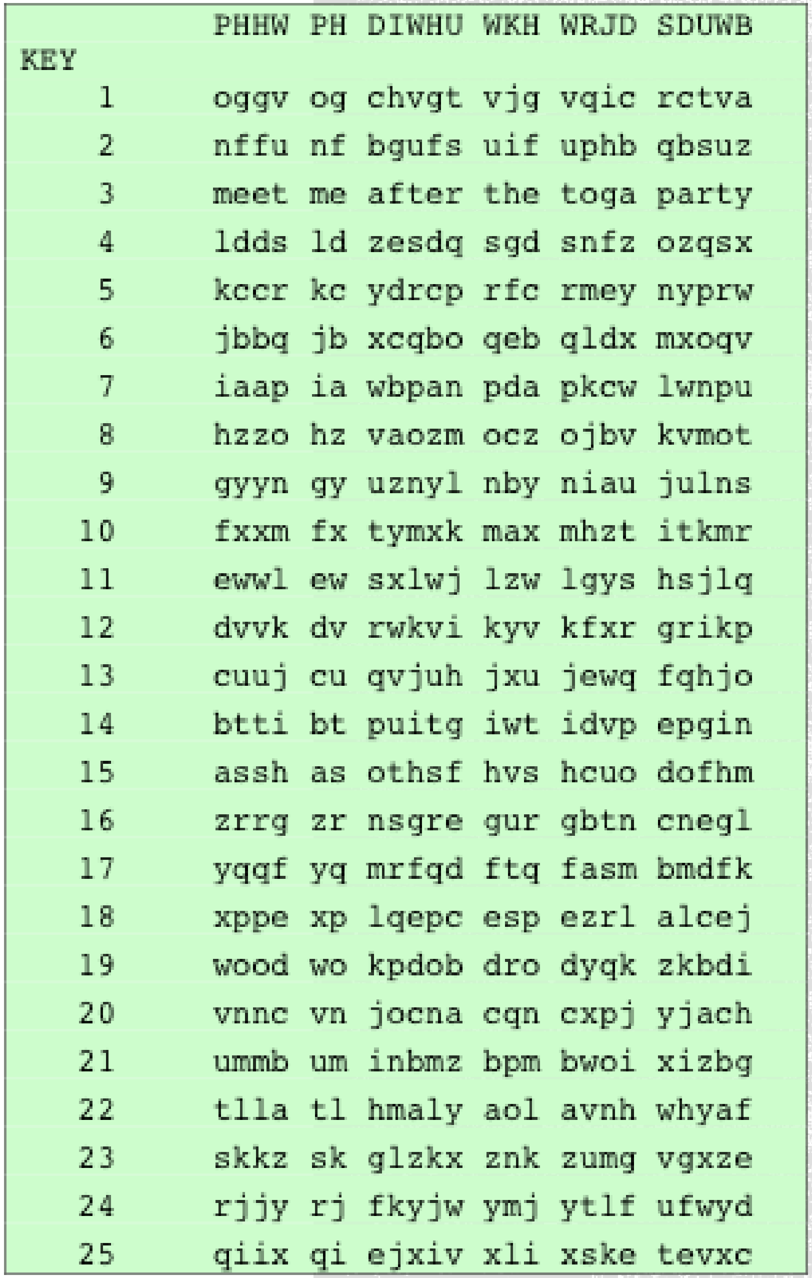
Type of Attack	Known to Cryptanalyst
Ciphertext Only	• Encryption algorithm • Ciphertext
Known Plaintext	• Encryption algorithm • Ciphertext • One or more plaintext-ciphertext pairs formed with the secret key
Chosen Plaintext	• Encryption algorithm • Ciphertext • Plaintext message chosen by cryptanalyst,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

A CLASSIC EXAMPLE: SHIFT CIPHER

- Assume the English uppercase alphabet (no lowercase, punctuation, etc.)
  - View letters as numbers in {0, …, 25}
- The key is a random letter of the alphabet
- Encryption done by addition modulo 26

- Is this secure?
  - Exhaustive key search
  - Automated determination of the key

BRUTE-FORCE CRYPTANALYSIS OF SHIFT CIPHER



	PHHW PH DIWHU WKH WRJD SDUWB
KEY	
1	oggv og chvgt vjg vqic rctva
2	nffu nf bgufs uif uphb qbsuz
3	meet me after the toga party
4	ldds ld zesdq sgd snfz ozqsx
5	kccr kc ydrcp rfc rmey nyprw
6	jbbq jb xcqbo qeb qldx mxoqv
7	iaap ia wbpan pda pkcw lwnpu
8	hzzo hz vaozm ocz ojbv kvmot
9	gyyn gy uznyl nby niau julns
10	fxxm fx tymxk max mhzt itkmr
11	ewwl ew sxlwj lzw lgys hsjlq
12	dvvk dv rwkvi kyv kfxr grikp
13	cuuj cu qvjuh jxu jewq fqhjo
14	btti bt puitg iwt idvp epgin
15	assh as othsf hvs hcuo dofhm
16	zrrg zr nsgre gur gbtn cnegl
17	yqqf yq mrfqd ftq fasm bmdfk
18	xppe xp lqepc esp ezrl alcej
19	wood wo kpdob dro dyqk zkbdi
20	vnnc vn jocna cqn cxpj yjach
21	ummb um inbmz bpm bwoi xizbg
22	tlla tl hmaly aol avnh whyaf
23	skkz sk glzkx znk zumg vgxze
24	rjjy rj fkyjw ymj ytlf ufwyd
25	qiix qi ejxiv xli xske tevxc

Figure 2.3 Brute-Force Cryptanalysis of Caesar Cipher

MONOALPHABETIC CIPHER

- The key is a random permutation of the alphabet
  - Note: key space is huge!
- Encryption done in the natural way

- Is this secure?
  - Frequency analysis
- A large key space is necessary, but not sufficient, for security

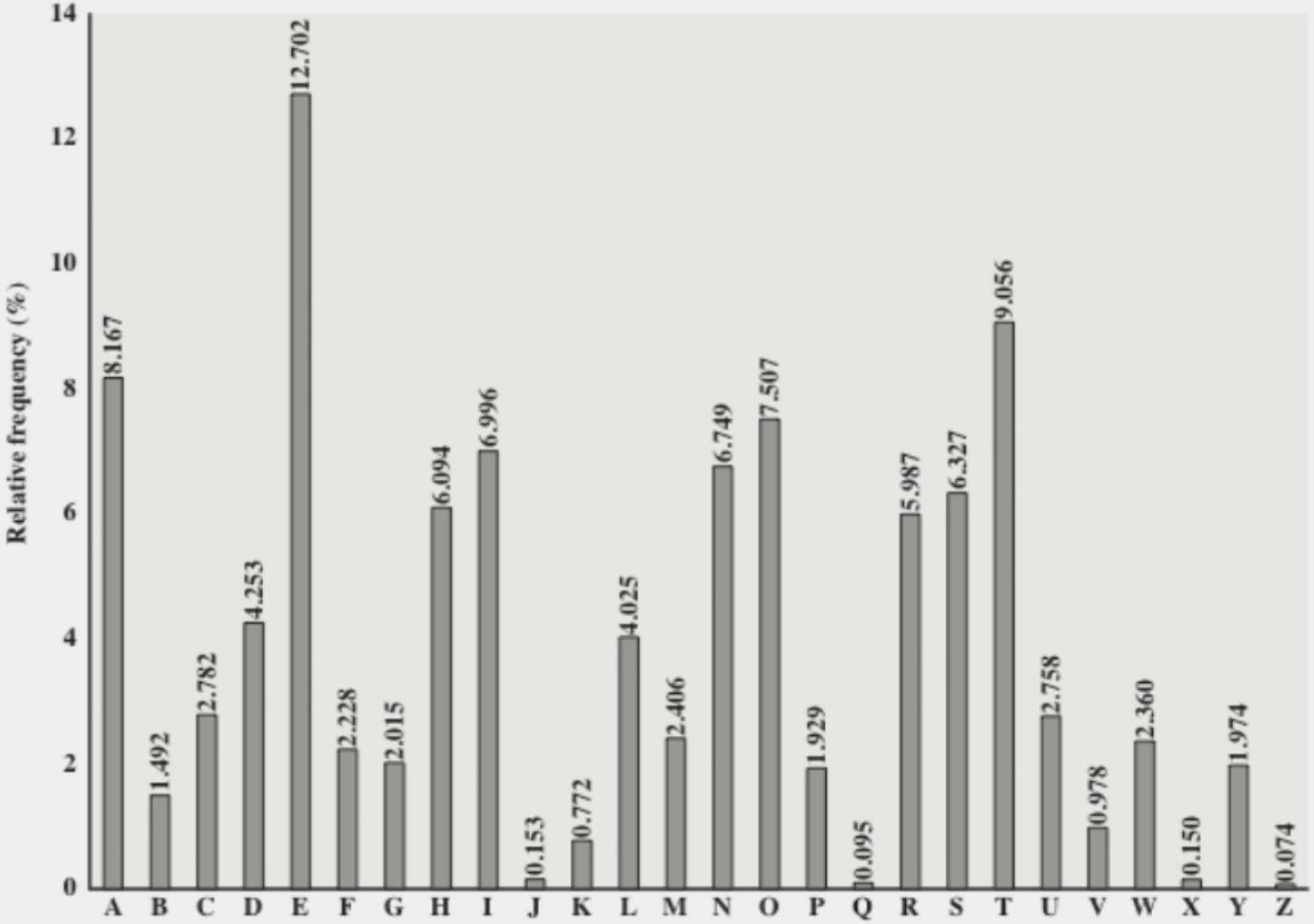
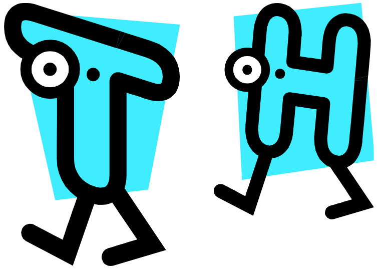
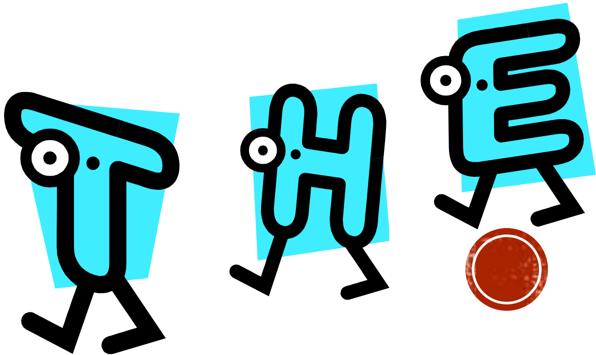


Figure 2.5 Relative Frequency of Letters in English Text

MONOALPHABETIC CIPHERS

- Easy to break because they reflect the frequency data of the original alphabet
- Countermeasure is to provide multiple substitutes (homophones) for a single letter
- Digram
  - Two-letter combination
  - Most common is *th*
- Trigram
  - Three-letter combination
  - Most frequent is *the*





ANOTHER EXAMPLE: VIGENERE CIPHER

- More complicated version of shift cipher
- Believed to be secure for over 100 years
- Is it secure?

POLYALPHABETIC CIPHERS

- Polyalphabetic substitution cipher
  - Improves on the simple monoalphabetic technique by using different monoalphabetic substitutions as one proceeds through the plaintext message

All these techniques have the following features in common:

- A set of related monoalphabetic substitution rules is used
- A key determines which particular rule is chosen for a given transformation

VIGENÈRE CIPHER

- Best known and one of the simplest polyalphabetic substitution ciphers
- In this scheme the set of related monoalphabetic substitution rules consists of the 26 Caesar ciphers with shifts of 0 through 25
- Each cipher is denoted by a key letter which is the ciphertext letter that substitutes for the plaintext letter a

EXAMPLE OF VIGENÈRE CIPHER

- To encrypt a message, a key is needed that is as long as the message
- Usually, the key is a repeating keyword
- For example, if the keyword is *deceptive*, the message “we are discovered save yourself” is encrypted as:

key:

deceptivedeceptivedeceptive

plaintext: wearediscoveredsaveyourself

ciphertext: ZICVTWQNGRZGVTWAVZHCQYGLMGJ

VIGENÈRE AUTOKEY SYSTEM

- A keyword is concatenated with the plaintext itself to provide a running key
- Example:

  key: deceptivewearediscoveredsav

  plaintext: wearediscoveredsaveyourself

  ciphertext: ZICVTWQNGKZEIIGASXSTSLVVWLA
- Even this scheme is vulnerable to cryptanalysis
  - Because the key and the plaintext share the same frequency distribution of letters, a statistical technique can be applied

ATTACKING THE VIGENERE CIPHER

- Let p_i (for i=0, …, 25) denote the frequency of letter i in English-language text
  - Known that $\Sigma\ p_i^2 \approx 0.065$
- For each candidate period t, compute frequencies $\{q_i\}$ of letters in the sequence $c_0, c_t, c_{2t}, \ldots$
- For the correct value of t, we expect $\Sigma\ q_i^2 \approx 0.065$
  - For incorrect values of t, we expect $\Sigma\ q_i^2 \approx 1/26$
- Once we have the period, can use frequency analysis as in the case of the shift cipher

MORAL OF THE STORY?

- Don’t use “simple” schemes
- Don’t use schemes that you design yourself
    - Use schemes that other people have already designed and analyzed…

A FUNDAMENTAL PROBLEM

- A fundamental problem with "classical" cryptography is that *no definition of security was ever specified*
  - It was not even clear what it meant for a scheme to be "secure"
- As a consequence, *proving security was not even an option*
  - So how can you *know* when something is secure?
  - (Or is at least based on well-studied, widely-believed assumptions)

SECURITY GOALS?

- Adversary unable to recover the key
  - Necessary, but meaningless on its own…
- Adversary unable to recover entire plaintext
  - Good, but is it enough?
- Adversary unable to determine any information at all about the plaintext
  - Formalize?
  - Sounds great!
  - Can we achieve it?

CMSC 414
Computer and Network Security

Lecture 3

Jonathan Katz

Modified by: Dr. Ramzi Saifan

Perfect secrecy

Defining secrecy (take 1)

- Even an adversary running for an *unbounded* amount of time learns *nothing* about the message from the ciphertext
  - (Except the length)
- *Perfect secrecy*
- Formally, for all distributions over the message space, all m, and all c:

$$\Pr[M=m \mid C=c] = \Pr[M=m]$$

Properties of the one-time pad?

- Achieves perfect secrecy
  - No eavesdropper (no matter how powerful) can determine any information whatsoever about the plaintext
- was developed by Gilbert Vernam in 1918.
- Stream cipher: The message is represented as a binary string.
- The key is a truly random sequence of 0’s and 1’s of the same length as the message.
- The encryption is done by *XOR the key and the message.*

Why OTP is perfect secure?

- The security depends on the randomness of the key.
- In cryptographic context, we seek two fundamental properties in a binary random key sequence:
  - *Unpredictability:* the probability of a certain bit being 1 or 0 is exactly equal to ½ even if you have all previous bits.
  - *Balanced (Equal Distribution):*
    - The number of 1’s and 0’s should be equal.

Mathematical Proof

- the probability of a key bit being 1 or 0 is exactly equal to ½.
- The plaintext bits are not balanced. Let the probability of 0 be *x* and then the probability of 1 turns out to be 1-*x*.
- Let us calculate the probability of ciphertext bits.

Mathematical Proof

m_i	prob.	k_i	prob.	c_i	prob.
0	x	0	½	0	½ x
0	x	1	½	1	½ x
1	1-x	0	½	1	½ (1-x)
1	1-x	1	½	0	½ (1-x)

- We find out the probability of a ciphertext bit being 1 or 0 is equal to (½)x + (½)(1-x) = ½. Ciphertext looks like a random sequence.

Disadvantages

- (Essentially) useless in practice…
  - Long key length
  - Can only be used once (hence the name!)
  - Insecure against known-plaintext attacks
  - Key distribution & Management difficult.

- These are *inherent* limitations of perfect secrecy

A computationally secure scheme

- A *pseudorandom (number) generator* (PRNG) is a deterministic function that takes as input a *seed* and outputs a string
- If seed chosen at random, output of the PRNG should "look random" (i.e., be *pseudorandom*)

Notes

- Pseudo-randomness must be indistinguishable from random for all efficient algorithms
  - General-purpose PRNGs *not sufficient* for crypto
- Pseudorandomness of the PRNG depends on the seed being chosen “at random”
  - Note in particular that if a seed is re-used then the output of the PRNG remains the same!
  - In practice: from physical processes and/or user behavior

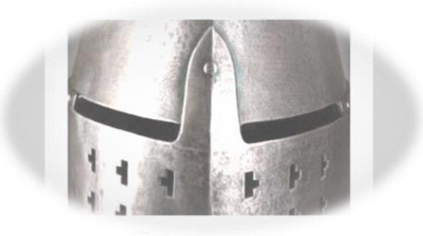
Computational secrecy

Computational secrecy

- We can overcome the limitations of perfect secrecy by (slightly) relaxing the definition
- Instead of requiring *total* secrecy against *unbounded* adversaries, require secrecy against *time-bounded* adversaries except with some *small probability*
  - E.g., secrecy for 100 years, except with probability 2^{-80}

The take-home message

- Weakening the definition slightly allows us to construct much more efficient schemes!
- Strictly speaking, no longer 100% absolutely guaranteed to be secure
  - Security of encryption now depends on security of building blocks (which are analyzed extensively, and are believed to be secure)
  - Given enough time and/or resources, the scheme can be broken



Block Ciphers and the Data Encryption Standard (DES)

Modified by: Dr. Ramzi Saifan

Block ciphers

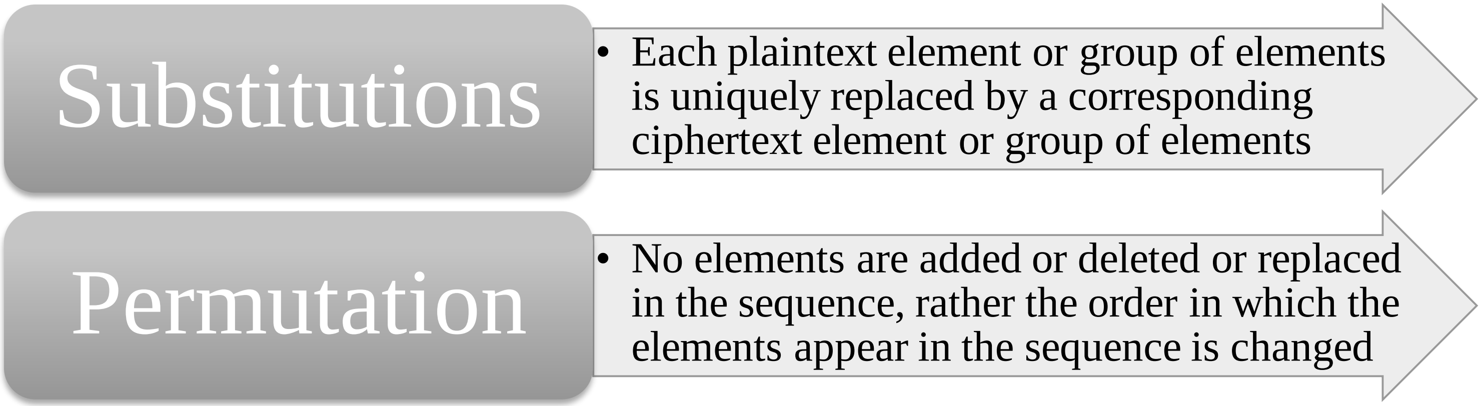
- Keyed, invertible
- Large key space, large block size
- A block of plaintext is treated as a whole and used to produce a ciphertext block of equal length
- Typically a block size of 64 or 128 bits is used
- The majority of network-based symmetric cryptographic applications make use of block ciphers

Data Encryption Standard (DES)

- Developed in 1970s by IBM / NSA / NBS
  - Non-public design process
- Block size = 64-bit input/output
- Key size = 56 bits out of a 64 bits
  - One bit in each octet is a parity-check bit
- Was the most widely used encryption scheme until the introduction of the Advanced Encryption Standard (AES) in 2001

Feistel Cipher

- Proposed the use of a cipher that alternates substitutions and permutations



Substitutions

- Each plaintext element or group of elements is uniquely replaced by a corresponding ciphertext element or group of elements

Permutation

- No elements are added or deleted or replaced in the sequence, rather the order in which the elements appear in the sequence is changed

– Is the structure used by many significant symmetric block ciphers currently in use.

Feistel Cipher Structure

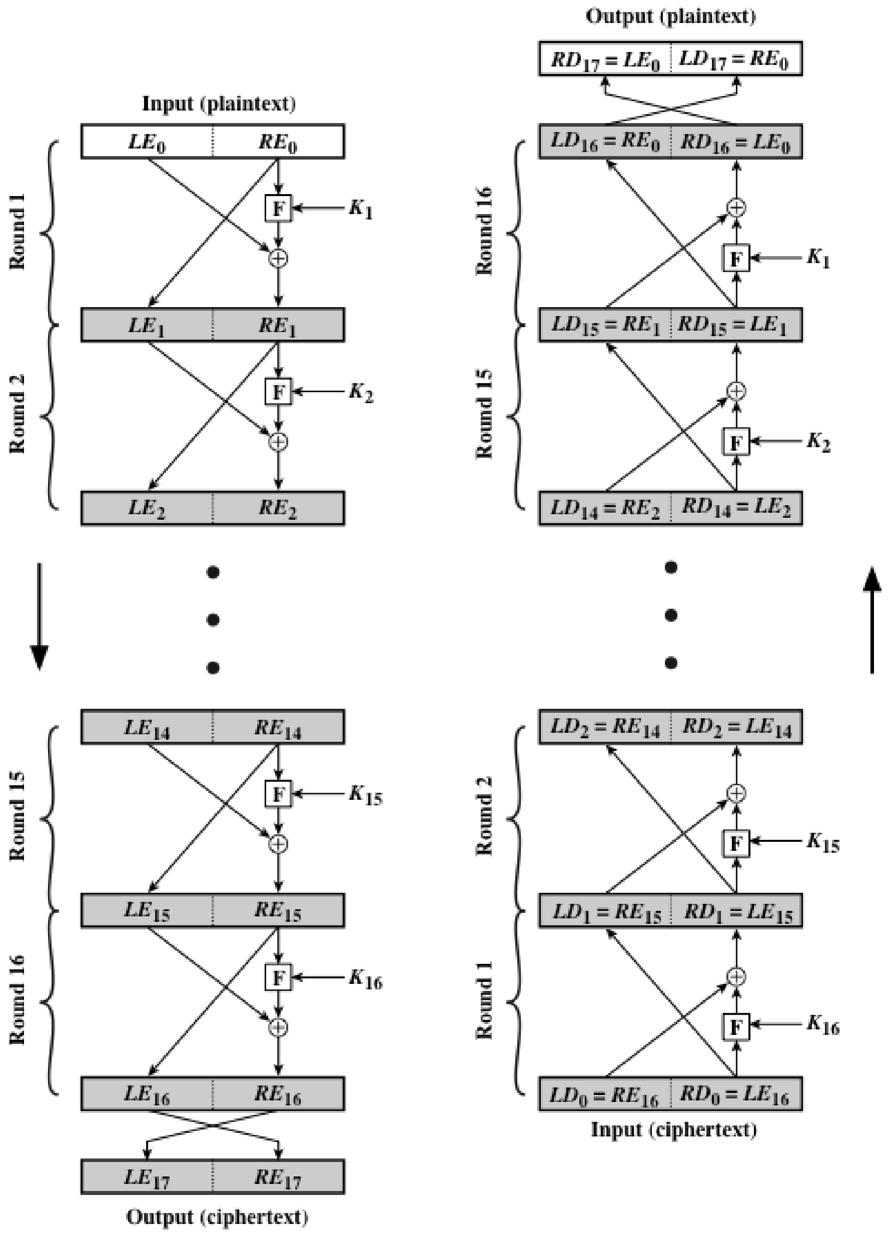


Figure 3.3 Feistel Encryption and Decryption (16 rounds)

Feistel Cipher Design Features

- Block size
  - Larger block sizes mean greater security but reduced encryption/decryption speed for a given algorithm
- Key size
  - Larger key size means greater security but may decrease encryption/decryption speeds
- Number of rounds
  - The essence of the Feistel cipher is that a single round offers inadequate security but that multiple rounds offer increasing security
- Subkey generation algorithm
  - Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis
- Round function F
  - Greater complexity generally means greater resistance to cryptanalysis
- Fast software encryption/decryption
  - In many cases, encrypting is embedded in applications or utility functions in such a way as to preclude a hardware implementation; accordingly, the speed of execution of the algorithm becomes a concern
- Ease of analysis
  - If the algorithm can be concisely and clearly explained, it is easier to analyze that algorithm for cryptanalytic vulnerabilities and therefore develop a higher level of assurance as to its strength

Feistel Example

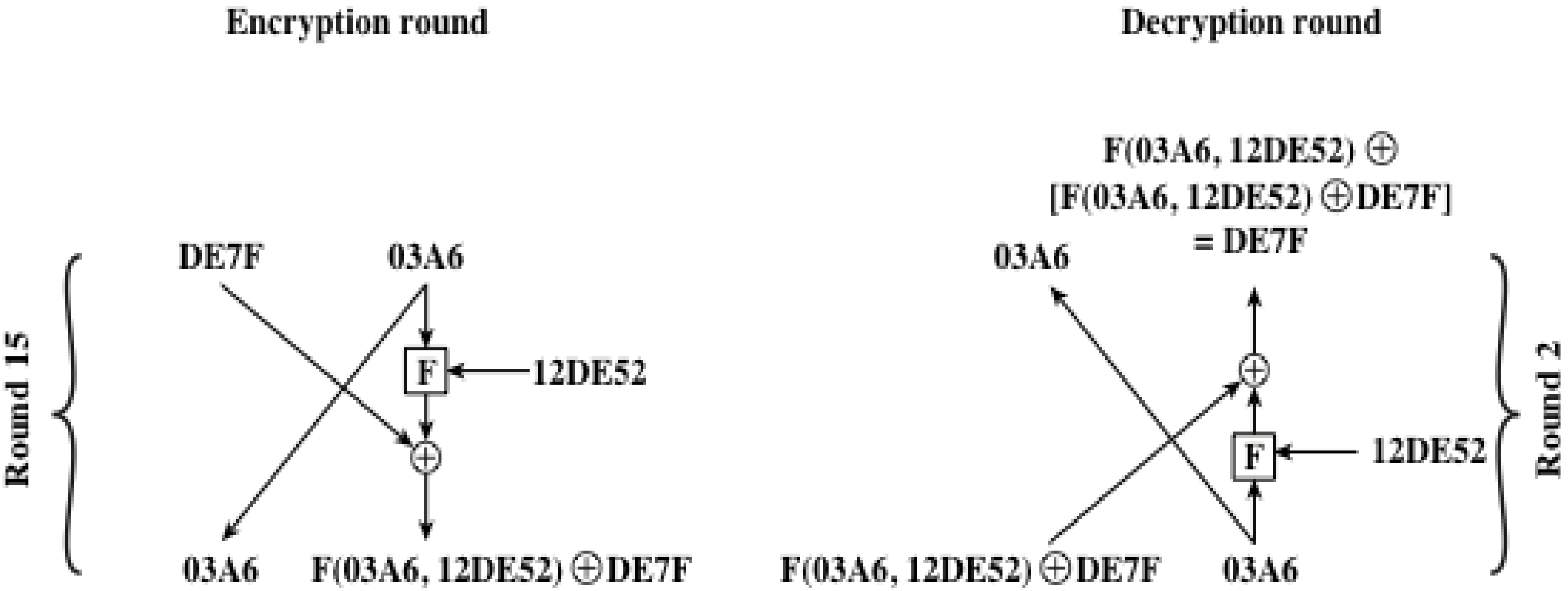


Figure 3.4 Feistel Example

DES Encryption Algorithm

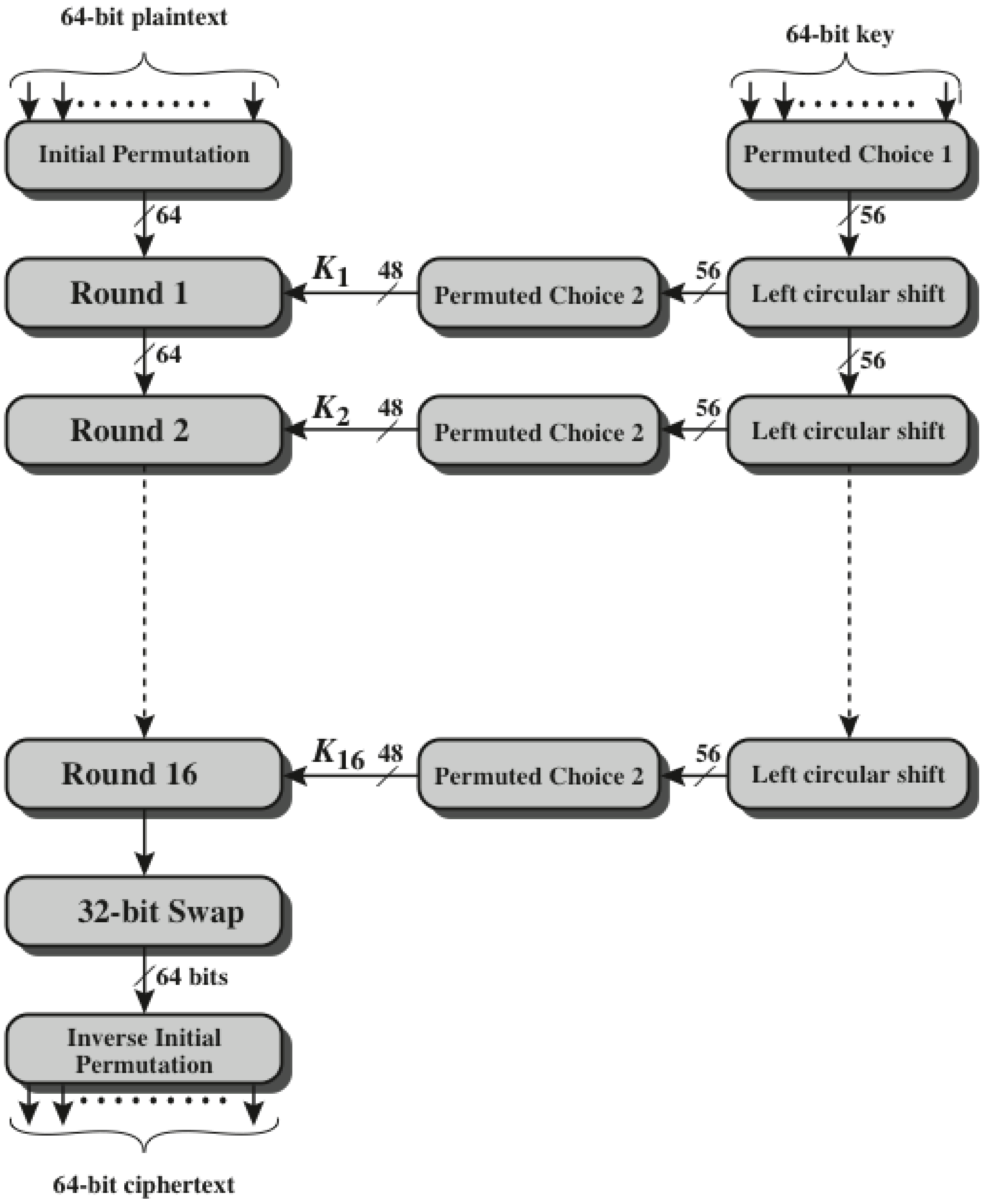
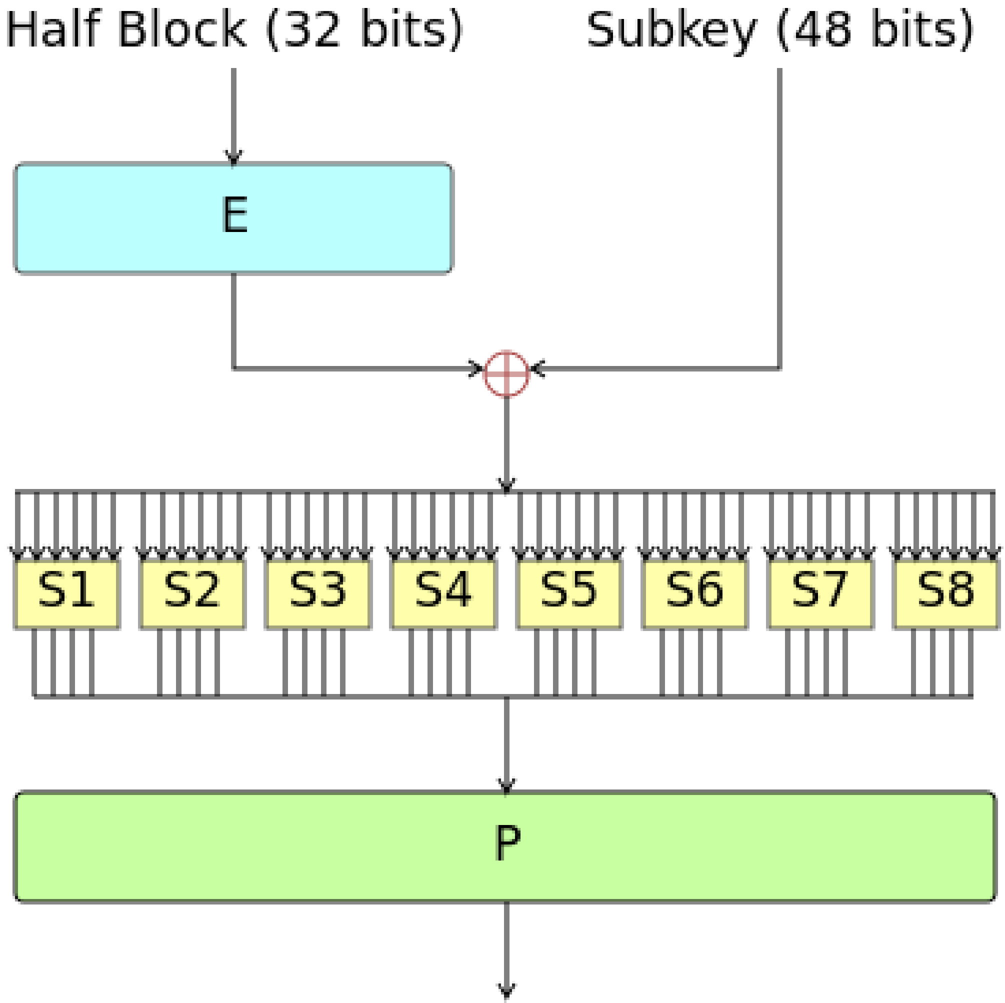
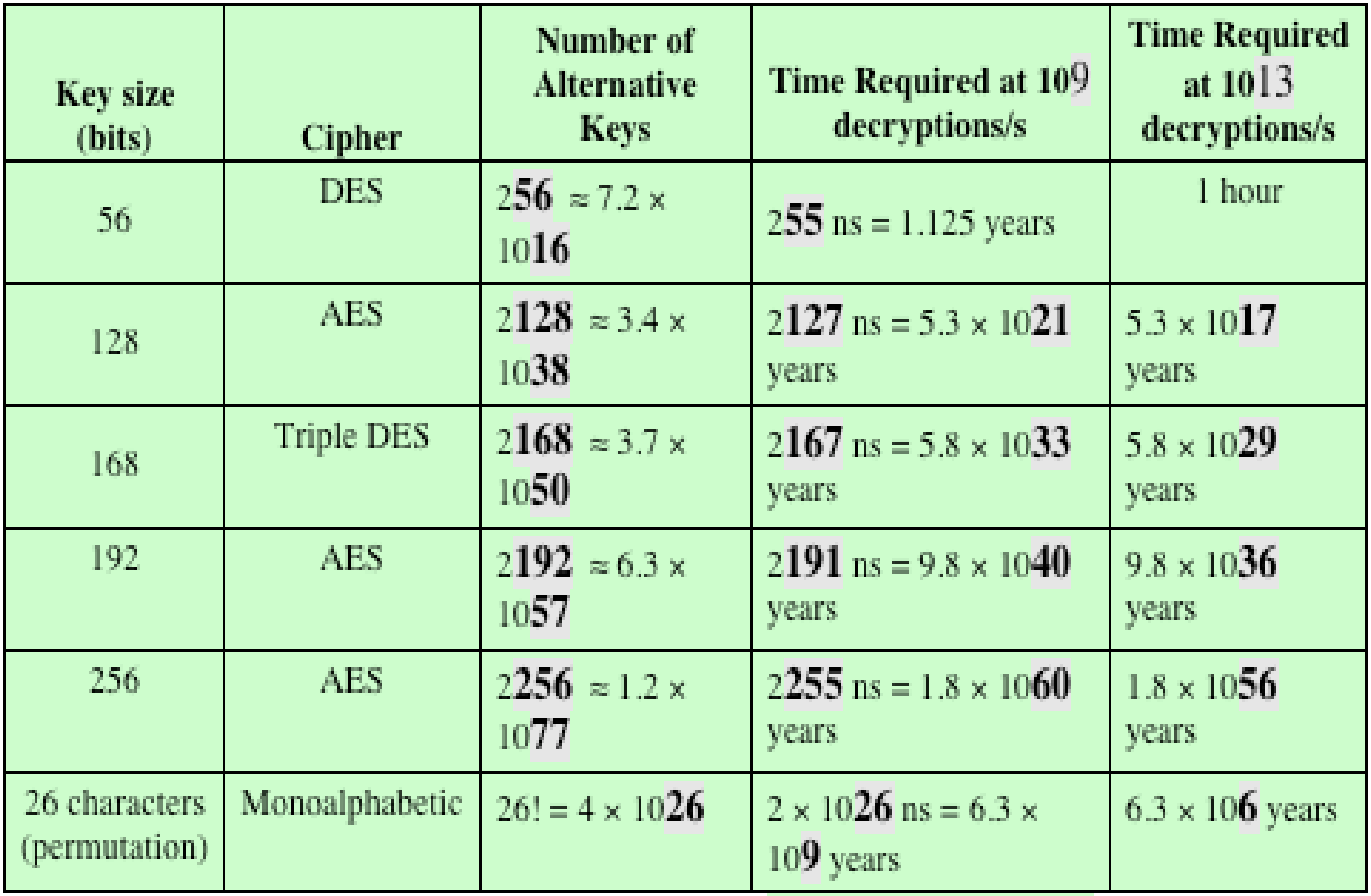


Figure 3.5 General Depiction of DES Encryption Algorithm

Round Function



Average Time Required for Exhaustive Key Search

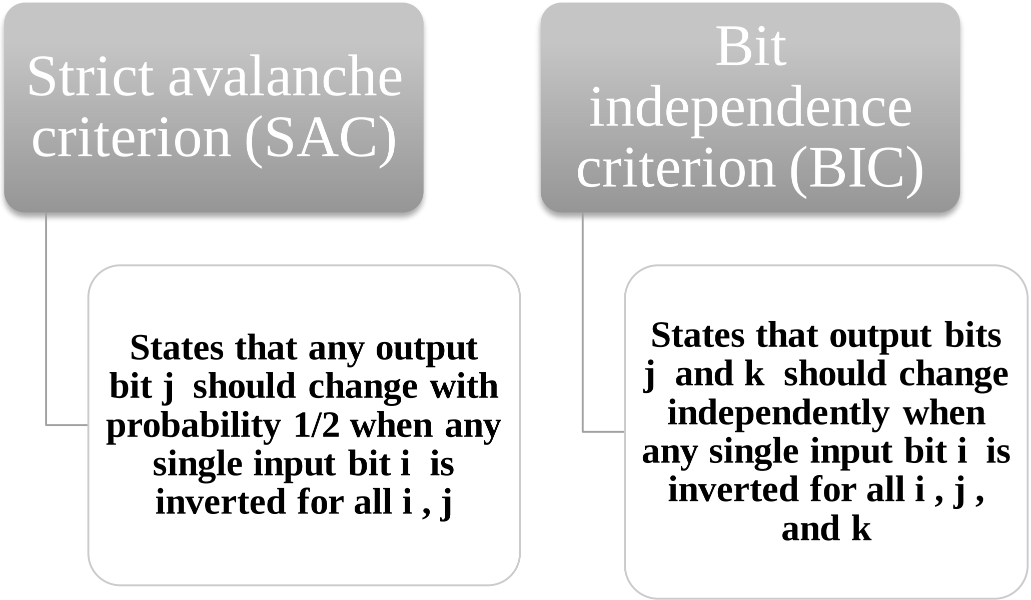


Key size (bits)	Cipher	Number of Alternative Keys	Time Required at 109 decryptions/s	Time Required at 1013 decryptions/s
56	DES	256 ≈ 7.2 × 1016	255 ns = 1.125 years	1 hour
128	AES	2128 ≈ 3.4 × 1038	2127 ns = 5.3 × 1021 years	5.3 × 1017 years
168	Triple DES	2168 ≈ 3.7 × 1050	2167 ns = 5.8 × 1033 years	5.8 × 1029 years
192	AES	2192 ≈ 6.3 × 1057	2191 ns = 9.8 × 1040 years	9.8 × 1036 years
256	AES	2256 ≈ 1.2 × 1077	2255 ns = 1.8 × 1060 years	1.8 × 1056 years
26 characters (permutation)	Monoalphabetic	26! = 4 × 1026	2 × 1026 ns = 6.3 × 109 years	6.3 × 106 years

Block Cipher Design Principles: Design of Function F

- The heart of a Feistel block cipher is the function F
- The more nonlinear F, the more difficult any type of cryptanalysis will be
- The SAC and BIC criteria appear to strengthen the effectiveness of the confusion function

The algorithm should have good avalanche properties



Strict avalanche criterion (SAC)

States that any output bit j should change with probability 1/2 when any single input bit i is inverted for all i , j

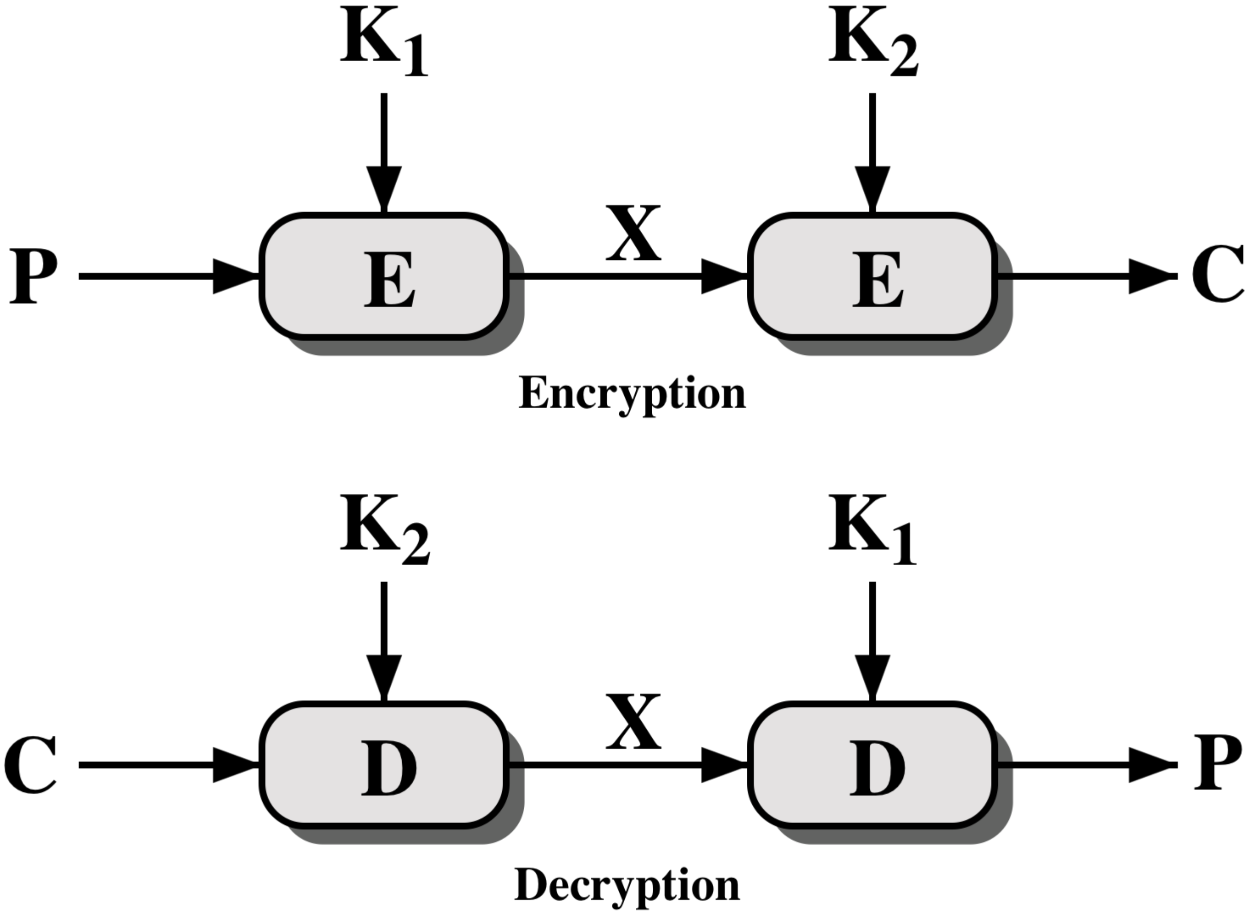
Bit independence criterion (BIC)

States that output bits j and k should change independently when any single input bit i is inverted for all i , j , and k

Concerns about DES

- Short key length
  - DES “cracker”, can break DES in days
  - Computation can be distributed to make it faster
  - Does not mean “DES is insecure”; depends on desired security
- Short block length
  - Repeated blocks happen “too frequently”
- Some (theoretical) attacks have been found
  - Claimed known to DES designers 15 years before public discovery!

Double DES

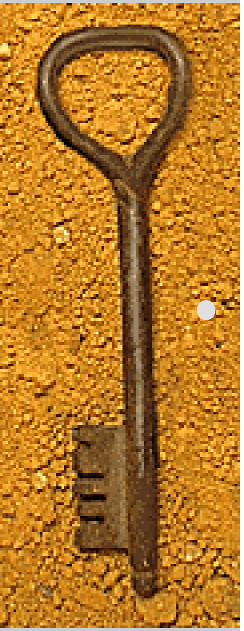


(a) Double Encryption

Meet-in-the-Middle Attack

The use of double DES results in a mapping that is not equivalent to a single DES encryption

The meet-in-the-middle attack algorithm will attack this scheme and does not depend on any particular property of DES but will work against any block encryption cipher

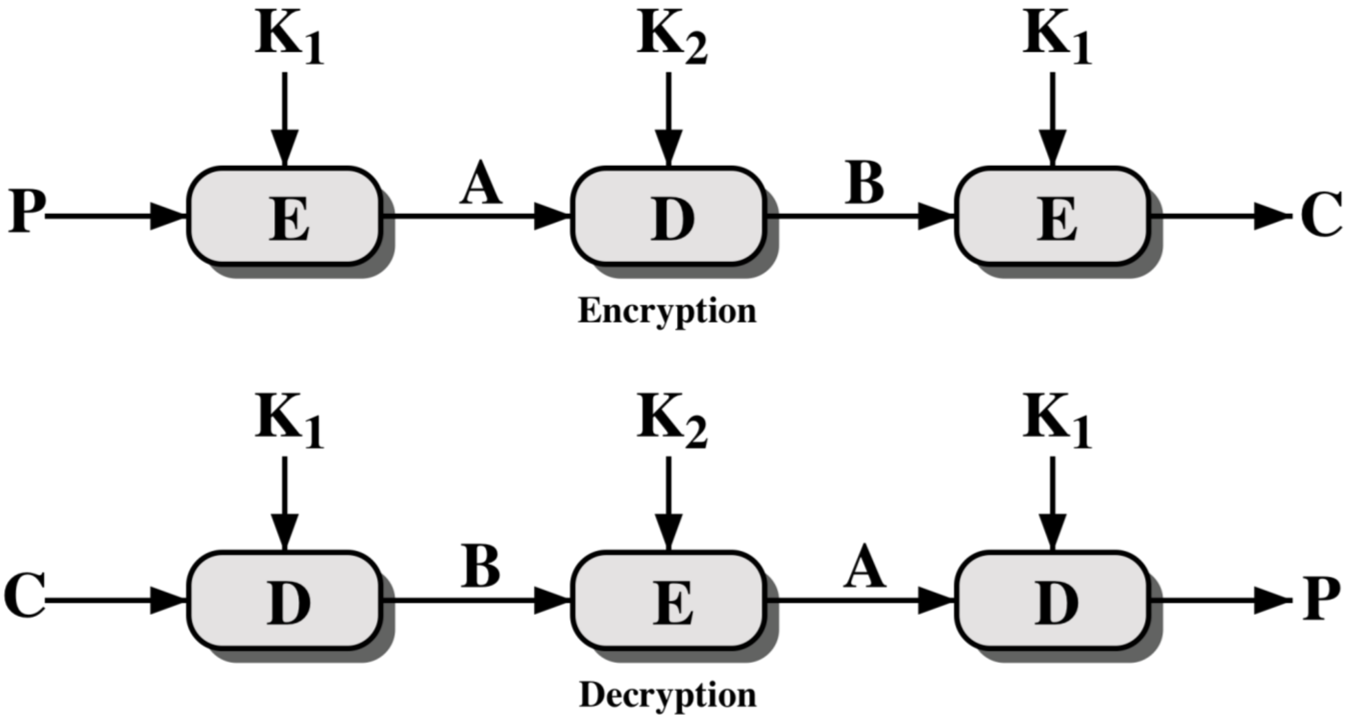


Triple-DES with Two-Keys

- Obvious counter to the meet-in-the-middle attack is to use three stages of encryption with three different keys
  - This raises the cost of the meet-in-the-middle attack to 2^{112}, which is beyond what is practical
  - Has the drawback of requiring a key length of 56 x 3 = 168 bits, which may be somewhat unwieldy
  - As an alternative Tuchman proposed a triple encryption method that uses only two keys

3DES with two keys is a relatively popular alternative to DES and has been adopted for use in the key management standards ANSI X9.17 and ISO 8732

Multiple Encryption



(b) Triple Encryption

Figure 6.1 Multiple Encryption

Triple DES with Three Keys

- Many researchers now feel that three-key 3DES is the preferred alternative

Three-key 3DES has an effective key length of 168 bits and is defined as:

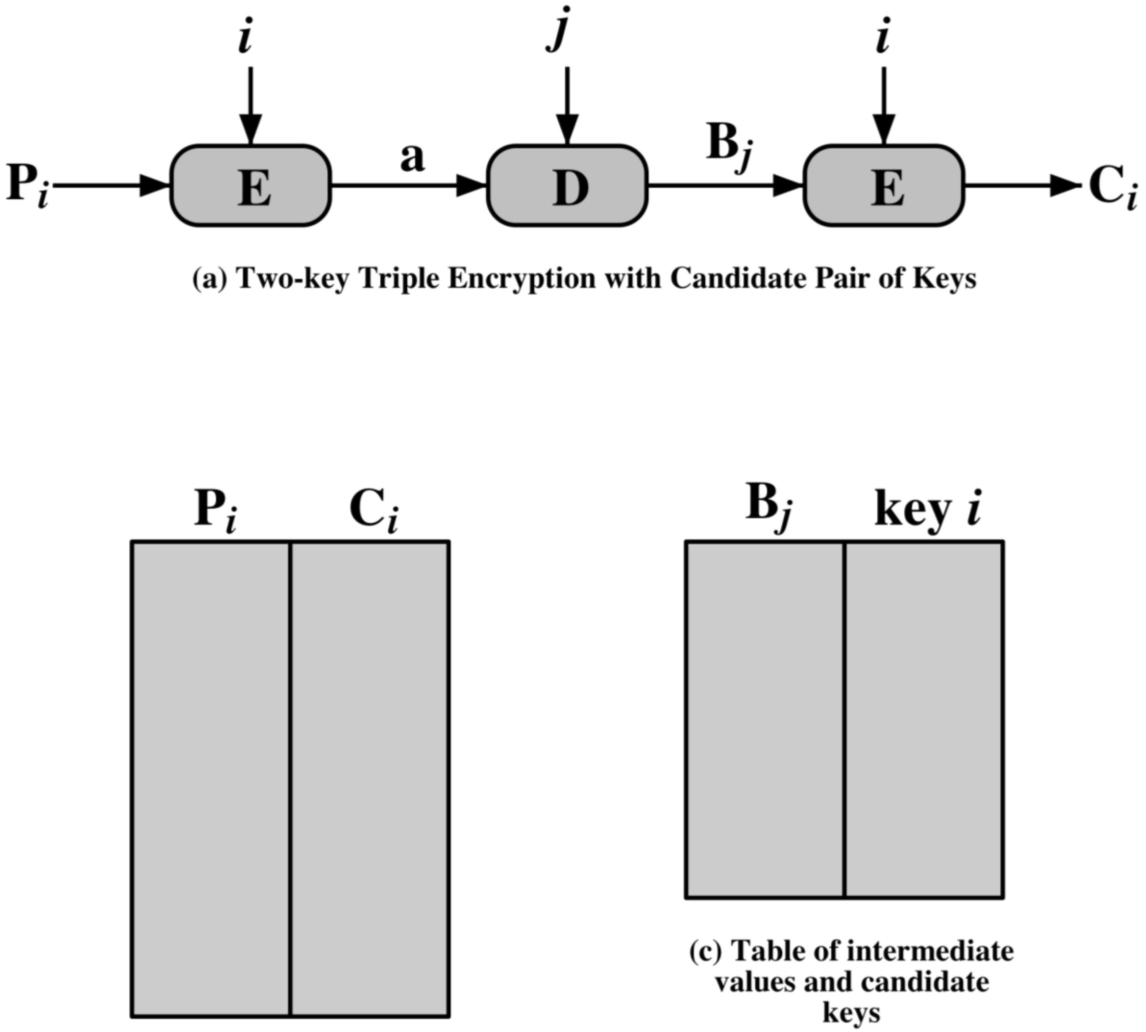
- $C = \mathrm{E}(K_3, \mathrm{D}(K_2, \mathrm{E}(K_1, P)))$

Backward compatibility with DES is provided by putting:

- $K_3 = K_2$ or $K_1 = K_2$

- A number of Internet-based applications have adopted three-key 3DES including PGP and S/MIME

Next is AES

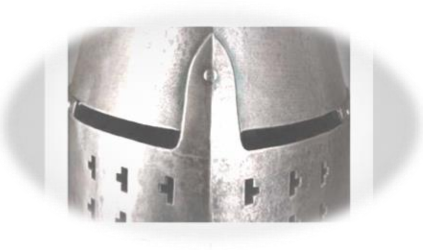


(a) Two-key Triple Encryption with Candidate Pair of Keys

(b) Table of n known plaintext-ciphertext pairs, sorted on P

(c) Table of intermediate values and candidate keys

Figure 6.2 Known-Plaintext Attack on Triple DES



Advanced Encryption Standard

Modified by: Dr. Ramzi Saifan

Why AES?

- Symmetric block cipher, published in 2001
- Intended to replace DES and 3DES

  DES is vulnerable to multiple attacks

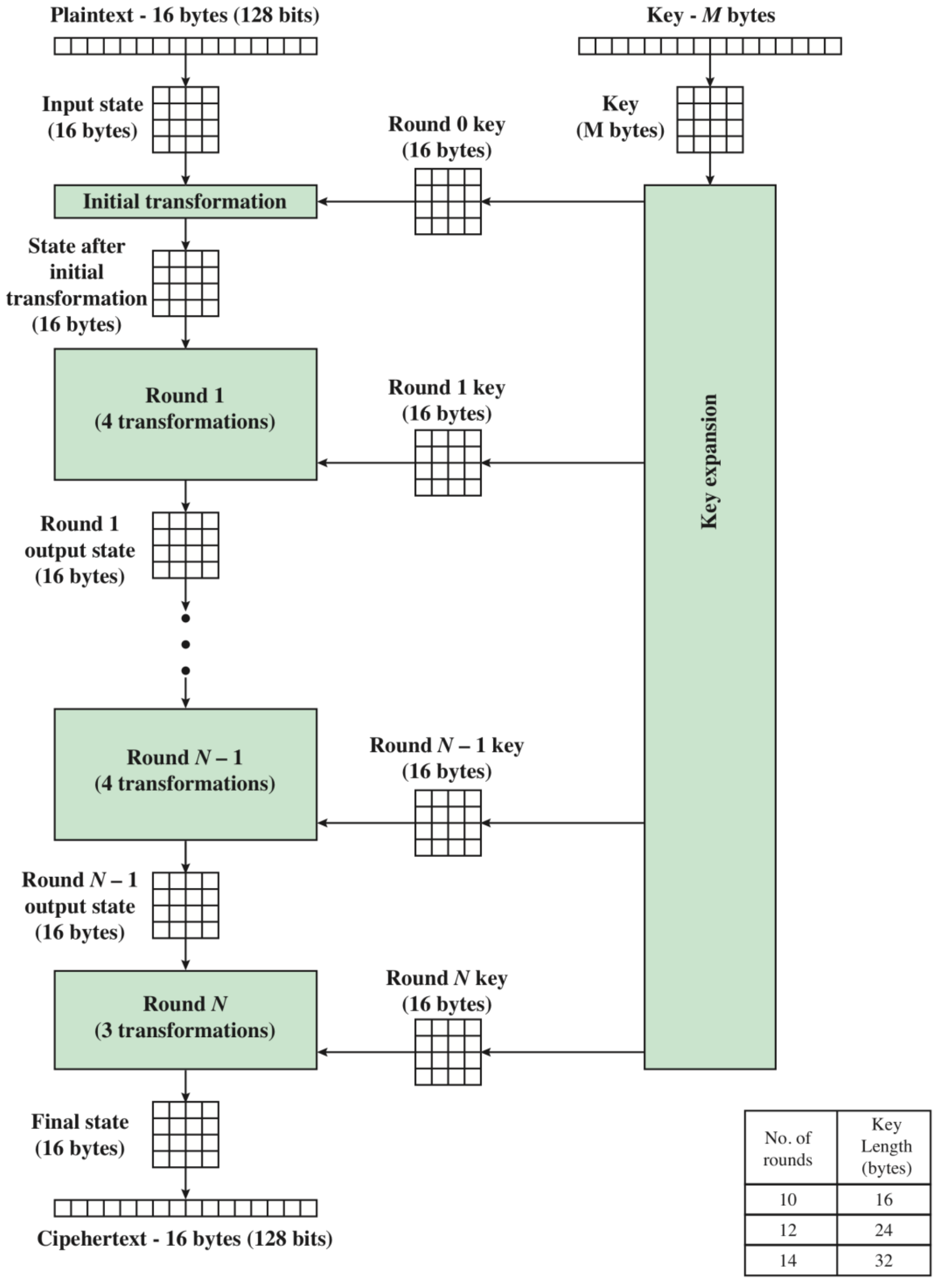
  3DES has slow performances

NIST Criteria to Evaluate Potential Candidates

- Security: The effort to crypt analyze an algorithm.
- Cost: The algorithm should be practical in a wide range of applications.
- Algorithm and Implementation Characteristics : Flexibility, simplicity etc.

5 final candidates have been chosen out of 15

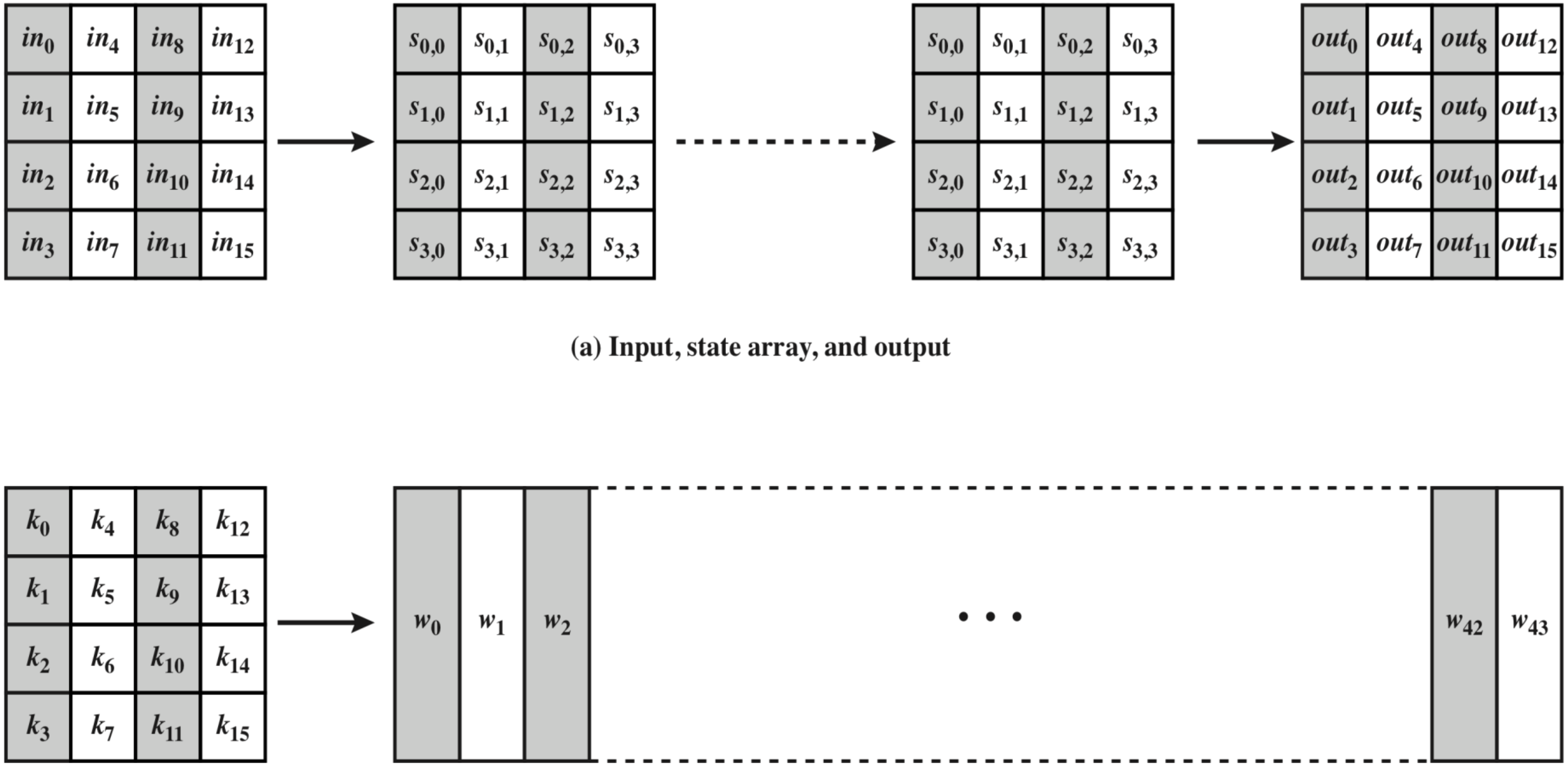
AES Encryption Process


Plaintext - 16 bytes (128 bits)
Key - *M* bytes
Input state (16 bytes)
Key (M bytes)
Round 0 key (16 bytes)
Initial transformation
State after initial transformation (16 bytes)
Round 1 (4 transformations)
Round 1 key (16 bytes)
Round 1 output state (16 bytes)
Key expansion
Round *N* – 1 (4 transformations)
Round *N* – 1 key (16 bytes)
Round *N* – 1 output state (16 bytes)
Round *N* (3 transformations)
Round *N* key (16 bytes)
Final state (16 bytes)
Cipehertext - 16 bytes (128 bits)

No. of rounds	Key Length (bytes)
10	16
12	24
14	32

Figure 5.1 AES Encryption Process

AES Data Structures



in_0	in_4	in_8	in_{12}
in_1	in_5	in_9	in_{13}
in_2	in_6	in_{10}	in_{14}
in_3	in_7	in_{11}	in_{15}

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

out_0	out_4	out_8	out_{12}
out_1	out_5	out_9	out_{13}
out_2	out_6	out_{10}	out_{14}
out_3	out_7	out_{11}	out_{15}

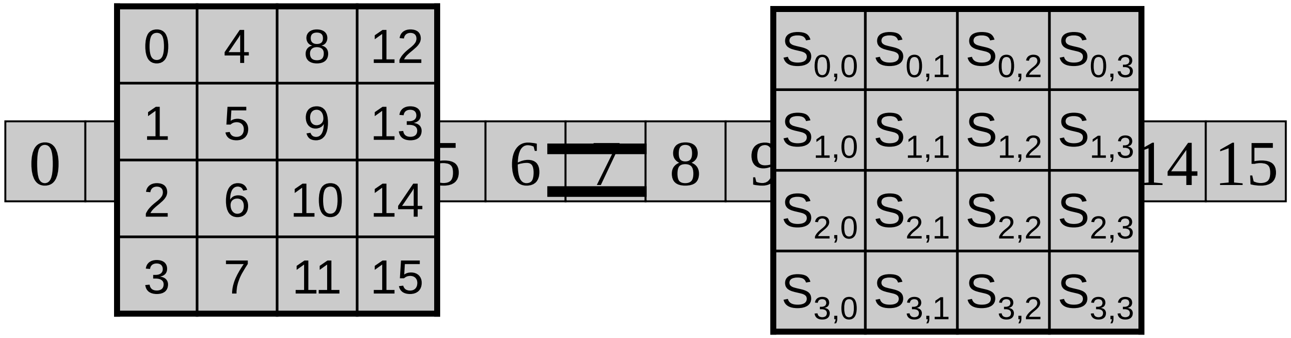
(a) Input, state array, and output

k_0	k_4	k_8	k_{12}
k_1	k_5	k_9	k_{13}
k_2	k_6	k_{10}	k_{14}
k_3	k_7	k_{11}	k_{15}

(b) Key and expanded key

Convert to State Array

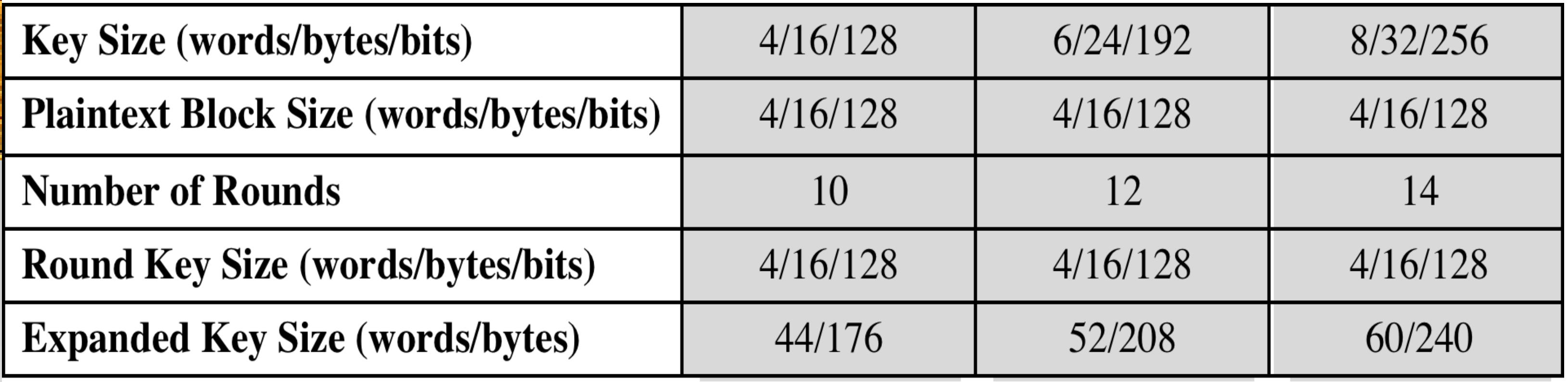
Input block:



0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

Table 5.1
AES Parameters



Key Size (words/bytes/bits)	4/16/128	6/24/192	8/32/256
Plaintext Block Size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Number of Rounds	10	12	14
Round Key Size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Expanded Key Size (words/bytes)	44/176	52/208	60/240

AES Encryption and Decryption

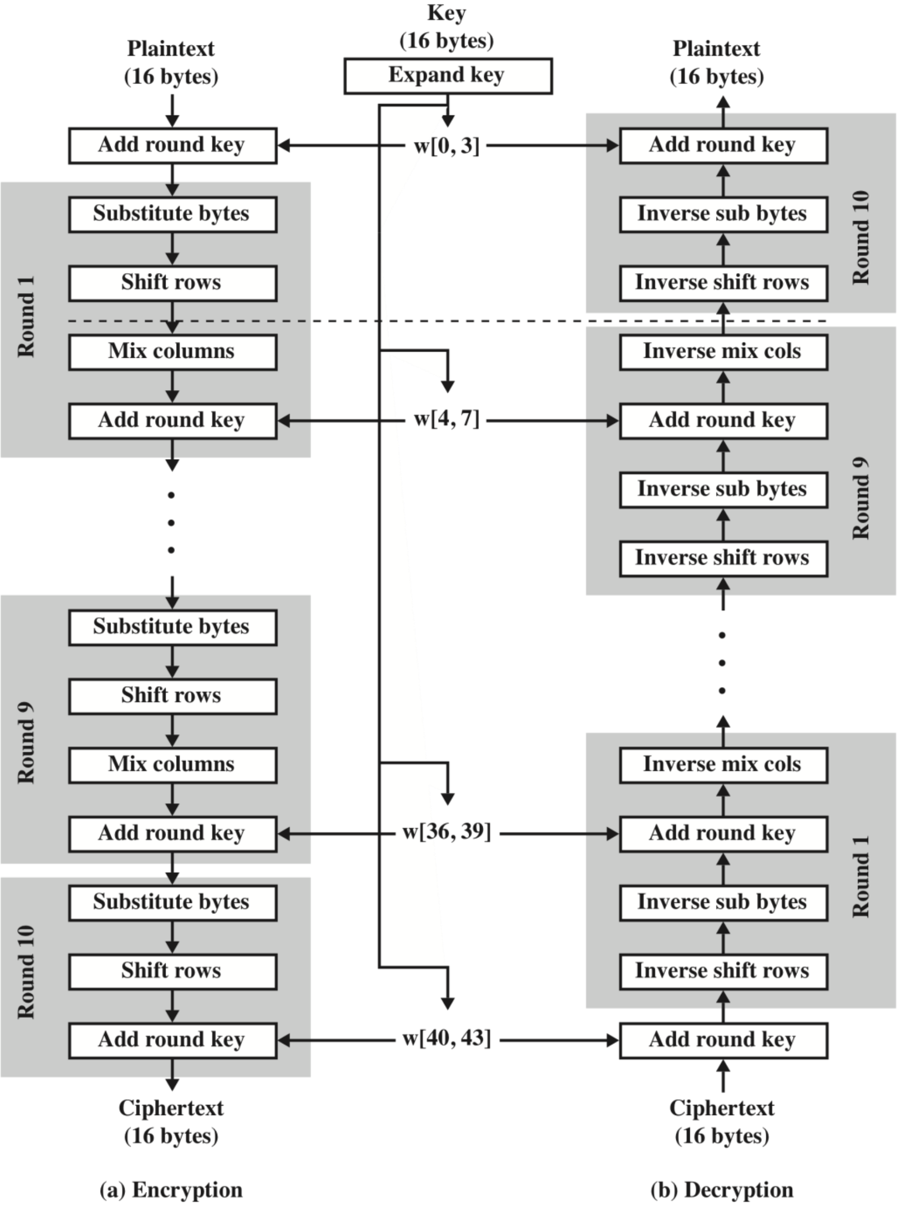


Figure 5.3 AES Encryption and Decryption

Detailed Structure

- The key that is provided as input is expanded into an array of forty-four 32-bit words, *w[i]*

Four different stages are used:

- Substitute bytes – uses an S-box to perform a byte-by-byte substitution of the block
- ShiftRows – a simple permutation
- MixColumns – a substitution that makes use of arithmetic
- AddRoundKey – a simple bitwise XOR of the current block with a portion of the expanded key

- Can view the cipher as alternating operations of XOR encryption (AddRoundKey) of a block, followed by scrambling of the block (the other three stages), followed by XOR encryption, and so on
- Each stage is easily reversible
- The decryption algorithm makes use of the expanded key in reverse order, however the decryption algorithm is not identical to the encryption algorithm
- Final round of both encryption and decryption consists of only three stages

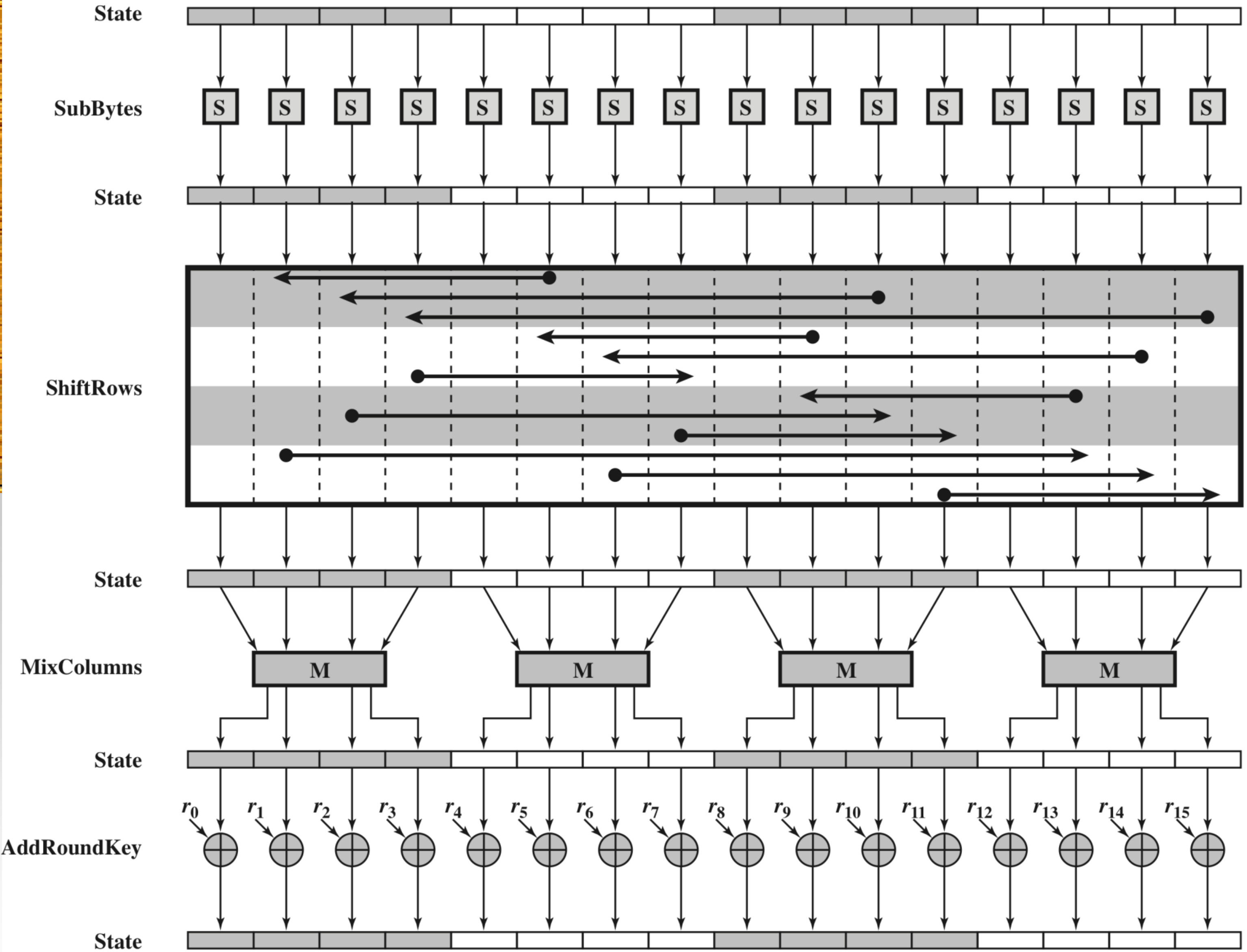
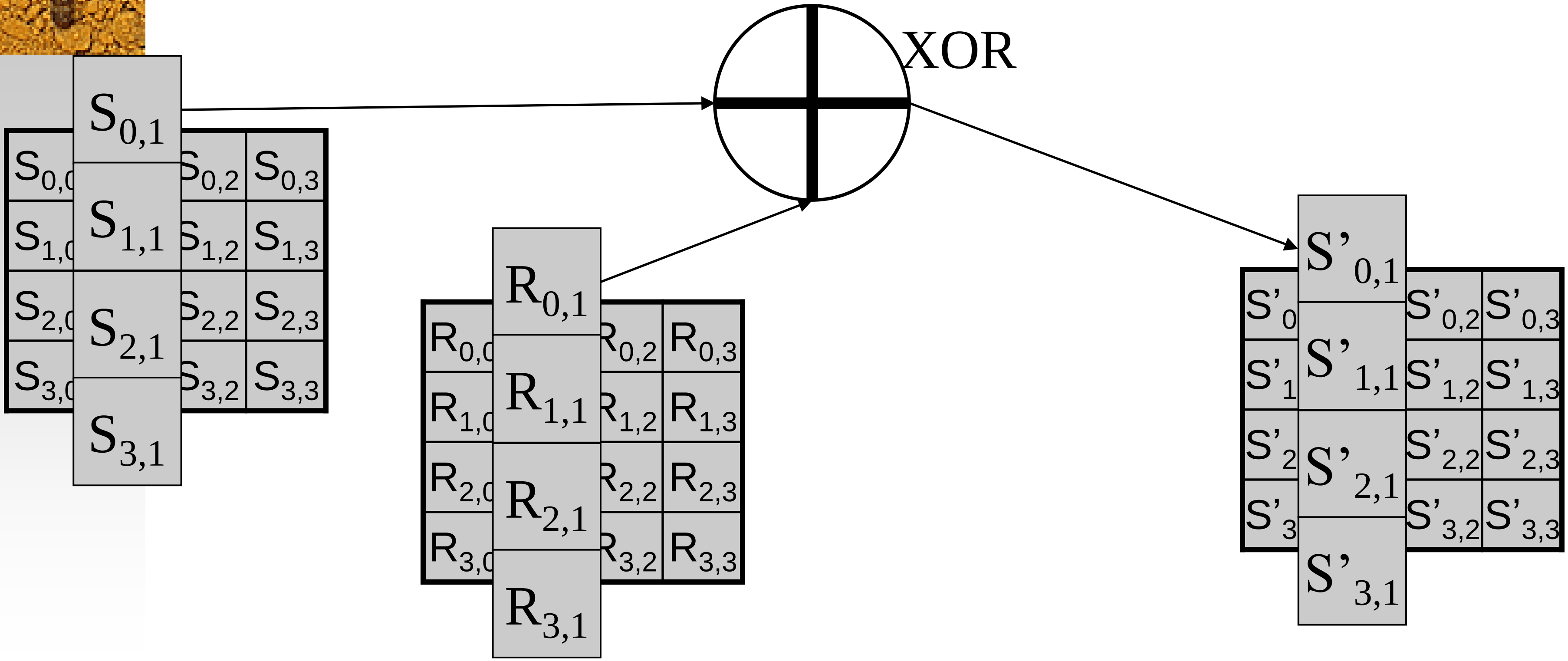

State
SubBytes
S
State
ShiftRows
State
MixColumns
M
State
AddRoundKey
r_0 r_1 r_2 r_3 r_4 r_5 r_6 r_7 r_8 r_9 r_{10} r_{11} r_{12} r_{13} r_{14} r_{15}
State

Figure 5.4 AES Encryption Round

AddRoundKey

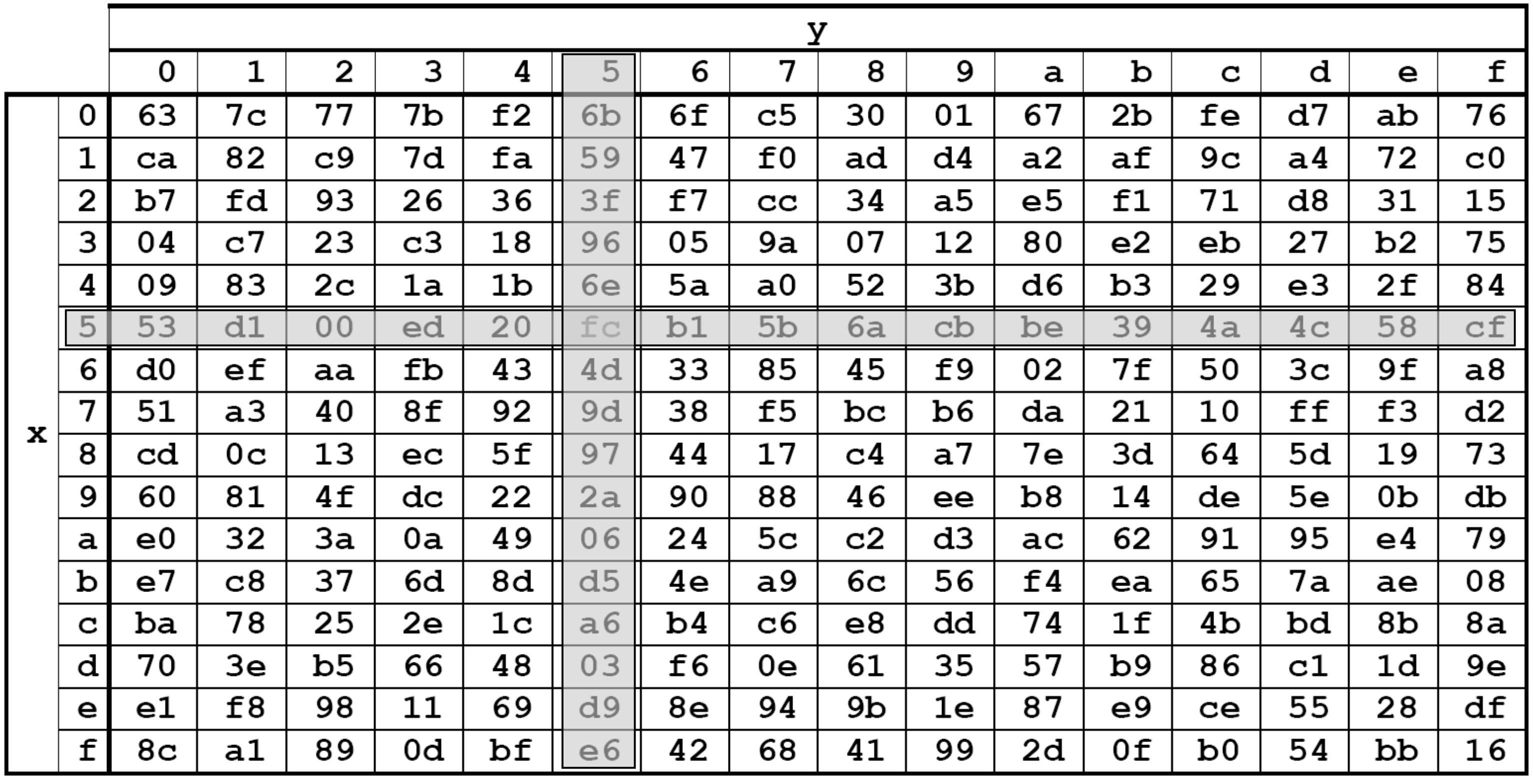
- XOR each byte of the round key with its corresponding byte in the state array



XOR
$S_{0,0}$ $S_{0,1}$ $S_{0,2}$ $S_{0,3}$
$S_{1,0}$ $S_{1,1}$ $S_{1,2}$ $S_{1,3}$
$S_{2,0}$ $S_{2,1}$ $S_{2,2}$ $S_{2,3}$
$S_{3,0}$ $S_{3,1}$ $S_{3,2}$ $S_{3,3}$
$R_{0,0}$ $R_{0,1}$ $R_{0,2}$ $R_{0,3}$
$R_{1,0}$ $R_{1,1}$ $R_{1,2}$ $R_{1,3}$
$R_{2,0}$ $R_{2,1}$ $R_{2,2}$ $R_{2,3}$
$R_{3,0}$ $R_{3,1}$ $R_{3,2}$ $R_{3,3}$
S'_{0} $S'_{0,1}$ $S'_{0,2}$ $S'_{0,3}$
S'_{1} $S'_{1,1}$ $S'_{1,2}$ $S'_{1,3}$
S'_{2} $S'_{2,1}$ $S'_{2,2}$ $S'_{2,3}$
S'_{3} $S'_{3,1}$ $S'_{3,2}$ $S'_{3,3}$

SubBytes

- Replace each byte in the state array with its corresponding value from the 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

(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

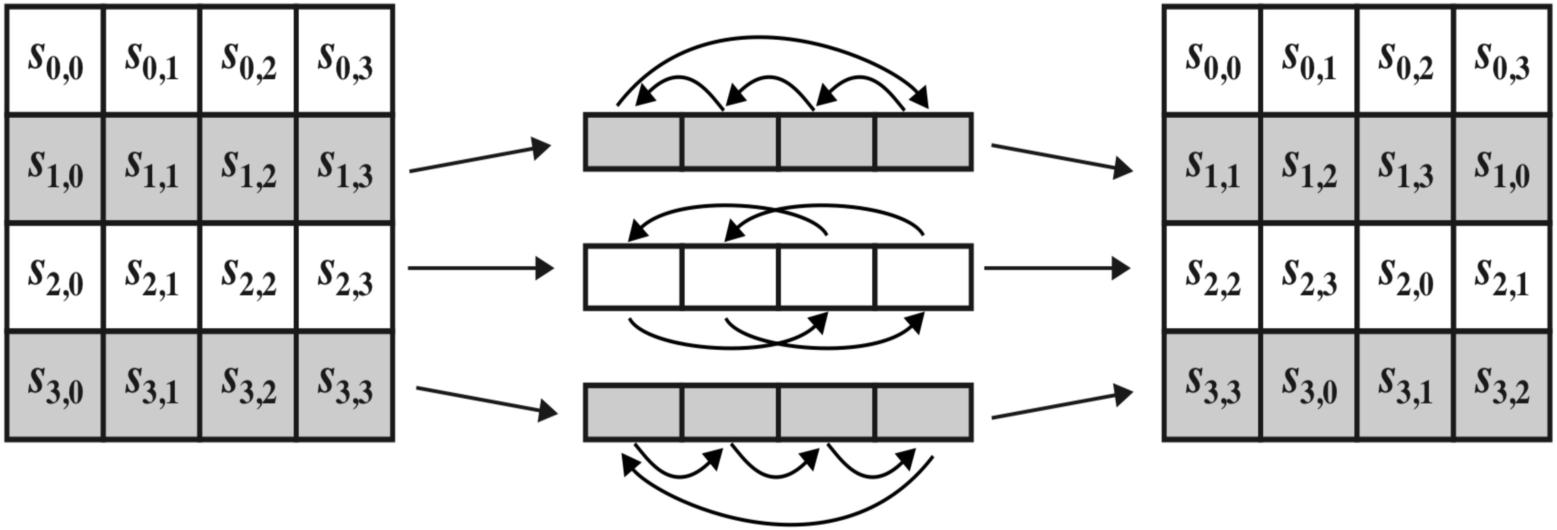
(b) Inverse S-box

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

S-Box Rationale

- The S-box is designed to be resistant to known cryptanalytic attacks
- The Rijndael developers sought a design that has a low correlation between input bits and output bits and the property that the output is not a linear mathematical function of the input

Shift Row Transformation



$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

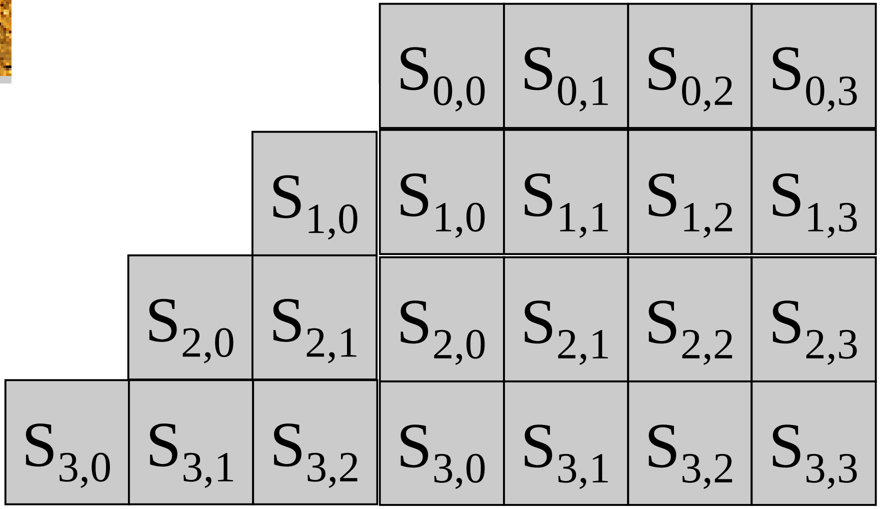
$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,1}$	$s_{1,2}$	$s_{1,3}$	$s_{1,0}$
$s_{2,2}$	$s_{2,3}$	$s_{2,0}$	$s_{2,1}$
$s_{3,3}$	$s_{3,0}$	$s_{3,1}$	$s_{3,2}$

(a) Shift row transformation

AES Row and Column Operations

ShiftRows

- Last three rows are cyclically shifted

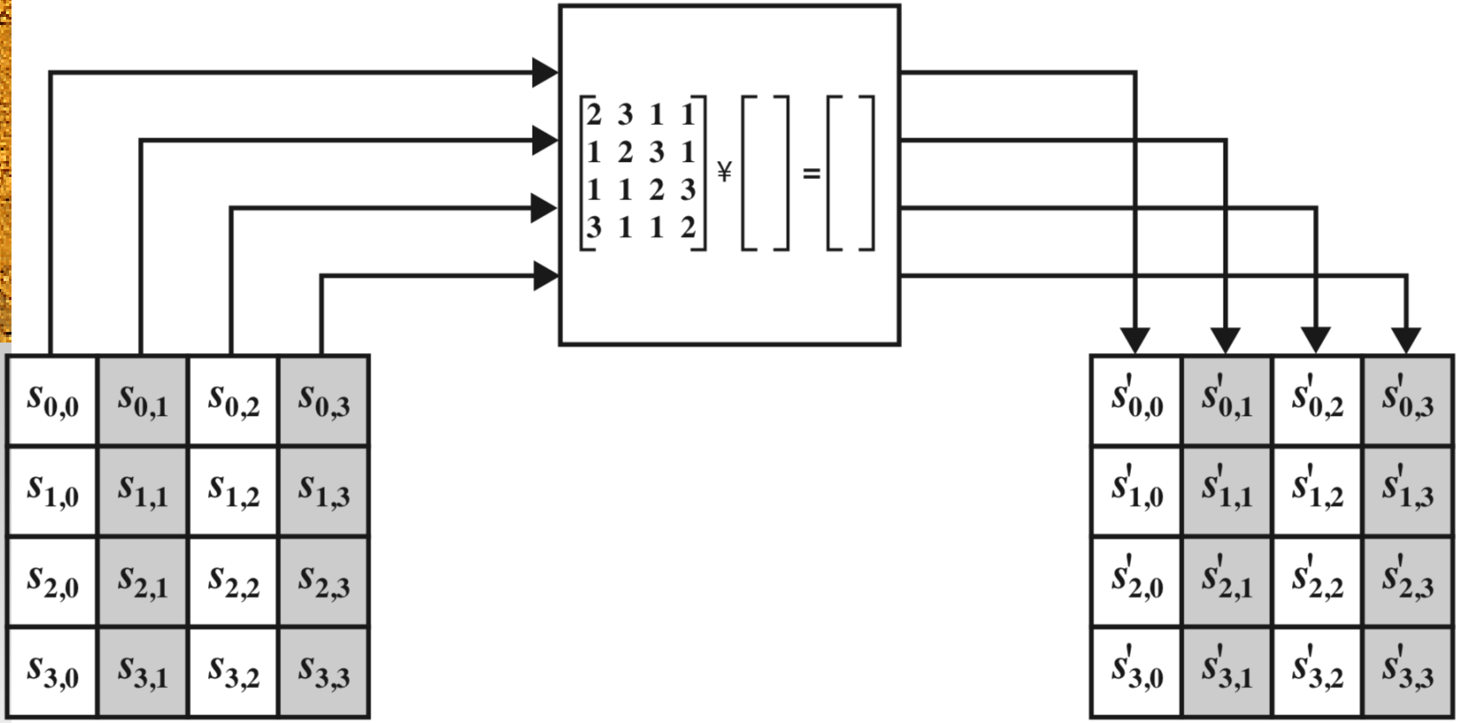


			$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
		$S_{1,0}$	$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
	$S_{2,0}$	$S_{2,1}$	$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

Shift Row Rationale

- On encryption, the first 4 bytes of the plaintext are copied to the first column of State, and so on
- The round key is applied to State column by column
  - Thus, a row shift moves an individual byte from one column to another, which is a linear distance of a multiple of 4 bytes
- Transformation ensures that the 4 bytes of one column are spread out to four different columns

MixColumn Transformation


2 3 1 1
1 2 3 1
1 1 2 3
3 1 1 2
¥
=
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

(b) Mix column transformation

Figure 5.7 AES Row and Column Operations

(Figure can be found on page 144 in textbook)

MixColumns

- Apply MixColumn transformation to each column

$$S'_{0,c} = (\{02\} \bullet S_{0,c}) \oplus (\{03\} \bullet S_{1,c}) \oplus S_{2,c} \oplus S_{3,c}$$

$$S'_{1,c} = S_{0,c} \oplus (\{02\} \bullet S_{1,c}) \oplus (\{03\} \bullet S_{2,c}) \oplus S_{3,c}$$

$$S'_{2,c} = S_{0,c} \oplus S_{1,c} \oplus (\{02\} \bullet S_{2,c}) \oplus (\{03\} \bullet S_{3,c})$$

$$S'_{3,c} = (\{03\} \bullet S_{0,c}) \oplus S_{1,c} \oplus S_{2,c} \oplus (\{02\} \bullet S_{3,c}$$

Mix Columns Rationale

- Coefficients of a matrix based on a linear code with maximal distance between code words ensures a good mixing among the bytes of each column
- The mix column transformation combined with the shift row transformation ensures that after a few rounds all output bits depend on all input bits

AddRoundKey Transformation

- The 128 bits of State are bitwise XORed with the 128 bits of the round key
- Operation is viewed as a columnwise operation between the 4 bytes of a State column and one word of the round key
  - Can also be viewed as a byte-level operation

Rationale:

Is as simple as possible and affects every bit of State

The complexity of the round key expansion plus the complexity of the other stages of AES ensure security

Inputs for Single AES Round

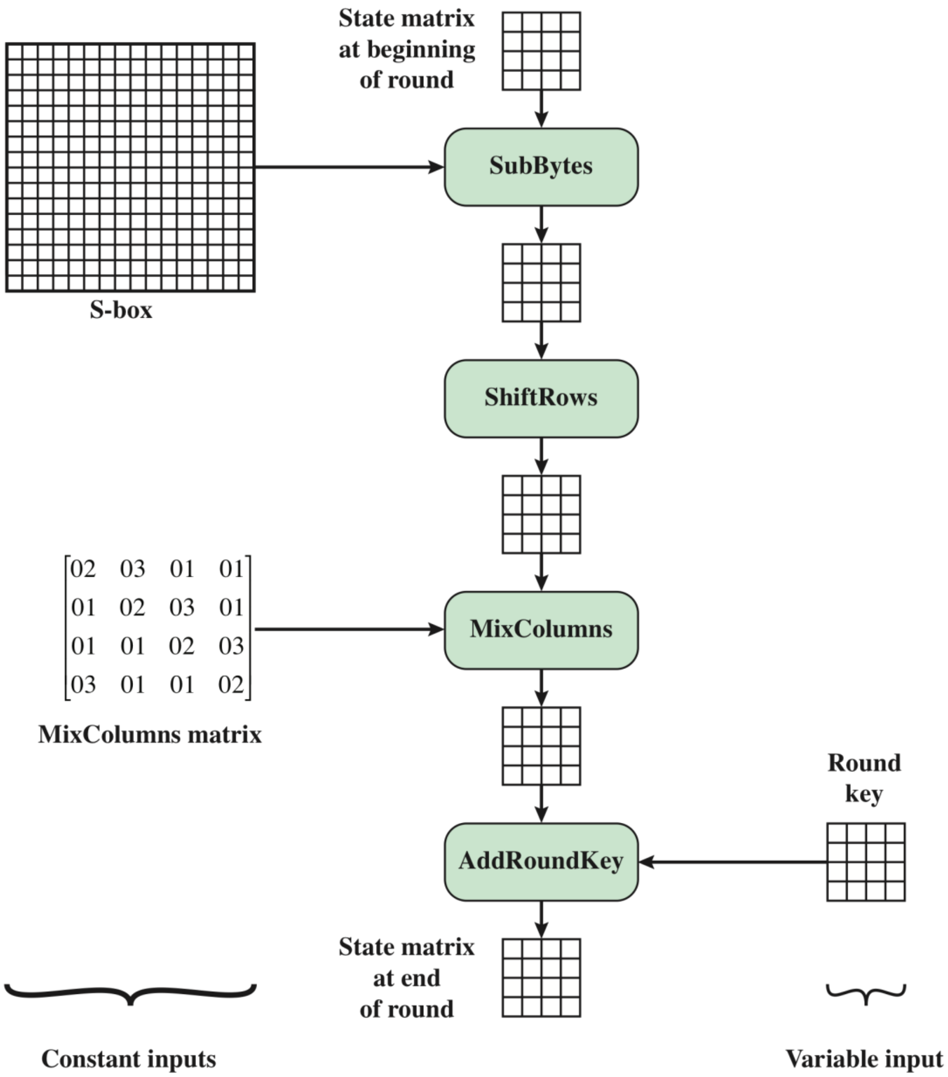
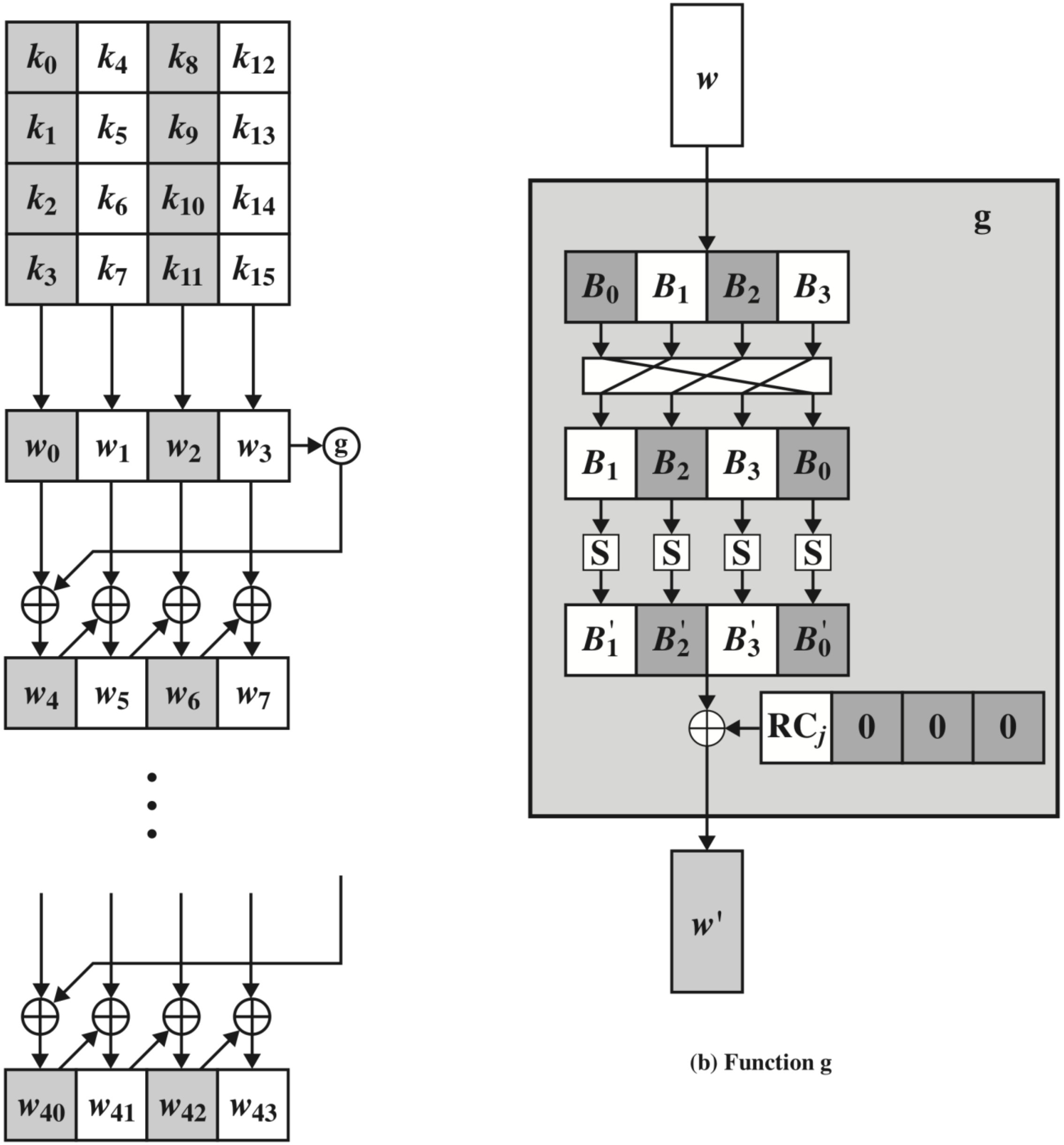


Figure 5.8 Inputs for Single AES Round

AES Key Expansion

- Takes as input a four-word (16 byte) key and produces a linear array of 44 words (176) bytes
  - This is sufficient to provide a four-word round key for the initial AddRoundKey stage and each of the 10 rounds of the cipher
- Key is copied into the first four words of the expanded key
  - The remainder of the expanded key is filled in four words at a time
- Each added word *w*[i] depends on the immediately preceding word, *w[i – 1]*, and the word four positions back, w[i – 4]
  - In three out of four cases a simple XOR is used
  - For a word whose position in the *w* array is a multiple of 4, a more complex function is used

AES Key Expansion



k_0 k_4 k_8 k_{12}
k_1 k_5 k_9 k_{13}
k_2 k_6 k_{10} k_{14}
k_3 k_7 k_{11} k_{15}
w_0 w_1 w_2 w_3
g
w_4 w_5 w_6 w_7
w_{40} w_{41} w_{42} w_{43}
w
g
B_0 B_1 B_2 B_3
B_1 B_2 B_3 B_0
S S S S
B'_1 B'_2 B'_3 B'_0
RC_j 0 0 0
w'

(b) Function g

(a) Overall algorithm

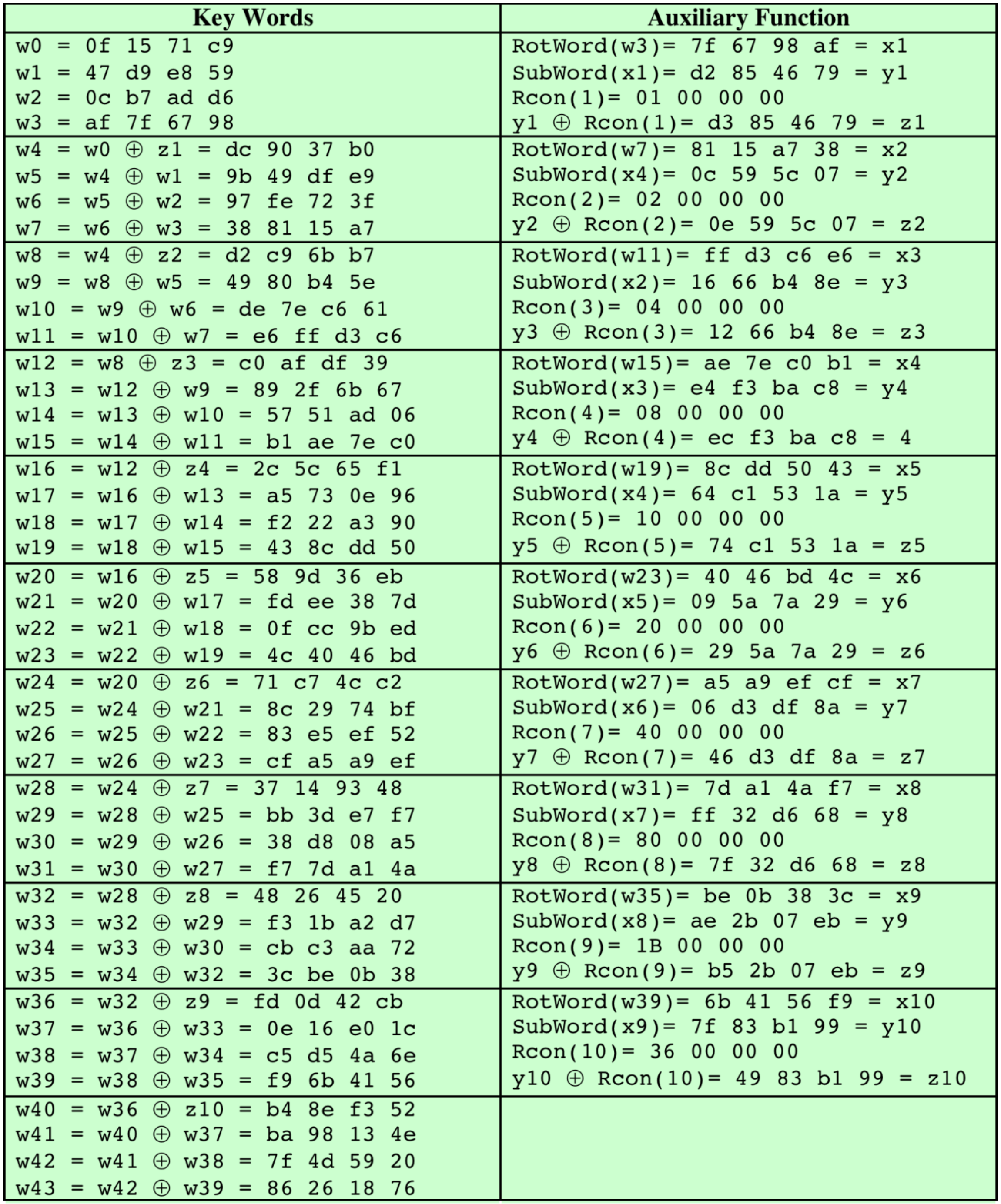
Figure 5.9 AES Key Expansion

Key Expansion Rationale

- The Rijndael developers designed the expansion key algorithm to be resistant to known cryptanalytic attacks

- Inclusion of a round-dependent round constant eliminates the symmetry between the ways in which round keys are generated in different rounds

AES Example Key Expansion



Key Words	Auxiliary Function
w0 = 0f 15 71 c9	RotWord(w3)= 7f 67 98 af = x1
w1 = 47 d9 e8 59	SubWord(x1)= d2 85 46 79 = y1
w2 = 0c b7 ad d6	Rcon(1)= 01 00 00 00
w3 = af 7f 67 98	y1 ⊕ Rcon(1)= d3 85 46 79 = z1
w4 = w0 ⊕ z1 = dc 90 37 b0	RotWord(w7)= 81 15 a7 38 = x2
w5 = w4 ⊕ w1 = 9b 49 df e9	SubWord(x4)= 0c 59 5c 07 = y2
w6 = w5 ⊕ w2 = 97 fe 72 3f	Rcon(2)= 02 00 00 00
w7 = w6 ⊕ w3 = 38 81 15 a7	y2 ⊕ Rcon(2)= 0e 59 5c 07 = z2
w8 = w4 ⊕ z2 = d2 c9 6b b7	RotWord(w11)= ff d3 c6 e6 = x3
w9 = w8 ⊕ w5 = 49 80 b4 5e	SubWord(x2)= 16 66 b4 8e = y3
w10 = w9 ⊕ w6 = de 7e c6 61	Rcon(3)= 04 00 00 00
w11 = w10 ⊕ w7 = e6 ff d3 c6	y3 ⊕ Rcon(3)= 12 66 b4 8e = z3
w12 = w8 ⊕ z3 = c0 af df 39	RotWord(w15)= ae 7e c0 b1 = x4
w13 = w12 ⊕ w9 = 89 2f 6b 67	SubWord(x3)= e4 f3 ba c8 = y4
w14 = w13 ⊕ w10 = 57 51 ad 06	Rcon(4)= 08 00 00 00
w15 = w14 ⊕ w11 = b1 ae 7e c0	y4 ⊕ Rcon(4)= ec f3 ba c8 = 4
w16 = w12 ⊕ z4 = 2c 5c 65 f1	RotWord(w19)= 8c dd 50 43 = x5
w17 = w16 ⊕ w13 = a5 73 0e 96	SubWord(x4)= 64 c1 53 1a = y5
w18 = w17 ⊕ w14 = f2 22 a3 90	Rcon(5)= 10 00 00 00
w19 = w18 ⊕ w15 = 43 8c dd 50	y5 ⊕ Rcon(5)= 74 c1 53 1a = z5
w20 = w16 ⊕ z5 = 58 9d 36 eb	RotWord(w23)= 40 46 bd 4c = x6
w21 = w20 ⊕ w17 = fd ee 38 7d	SubWord(x5)= 09 5a 7a 29 = y6
w22 = w21 ⊕ w18 = 0f cc 9b ed	Rcon(6)= 20 00 00 00
w23 = w22 ⊕ w19 = 4c 40 46 bd	y6 ⊕ Rcon(6)= 29 5a 7a 29 = z6
w24 = w20 ⊕ z6 = 71 c7 4c c2	RotWord(w27)= a5 a9 ef cf = x7
w25 = w24 ⊕ w21 = 8c 29 74 bf	SubWord(x6)= 06 d3 df 8a = y7
w26 = w25 ⊕ w22 = 83 e5 ef 52	Rcon(7)= 40 00 00 00
w27 = w26 ⊕ w23 = cf a5 a9 ef	y7 ⊕ Rcon(7)= 46 d3 df 8a = z7
w28 = w24 ⊕ z7 = 37 14 93 48	RotWord(w31)= 7d a1 4a f7 = x8
w29 = w28 ⊕ w25 = bb 3d e7 f7	SubWord(x7)= ff 32 d6 68 = y8
w30 = w29 ⊕ w26 = 38 d8 08 a5	Rcon(8)= 80 00 00 00
w31 = w30 ⊕ w27 = f7 7d a1 4a	y8 ⊕ Rcon(8)= 7f 32 d6 68 = z8
w32 = w28 ⊕ z8 = 48 26 45 20	RotWord(w35)= be 0b 38 3c = x9
w33 = w32 ⊕ w29 = f3 1b a2 d7	SubWord(x8)= ae 2b 07 eb = y9
w34 = w33 ⊕ w30 = cb c3 aa 72	Rcon(9)= 1B 00 00 00
w35 = w34 ⊕ w32 = 3c be 0b 38	y9 ⊕ Rcon(9)= b5 2b 07 eb = z9
w36 = w32 ⊕ z9 = fd 0d 42 cb	RotWord(w39)= 6b 41 56 f9 = x10
w37 = w36 ⊕ w33 = 0e 16 e0 1c	SubWord(x9)= 7f 83 b1 99 = y10
w38 = w37 ⊕ w34 = c5 d5 4a 6e	Rcon(10)= 36 00 00 00
w39 = w38 ⊕ w35 = f9 6b 41 56	y10 ⊕ Rcon(10)= 49 83 b1 99 = z10
w40 = w36 ⊕ z10 = b4 8e f3 52	
w41 = w40 ⊕ w37 = ba 98 13 4e	
w42 = w41 ⊕ w38 = 7f 4d 59 20	
w43 = w42 ⊕ w39 = 86 26 18 76	

AES Example

Start of round	After SubBytes	After ShiftRows	After MixColumns	Round Key
01 89 fe 76 23 ab dc 54 45 cd ba 32 67 ef 98 10				0f 47 0c af 15 d9 b7 7f 71 e8 ad 67 c9 59 d6 98
0e ce f2 d9 36 72 6b 2b 34 25 17 55 ae b6 4e 88	ab 8b 89 35 05 40 7f f1 18 3f f0 fc e4 4e 2f c4	ab 8b 89 35 40 7f f1 05 f0 fc 18 3f c4 e4 4e 2f	b9 94 57 75 e4 8e 16 51 47 20 9a 3f c5 d6 f5 3b	dc 9b 97 38 90 49 fe 81 37 df 72 15 b0 e9 3f a7
65 0f c0 4d 74 c7 e8 d0 70 ff e8 2a 75 3f ca 9c	4d 76 ba e3 92 c6 9b 70 51 16 9b e5 9d 75 74 de	4d 76 ba e3 c6 9b 70 92 9b e5 51 16 de 9d 75 74	8e 22 db 12 b2 f2 dc 92 df 80 f7 c1 2d c5 1e 52	d2 49 de e6 c9 80 7e ff 6b b4 c6 d3 b7 5e 61 c6
5c 6b 05 f4 7b 72 a2 6d b4 34 31 12 9a 9b 7f 94	4a 7f 6b bf 21 40 3a 3c 8d 18 c7 c9 b8 14 d2 22	4a 7f 6b bf 40 3a 3c 21 c7 c9 8d 18 22 b8 14 d2	b1 c1 0b cc ba f3 8b 07 f9 1f 6a c3 1d 19 24 5c	c0 89 57 b1 af 2f 51 ae df 6b ad 7e 39 67 06 c0
71 48 5c 7d 15 dc da a9 26 74 c7 bd 24 7e 22 9c	a3 52 4a ff 59 86 57 d3 f7 92 c6 7a 36 f3 93 de	a3 52 4a ff 86 57 d3 59 c6 7a f7 92 de 36 f3 93	d4 11 fe 0f 3b 44 06 73 cb ab 62 37 19 b7 07 ec	2c a5 f2 43 5c 73 22 8c 65 0e a3 dd f1 96 90 50
f8 b4 0c 4c 67 37 24 ff ae a5 c1 ea e8 21 97 bc	41 8d fe 29 85 9a 36 16 e4 06 78 87 9b fd 88 65	41 8d fe 29 9a 36 16 85 78 87 e4 06 65 9b fd 88	2a 47 c4 48 83 e8 18 ba 84 18 27 23 eb 10 0a f3	58 fd 0f 4c 9d ee cc 40 36 38 9b 46 eb 7d ed bd
72 ba cb 04 1e 06 d4 fa b2 20 bc 65 00 6d e7 4e	40 f4 1f f2 72 6f 48 2d 37 b7 65 4d 63 3c 94 2f	40 f4 1f f2 6f 48 2d 72 65 4d 37 b7 2f 63 3c 94	7b 05 42 4a 1e d0 20 40 94 83 18 52 94 c4 43 fb	71 8c 83 cf c7 29 e5 a5 4c 74 ef a9 c2 bf 52 ef
0a 89 c1 85 d9 f9 c5 e5 d8 f7 f7 fb 56 7b 11 14	67 a7 78 97 35 99 a6 d9 61 68 68 0f b1 21 82 fa	67 a7 78 97 99 a6 d9 35 68 0f 61 68 fa b1 21 82	ec 1a c0 80 0c 50 53 c7 3b d7 00 ef b7 22 72 e0	37 bb 38 f7 14 3d d8 7d 93 e7 08 a1 48 f7 a5 4a
db a1 f8 77 18 6d 8b ba a8 30 08 4e ff d5 d7 aa	b9 32 41 f5 ad 3c 3d f4 c2 04 30 2f 16 03 0e ac	b9 32 41 f5 3c 3d f4 ad 30 2f c2 04 ac 16 03 0e	b1 1a 44 17 3d 2f ec b6 0a 6b 2f 42 9f 68 f3 b1	48 f3 cb 3c 26 1b c3 be 45 a2 aa 0b 20 d7 72 38
f9 e9 8f 2b 1b 34 2f 08 4f c9 85 49 bf bf 81 89	99 1e 73 f1 af 18 15 30 84 dd 97 3b 08 08 0c a7	99 1e 73 f1 18 15 30 af 97 3b 84 dd a7 08 08 0c	31 30 3a c2 ac 71 8c c4 46 65 48 eb 6a 1c 31 62	fd 0e c5 f9 0d 16 d5 6b 42 e0 4a 41 cb 1c 6e 56
cc 3e ff 3b a1 67 59 af 04 85 02 aa a1 00 5f 34	4b b2 16 e2 32 85 cb 79 f2 97 77 ac 32 63 cf 18	4b b2 16 e2 85 cb 79 32 77 ac f2 97 18 32 63 cf	4b 86 8a 36 b1 cb 27 5a fb f2 f2 af cc 5a 5b cf	b4 ba 7f 86 8e 98 4d 26 f3 13 59 18 52 4e 20 76
ff 08 69 64 0b 53 34 14 84 bf ab 8f 4a 7c 43 b9				

Avalanche Effect in AES: Change in Plaintext

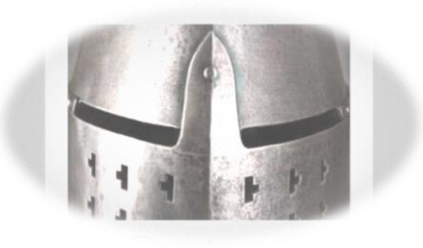
Round		Number of Bits that Differ
	0123456789abcdeffedcba9876543210 0023456789abcdeffedcba9876543210	1
0	0e3634aece7225b6f26b174ed92b5588 0f3634aece7225b6f26b174ed92b5588	1
1	657470750fc7ff3fc0e8e8ca4dd02a9c c4a9ad090fc7ff3fc0e8e8ca4dd02a9c	20
2	5c7bb49a6b72349b05a2317ff46d1294 fe2ae569f7ee8bb8c1f5a2bb37ef53d5	58
3	7115262448dc747e5cdac7227da9bd9c ec093dfb7c45343d689017507d485e62	59
4	f867aee8b437a5210c24c1974cffeabc 43efdb697244df808e8d9364ee0ae6f5	61
5	721eb200ba06206dcbd4bce704fa654e 7b28a5d5ed643287e006c099bb375302	68
6	0ad9d85689f9f77bc1c5f71185e5fb14 3bc2d8b6798d8ac4fe36a1d891ac181a	64
7	db18a8ffa16d30d5f88b08d777ba4eaa 9fb8b5452023c70280e5c4bb9e555a4b	67
8	f91b4fbfe934c9bf8f2f85812b084989 20264e1126b219aef7feb3f9b2d6de40	65
9	cca104a13e678500ff59025f3bafaa34 b56a0341b2290ba7dfdfbddcd8578205	61
10	ff0b844a0853bf7c6934ab4364148fb9 612b89398d0600cde116227ce72433f0	58

Avalanche Effect in AES: Change in Key

Round		Number of Bits that Differ
	0123456789abcdeffedcba9876543210 0123456789abcdeffedcba9876543210	0
0	0e3634aece7225b6f26b174ed92b5588 0f3634aece7225b6f26b174ed92b5588	1
1	657470750fc7ff3fc0e8e8ca4dd02a9c c5a9ad090ec7ff3fc1e8e8ca4cd02a9c	22
2	5c7bb49a6b72349b05a2317ff46d1294 90905fa9563356d15f3760f3b8259985	58
3	7115262448dc747e5cdac7227da9bd9c 18aeb7aa794b3b66629448d575c7cebf	67
4	f867aee8b437a5210c24c1974cffeabc f81015f993c978a876ae017cb49e7eec	63
5	721eb200ba06206dcbd4bce704fa654e 5955c91b4e769f3cb4a94768e98d5267	81
6	0ad9d85689f9f77bc1c5f71185e5fb14 dc60a24d137662181e45b8d3726b2920	70
7	db18a8ffa16d30d5f88b08d777ba4eaa fe8343b8f88bef66cab7e977d005a03c	74
8	f91b4fbfe934c9bf8f2f85812b084989 da7dad581d1725c5b72fa0f9d9d1366a	67
9	cca104a13e678500ff59025f3bafaa34 0ccb4c66bbfd912f4b511d72996345e0	59
10	ff0b844a0853bf7c6934ab4364148fb9 fc8923ee501a7d207ab670686839996b	53

Implementation Aspects

- AddRoundKey is a bytewise XOR operation
- ShiftRows is a simple byte-shifting operation
- SubBytes operates at the byte level and only requires a table of 256 bytes
- MixColumns requires matrix multiplication
- MixColumns only requires multiplication by {02} and {03}, which can be converted to shifts and XORs.
- Designers believe this very efficient implementation was a key factor in its selection as the AES cipher.



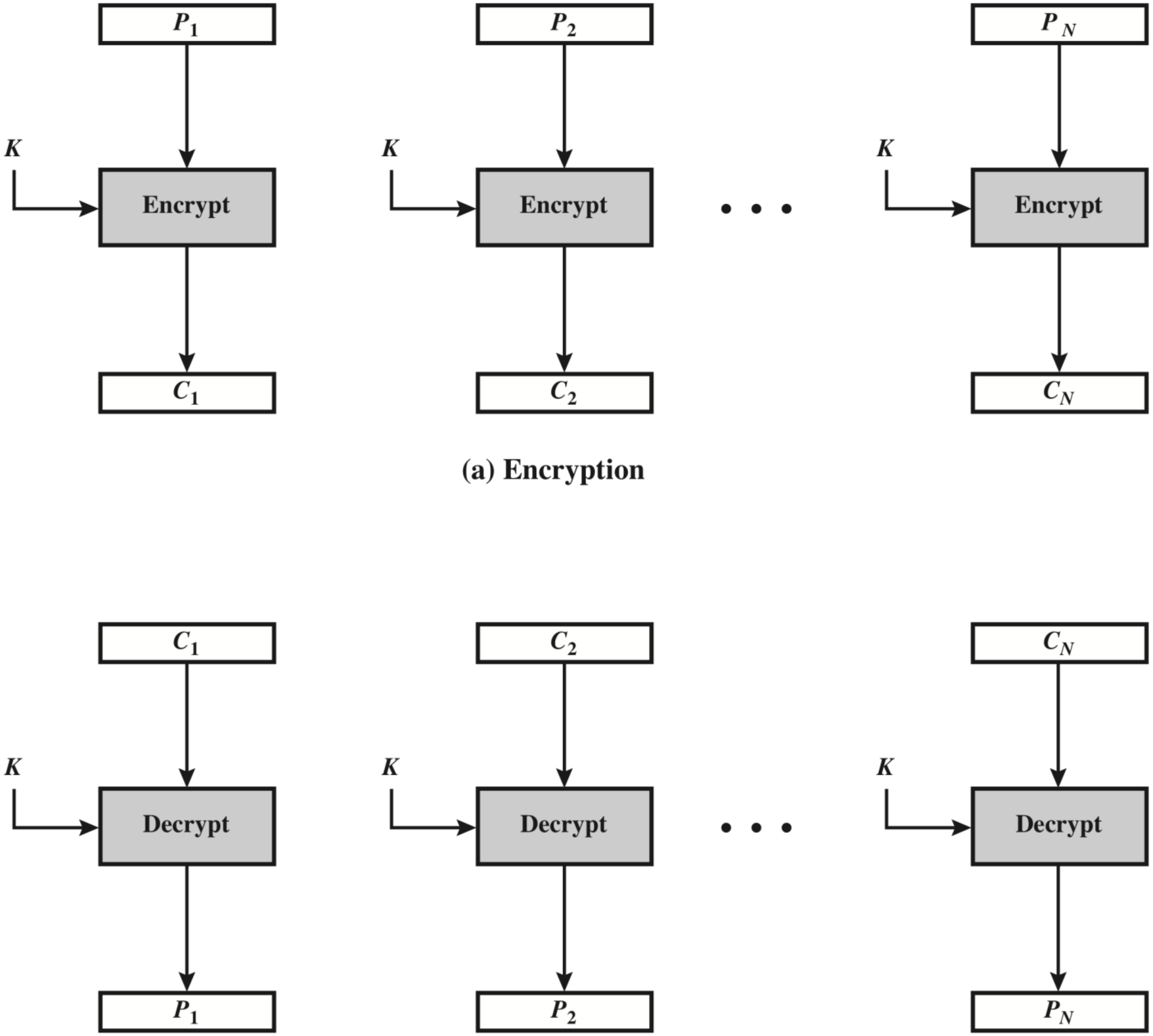
Modes of Operation

Modified by: Dr. Ramzi Saifan

Modes of Operation

- To apply a block cipher in a variety of applications, five *modes of operation* have been defined by NIST.
  - The five modes are intended to cover a wide variety of applications of encryption for which a block cipher could be used
  - These modes are intended for use with any symmetric block cipher, including triple DES and AES

Electronic Codebook Mode (ECB)



(a) Encryption

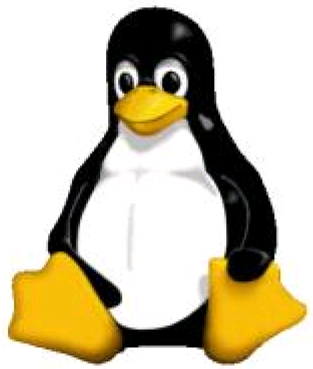
(b) Decryption

$C_i = E_K(P_i)$; the cipher text is $(C_1, \ldots, C_n)$

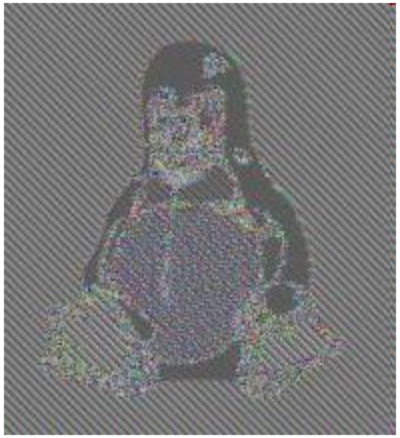
Security?

- ECB should *not* be used
  - Why?

The effect of ECB mode



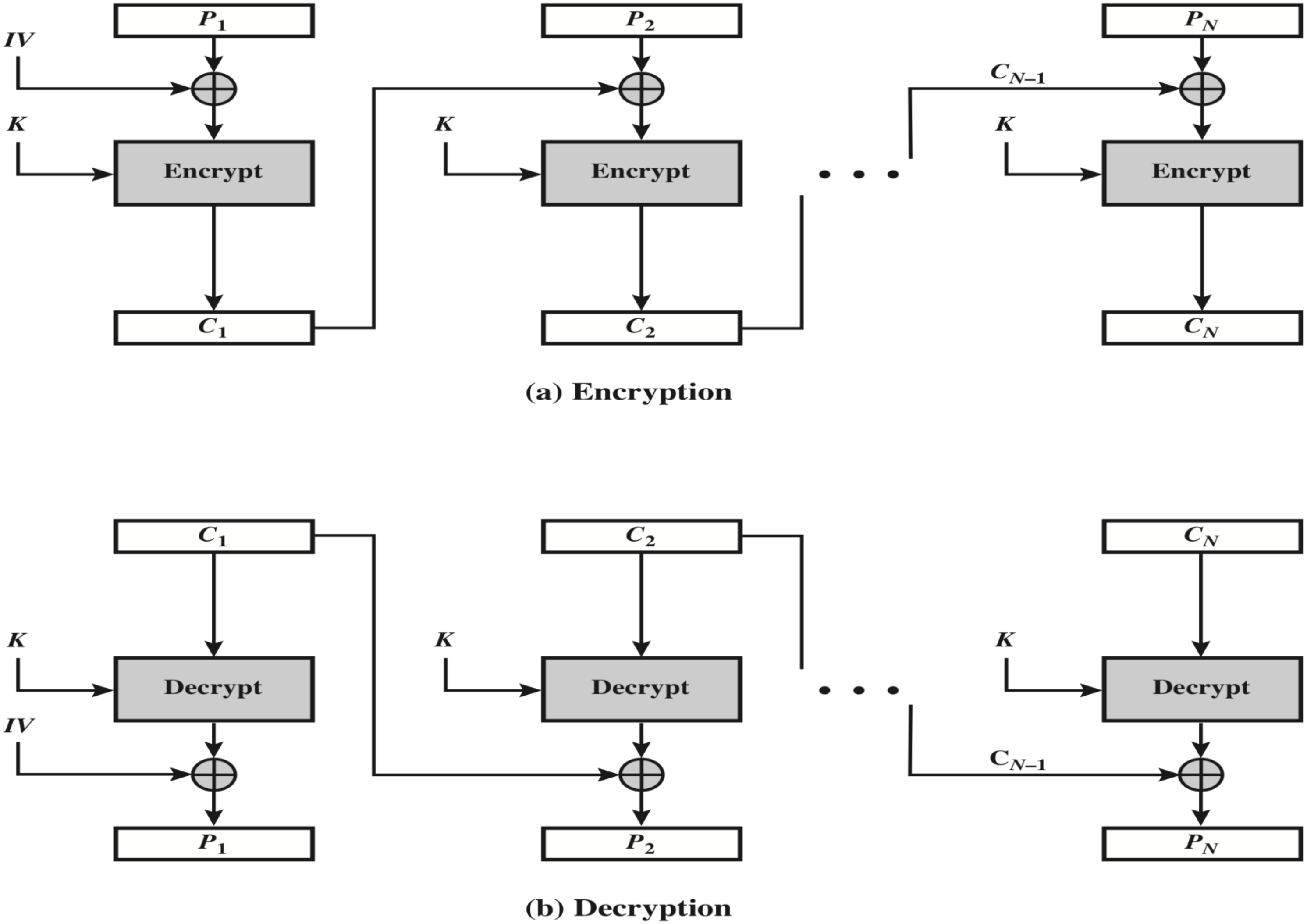
original



encrypted using ECB mode

*Images from Wikipedia

Cipher Block Chaining (CBC)



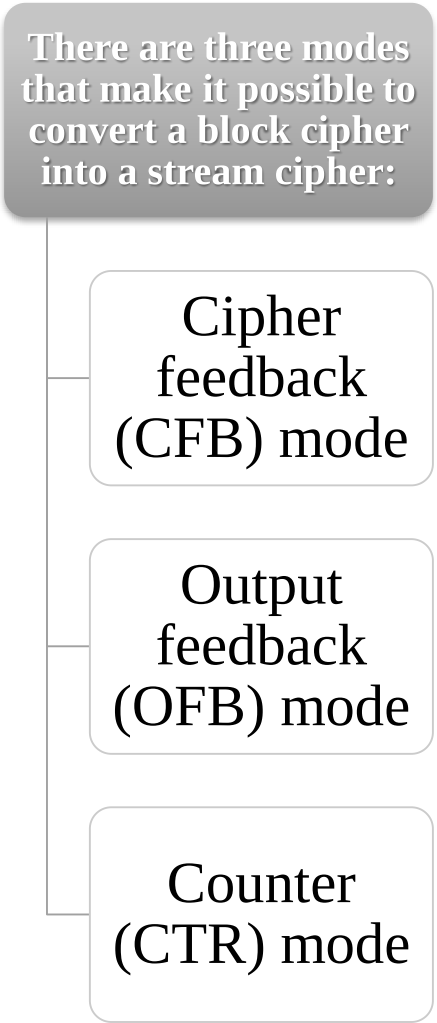
(a) Encryption

(b) Decryption

IV; $C_i = E_K(m_i \oplus C_{i-1})$; the ciphertext is $(IV, C_1, \ldots, C_n)$

Cipher Feedback Mode

- For AES, DES, or any block cipher, encryption is performed on a block of *b* bits
  - In the case of DES *b*= 64
  - In the case of AES *b*= 128



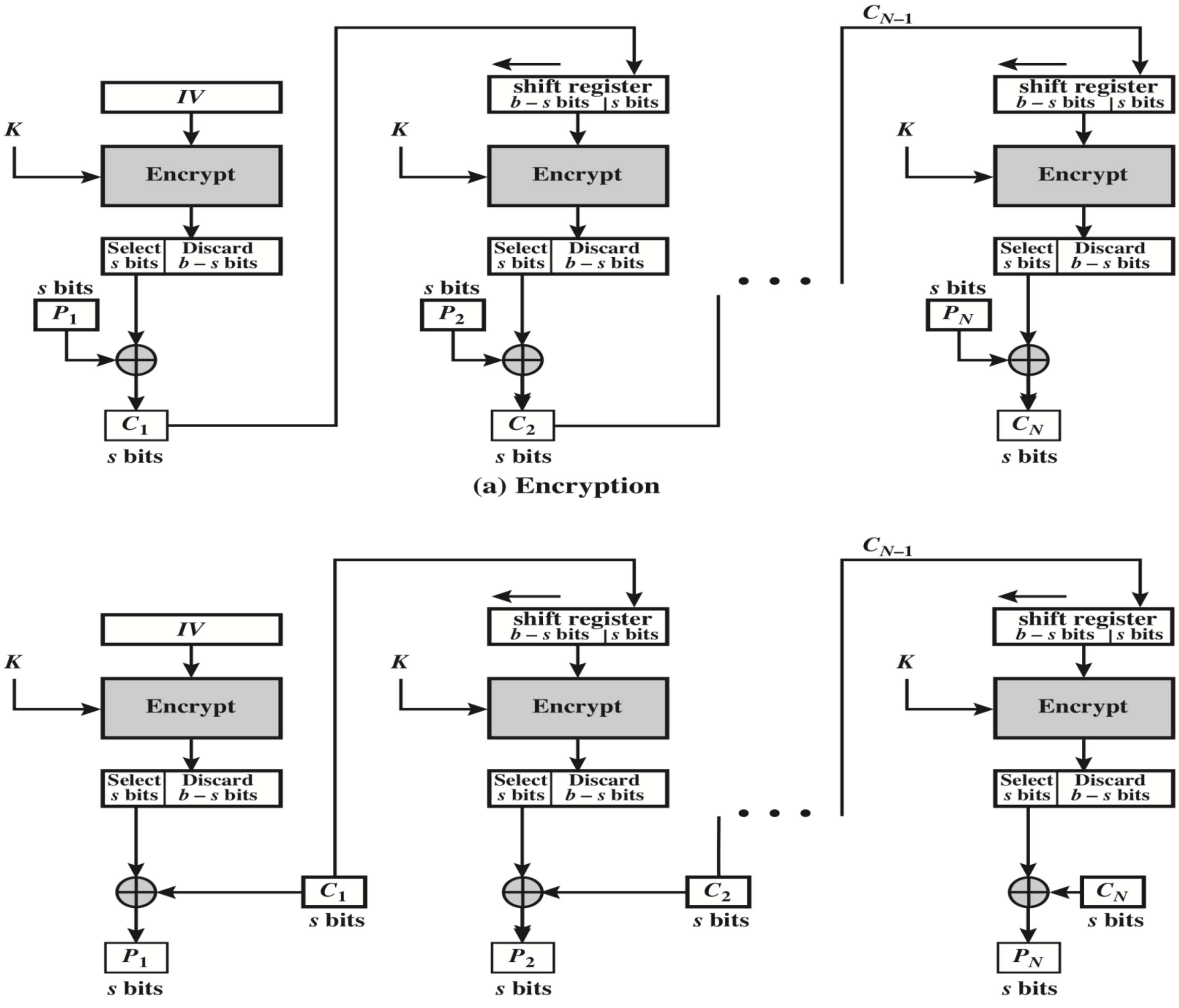
There are three modes that make it possible to convert a block cipher into a stream cipher:

Cipher feedback (CFB) mode

Output feedback (OFB) mode

Counter (CTR) mode

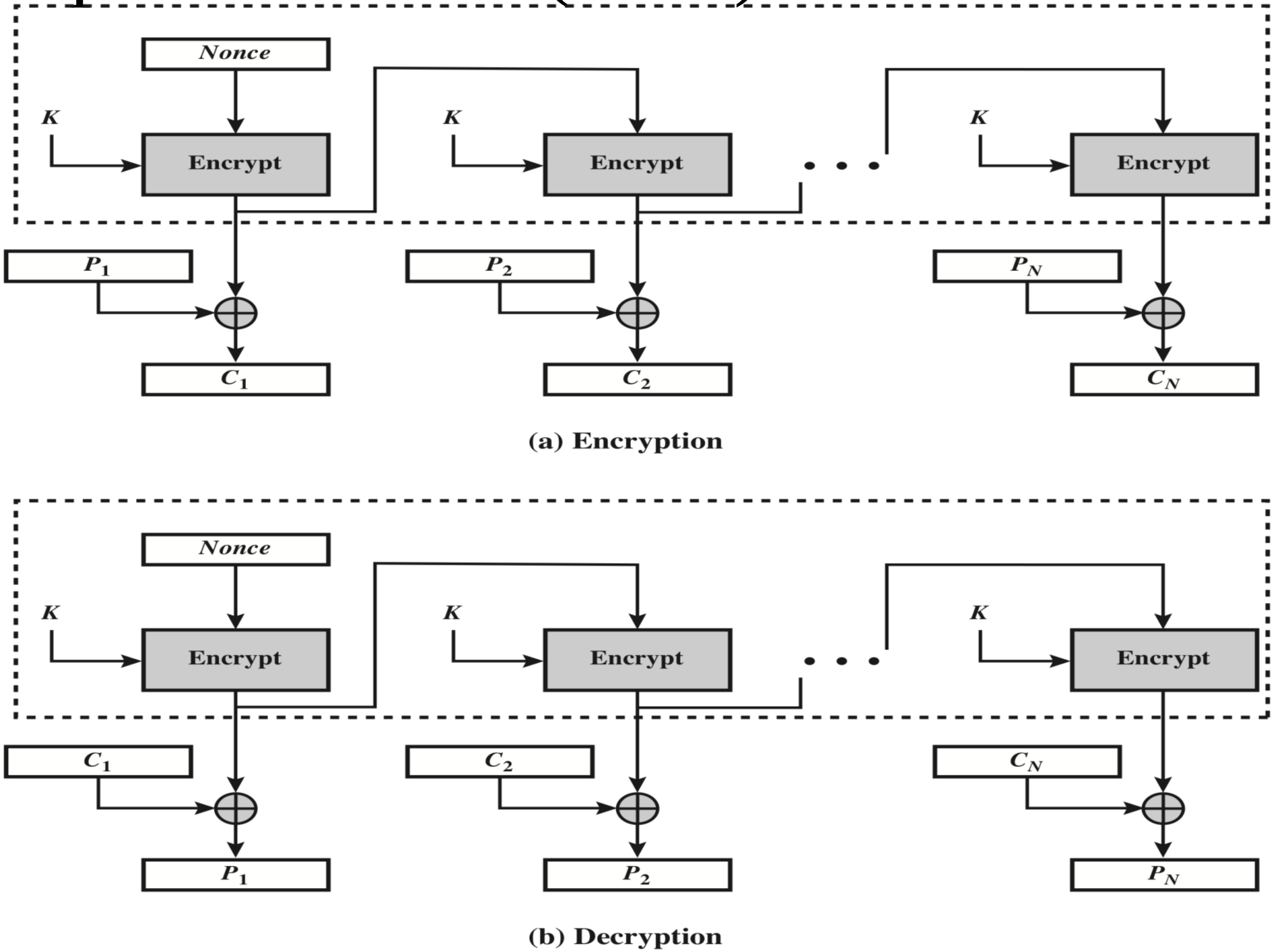
s-bit Cipher Feedback (CFB) Mode



(a) Encryption

(b) Decryption

Output Feedback (OFB)Mode

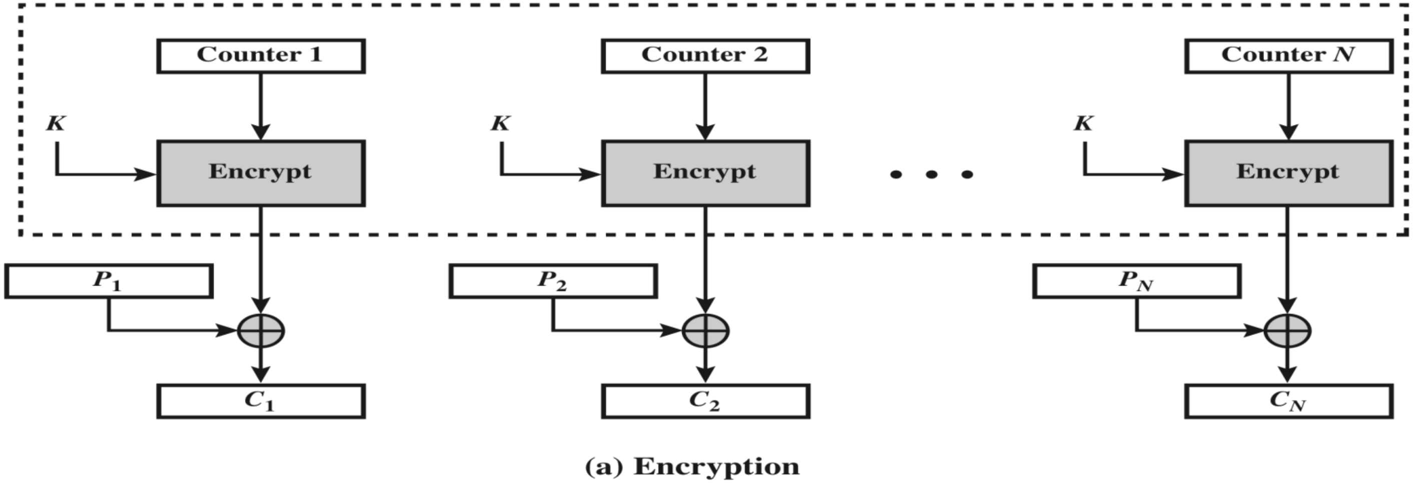


(a) Encryption

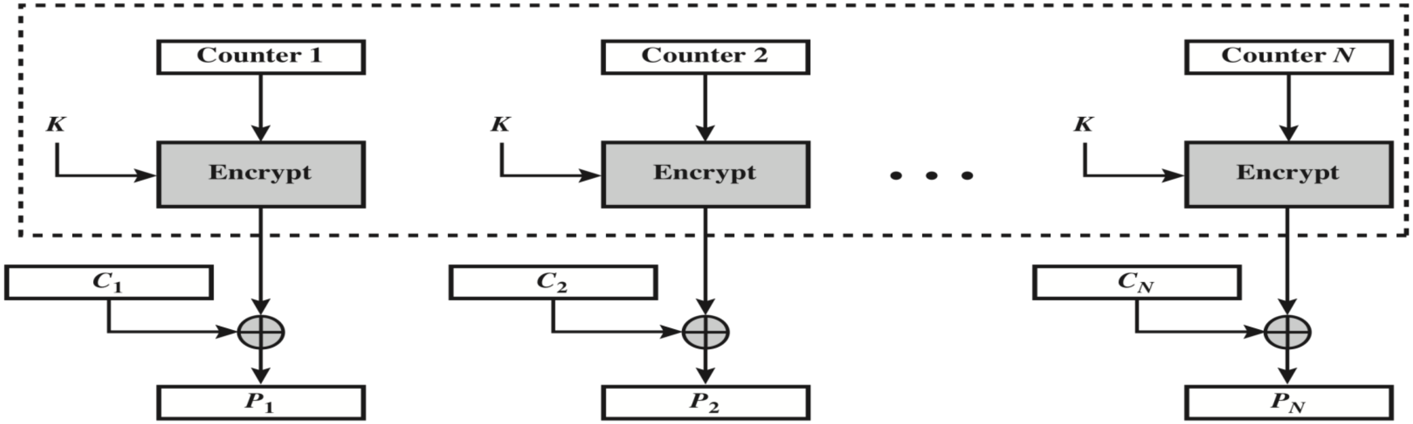
(b) Decryption

Nonce; $z_i = E_K(z_{i-1})$; $C_i = z_i \oplus m_i$; the ciphertext is (Nonce, $C_1, \ldots, C_n$)

Counter (CTR) Mode



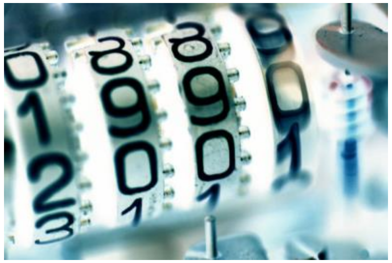
(a) Encryption



(b) Decryption

Counter1; $z_i = F_K(IV+i)$; $C_i = z_i \oplus m_i$; the ciphertext is (Counter1, C_1, .., C_n)

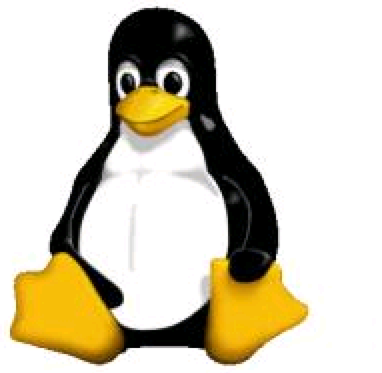
Advantages of CTR

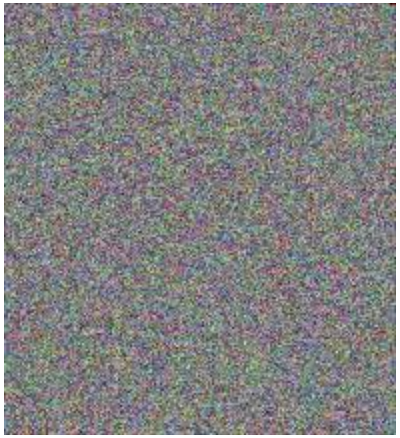


- Hardware efficiency
- Software efficiency
- Preprocessing
- Random access
- Provable security
- Simplicity

Security

- CBC, OFB, and CTR modes *are* secure against chosen-plaintext attacks





*Images from Wikipedia

Table 6.1 Block Cipher Modes of Operation

Mode	Description	Typical Application
Electronic Codebook (ECB)	Each block of plaintext bits is encoded independently using the same key.	•Secure transmission of single values (e.g., an encryption key)
Cipher Block Chaining (CBC)	The input to the encryption algorithm is the XOR of the next block of plaintext and the preceding block of ciphertext.	•General-purpose block-oriented transmission •Authentication
Cipher Feedback (CFB)	Input is processed *s* bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext.	•General-purpose stream-oriented transmission •Authentication
Output Feedback (OFB)	Similar to CFB, except that the input to the encryption algorithm is the preceding encryption output, and full blocks are used.	•Stream-oriented transmission over noisy channel (e.g., satellite communication)
Counter (CTR)	Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.	•General-purpose block-oriented transmission •Useful for high-speed requirements

Data Integrity

Modified by: Dr. Ramzi Saifan

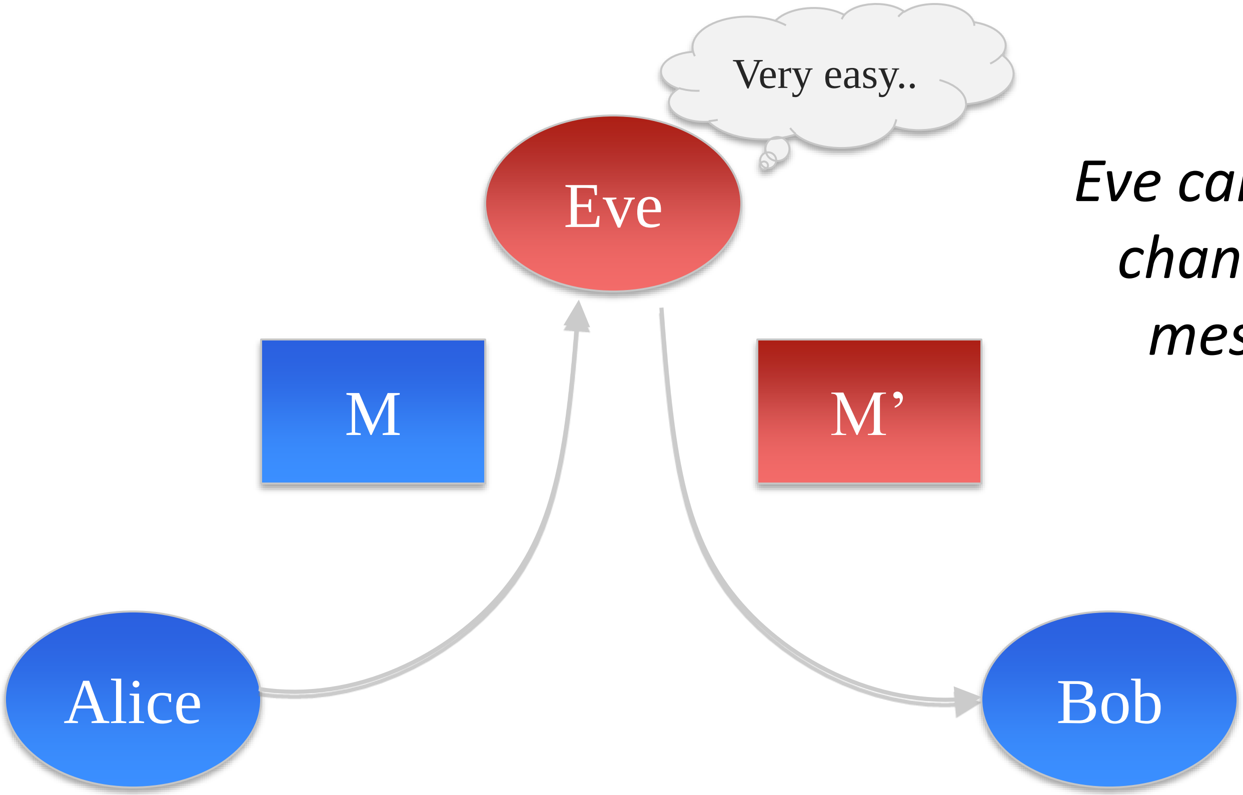
Encryption/Decryption

- Provides message confidentiality.
- Does it provide message authentication?

Message Authentication

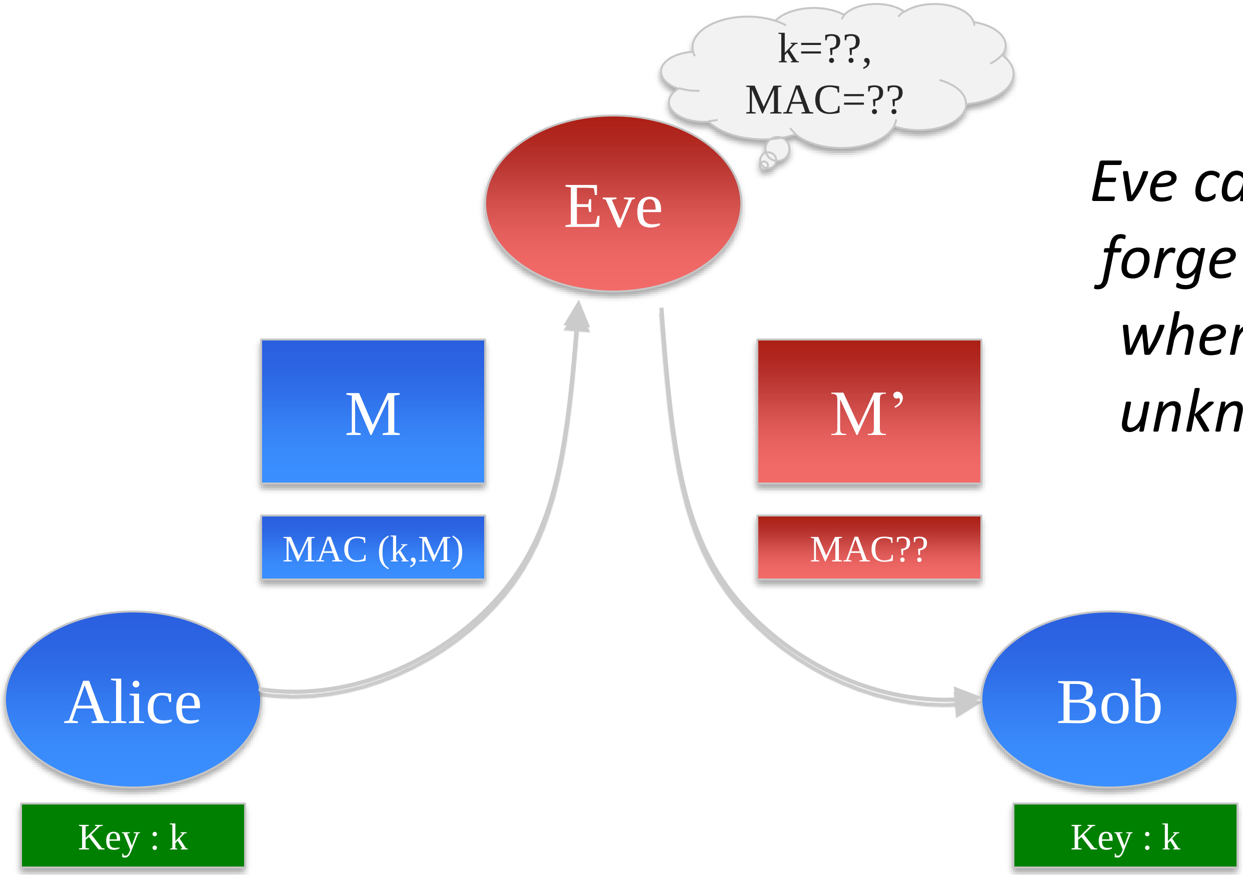
- Bob receives a message *m* from Alice, he wants to know
  - (Data origin authentication) whether the message was really sent by Alice;
  - (Data integrity) whether the message has been modified.
- Solutions:
  - Alice attaches a message authentication code (MAC) to the message.
  - Or she attaches a digital signature to the message.

Communication without authentication



Eve can simply change the message

Integrity Protection with MAC

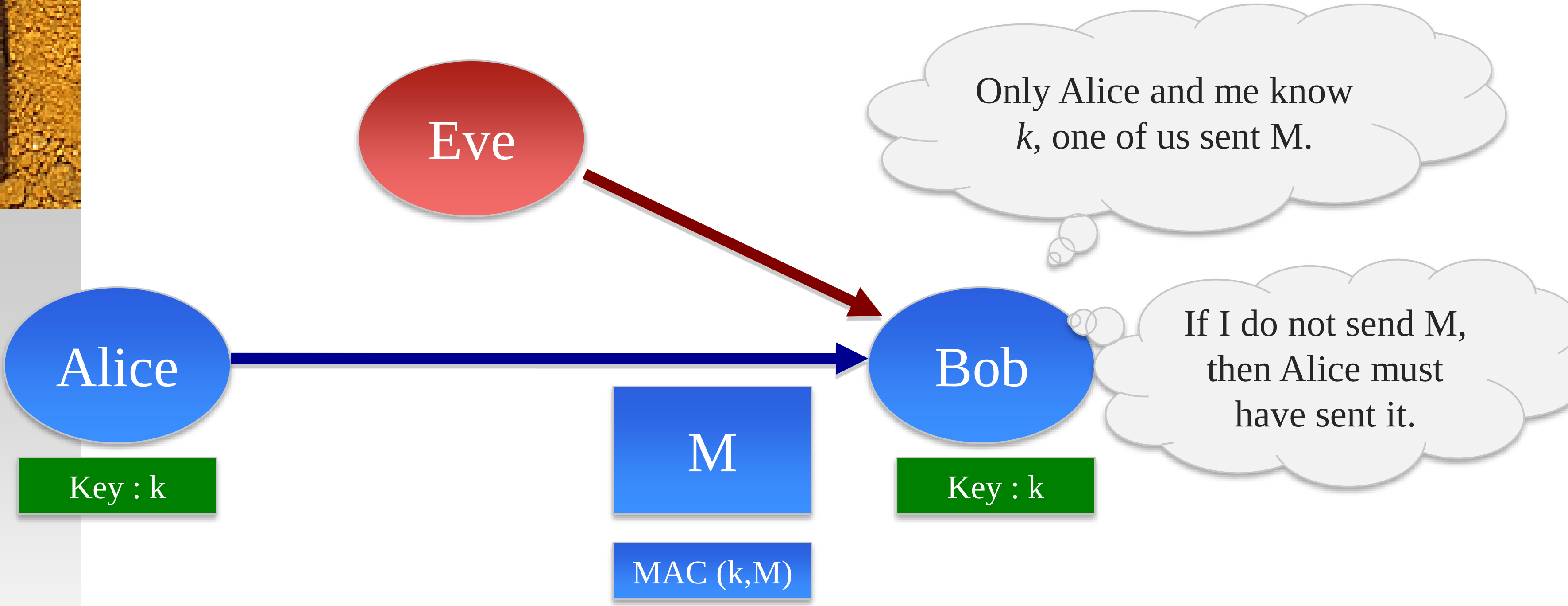


Eve can not forge MAC when k is unknown

Shared key *k* to generate authenticate message

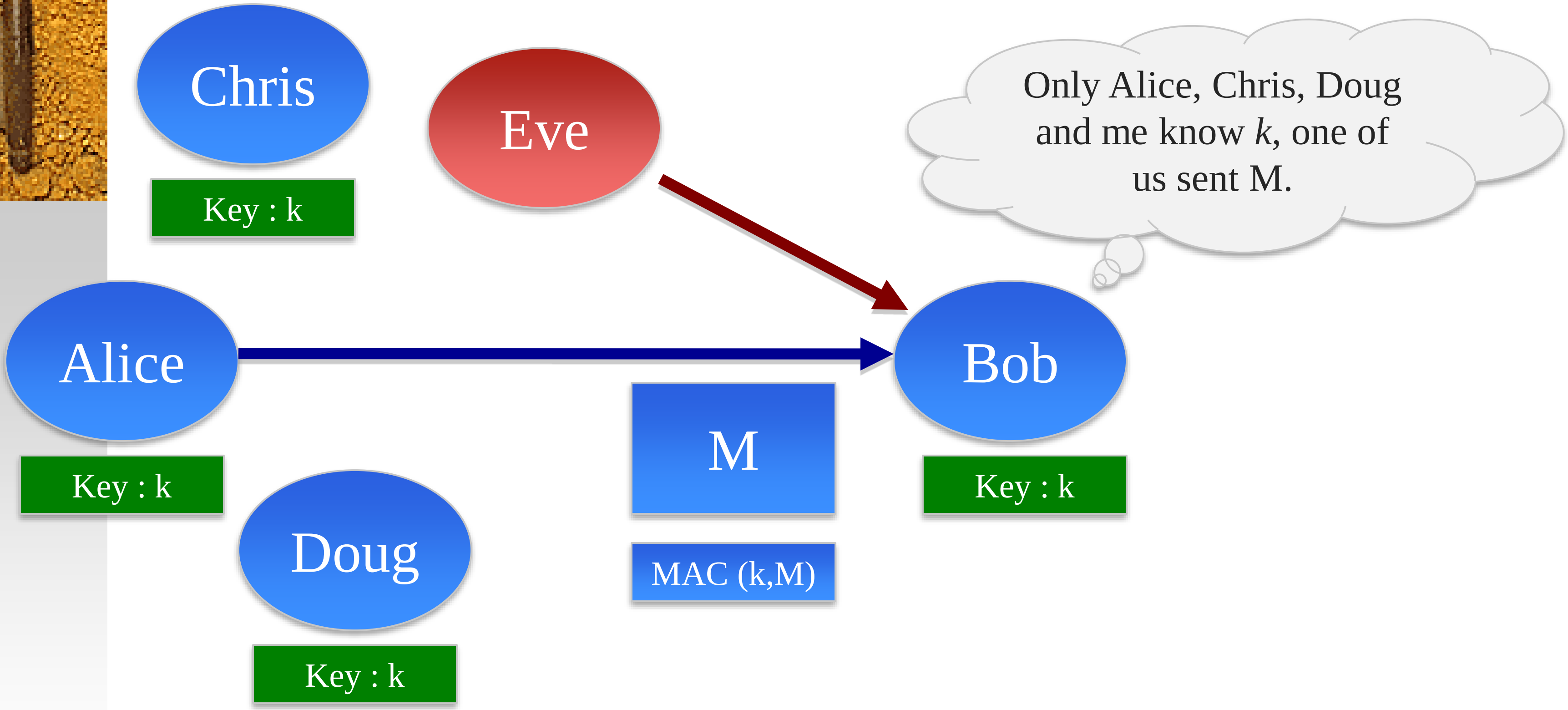
MAC Authentication (I)

- MAC allows **two** or more mutually trusting parties to authenticate messages sent between members

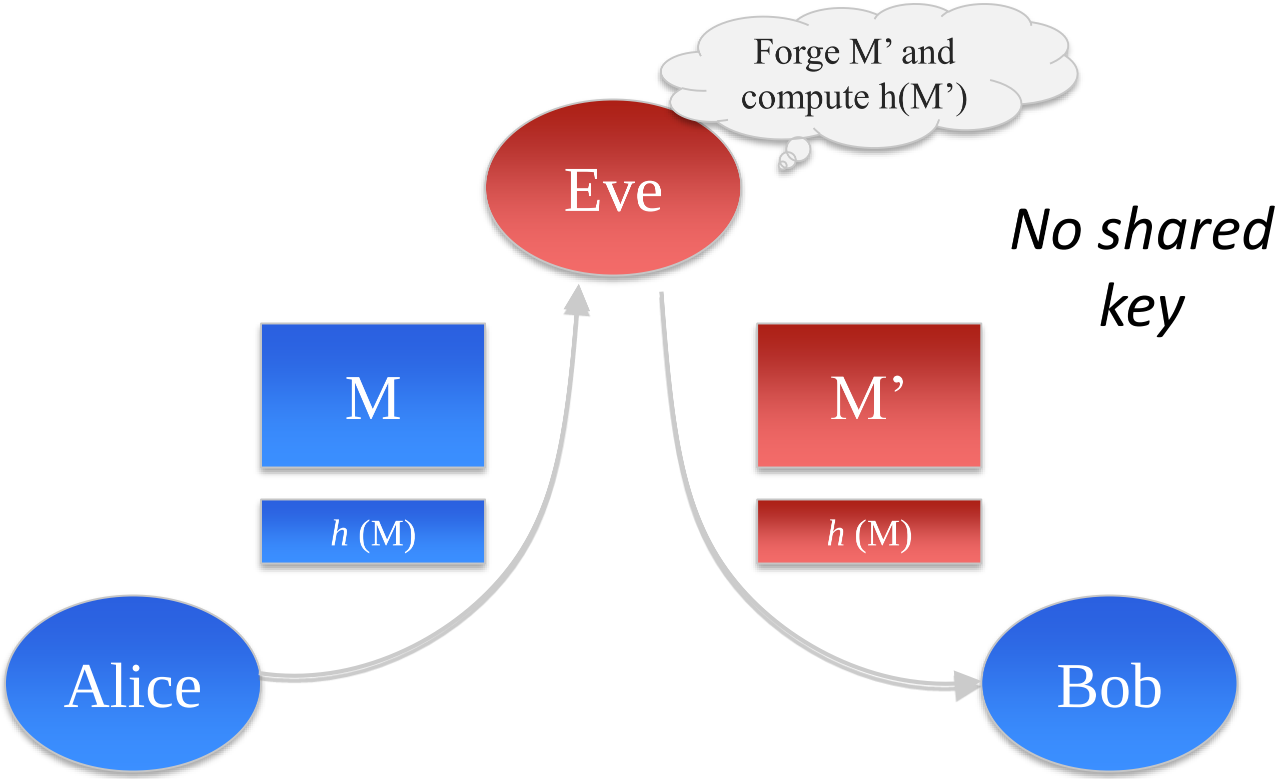

Eve
Only Alice and me know *k*, one of us sent M.
Alice
Bob
If I do not send M, then Alice must have sent it.
Key : k
M
Key : k
MAC (k,M)

MAC Authentication (II)

- MAC allows two or **more** mutually trusting parties to authenticate messages sent between members



Integrity with Hash



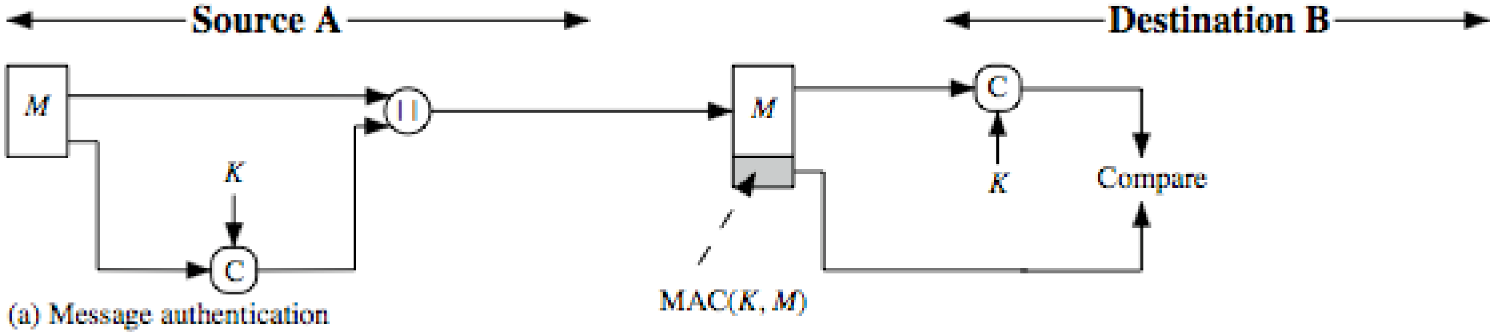
Can we simply send the hash with the message to serve message authentication ?

Ans: No, Eve can change the message and recompute the hash.

Using hash needs more appropriate procedure to guarantee integrity

Message Authentication Code

- A function of the message and a secret key that produces a fixed-length value that serves as the authenticator
- Generated by an algorithm :
  - generated from message + secret key : **MAC = F(K,M)**
  - A small fixed-sized block of data
  - appended to message as a **signature** when sent
- Receiver performs same computation on message and checks it matches the MAC



(a) Message authentication

MAC and Encryption

- ➢ As shown the MAC provides authentication
- ➢ But encryption can also provides authentication!
- ➢ Why use a MAC?
  - ● sometimes **only authentication is needed**
  - ● sometimes need authentication to persist longer than the encryption (eg. archival use)

MAC Properties

- A MAC is a cryptographic hash

$$MAC = C_K(M)$$

  - condenses a variable-length message M
  - using a secret key K
  - to a fixed-sized authenticator
- A many-to-one function
  - potentially many messages have same MAC
  - but finding these needs to be very difficult

	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

Keyed Hash Functions as MACs

- Want a MAC based on a hash function
  - because hash functions are generally faster
  - crypto hash function code is widely available
  - But **hashing is internally has no key!**
- Original proposal:

```
KeyedHash = Hash(Key|Message)
```

  - some weaknesses were found with this
- Eventually led to development of HMAC

Security requirements

- Pre-image: if $h(m) = y,\ m$ is a pre-image of y.
- Each hash value typically has multiple pre-images.
- Collision: a pair of $(m, m'),\ m \neq m',$ s.t. $h(m) = h(m')$.

A hash function is said to be:

- Pre-image resistant if it is computationally infeasible to find a pre-image of a hash value.
- Collision resistant if it is computationally infeasible to find a collision.
- A hash function is a cryptographic hash function if it is collision resistant.

To be useful for message authentication, a hash function H must have the following properties:

- Can be applied to a block of data of any size
- Produces a fixed-length output
- H(x) is relatively easy to compute for any given x
- One-way or pre-image resistant
  - Computationally infeasible to find x such that H(x) = h
- Computationally infeasible to find y ≠ x such that H(y) = H(x)
- Collision resistant or strong collision resistance
  - Computationally infeasible to find any pair (x,y) such that H(x) = H(y)

Birthday Problem

- Birthday problem: In a group of k people, what is the probability that at least two people have the same birthday?

  □Having the same birthday is a collision?

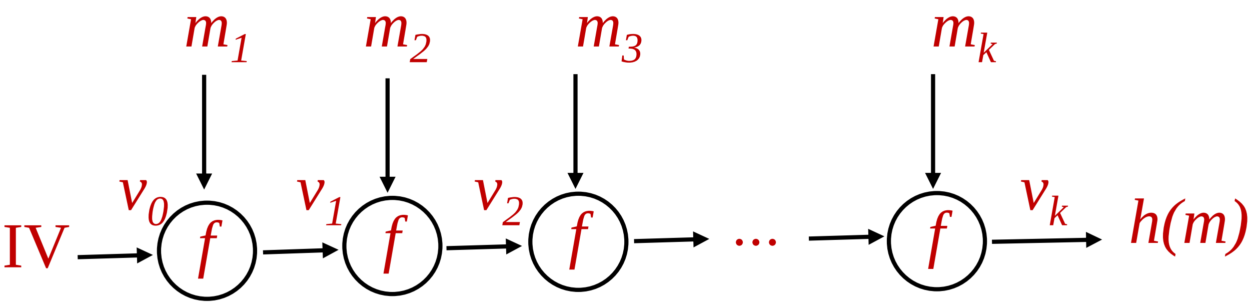
- Birthday paradox: $p \ge 1/2$ with k as small as 23.

- Consider a hash function $h:\{0,1\}^* \to \{0,1\}^n$.
- If we randomly generate k messages, the probability of having a collision depends on n.
- To resist birthday attack, we choose n to be sufficiently large that it will take an infeasibly large k to have a non-negligible probability of collision.

Collision-resistant hash functions

- Collision-resistant hash functions can be built from collision-resistant compression functions using Merkle-Damgard construction.

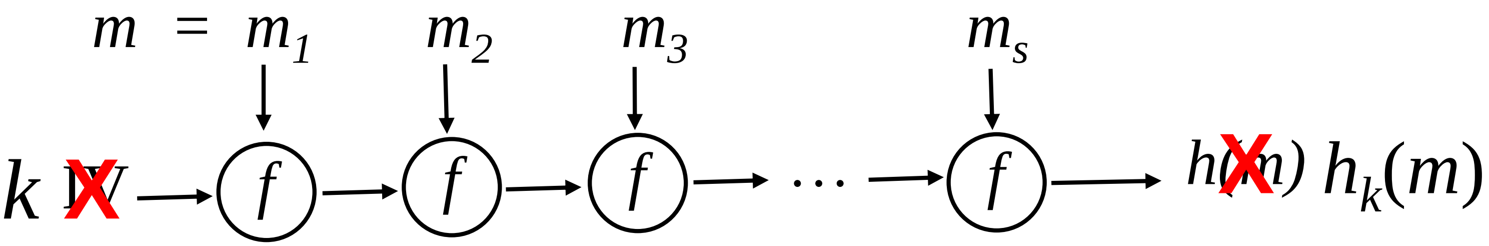
Merkle-Damgard Construction



Compression function $f:\{0,1\}^{n+b} \rightarrow \{0,1\}^n$

- Insecure: $MAC_k(m) = h(m)$ with IV = k.

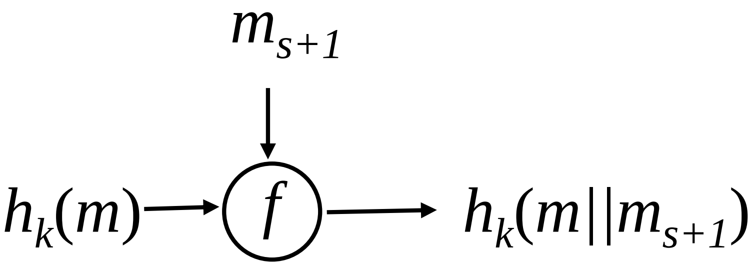
  (For simplicity, without padding)



- Easy to forge:

  $(m', h_k(m'))$,

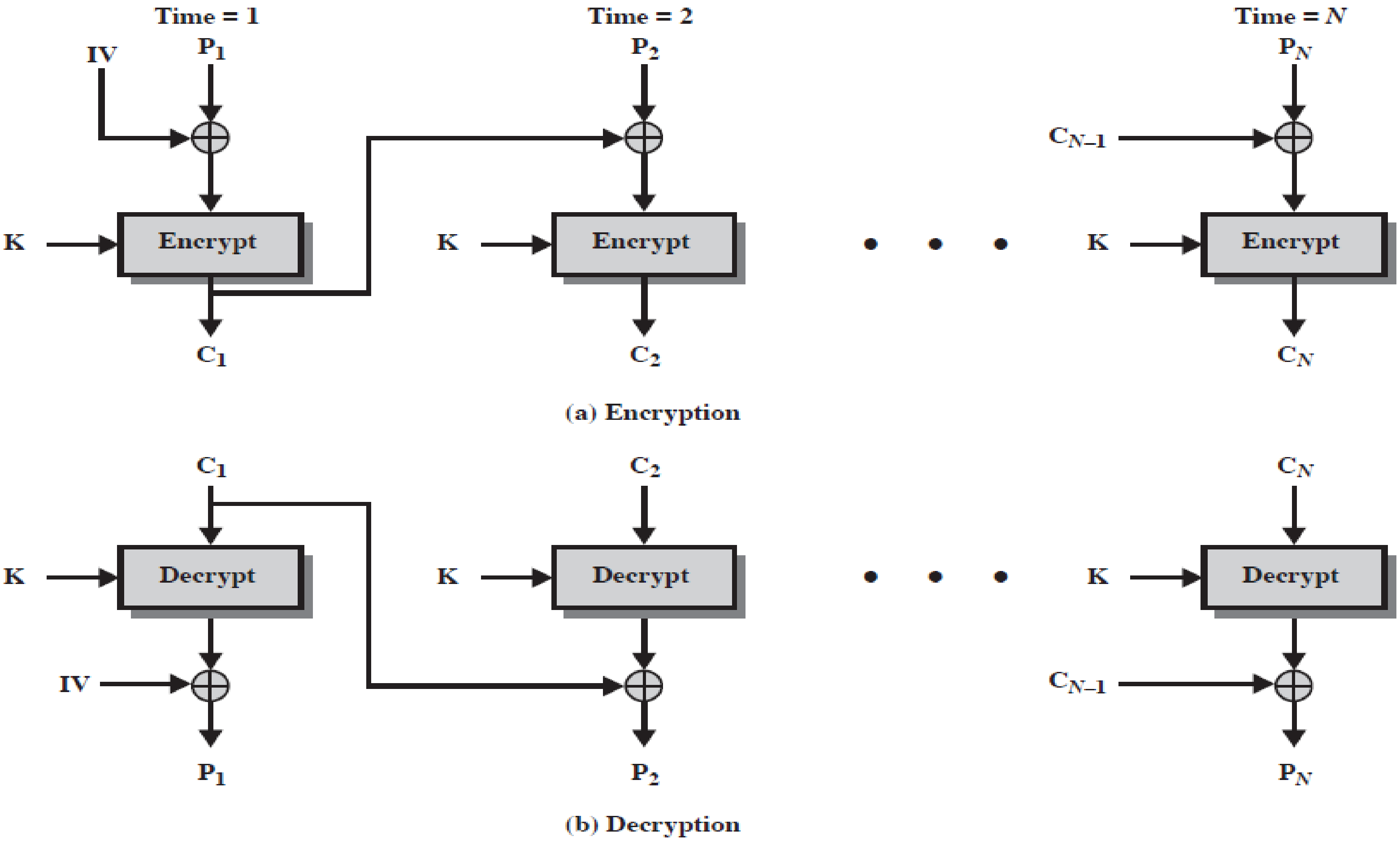
  where $m' = m \square m_{s+1}$



CMAC (Cipher-based MAC)

- “Hashless” MAC
  - Uses an encryption algorithm (DES, AES, etc.) to generate MAC
  - Based on same idea as cipher block chaining
- Compresses result to size of single block (unlike encryption

CBC CMAC Overview



(a) Encryption

(b) Decryption

CMAC Facts

- Advantages:
  - Can use existing encryption functions
  - Encryption functions have properties that resist preimage and collision attacks
  - Most exhibit strong avalanche effect – minor change in message gives great change in resulting MAC
- Disadvantage:
  - Encryption algorithms (particularly when chained) can be much slower than hash algorithms

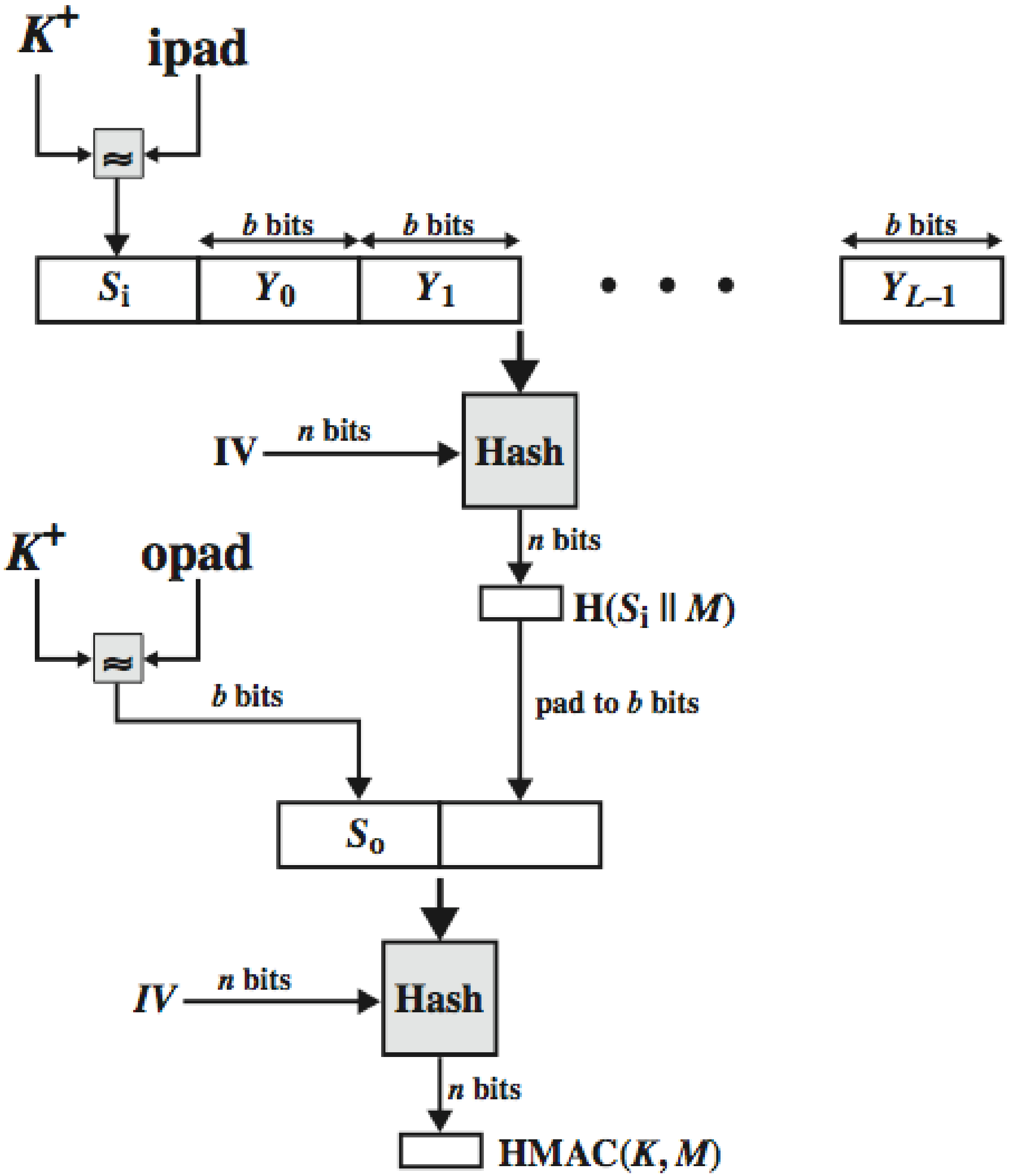
HMAC

- 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 deigned 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

- $HMAC(K,m) = H(\ \ (K' \oplus opad)\ \ ||\ \ H((K' \oplus ipad) \,||\, m\,)\ \ \ )$, where
  - *H* : is a cryptographic hash function, composed of multiple rounds with operations AND, OR, XOR, NOT, and SHIFT. Very efficient to compute.
  - *K:* is the secret key,
  - *M:* is the message to be authenticated,
  - *K'* : is another secret key, derived from the original key *K* (by padding *K* to the right with extra zeroes to the input block size of the hash function, or by hashing *K* if it is longer than that block size,
  - || denotes concatenation,
  - *opad* is the outer padding (0x5c5c5c…5c5c, one-block long constant),and
  - *ipad* is the inner padding (0x363636…3636, one-block long constant).

HMAC


K^+
ipad
≈
b bits
b bits
b bits
S_i
Y_0
Y_1
• • •
Y_{L-1}
IV
n bits
Hash
n bits
$H(S_i \| M)$
K^+
opad
≈
b bits
pad to b bits
S_o
IV
n bits
Hash
n bits
HMAC(K, M)

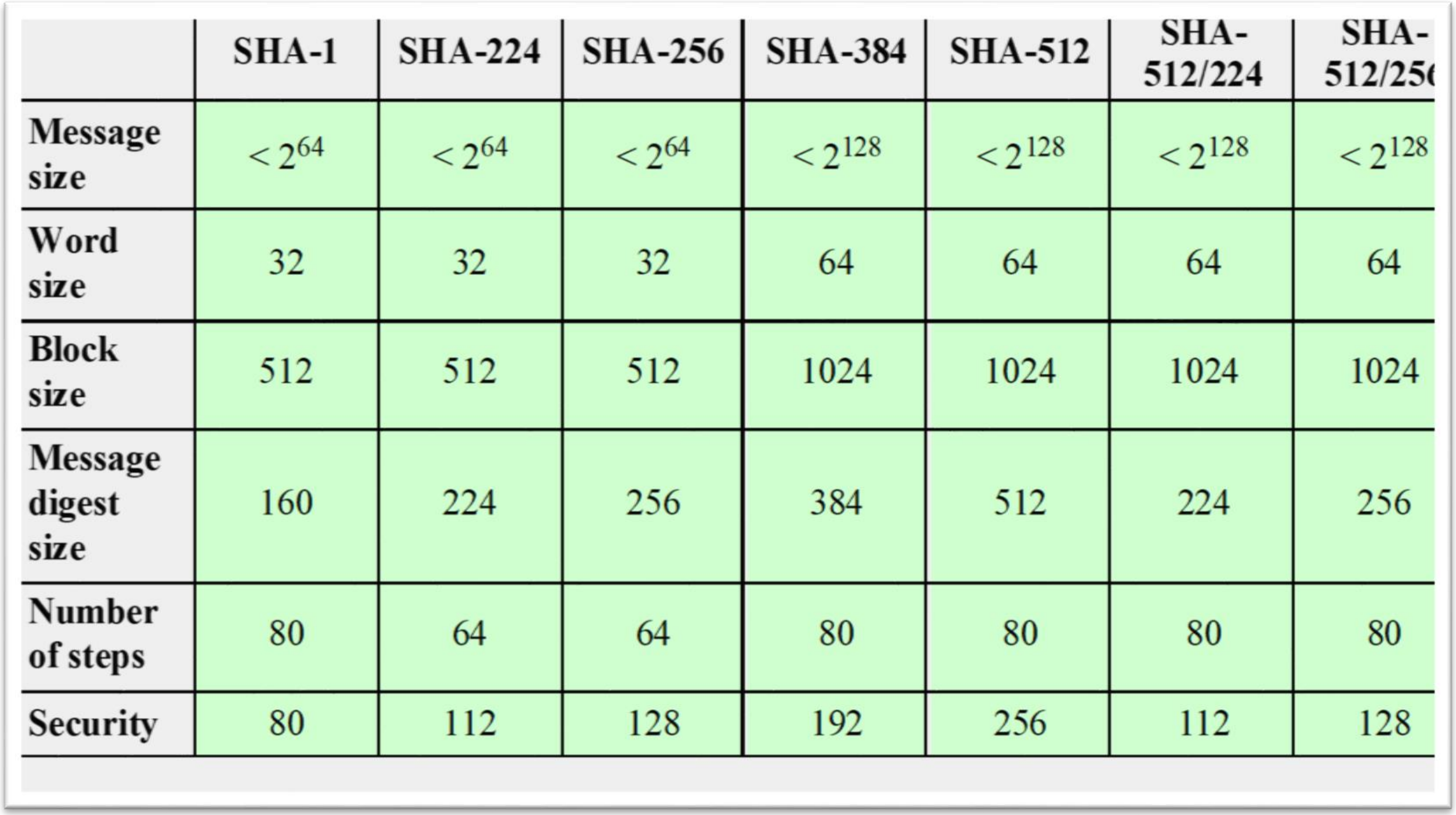
Hash functions in practice

- MD5
  - 128-bit output
  - Introduced in 1991…collision attacks found in 2004…several extensions and improvements since then
  - Still widely deployed(!)
- SHA-1
  - 160-bit output
  - No collisions known, but theoretical attacks exist
- SHA-2
  - 256-/512-bit outputs

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-512/25
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 $2^{n/2}$.

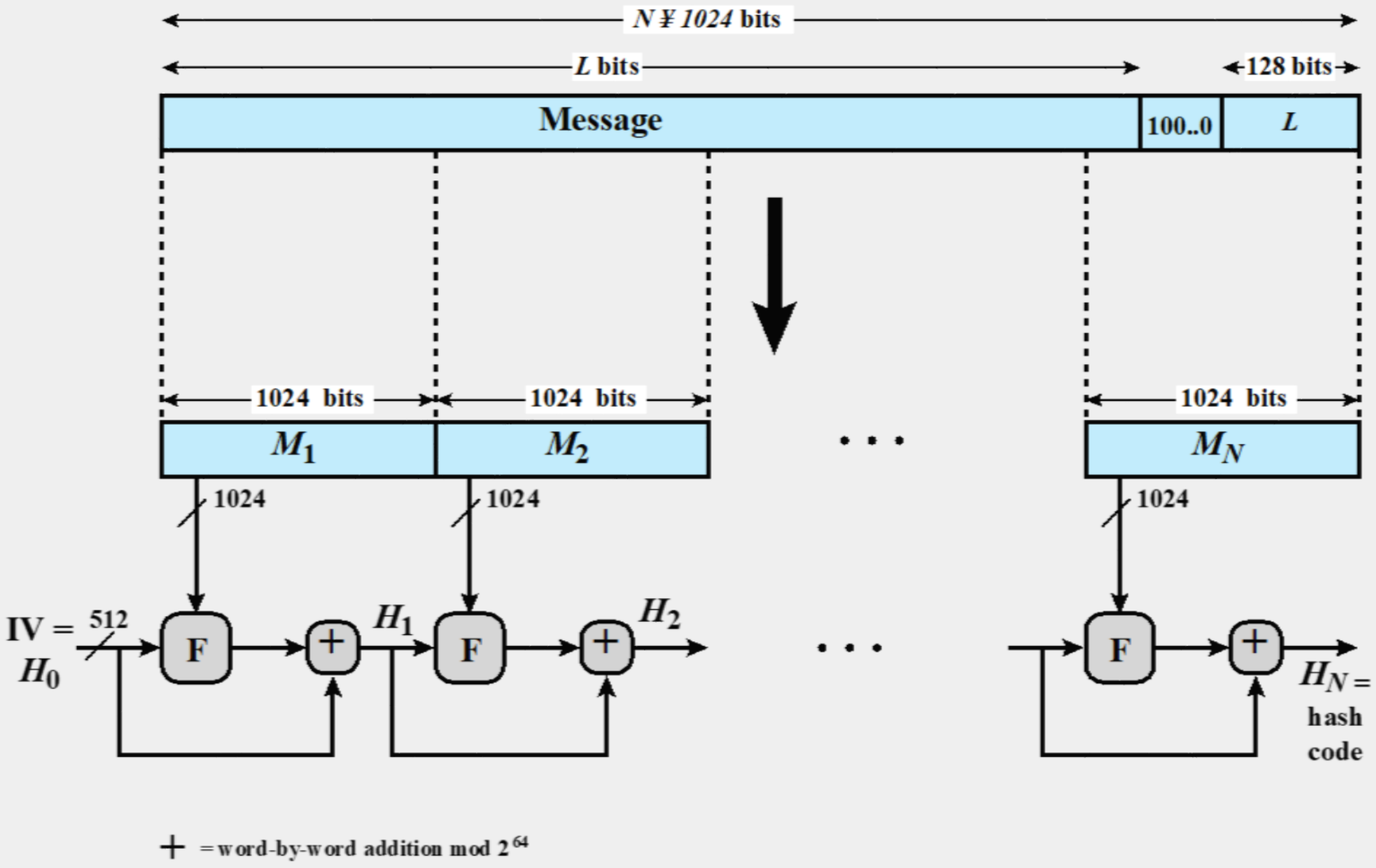


Figure 21.2 Message Digest Generation Using SHA-512

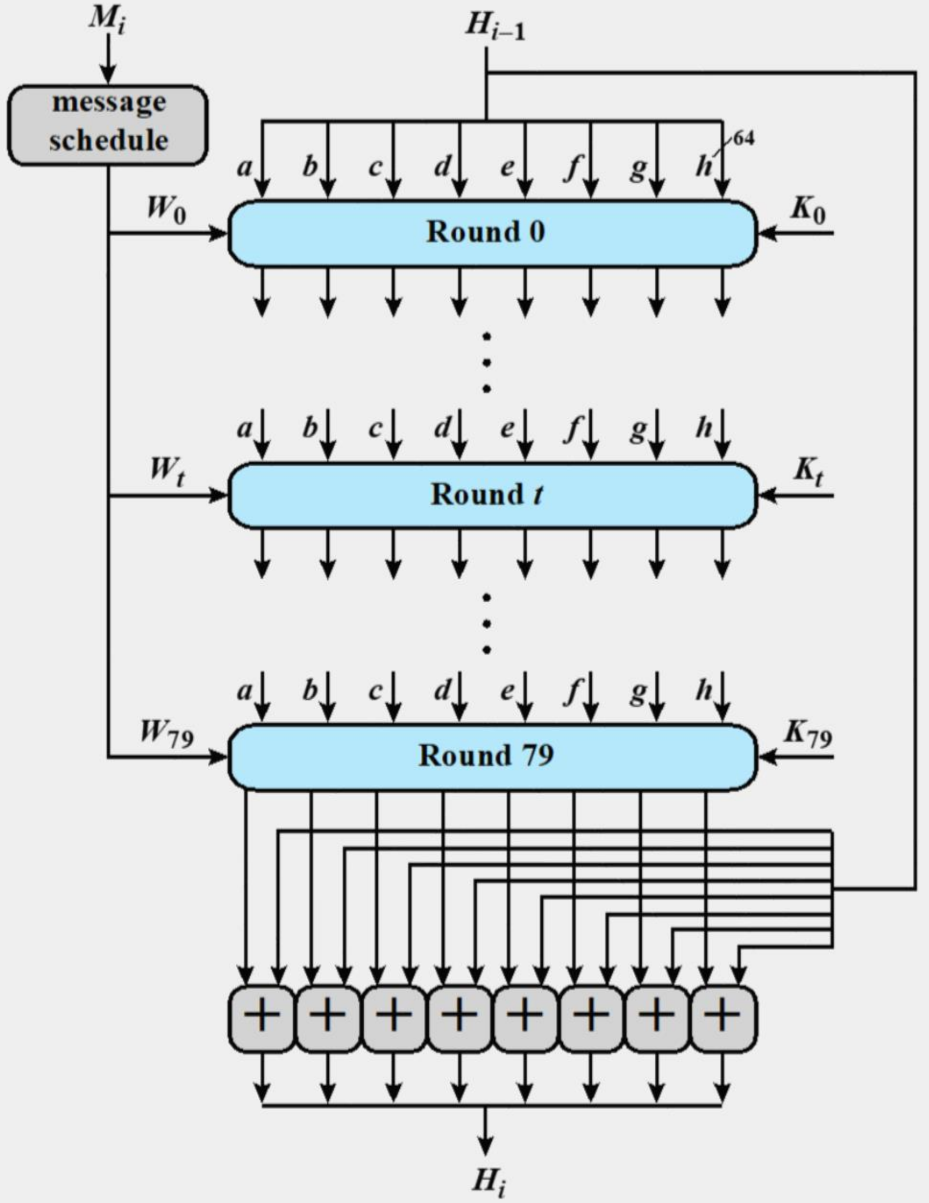


Figure 21.3 SHA-512 Processing of a Single 1024-Bit Block

SHA-3

- SHA-2 shares same structure and mathematical operations as its predecessors and causes concern

- Due to time required to replace SHA-2 should it become vulnerable, NIST announced in 2007 a competition to produce SHA-3

Requirements:

- Must support hash value lengths of 224, 256,384, and 512 bits
- Algorithm must process small blocks at a time instead of requiring the entire message to be buffered in memory before processing it

Hash Function

- The ideal cryptographic hash function has four main properties:
  1) it is quick to compute the hash value for any given message
  2) it is infeasible to generate a message from its hash value except by trying all possible messages
  3) a small change to a message should change the hash value so extensively
  4) it is infeasible to find two different messages with the same hash value

Encryption + integrity

- simultaneously protect confidentiality and authenticity of communications
  - often required but usually separate
- approaches
  - Hash-then-encrypt: $E_K(M \| H(M))$
  - MAC-then-encrypt: $E_{K2}(M \| MAC_{K1}(M))$
  - Encrypt-then-MAC: $(C=E_{K2}(M), T=MAC_{K1}(C)$
  - Encrypt-and-MAC: $(C=E_{K2}(M), T=MAC_{K1}(M)$
- decryption /verification straightforward
- but security vulnerabilities with all these

Replay attacks

- A MAC inherently cannot prevent replay attacks
- Replay attacks must be prevented at a higher level of the protocol!
  - (Note that whether a replay is ok is application-dependent.)
- Replay attacks can be prevented using nonces, timestamps, etc.

Public Key Encryption

Modified by: Dr. Ramzi Saifan

Prime Numbers

- Prime numbers only have divisors of 1 and itself
  - They cannot be written as a product of other numbers
- Any integer a > 1 can be factored in a unique way as

  $$a = {}_{p1}{}^{a1} * p_2{}^{a2} * \ . \ . \ . * p_{p1}{}^{a1}$$

  where $p_1 < p_2 < . . . < p_t$ are prime numbers and where each a_i is a positive integer
- This is known as the fundamental theorem of arithmetic

Table 8.1
Primes Under 2000

2	101	211	307	401	503	601	701	809	907	1009	1103	1201	1301	1409	1511	1601	1709	1801	1901
3	103	223	311	409	509	607	709	811	911	1013	1109	1213	1303	1423	1523	1607	1721	1811	1907
5	107	227	313	419	521	613	719	821	919	1019	1117	1217	1307	1427	1531	1609	1723	1823	1913
7	109	229	317	421	523	617	727	823	929	1021	1123	1223	1319	1429	1543	1613	1733	1831	1931
11	113	233	331	431	541	619	733	827	937	1031	1129	1229	1321	1433	1549	1619	1741	1847	1933
13	127	239	337	433	547	631	739	829	941	1033	1151	1231	1327	1439	1553	1621	1747	1861	1949
17	131	241	347	439	557	641	743	839	947	1039	1153	1237	1361	1447	1559	1627	1753	1867	1951
19	137	251	349	443	563	643	751	853	953	1049	1163	1249	1367	1451	1567	1637	1759	1871	1973
23	139	257	353	449	569	647	757	857	967	1051	1171	1259	1373	1453	1571	1657	1777	1873	1979
29	149	263	359	457	571	653	761	859	971	1061	1181	1277	1381	1459	1579	1663	1783	1877	1987
31	151	269	367	461	577	659	769	863	977	1063	1187	1279	1399	1471	1583	1667	1787	1879	1993
37	157	271	373	463	587	661	773	877	983	1069	1193	1283		1481	1597	1669	1789	1889	1997
41	163	277	379	467	593	673	787	881	991	1087		1289		1483		1693			1999
43	167	281	383	479	599	677	797	883	997	1091		1291		1487		1697			
47	173	283	389	487		683		887		1093		1297		1489		1699			
53	179	293	397	491		691				1097				1493					
59	181			499										1499					
61	191																		
67	193																		
71	197																		
73	199																		
79																			
83																			
89																			
97																			

Miller-Rabin Algorithm

- Typically used to test a large number for primality
- Algorithm is: TEST (*n*)

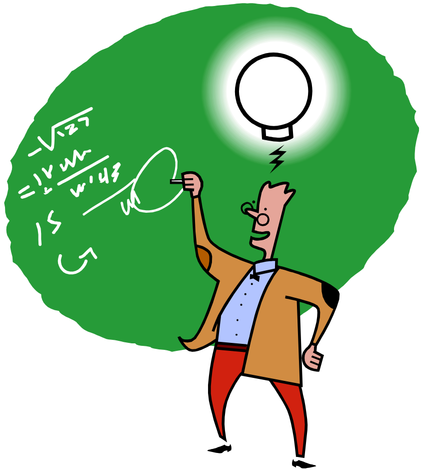
```
– Find integers $k$, $q$, with $k > 0$, $q$ odd, so that $(n – 1)=2^k q$ ;
– Select a random integer $a$, $1 < a < n – 1$ ;
– if $a^q \bmod n = 1$ then
    • return ("inconclusive") ;
– for $j = 0$ to $k – 1$ do
    • if $(a^{2^j q} \bmod n = n – 1)$ then
        • return ("inconclusive") ;
– return ("composite") ;
```

Miller Rabin Usage

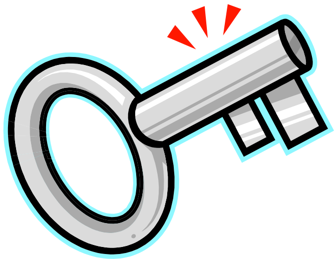
- It can be shown that given an odd number *n* that is not prime and a randomly chosen integer, *a* with $1 < a < n - 1$, the probability that TEST will return **inconclusive** (i.e., fail to detect that *n* is not prime) is less than 1/4.
- Thus, if *t* different values of *a* are chosen, the probability that all of them will pass TEST (return inconclusive) for *n* is less than $(1/4)^t$. For example, for $t = 10$, the probability that a nonprime number will pass all ten tests is less than 10^{-6} .
- Thus, for a sufficiently large value of t, we can be confident that n is prime if Miller's test always returns **inconclusive** .
- invoke TEST (*n*) using randomly chosen values for *a* . If, at any point, TEST returns composite , then *n* is determined to be nonprime. If TEST continues to return **inconclusive** for *t* tests, then for a sufficiently large value of *t* , assume that *n* is prime.

Deterministic Primality Algorithm

- Prior to 2002 there was no known method of efficiently proving the primality of very large numbers
- All of the algorithms in use produced a probabilistic result
- In 2002 Agrawal, Kayal, and Saxena developed an algorithm that efficiently determines whether a given large number is prime
  - Known as the AKS algorithm
  - Does not appear to be as efficient as Miller-Rabin algorithm



Public-Key Cryptography



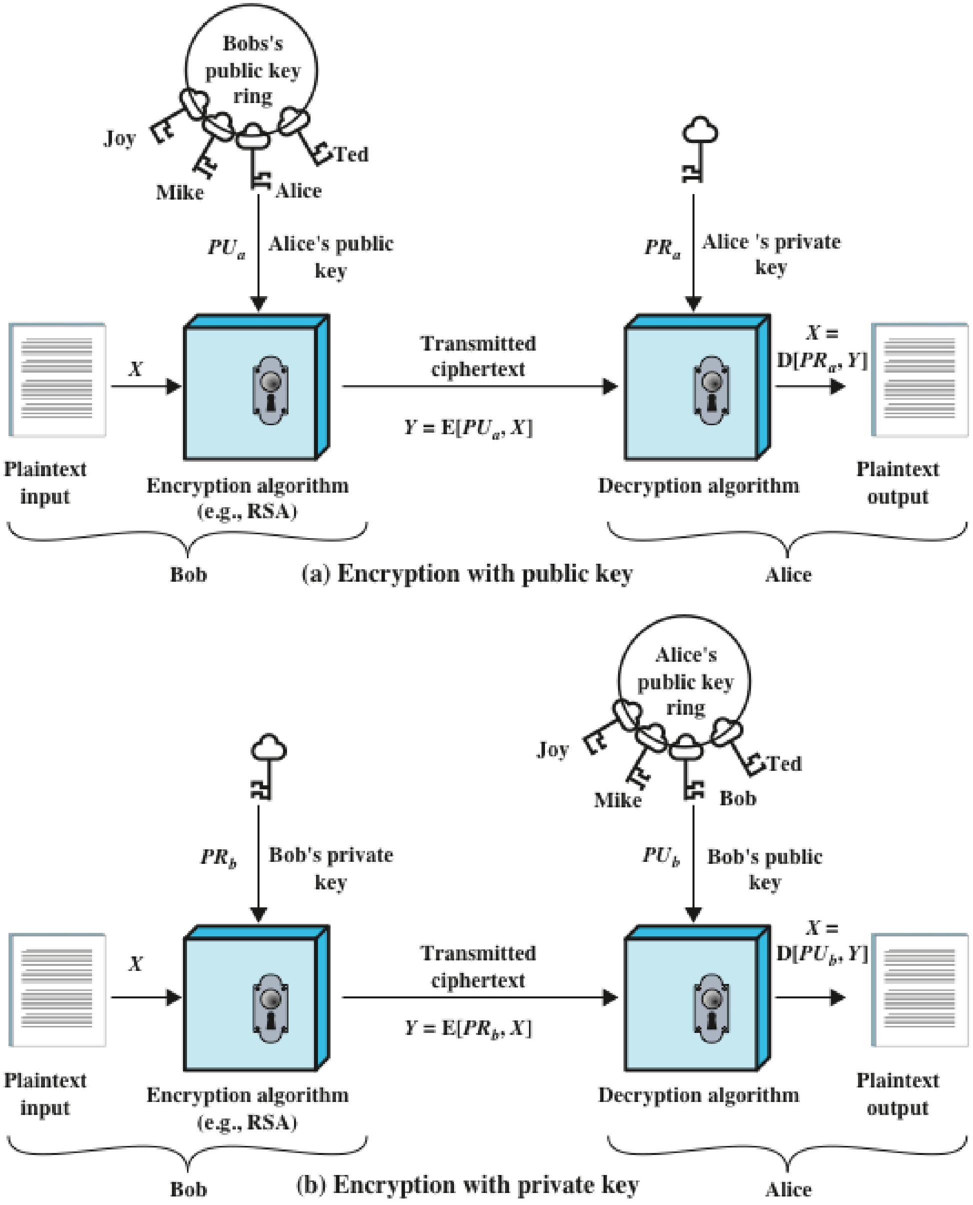
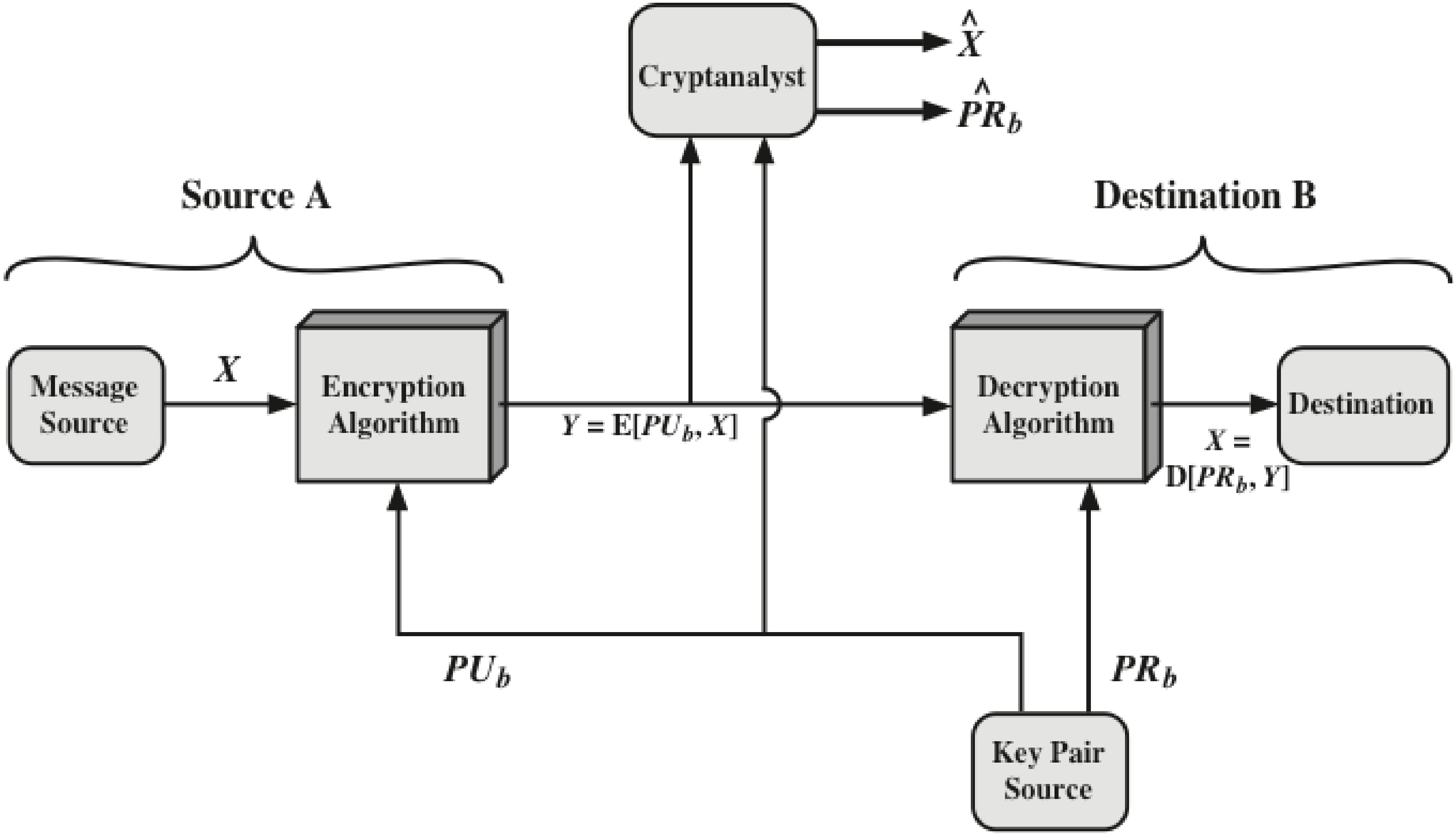
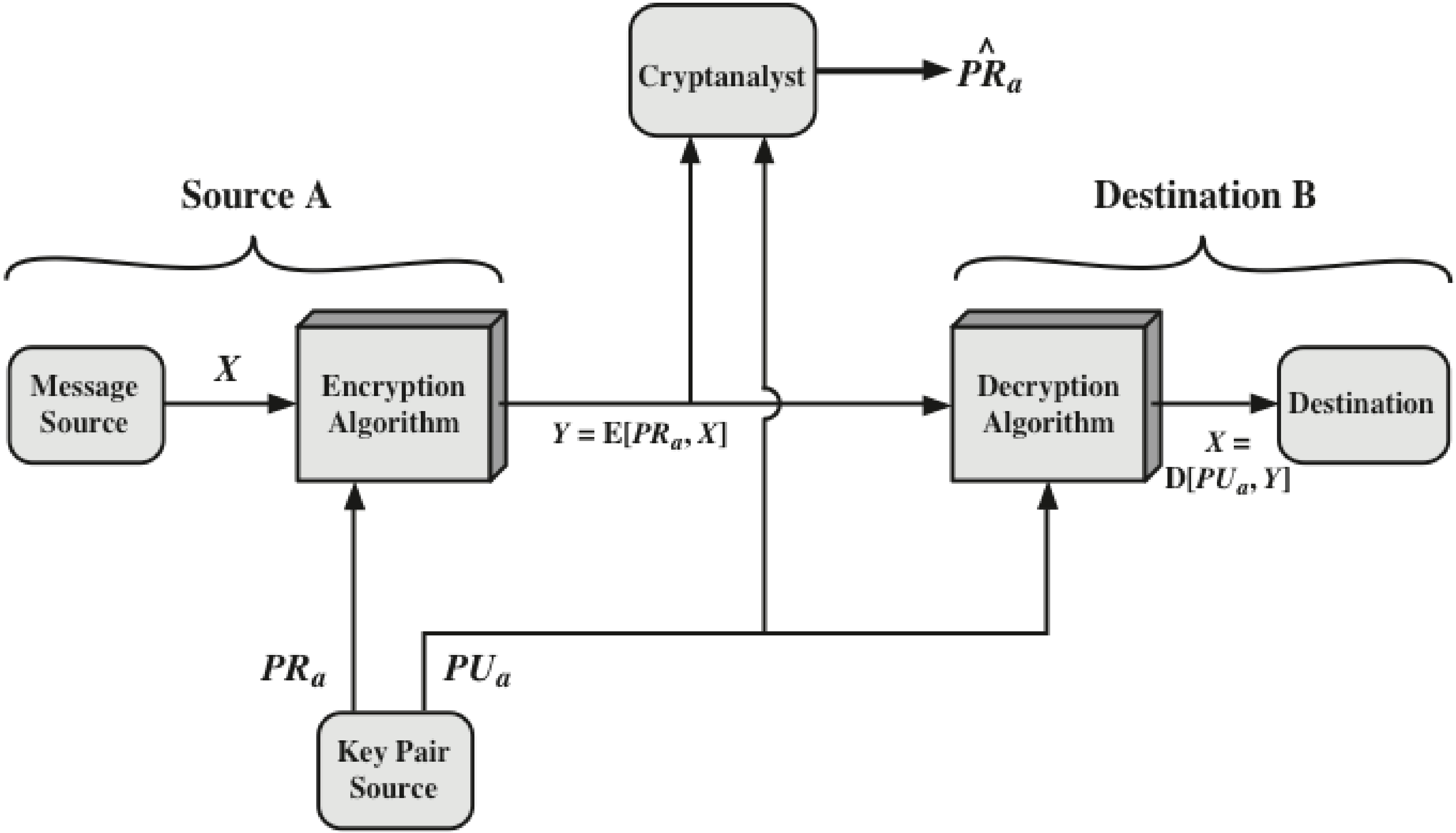


Figure 9.1 Public-Key Cryptography

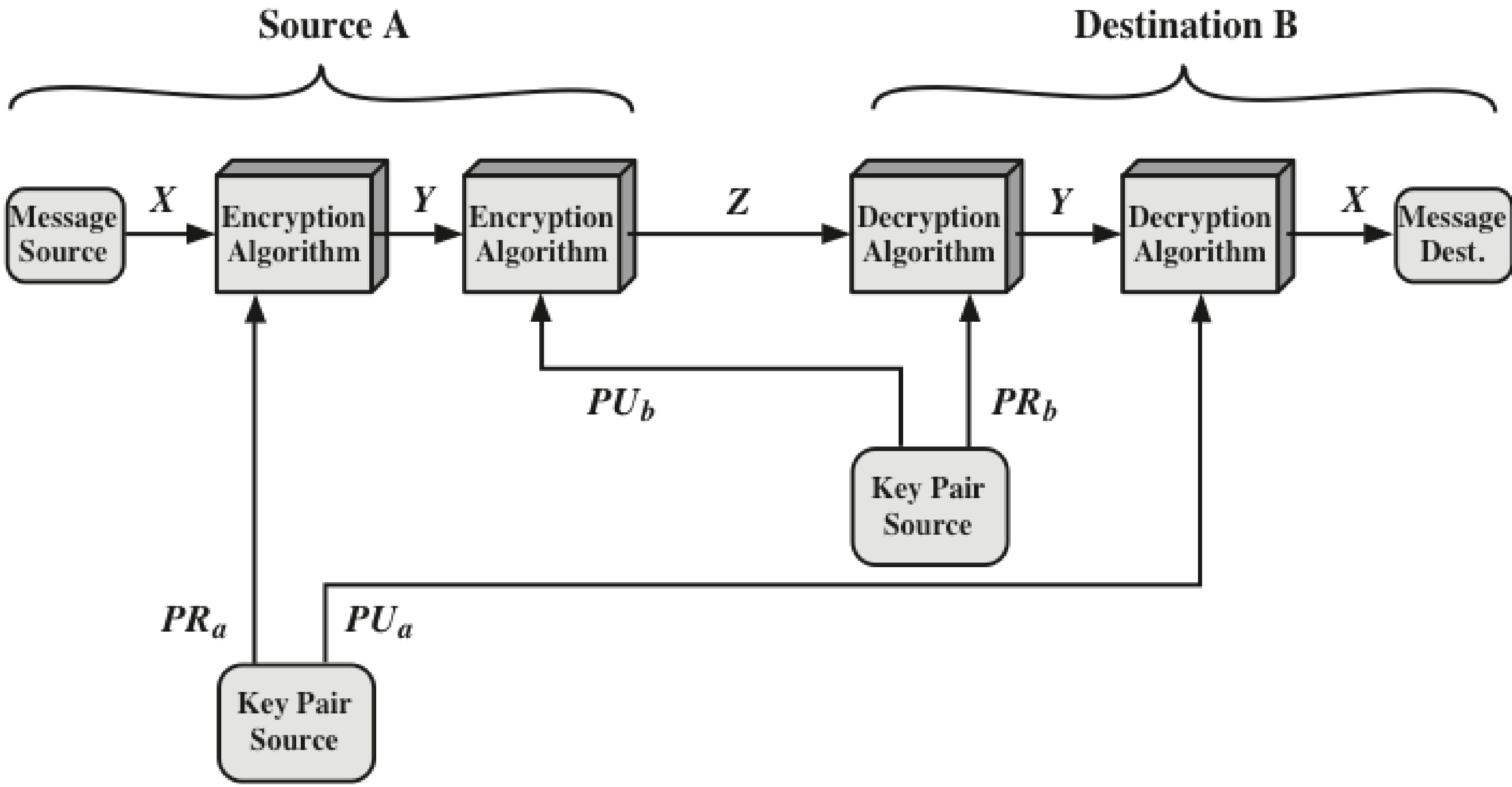
Public-Key Cryptosystem: Confidentiality

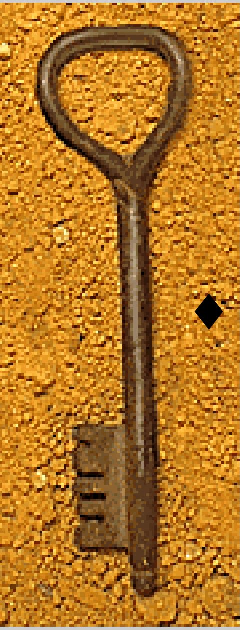


Public-Key Cryptosystem: Authentication



Public-Key Cryptosystem: Authentication and Confidentiality





Public-Key Requirements

- Conditions that these algorithms must fulfill:
  - It is computationally easy for a party B to generate a pair (public-key PU_b, private key PR_b)
  - It is computationally easy for a sender A, knowing the public key and the message to be encrypted, to generate the corresponding ciphertext
  - It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message
  - It is computationally infeasible for an adversary, knowing the public key, to determine the private key
  - It is computationally infeasible for an adversary, knowing the public key and a ciphertext, to recover the original message
  - The two keys can be applied in either order

Public-Key Requirements

- Need a trap-door one-way function
  - A one-way function is one that maps a domain into a range such that every function value has a unique inverse, with the condition that the calculation of the function is easy, whereas the calculation of the inverse is infeasible
    - $Y = f(X)$ easy
    - $X = f^{-1}(Y)$ infeasible
- A trap-door one-way function is a family of invertible functions f_k, such that
  - $Y = f_k(X)$ easy, if k and X are known
  - $X = f_k^{-1}(Y)$ easy, if k and Y are known
  - $X = f_k^{-1}(Y)$ infeasible, if Y known but k not known
- A practical public-key scheme depends on a suitable trap-door one-way function

Rivest-Shamir-Adleman (RSA) Scheme

- Developed in 1977 at MIT by Ron Rivest, Adi Shamir & Len Adleman
- Most widely used general-purpose approach to public-key encryption
- Is a cipher in which the plaintext and ciphertext are integers between 0 and *n* – 1 for some *n*
  - A typical size for *n* is 1024 bits, or 309 decimal digits

Table 8.2

Some Values of Euler's Totient Function *ø(n)*

n	$\phi(n)$
1	1
2	1
3	2
4	2
5	4
6	2
7	6
8	4
9	6
10	4
11	10
12	4
13	12
14	6
15	8
16	8
17	16
18	6
19	18
20	8
21	12
22	10
23	22
24	8
25	20
26	12
27	18
28	12
29	28
30	8

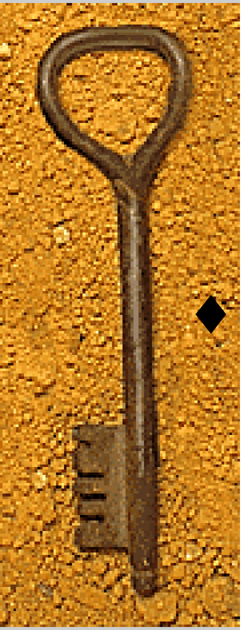
RSA Algorithm

- RSA makes use of an expression with exponentials
- Plaintext is encrypted in blocks with each block having a binary value less than some number *n*
- Encryption and decryption are of the following form, for some plaintext block *M* and ciphertextblock C

$$C = M^{e} \mathrm{mod}\ n$$

$$M = C^{d}\ mod\ n = (M^{e})^{d}\ mod\ n = M^{ed}\ mod\ n$$

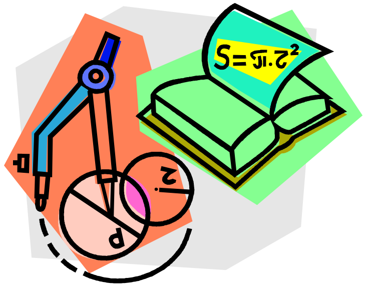
- Both sender and receiver must know the value of *n*
- The sender knows the value of *e*, and only the receiver knows the value of *d*
- This is a public-key encryption algorithm with a public key of *PU={e,n}* and a private key of *PR={d,n}*

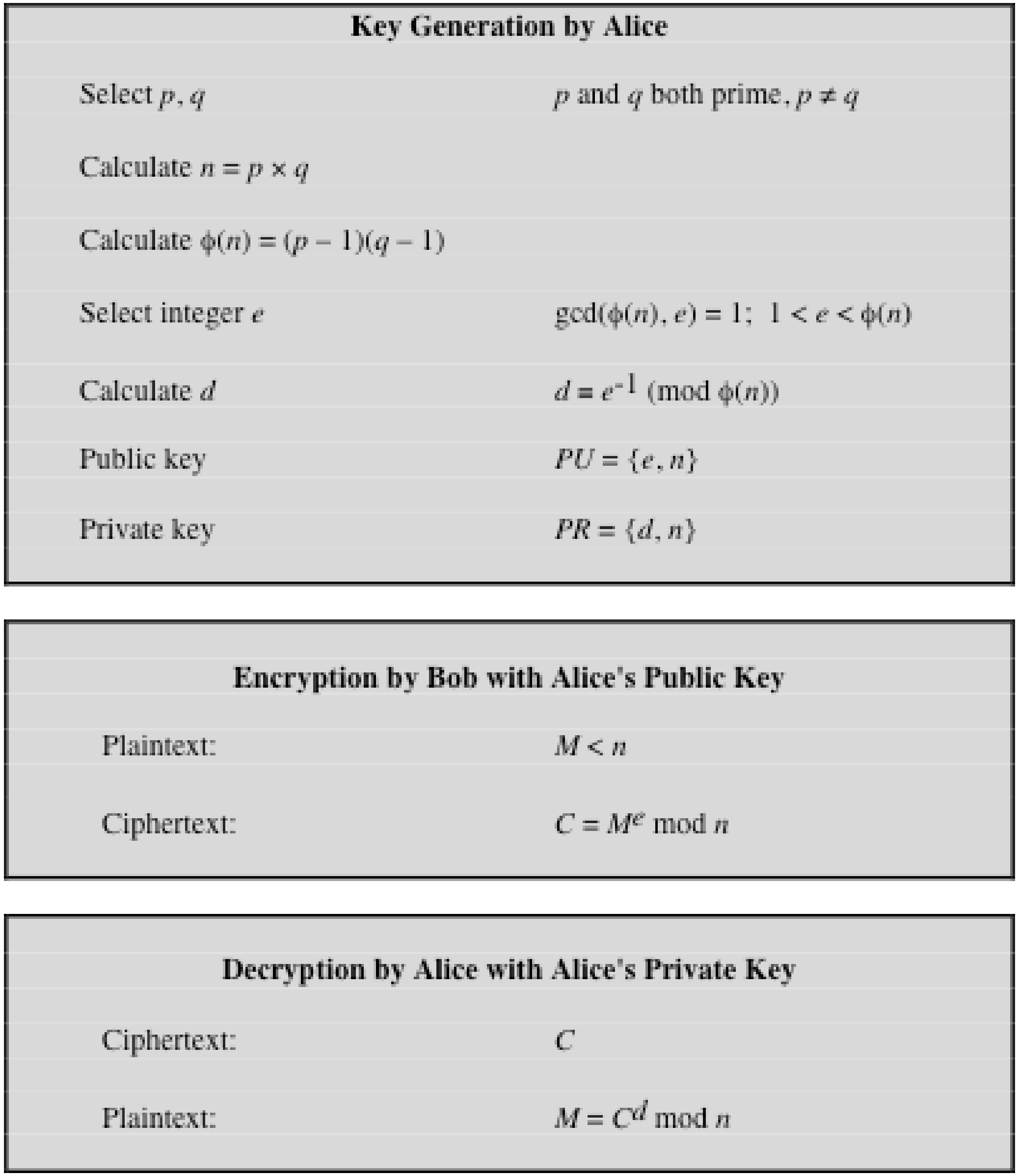


Algorithm Requirements

- For this algorithm to be satisfactory for public-key encryption, the following requirements must be met:

1. It is possible to find values of *e*, *d*, *n* such that M^{ed} mod $n = M$ for all $M<n$
2. It is relatively easy to calculate M^e mod n and C^d mod n for all values of $M < n$
3. It is infeasible to determine *d* given *e* and *n*





Key Generation by Alice	
Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1;\ 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

Encryption by Bob with Alice's Public Key	
Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$

Decryption by Alice with Alice's Private Key	
Ciphertext:	C
Plaintext:	$M = C^d \bmod n$

Figure 9.5 The RSA Algorithm

Example of RSA Algorithm

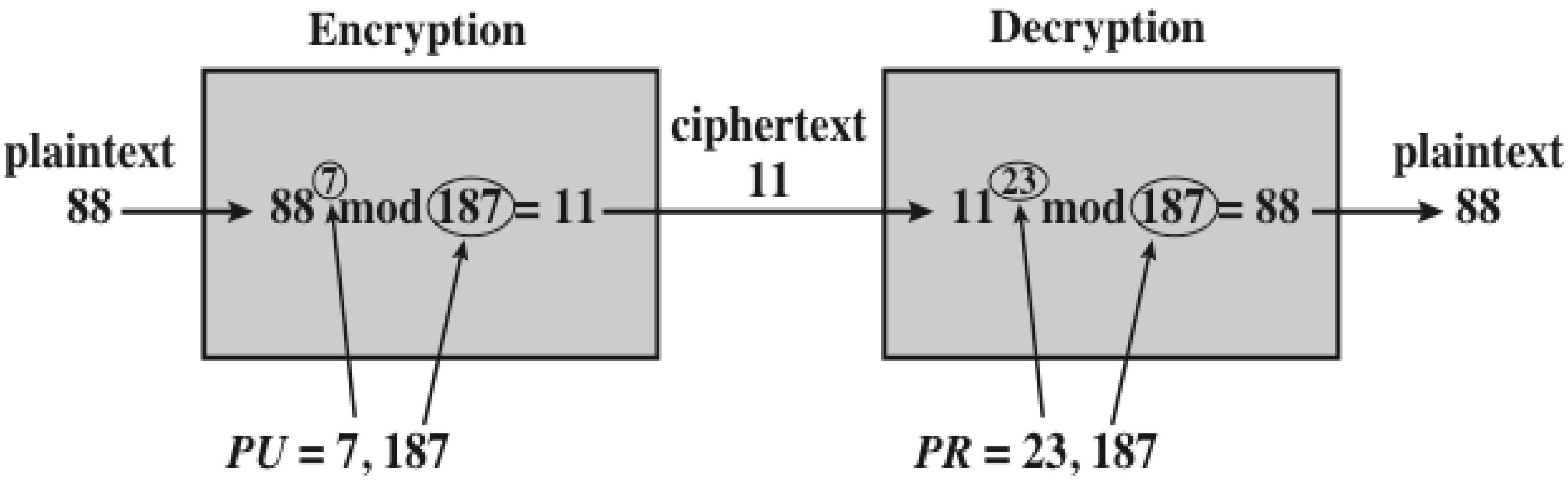
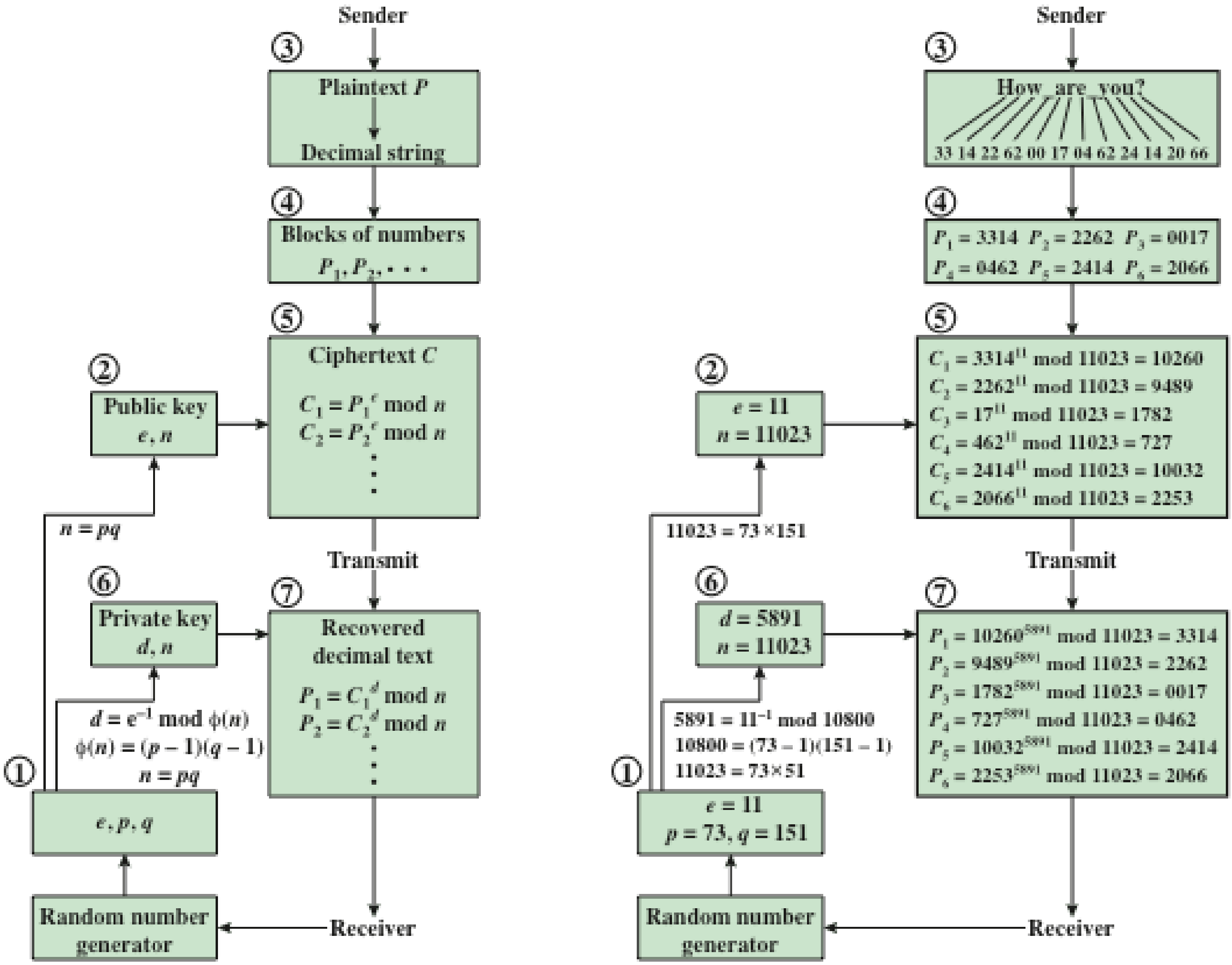


Figure 9.6 Example of RSA Algorithm



Sender

③

Plaintext P

Decimal string

④

Blocks of numbers

$P_1, P_2, \ldots$

⑤

Ciphertext C

$C_1 = P_1^e \bmod n$

$C_2 = P_2^e \bmod n$

②

Public key

e, n

$n = pq$

Transmit

⑥

Private key

d, n

⑦

Recovered decimal text

$P_1 = C_1^d \bmod n$

$P_2 = C_2^d \bmod n$

$d = e^{-1} \bmod \phi(n)$

$\phi(n) = (p-1)(q-1)$

$n = pq$

①

e, p, q

Random number generator

Receiver

Sender

③

How are you?

33 14 22 62 00 17 04 62 24 14 20 66

④

$P_1 = 3314$ $P_2 = 2262$ $P_3 = 0017$

$P_4 = 0462$ $P_5 = 2414$ $P_6 = 2066$

⑤

$C_1 = 3314^{11} \bmod 11023 = 10260$

$C_2 = 2262^{11} \bmod 11023 = 9489$

$C_3 = 17^{11} \bmod 11023 = 1782$

$C_4 = 462^{11} \bmod 11023 = 727$

$C_5 = 2414^{11} \bmod 11023 = 10032$

$C_6 = 2066^{11} \bmod 11023 = 2253$

②

$e = 11$

$n = 11023$

$11023 = 73 \times 151$

Transmit

⑥

$d = 5891$

$n = 11023$

⑦

$P_1 = 10260^{5891} \bmod 11023 = 3314$

$P_2 = 9489^{5891} \bmod 11023 = 2262$

$P_3 = 1782^{5891} \bmod 11023 = 0017$

$P_4 = 727^{5891} \bmod 11023 = 0462$

$P_5 = 10032^{5891} \bmod 11023 = 2414$

$P_6 = 2253^{5891} \bmod 11023 = 2066$

$5891 = 11^{-1} \bmod 10800$

$10800 = (73 - 1)(151 - 1)$

$11023 = 73 \times 51$

①

$e = 11$

$p = 73, q = 151$

Random number generator

Receiver

(a) General approach

(b) Example

Figure 9.7 RSA Processing of Multiple Blocks

Fermat's Theorem

- States the following:
  - If p is prime and a is a positive integer not divisible by p then

$$a^{p-1} = 1 \ (\mathrm{mod}\ p)$$

- Sometimes referred to as Fermat’s Little Theorem
- An alternate form is:
  - If p is prime and a is a positive integer then

$$a^{p} = a \ (\mathrm{mod}\ p)$$

- Plays an important role in public-key cryptography

Euler's Theorem

- States that for every *a* and *n* that are relatively prime:

$$a^{ø(\mathrm{n})} = 1(\mathrm{mod}\ n)$$

- An alternative form is:

$$a^{ø(\mathrm{n})+1} = a(\mathrm{mod}\ n)$$

Chinese Remainder Thm

- If p and q are prime, then for all x and a:
- x = a(mod p) and x = a(mod q) iff x=a mod(pq)

- Example:
- Suppose that n = 2501 = 61 * 41
- To calculate *V* mod 2501:
  - *V* mod 61
  - *V* mod 41

Correctness of RSA

- To show RSA is correct, we must show that encryption and decryption are inverse functions:
  - $En(De(M)) = De(En(M)) = M = M^{ed} \pmod{n}$
  - Since d and e are multiplicative inverses mod $\phi(n)$, there is a *k* such that:
    - $ed = 1 + k * \phi(n), = 1 + k(p-1)(q-1)$
    - $M^{ed} = M^{1+k(p-1)(q-1)} = M*(M^{p-1})^{k(q-1)}$
    - By Fermat: $M^{p-1} = 1 \pmod{p}$
    - $M^{ed} = M(1)^{k(q-1)} \pmod{p} = M \pmod{p}$

Correctness of RSA

- $M^{ed} = M(1)^{k(q-1)}(\text{mod } p) = M(\text{mod } p)$
- $M^{ed} = M(1)^{k(q-1)}(\text{mod } q) = M(\text{mod } q)$
- By Chinese Remainder Thm, we get:
- M^{ed} = M (mod p)=M (mod q) =
  M (mod pq) = M (mod n)

- Therefore, RSA reproduces the original message and is correct.

Exponentiation in Modular Arithmetic

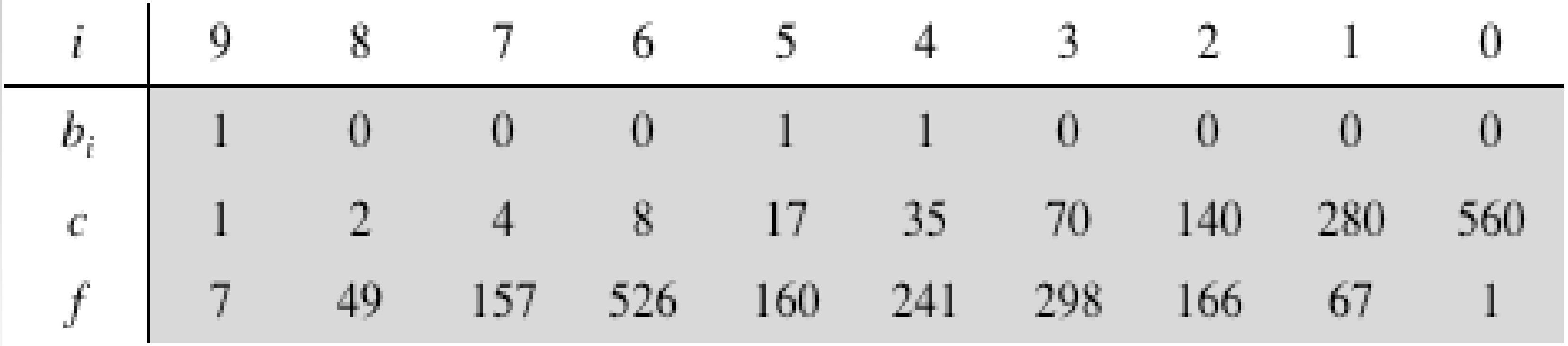
- Both encryption and decryption in RSA involve raising an integer to an integer power, mod *n*
- Can make use of a property of modular arithmetic:

  $[(a \bmod n)\ x\ (b \bmod n)] \bmod n = (a\ x\ b) \bmod n$
- With RSA you are dealing with potentially large exponents so efficiency of exponentiation is a consideration

Fast Exponentiation Algorithm

```
f = 1
for (i=k ; i>0 ; i--)
        f = (f * f)    mod  n;
        if (b_i == 1)
                 f = (f * a) mod n;
return f;
```

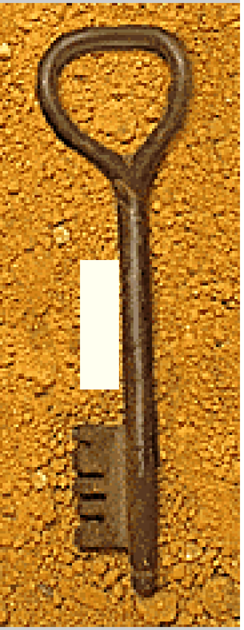
Algorithm for computing a^b mod n, b is expressed as a binary $b_k\, b_{k-1} \ldots b_0$



i	9	8	7	6	5	4	3	2	1	0
b_i	1	0	0	0	1	1	0	0	0	0
c	1	2	4	8	17	35	70	140	280	560
f	7	49	157	526	160	241	298	166	67	1

Result of the Fast Modular Exponentiation Algorithm for a^b mod n, where $a = 7$, $b = 560 = 1000110000$, and $n = 561$

Euclidean Algorithm



INPUT: Two non-negative integers *a* and *b* with a ≥ b.
OUTPUT: gcd(a, b).

```
1.While b > 0, do
    1.Set r = a mod b,
    2.a = b,
    3.b = r
2.Return a.
```

Ex: Find gcd(421, 111). use the Euclidean algorithm as follows:

$421 = 111 \times 3 + 88$

$111 = 88 \times 1 + 23$

$88 = 23 \times 3 + 19$

$23 = 19 \times 1 + 4$

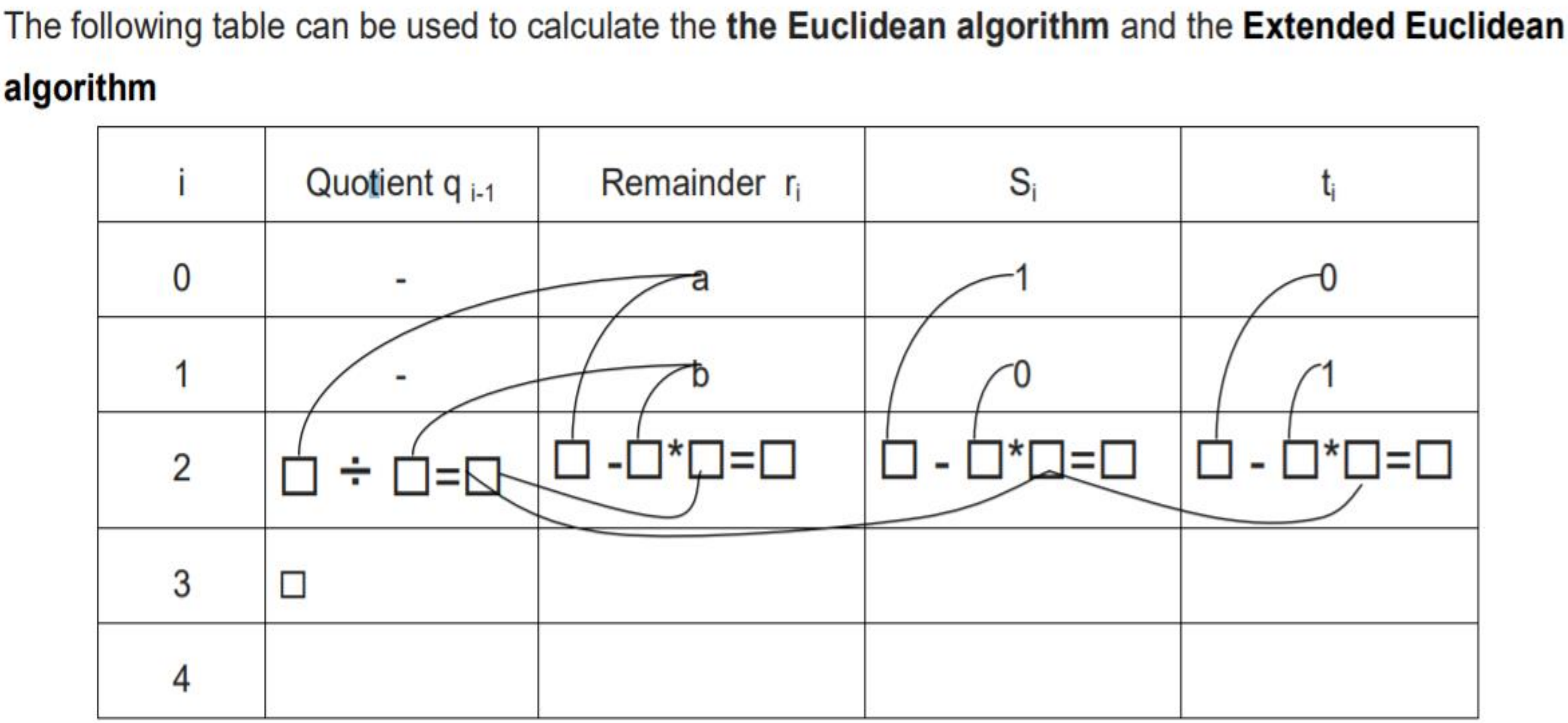
$19 = 4 \times 4 + 3$

$4 = 3 \times 1 + 1$

$3 = 1 \times 3 + 0$

The last non-zero remainder is 1 and therefore gcd(421, 111) = 1.

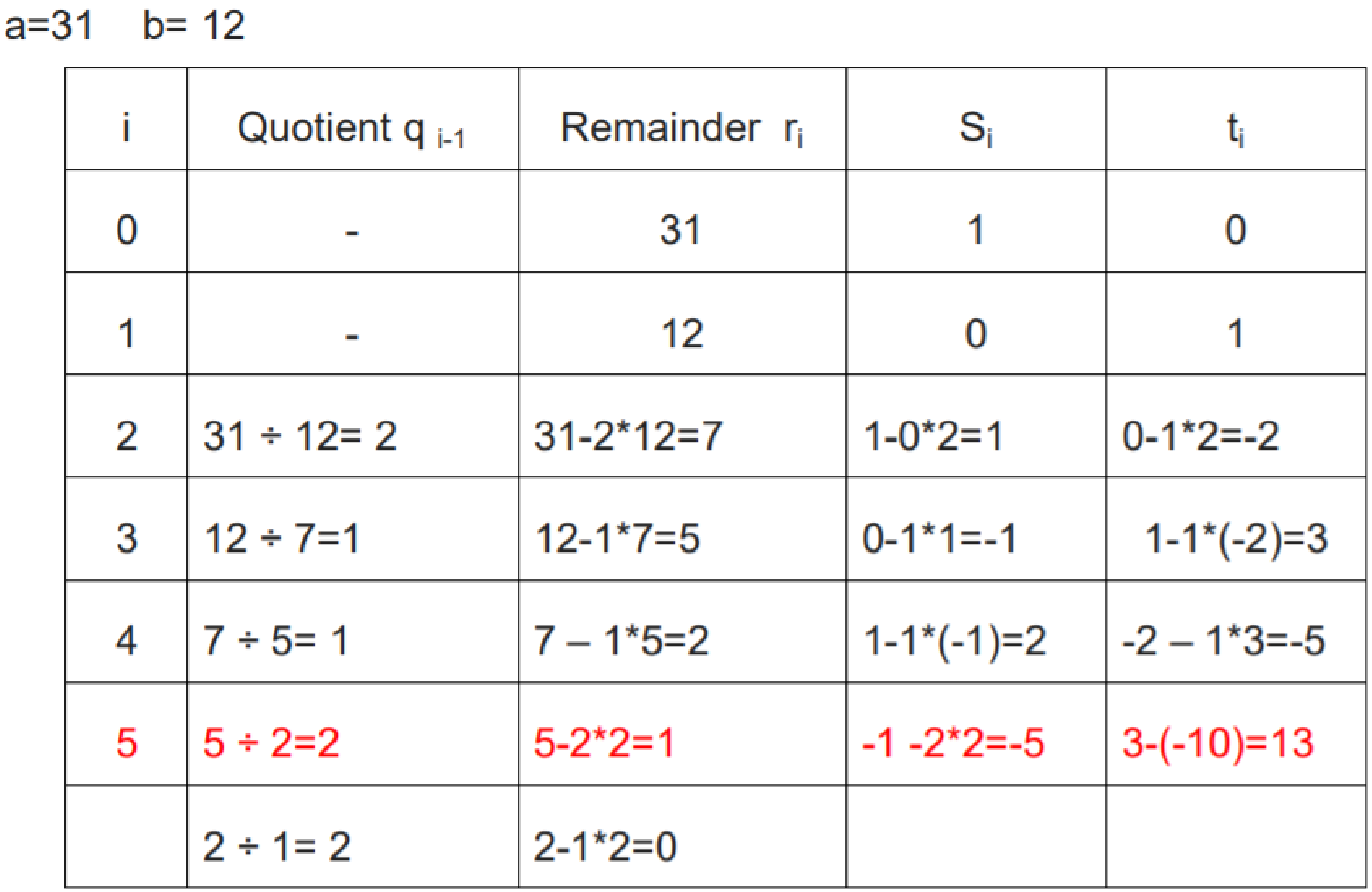
Extended Euclidean Algorithm



The following table can be used to calculate the **the Euclidean algorithm** and the **Extended Euclidean algorithm**

i	Quotient q_{i-1}	Remainder r_i	S_i	t_i
0	-	a	1	0
1	-	b	0	1
2	□ ÷ □=□	□ -□*□=□	□ - □*□=□	□ - □*□=□
3	□			
4				

Example

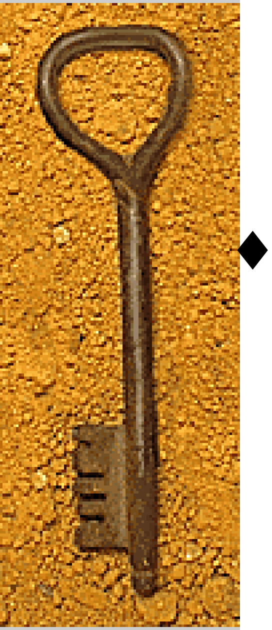


a=31   b= 12

i	Quotient q_{i-1}	Remainder r_i	S_i	t_i
0	-	31	1	0
1	-	12	0	1
2	31 ÷ 12= 2	31-2*12=7	1-0*2=1	0-1*2=-2
3	12 ÷ 7=1	12-1*7=5	0-1*1=-1	1-1*(-2)=3
4	7 ÷ 5= 1	7 – 1*5=2	1-1*(-1)=2	-2 – 1*3=-5
5	5 ÷ 2=2	5-2*2=1	-1 -2*2=-5	3-(-10)=13
	2 ÷ 1= 2	2-1*2=0		

Efficient Operation Using the Public Key

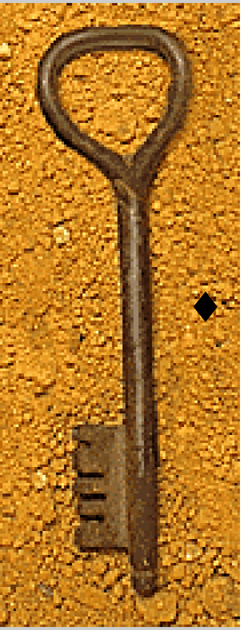
- To speed up the operation of the RSA algorithm using the public key, a specific choice of *e* is usually made
- The most common choice is 65537 ($2^{16} + 1$)
  - Two other popular choices are *e*=3 and *e*=17
  - Each of these choices has only two 1 bits, so the number of multiplications required to perform exponentiation is minimized
  - With a very small public key, such as *e* = 3, RSA becomes vulnerable to a simple attack



Key Generation

- Before the application of the public-key cryptosystem each participant must generate a pair of keys:
  - Determine two prime numbers *p* and *q*
  - Select either *e* or *d* and calculate the other

- Because the value of $n = pq$ will be known to any potential adversary, primes must be chosen from a sufficiently large set
  - The method used for finding large primes must be reasonably efficient

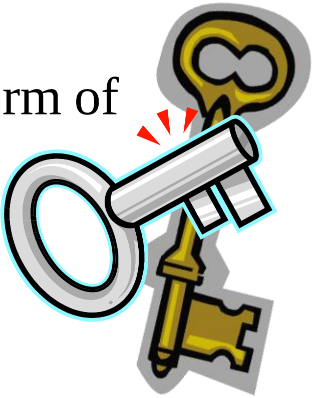


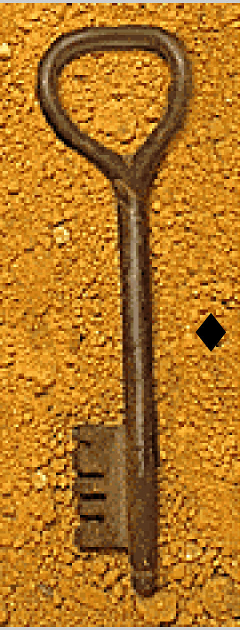
Public-Key Cryptanalysis

- A public-key encryption scheme is vulnerable to a brute-force attack
  - Countermeasure: use large keys
  - Key size must be small enough for practical encryption and decryption
  - Key sizes that have been proposed result in encryption/decryption speeds that are too slow for general-purpose use
  - Public-key encryption is currently confined to key management and signature applications

- Another form of attack is to find some way to compute the private key given the public key
  - To date it has not been mathematically proven that this form of attack is infeasible for a particular public-key algorithm

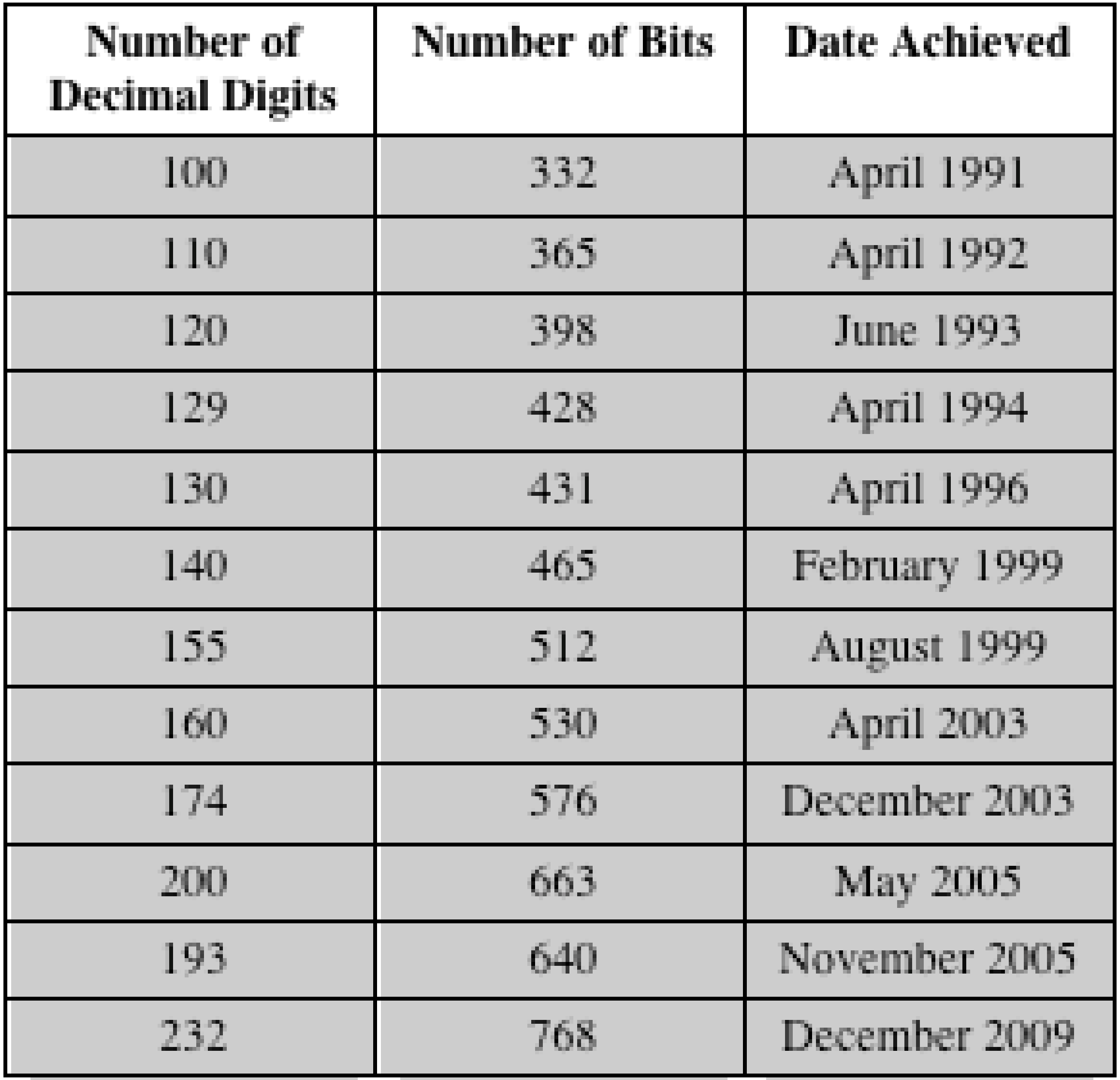
- Finally, there is a probable-message attack
  - This attack can be thwarted by appending some random bits to simple messages





Factoring Problem

- We can identify three approaches to attacking RSA mathematically:
  - Factor n into its two prime factors. This enables calculation of $\o(n) = (p - 1)\ x\ (q - 1)$, which in turn enables determination of $d = e^{-1}\ (mod\ \o(n))$
  - Determine $\o(n)$ directly without first determining p and q. Again this enables determination of $d = e^{-1}\ (mod\ \o(n))$
  - Determine d directly without first determining $\o(n)$



Number of Decimal Digits	Number of Bits	Date Achieved
100	332	April 1991
110	365	April 1992
120	398	June 1993
129	428	April 1994
130	431	April 1996
140	465	February 1999
155	512	August 1999
160	530	April 2003
174	576	December 2003
200	663	May 2005
193	640	November 2005
232	768	December 2009

Table 9.5 Progress in RSA Factorization

MIPS-Years Needed to Factor

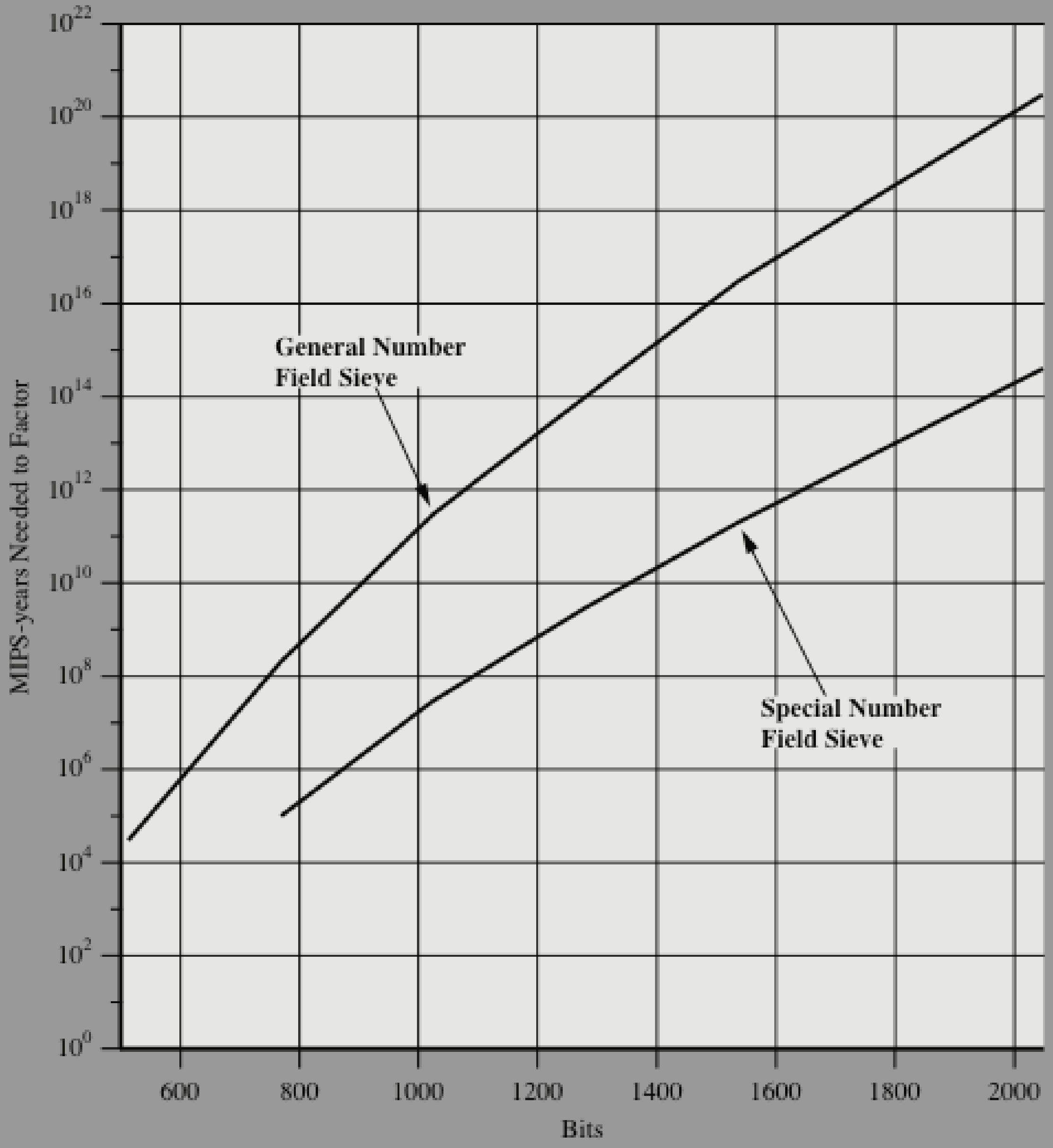


Figure 9.9 MIPS-years Needed to Factor

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
- Are applicable not just to RSA but to other public-key cryptography systems
- Are alarming for two reasons:
  - It comes from a completely unexpected direction
  - It is a ciphertext-only 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**

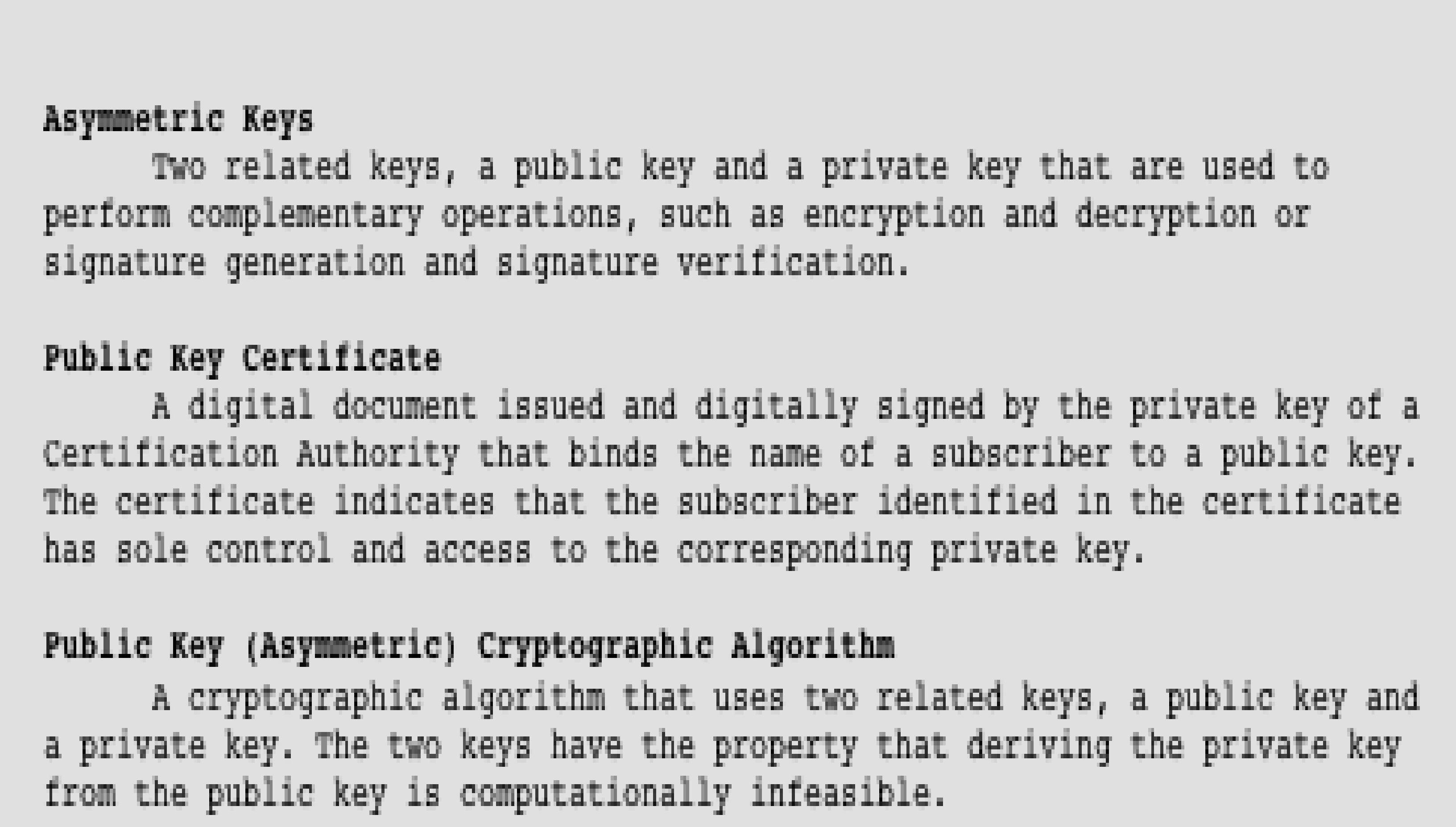
Blinding

- **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**

Misconceptions Concerning Public-Key Encryption

- Public-key encryption is more secure from cryptanalysis than symmetric encryption
- Public-key encryption is a general-purpose technique that has made symmetric encryption obsolete
- There is a feeling that key distribution is trivial when using public-key encryption, compared to the cumbersome handshaking involved with key distribution centers for symmetric encryption

Terminology Related to Asymmetric Encryption



Asymmetric Keys

Two related keys, a public key and a private key that are used to perform complementary operations, such as encryption and decryption or signature generation and signature verification.

Public Key Certificate

A digital document issued and digitally signed by the private key of a Certification Authority that binds the name of a subscriber to a public key. The certificate indicates that the subscriber identified in the certificate has sole control and access to the corresponding private key.

Public Key (Asymmetric) Cryptographic Algorithm

A cryptographic algorithm that uses two related keys, a public key and a private key. The two keys have the property that deriving the private key from the public key is computationally infeasible.

Source: *Glossary of Key Information Security Terms*, NIST IR 7298 [KISS06]

Principles of Public-Key Cryptosystems

- The concept of public-key cryptography evolved from an attempt to attack two of the most difficult problems associated with symmetric encryption:

Key distribution

- **How to have secure communications in general without having to trust a KDC with your key**

Digital signatures

- **How to verify that a message comes intact from the claimed sender**

- Whitfield Diffie and Martin Hellman from Stanford University achieved a breakthrough in 1976 by coming up with a method that addressed both problems and was radically different from all previous approaches to cryptography

End

Questions

The Diffie-Hellman Algorithm

Modified by: Dr. Ramzi Saifan

Introduction

- Discovered by Whitfield Diffie and Martin Hellman
  - "New Directions in Cryptography"
- The point is to agree on a key that two parties can use for a symmetric encryption, in such a way that an eavesdropper cannot obtain the key.
- Diffie-Hellman key agreement protocol
  - Exponential key agreement
  - Allows two users to exchange a secret key
  - Requires no prior secrets
  - Real-time over an un-trusted network

Introduction

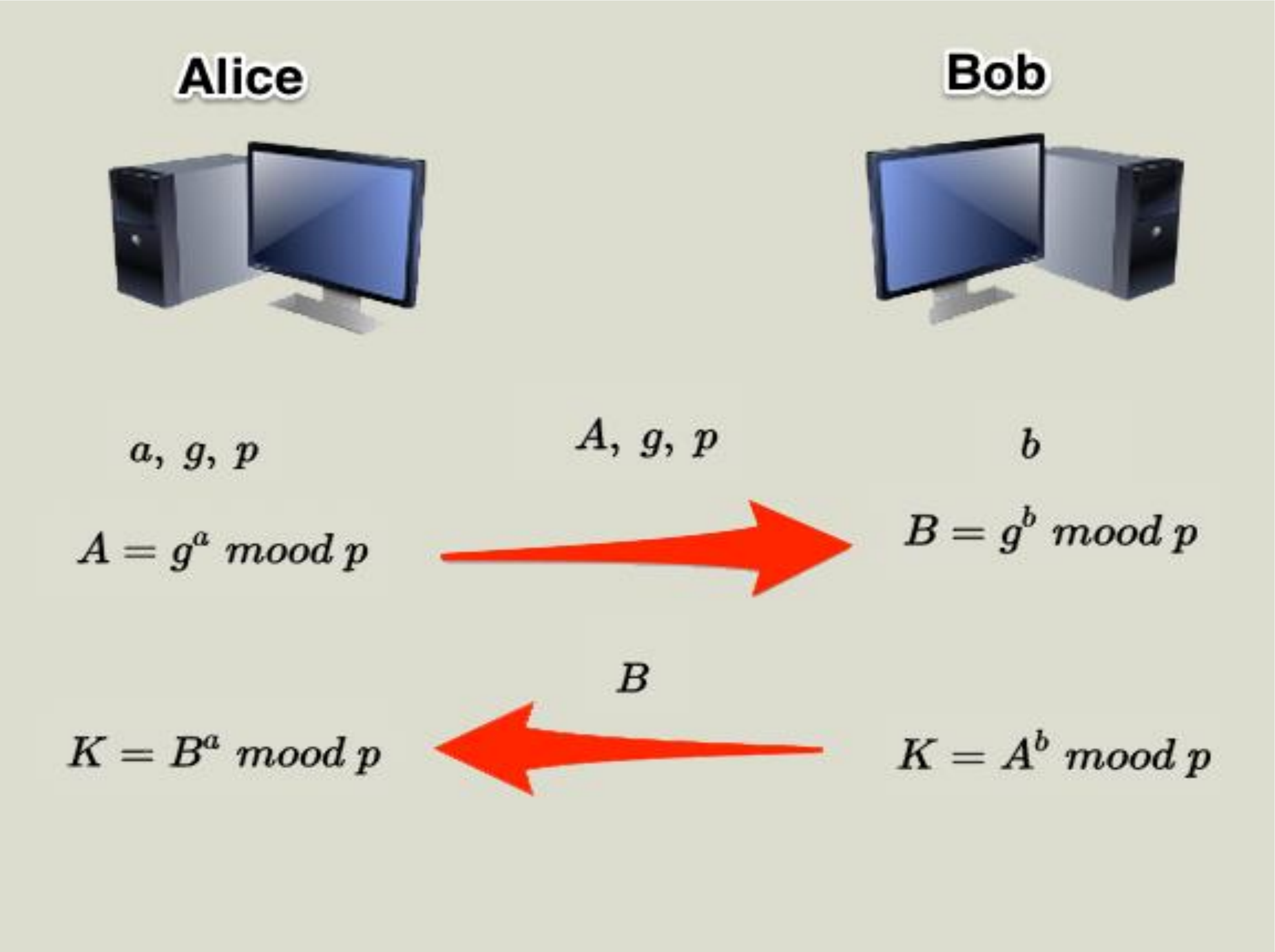
- Based on the difficulty of computing discrete logarithms of large numbers.
- Requires two large numbers, one prime (P), and (G), a primitive root of P

Implementation

- p and g are both publicly available numbers
  - P is at least 512 bits
- Alice picks a private value "a" and send to Bob
  - A = g^a mod p
- Bob picks a private value "b" and sends to Alice:
  - B = g^b mod p

Implementation

- Compute shared, private key:
  - Alice received B and knows a, p and g, so she calculates:
    - $K_a = B^a \bmod p$
  - Bob received A and knows b, p and g, so he calculates:
    - $K_b = A^b \bmod p$
- Algebraically it can be shown that $K_a = K_b = K$
  - Users now have a symmetric secret key to encrypt



Alice

Bob

$a,\ g,\ p$

$A,\ g,\ p$

b

$A = g^a \ mood\ p$

$B = g^b \ mood\ p$

B

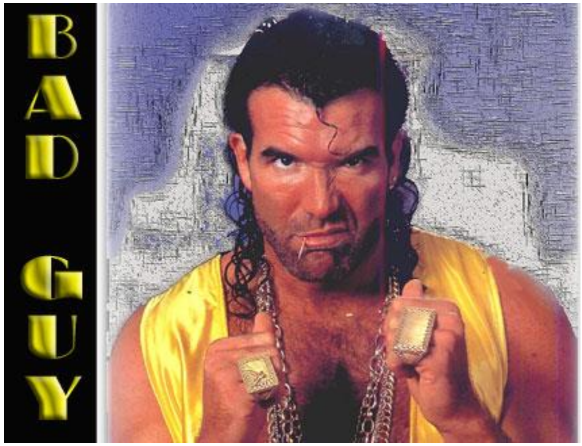
$K = B^a \ mood\ p$

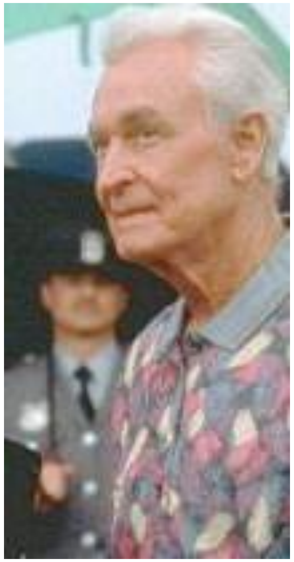
$K = A^b \ mood\ p$

Example

- Bob and Alice are unable to talk on the untrusted network.

  –Who knows who’s listening?





Example

- Alice and Bob get public numbers
  - P = 23, G = 9

- Alice and Bob compute public values
  - A = 9^4 mod 23 = 6561 mod 23 = 6
  - B = 9^3 mod 23 = 729 mod 23 = 16

- Alice and Bob exchange public numbers

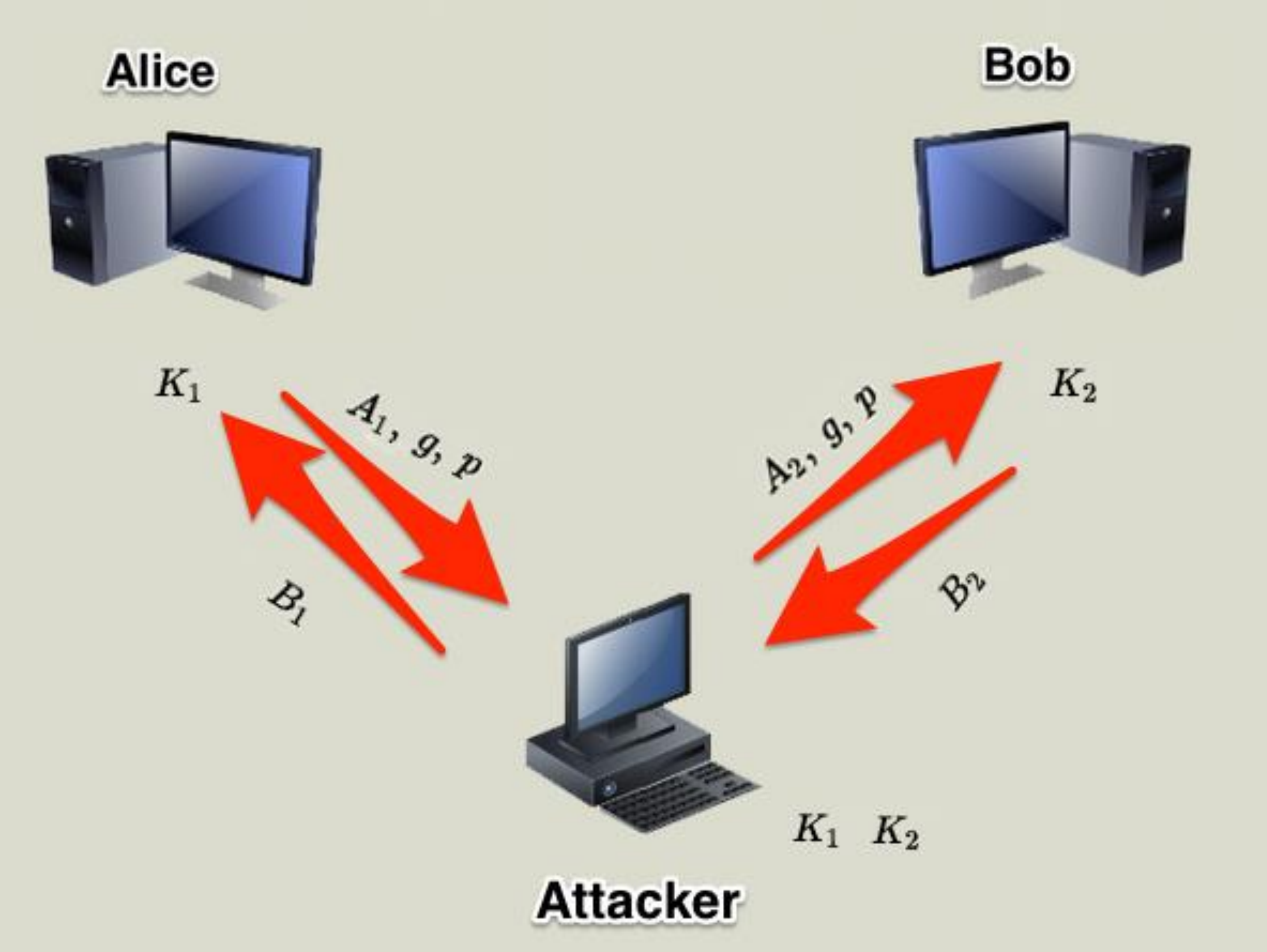
Example

- Alice and Bob compute symmetric keys
  - $k_a = B^a \bmod p = 16^4 \bmod 23 = 9$
  - $k_b = A^b \bmod p = 6^3 \bmod 23 = 9$
- Alice and Bob now can talk securely!

Security of DH

- Suppose **p** is a prime of around 300 digits,
- and **a** and **b** at least 100 digits each.
- Discovering the shared secret given **g**, **p**, $\mathbf{g^a}$ **mod p,** and $\mathbf{g^b}$ **mod p** would take longer than the lifetime of the universe, using the best known algorithm.
- This is called the discrete logarithm problem.

Man in the middle attack



Applications

- Diffie-Hellman is currently used in many protocols, namely:
  - Secure Sockets Layer (SSL)/Transport Layer Security (TLS)
  - Secure Shell (SSH)
  - Internet Protocol Security (IPSec)
  - Public Key Infrastructure (PKI)

User Authentication

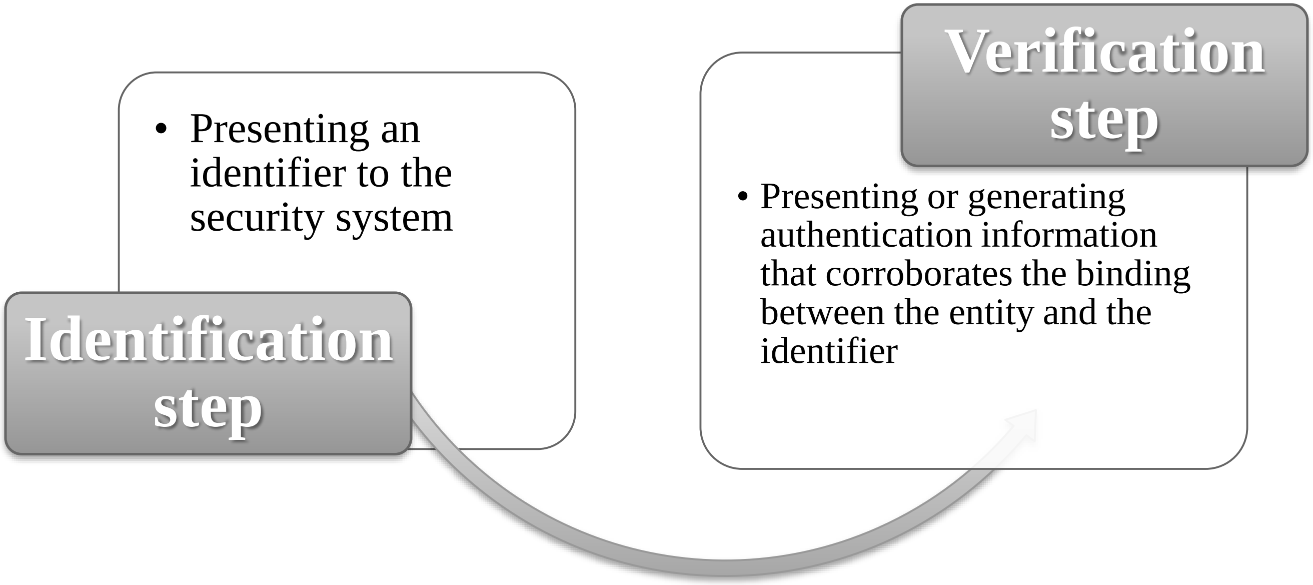
Modified By: Dr. Ramzi Saifan

Authentication

- Verifying the identity of another entity
  - Computer authenticating to another computer
  - Person authenticating to a local/remote computer
- Important to be clear about what is being authenticated
  - The user?
  - The machine? A specific application on the machine?
  - The data?
- Mutual authentication vs. unidirectional authentication

Remote User-Authentication Principles

- An authentication process consists of two steps:



Identification step

- Presenting an identifier to the security system

Verification step

- Presenting or generating authentication information that corroborates the binding between the entity and the identifier

Authentication

- Authentication may be based on
  1. What you know
  2. What you have
  3. What you are
  4. What you do
  - – Examples? Tradeoffs?
  - – Others?

Address-based authentication

- Is sometimes used
- Generally not very secure
  - Relatively easy to forge source addresses of network packets
- But can be useful if the adversary does not know what IP address to forge
  - E.g., IP address of a user’s home computer

Multi-factor Authentication

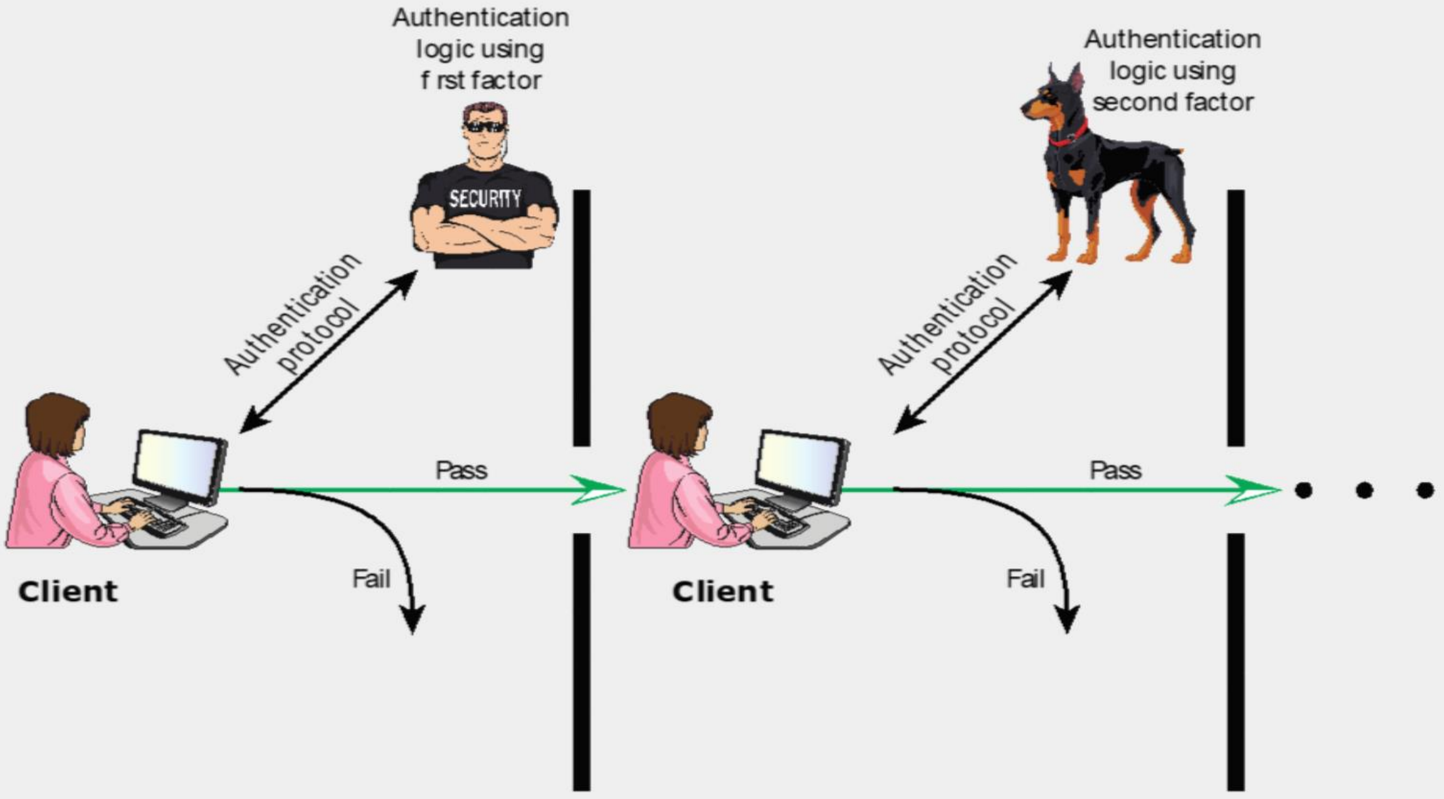
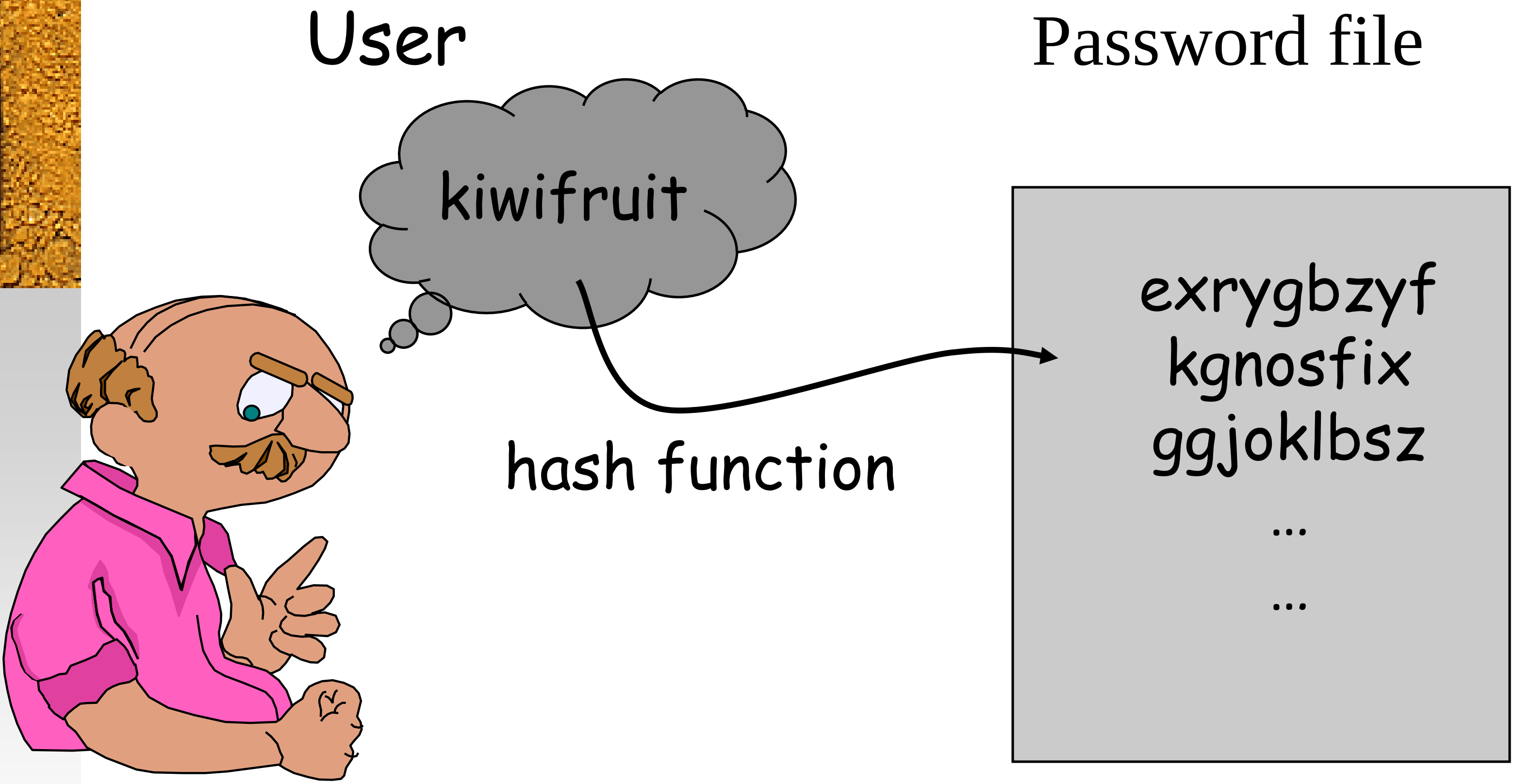


Figure 3.2 Multifactor Authentication

Password-based protocols

- Basic idea
  - User has a secret password
  - System checks password to authenticate user
- Issues
  - How is password stored?
  - How does system check password?
  - How easy is it to guess a password?
    - Difficult to keep password file secret, so best if it is hard to guess password even if you have the password file
- Distinguish *on-line attacks* vs. *off-line attacks*

Basic password scheme



User

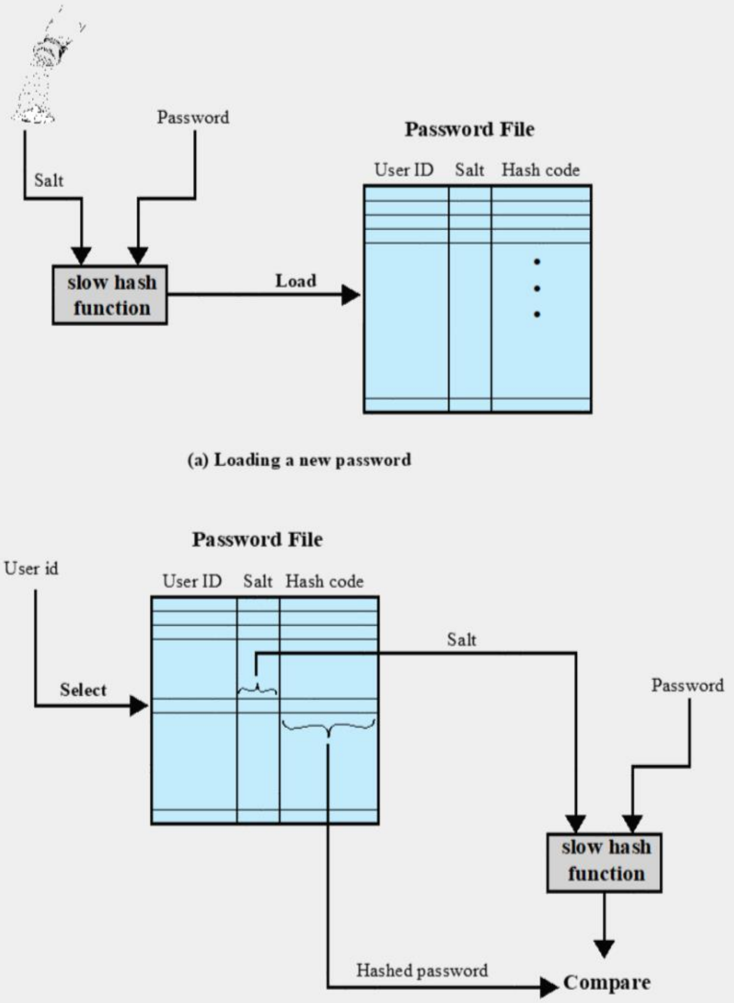
Password file

Basic password scheme

- Hash function h : strings → strings
  - Given h(password), hard to find password
  - No known algorithm better than trial and error
- User password stored as h(password)
- When user enters password
  - System computes h(password)
  - Compares with entry in password file
- No passwords stored on disk

Unix password system

- In past UNIX systems, password used modified DES (encryption algorithm) as if it were a hash function
  - Encrypts NULL string using password as the key (truncates passwords to 8 characters!)
  - Caused artificial slowdown: ran DES 25 times
- Also stored password file in directory: /etc/passwd/
  - World-readable (anyone who accessed the machine would be able to copy the password file to crack at their leisure)
  - Contained userIDs/groupIDs used by many system programs
  - Can instruct modern UNIXes to use MD5 hash function

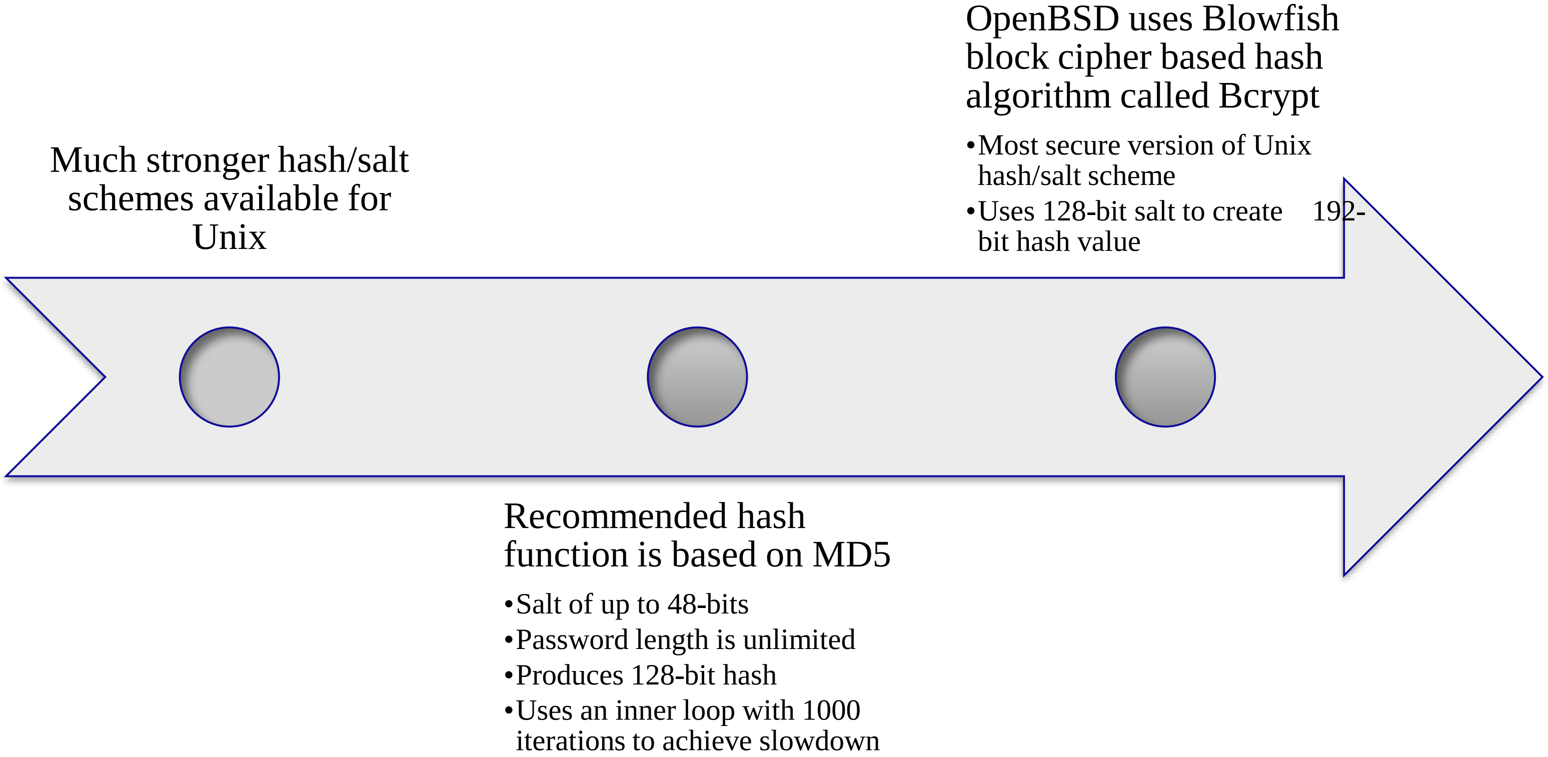


(a) Loading a new password

(b) Verifying a password

Figure 3.3 UNIX Password Scheme

Improved Implementations



Much stronger hash/salt schemes available for Unix

Recommended hash function is based on MD5

- Salt of up to 48-bits
- Password length is unlimited
- Produces 128-bit hash
- Uses an inner loop with 1000 iterations to achieve slowdown

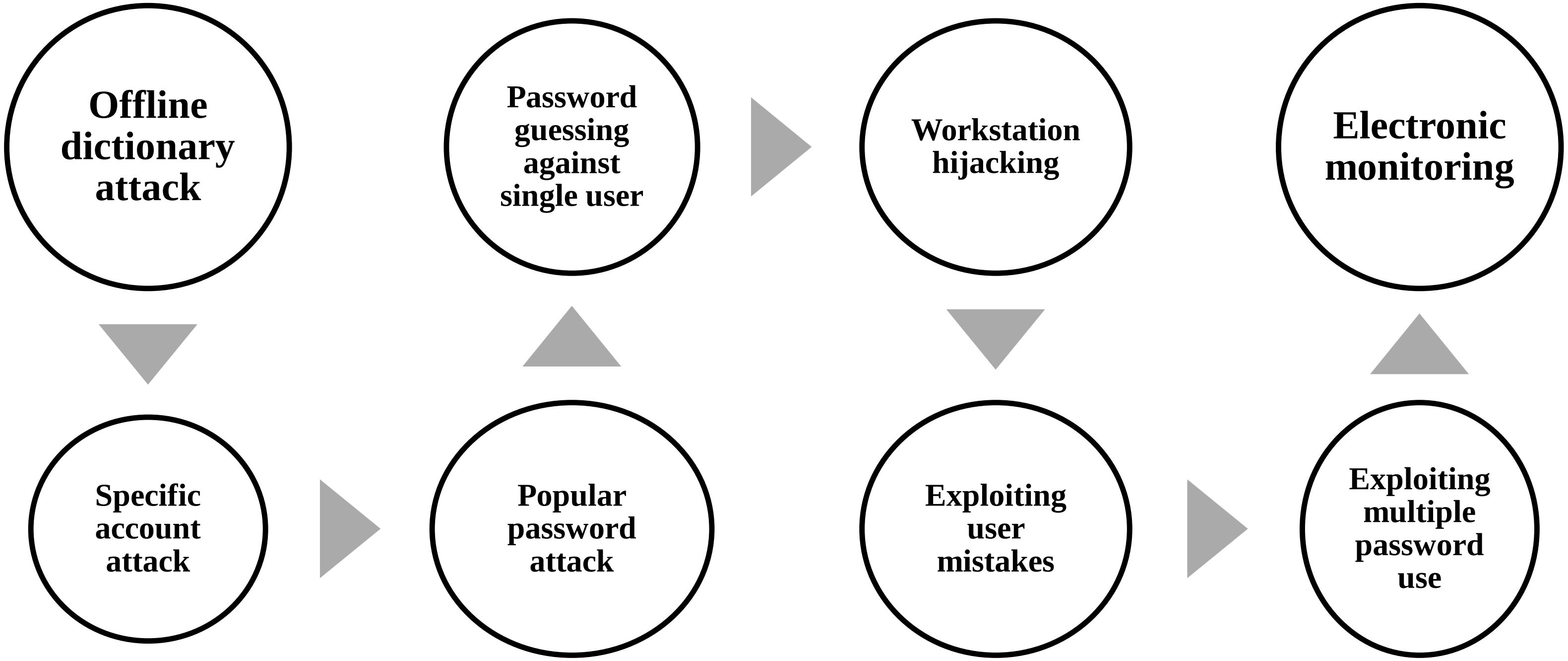
OpenBSD uses Blowfish block cipher based hash algorithm called Bcrypt

- Most secure version of Unix hash/salt scheme
- Uses 128-bit salt to create 192-bit hash value

Windows NT/2k/XP/Vista Password

- Uses 2 functions for “hashing” passwords:
  1. LAN Manager hash (LM hash)
     - Password is padded with zeros until there are 14 characters.
     - It is then converted to uppercase and split into two 7-character pieces
     - Each half is encrypted using an 8-byte DES (data encryption standard) key
     - Result is combined into a 16-byte, one way hash value
  2. NT hash (NT hash)
     - Converts password to Unicode and uses MD4 hash algorithm to obtain a 16-byte value
- Hashes stored in Security Accounts Manager (SAM)
  - Locked within system kernel when system is running.
  - Location - C:\WINNT\SYSTEM32\CONFIG
- SYSKEY
  - Utility which moves the encryption key for the SAM database off of the computer

Password Vulnerabilities



Offline dictionary attack

Specific account attack

Popular password attack

Password guessing against single user

Workstation hijacking

Exploiting user mistakes

Exploiting multiple password use

Electronic monitoring

Password selection

- User selection of passwords is typically very poor
  - Lower entropy password makes dictionary attacks easier
- Typical passwords:
  - Derived from account names or usernames
  - Dictionary words, reversed dictionary words, or small modifications of dictionary words
- Users typically use the same password for multiple accounts
  - Weakest account determines the security!

Better password selection

- Non-alphanumeric characters
- Longer phrases
- Can try to enforce good password selection…
- …but these types of passwords are difficult for people to memorize and type!

Dictionary Attack – some numbers

- Typical password dictionary
  - 1,000,000 entries of common passwords
    - people's names, common pet names, and ordinary words.
  - Suppose you generate and analyze 10 guesses per second
    - This may be reasonable for a web site; offline is *much* faster
  - Dictionary attack in at most 100,000 seconds = 28 hours, or 14 hours on average
- If passwords were random
  - Assume six-character password
    - Upper- and lowercase letters, digits, 32 punctuation characters
    - 689,869,781,056 password combinations.
    - Exhaustive search requires 1,093 years on average

Password-based protocols

- Any password-based protocol is potentially vulnerable to an "on-line" dictionary attack
  - On-line attacks can be detected and limited
- How?
  - "Three strikes"
  - Ratio of successful to failed logins
  - Gradually slow login response time
- Potential DoS
  - Cache IP address of last successful login

From passwords to keys?

- Can potentially use passwords to derive symmetric or public keys
- What is the entropy of the resulting key?

Password-based protocols

- *Off-line attacks* can never be ‘prevented’, but protocols can be made secure against such attacks
- Any password-based protocol is vulnerable to off-line attack if the server is compromised
  - Once the server is compromised, why do we care?

Password storage

- “Salt”-ed hash of password
  - Makes dictionary attacks harder,
  - Prevents using ‘rainbow tables’

Advantages of salt

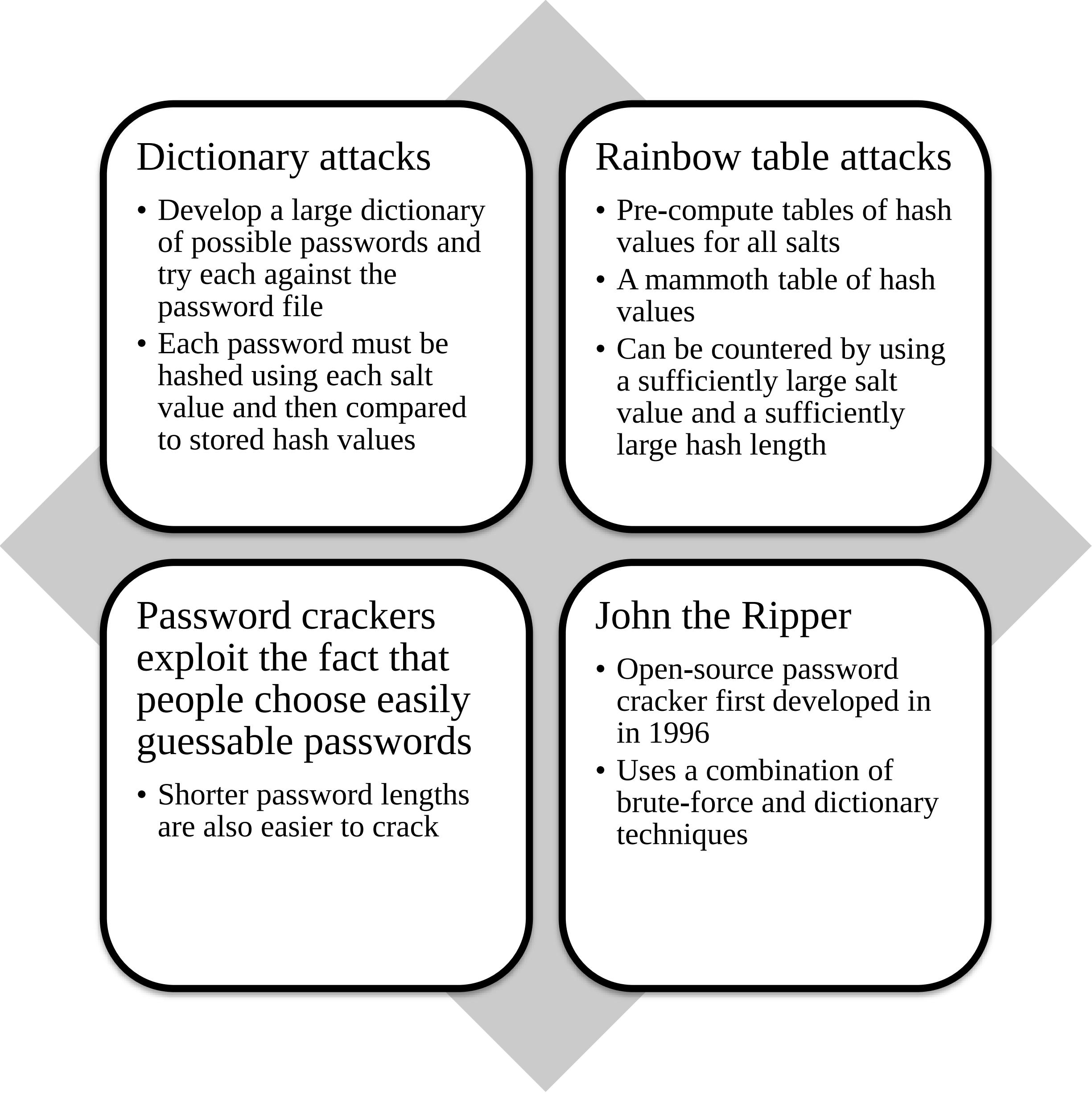
- Without salt
  - Same hash functions on all machines
    - Compute hash of all common strings once
    - Compare hash file with all known password files
- With 12 bits salt
  - One password hashed 2^{12} different ways
    - Precompute hash file?
      - Need much larger file to cover all common strings
    - Dictionary attack on known password file
      - For each salt found in file, try all common strings

One-time password

- New password obtained by passing user-password through one-way function n times which keeps incrementing

- Protects against replay as well as eavesdropping

Password Cracking



Dictionary attacks

- Develop a large dictionary of possible passwords and try each against the password file
- Each password must be hashed using each salt value and then compared to stored hash values

Rainbow table attacks

- Pre-compute tables of hash values for all salts
- A mammoth table of hash values
- Can be countered by using a sufficiently large salt value and a sufficiently large hash length

Password crackers exploit the fact that people choose easily guessable passwords

- Shorter password lengths are also easier to crack

John the Ripper

- Open-source password cracker first developed in in 1996
- Uses a combination of brute-force and dictionary techniques

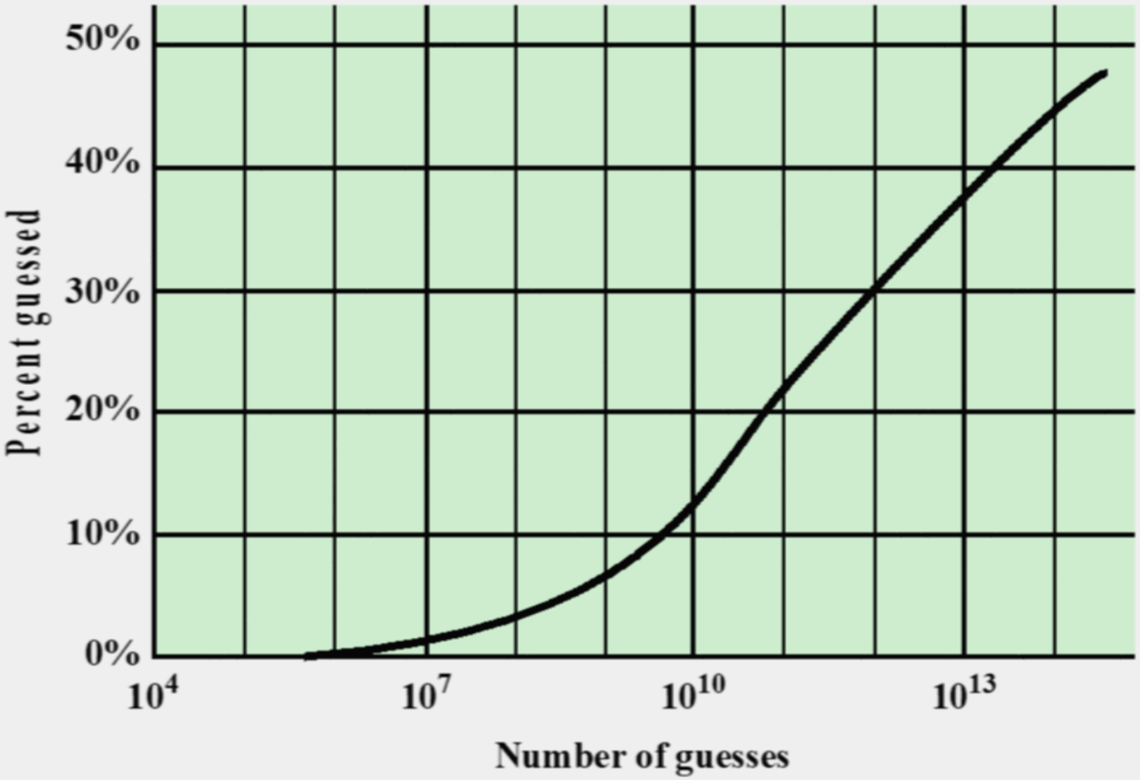


Figure 3.4 The Percentage of Passwords Guessed After a Given Number of Guesses

Passwords

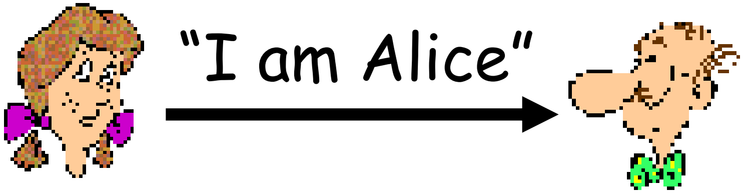
Improving Security

- Password complexity
  - Case-sensitivity
  - Use of special characters, numbers, and both upper and lower-case letters
  - Minimum length requirements
- Security questions
  - Ask personal questions which need to be verified
  - Some questions are very easy to discover answers
- Virtual keyboard
  - Person clicks on-screen keyboard to enter
  - password (prevents keylogging)

Challenge-response Authentication

Goal: Bob wants Alice to “prove” her identity to him

Protocol ap1.0: Alice says “I am Alice”

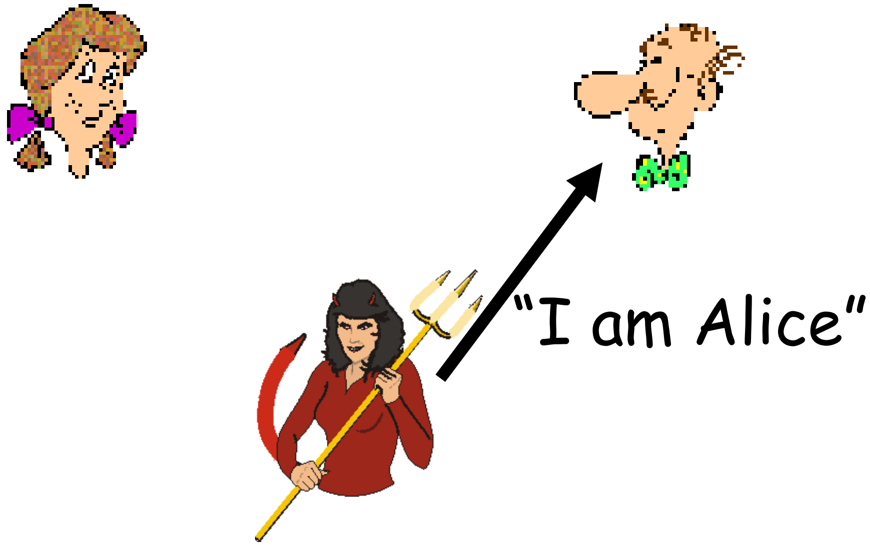


Failure scenario??

Authentication

Goal: Bob wants Alice to “prove” her identity to him

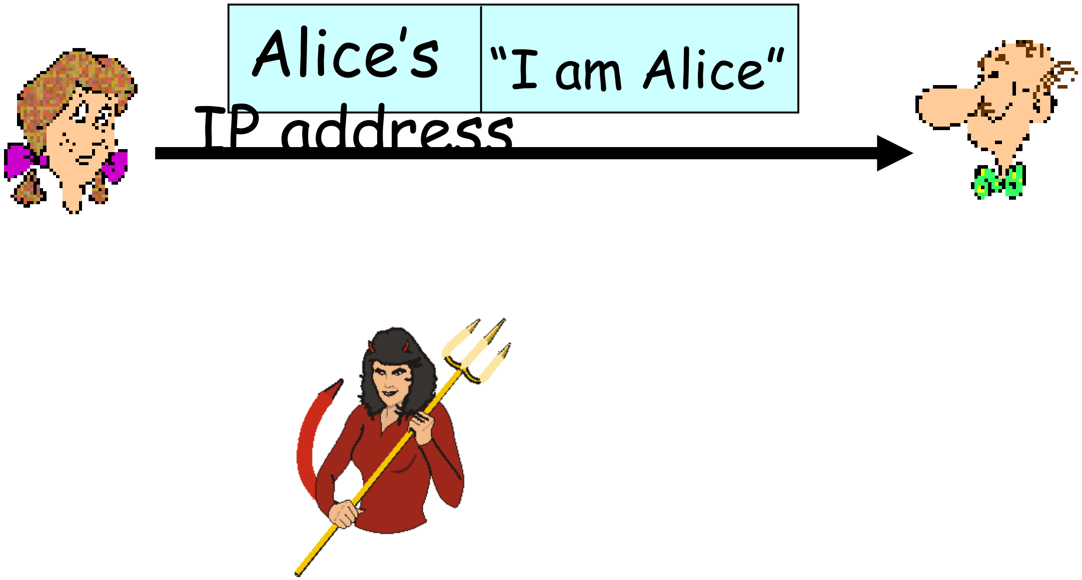
Protocol ap1.0: Alice says “I am Alice”



in a network,
Bob can not “see”
Alice, so Trudy simply
declares
herself to be Alice

Authentication: another try

Protocol ap2.0: Alice says “I am Alice” in an IP packet containing her source IP address



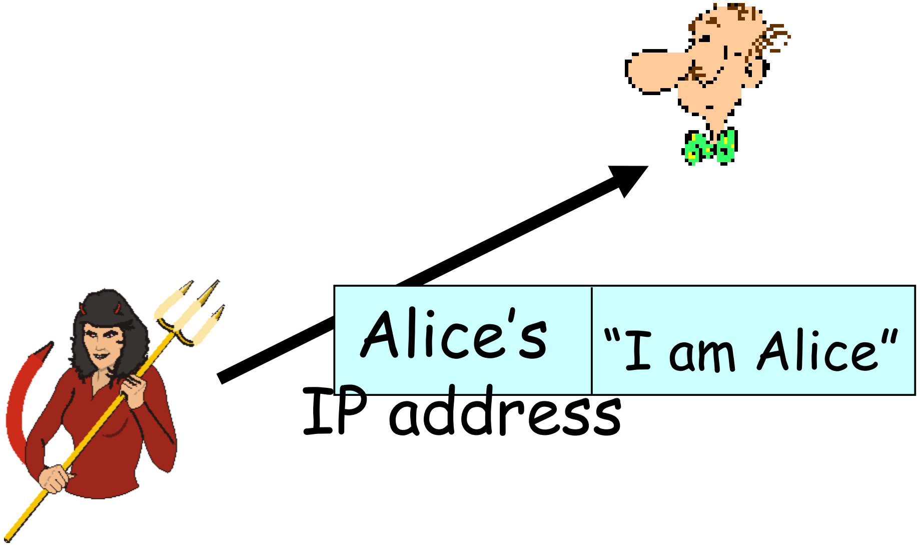
Alice's IP address

“I am Alice”

Failure scenario??

Authentication: another try

Protocol ap2.0: Alice says “I am Alice” in an IP packet
containing her source IP address



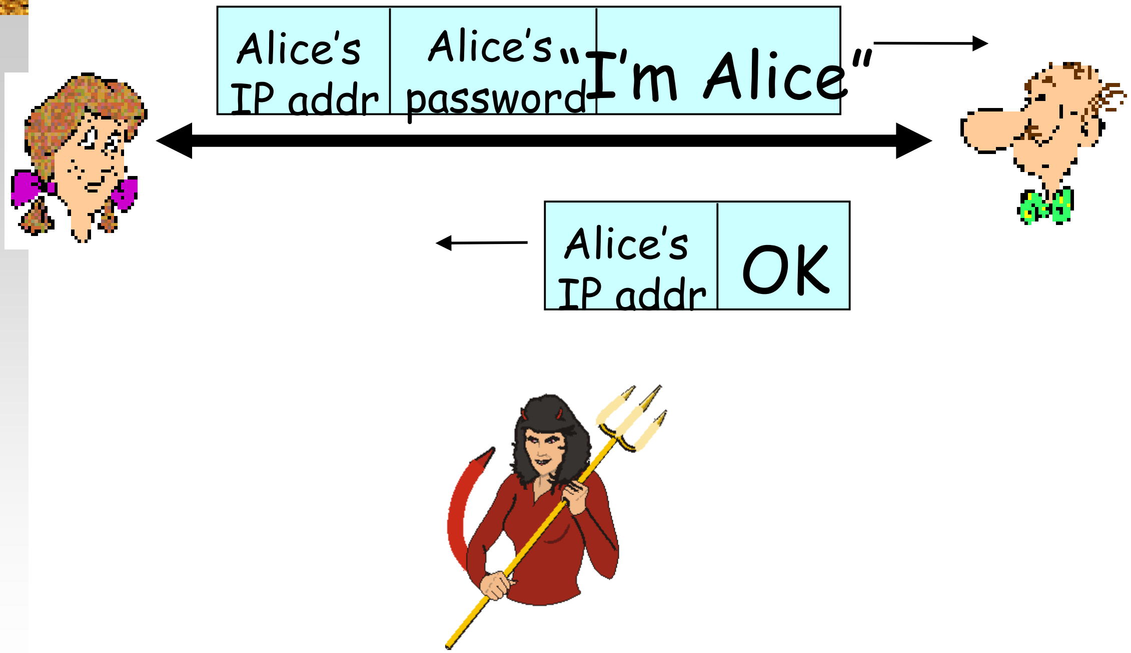
Alice’s IP address

“I am Alice”

Trudy can creat
a packet
“spoofing”
Alice’s address

Authentication: another try

Protocol ap3.0: Alice says “I am Alice” and sends her secret password to “prove” it.



Alice's IP addr

Alice's password

“I'm Alice”

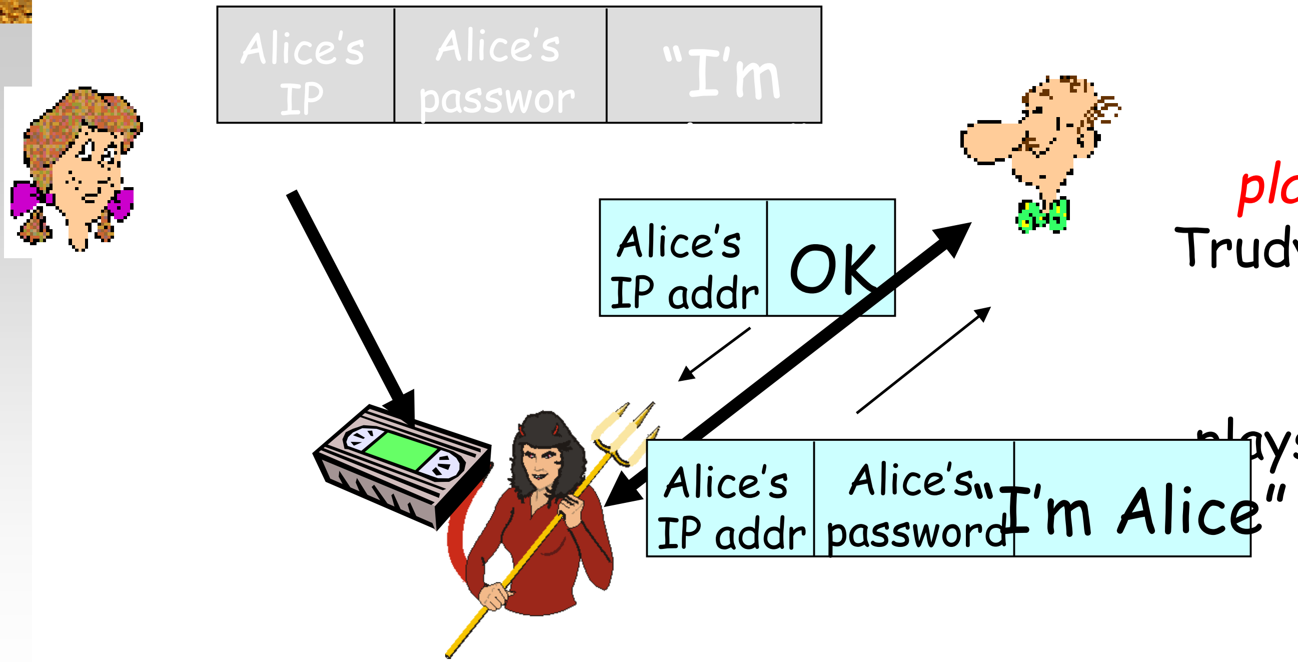
Alice's IP addr

OK

Failure scenario??

Authentication: another try

Protocol ap3.0: Alice says “I am Alice” and sends her
secret password to “prove” it.



Alice's IP | Alice's passwor | “I'm

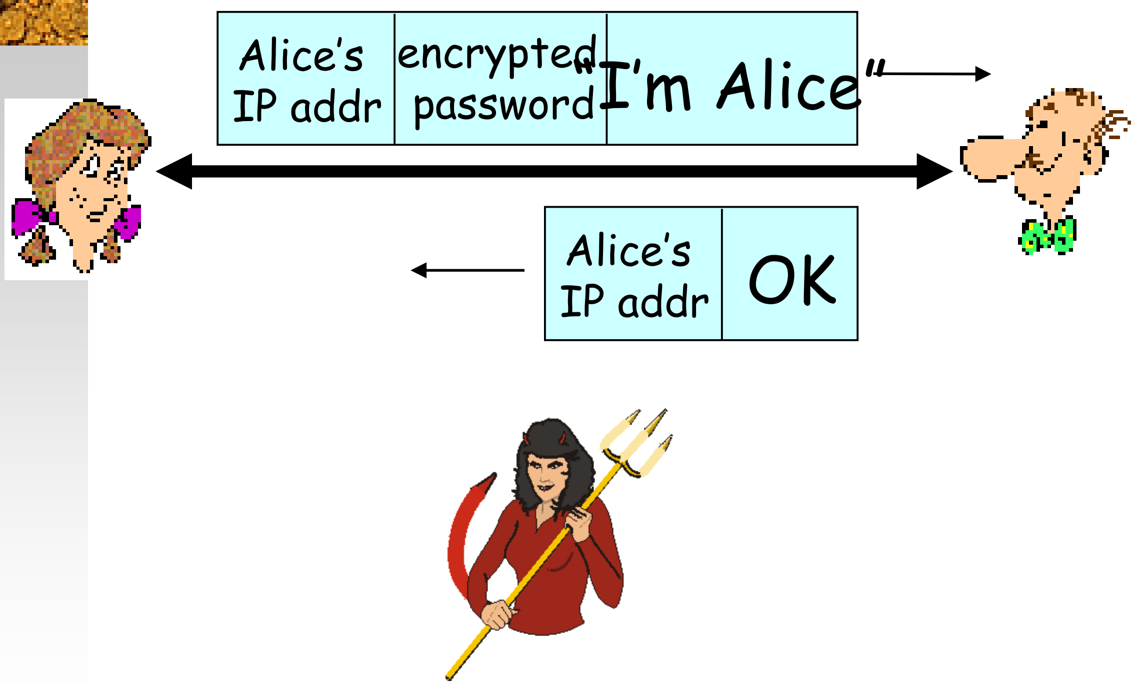
Alice's IP addr | OK

Alice's IP addr | Alice's password | “I'm Alice”

playback attack:
Trudy records Alice's packet
and later
plays it back to Bob

Authentication: yet another try

Protocol ap3.1: Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.



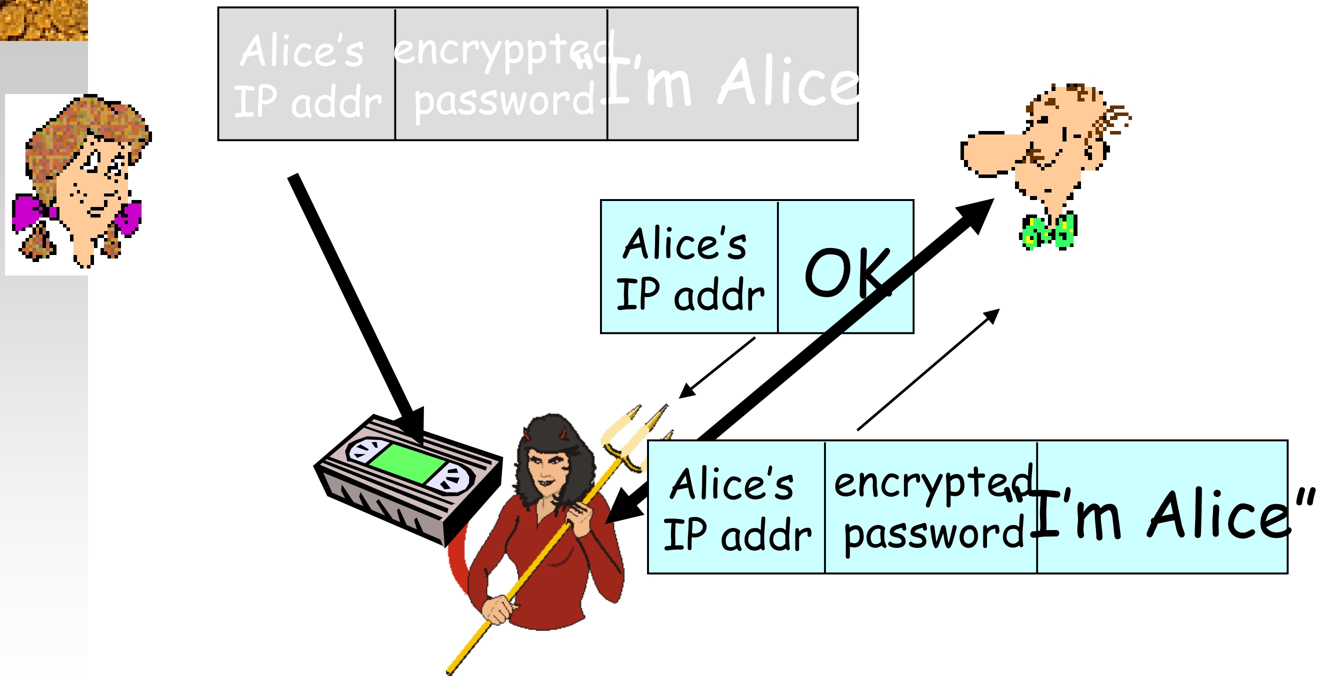
Alice's IP addr | encrypted password | "I'm Alice"

Alice's IP addr | OK

Failure scenario??

Authentication: another try

Protocol ap3.1: Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.



Alice's IP addr | encrypppted password | "I'm Alice"

Alice's IP addr | OK

Alice's IP addr | encrypted password | "I'm Alice"

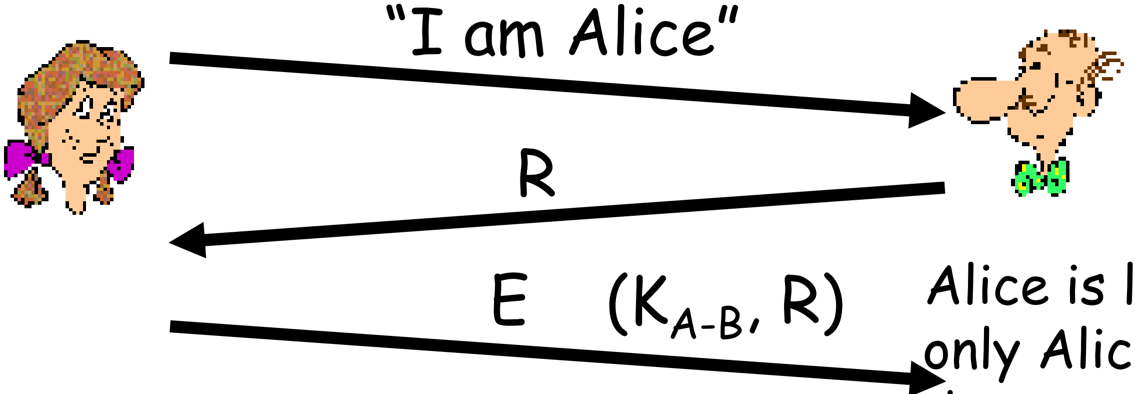
record and playback still works!

Authentication: yet another try

Goal: avoid playback attack

Nonce: number (R) used only *once –in-a-lifetime*

ap4.0: to prove Alice "live", Bob sends Alice nonce, R. Alice must return R, encrypted with shared secret key



Alice is live, and only Alice knows key to encrypt nonce, so it must be Alice!

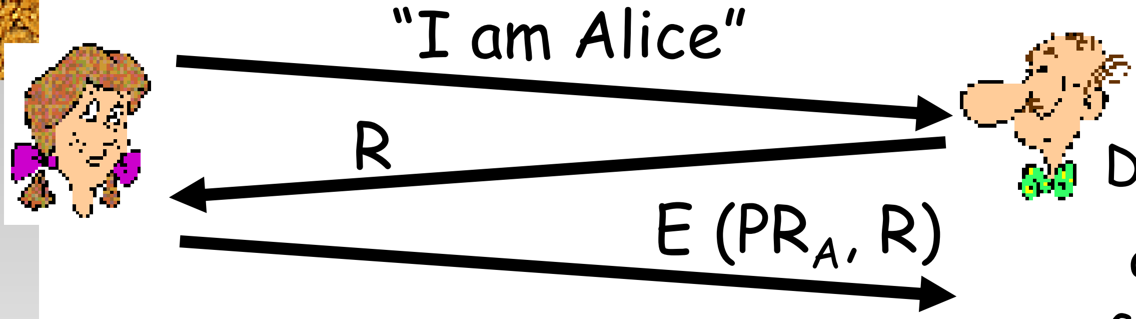
Failures, drawbacks?

Authentication: ap5.0

ap4.0 doesn't protect against server database reading

- can we authenticate using public key techniques?

ap5.0: use nonce, public key cryptography



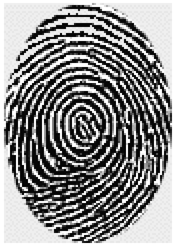
Bob computes

$D(PU_A , E(PR_A, R)) = R$

and knows only Alice could have the private key, that encrypted R such that

Biometrics

- ♦ Use a person’s physical characteristics
  - – fingerprint, voice, face, …
- ♦ Advantages
  - – Cannot be disclosed, lost, forgotten
- ♦ Disadvantages
  - – Cost, installation, maintenance
  - – Reliability of comparison algorithms
    - False positive: Allow access to unauthorized person
    - False negative: Disallow access to authorized person
  - – Privacy?
  - – If forged, how do you revoke?



Biometric Authentication

- Attempts to authenticate an individual based on unique physical characteristics
- Based on pattern recognition
- Is technically complex and expensive when compared to passwords and tokens
- Physical characteristics used include:
  - Facial characteristics
  - Fingerprints
  - Hand geometry
  - Retinal pattern
  - Iris
  - Signature
  - Voice

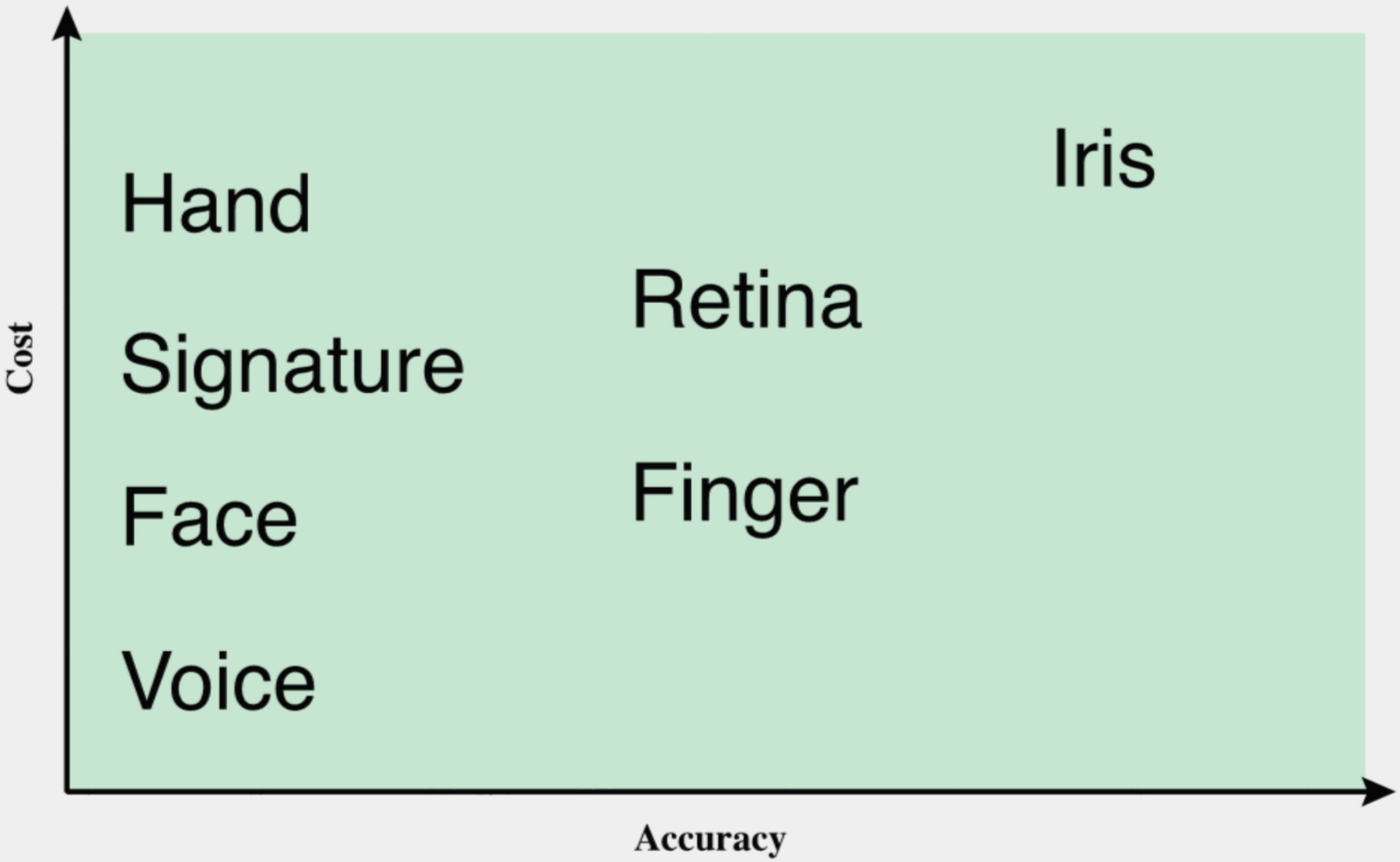


Figure 3.8 Cost Versus Accuracy of Various Biometric Characteristics in User Authentication Schemes.

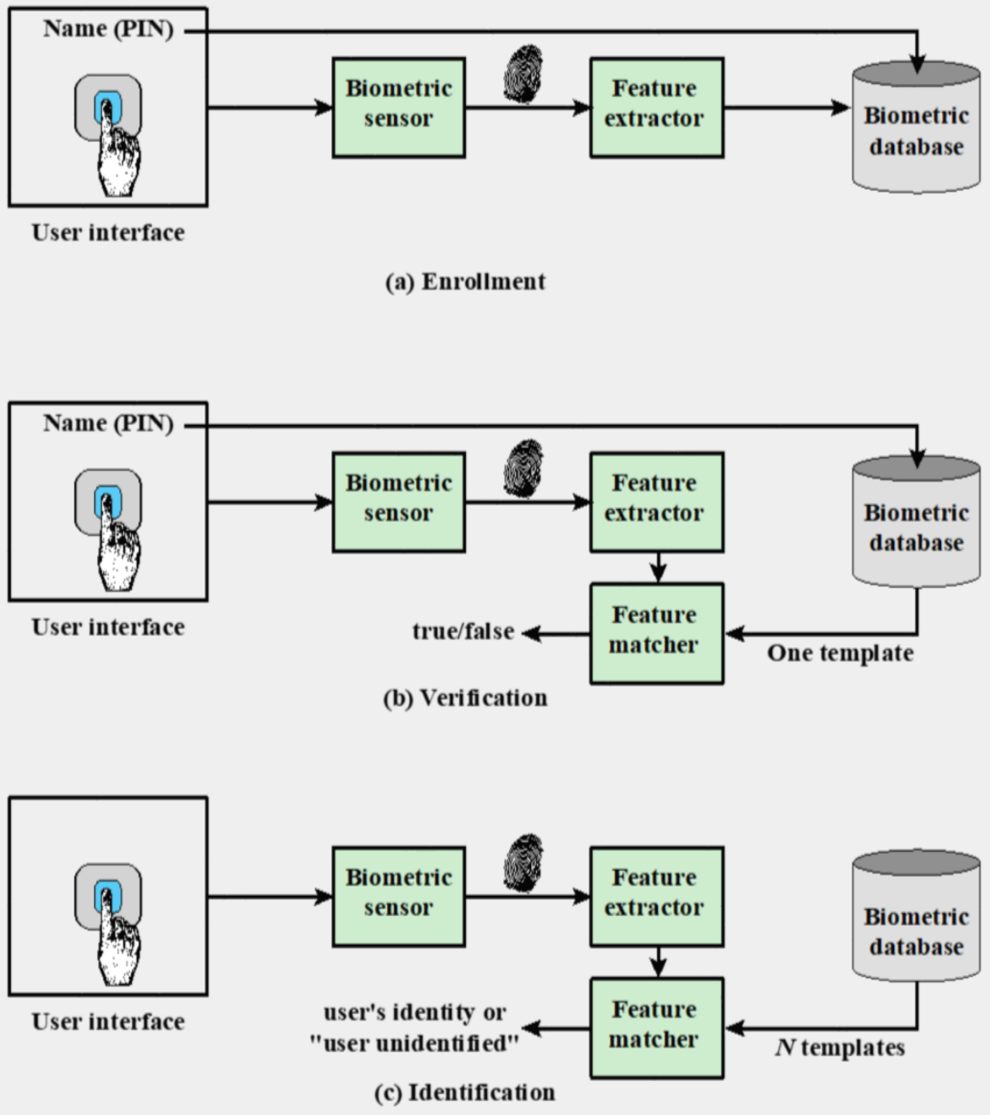


Figure 3.9 A Generic Biometric System. Enrollment creates an association between a user and the user's biometric characteristics. Depending on the application, user authentication either involves verifying that a claimed user is the actual user or identifying an unknown user.

Biometrics

- Common uses
  - Specialized situations, physical security
  - Combine
    - Multiple biometrics
    - Biometric and PIN
    - Biometric and token

Token-based Authentication

Smart Card

- ♦ With embedded CPU and memory
  - – Carries conversation w/ a small card reader
- ♦ Various forms
  - – PIN protected memory card
    - Enter PIN to get the password
  - – Cryptographic challenge/response cards
    - Computer create a random challenge
    - Enter PIN to encrypt/decrypt the challenge w/ the card

Key Distribution

- given parties A and B have various **key distribution** alternatives:

1. A can select key and physically deliver to B
2. third party can select & deliver key to A & B
3. if A & B have communicated previously can use previous key to encrypt a new key
4. if A & B have secure communications with a third party C, C can relay key between A & B

5. Using public key encryption

Trusted Intermediaries

Symmetric key problem:

- How do two entities establish shared secret key over network?

Solution:

- trusted key distribution center (KDC) acting as intermediary between entities

Public key problem:

- When Alice obtains Bob’s public key (from web site, e-mail, diskette), how does she know it is Bob’s public key, not Trudy’s?

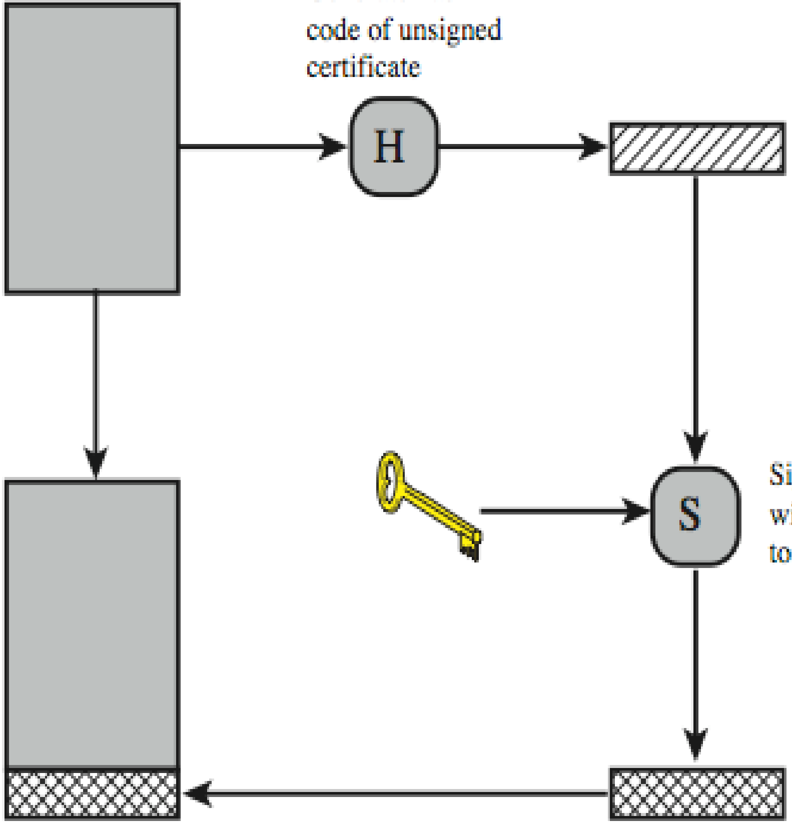
Solution:

- trusted certification authority (CA)

X.509 Certificate Use

Unsigned certificate:
contains user ID,
user's public key

Generate hash
code of unsigned
certificate



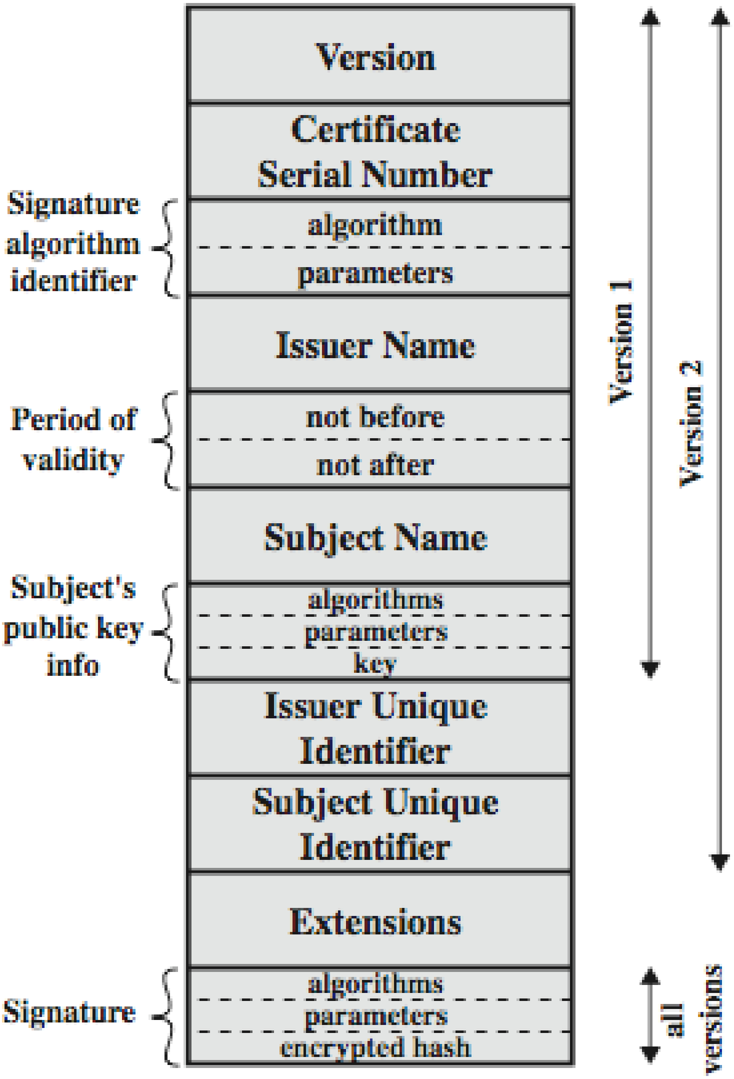
Sign hash code
with CA's private key
to form signature

Signed certificate:
Recipient can verify
signature using CA's
public key.

X.509 Certificates

- issued by a Certification Authority (CA), containing:
  - version V (1, 2, or 3)
  - serial number SN (unique within CA) identifying certificate
  - signature algorithm identifier AI
  - issuer (X.500 name CA)
  - period of validity TA (from - to dates)
  - subject X.500 name A (name of owner)
  - subject public-key info Ap (algorithm, parameters, key)
  - issuer unique identifier (v2+)
  - subject unique identifier (v2+)
  - extension fields (v3)
  - signature (of hash of all fields in certificate)
- notation `CA<<A>>` denotes certificate for A signed by CA

X.509 Certificates



Version

Certificate Serial Number

Signature algorithm identifier: algorithm, parameters

Issuer Name

Period of validity: not before, not after

Subject Name

Subject's public key info: algorithms, parameters, key

Issuer Unique Identifier

Subject Unique Identifier

Extensions

Signature: algorithms, parameters, encrypted hash

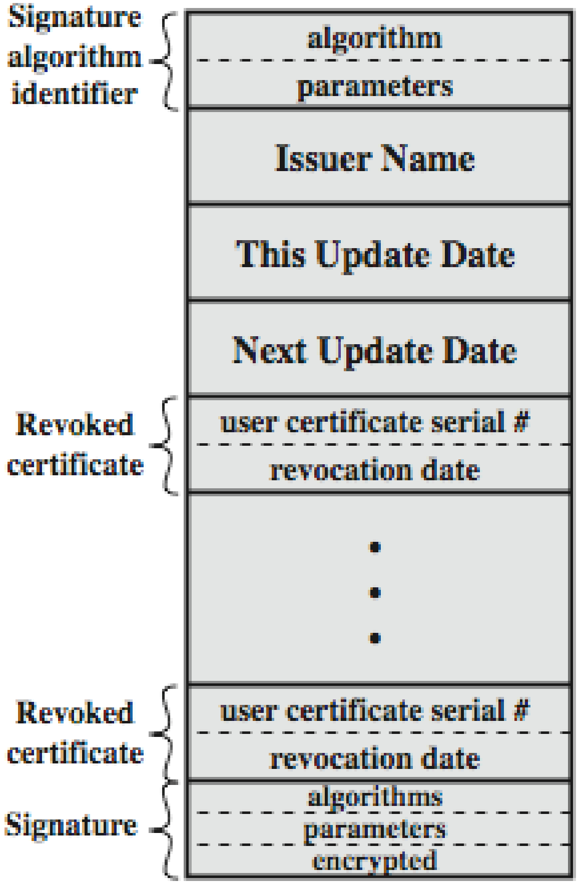
Version 1

Version 2

Version 3

all versions

(a) X.509 Certificate



Signature algorithm identifier: algorithm, parameters

Issuer Name

This Update Date

Next Update Date

Revoked certificate: user certificate serial #, revocation date

•
•
•

Revoked certificate: user certificate serial #, revocation date

Signature: algorithms, parameters, encrypted

(b) Certificate Revocation List

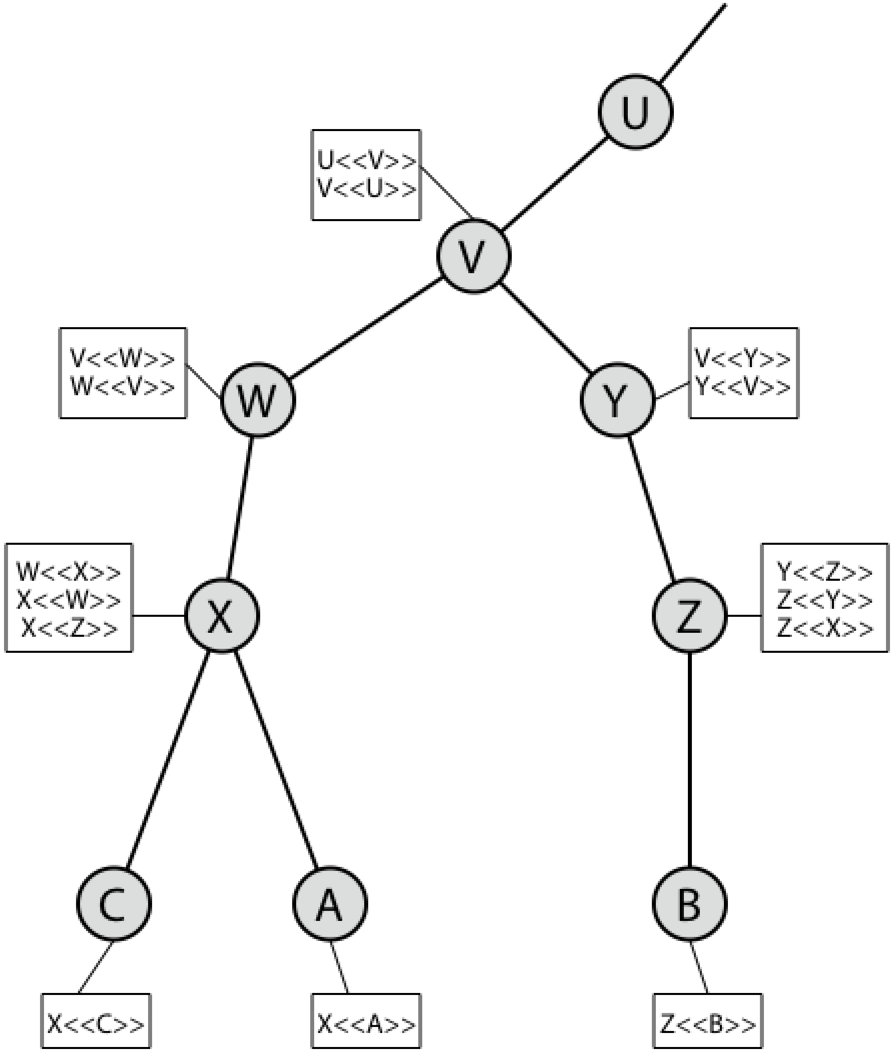
Obtaining a Certificate

- any user with access to CA can get any certificate from it
- only the CA can modify a certificate
- because cannot be forged, certificates can be placed in a public directory

CA Hierarchy

- if both users share a common CA then they are assumed to know its public key
- otherwise CA's must form a hierarchy
- use certificates linking members of hierarchy to validate other CA's
  - each CA has certificates for clients (forward) and parent (backward)
- each client trusts parents certificates
- enable verification of any certificate from one CA by users of all other CAs in hierarchy

CA Hierarchy Use



Certificate Revocation

- ♦ certificates have a period of validity
- ♦ may need to revoke before expiry, eg:
  1. user's private key is compromised
  2. user is no longer certified by this CA
  3. CA's certificate is compromised
- ♦ CA’s maintain list of revoked certificates
  - – the Certificate Revocation List (CRL)
- ♦ users should check certificates with CA’s CRL

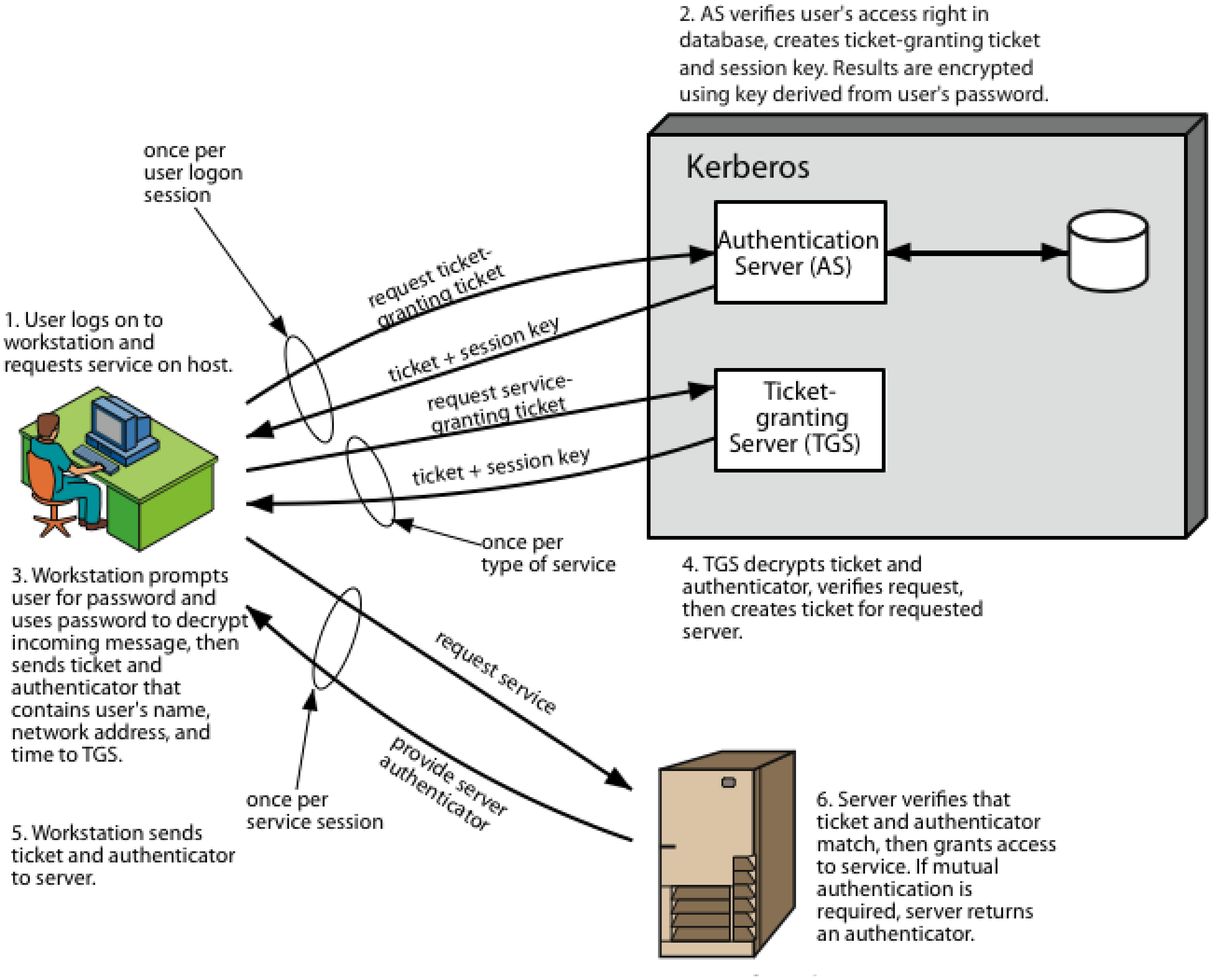
Kerberos

- trusted key server system from MIT
- provides centralised private-key third-party authentication in a distributed network
  - allows users access to services distributed through network
  - without needing to trust all workstations
  - rather all trust a central authentication server
- two versions in use: 4 & 5

Kerberos v4 Overview

- a basic third-party authentication scheme
- have an Authentication Server (AS)
  - users initially negotiate with AS to identify self
  - AS provides a non-corruptible authentication credential (ticket granting ticket TGT)
- have a Ticket Granting server (TGS)
  - users subsequently request access to other services from TGS on basis of users TGT
- using DES

Kerberos 4 Overview



1. User logs on to workstation and requests service on host.

2. AS verifies user's access right in database, creates ticket-granting ticket and session key. Results are encrypted using key derived from user's password.

3. Workstation prompts user for password and uses password to decrypt incoming message, then sends ticket and authenticator that contains user's name, network address, and time to TGS.

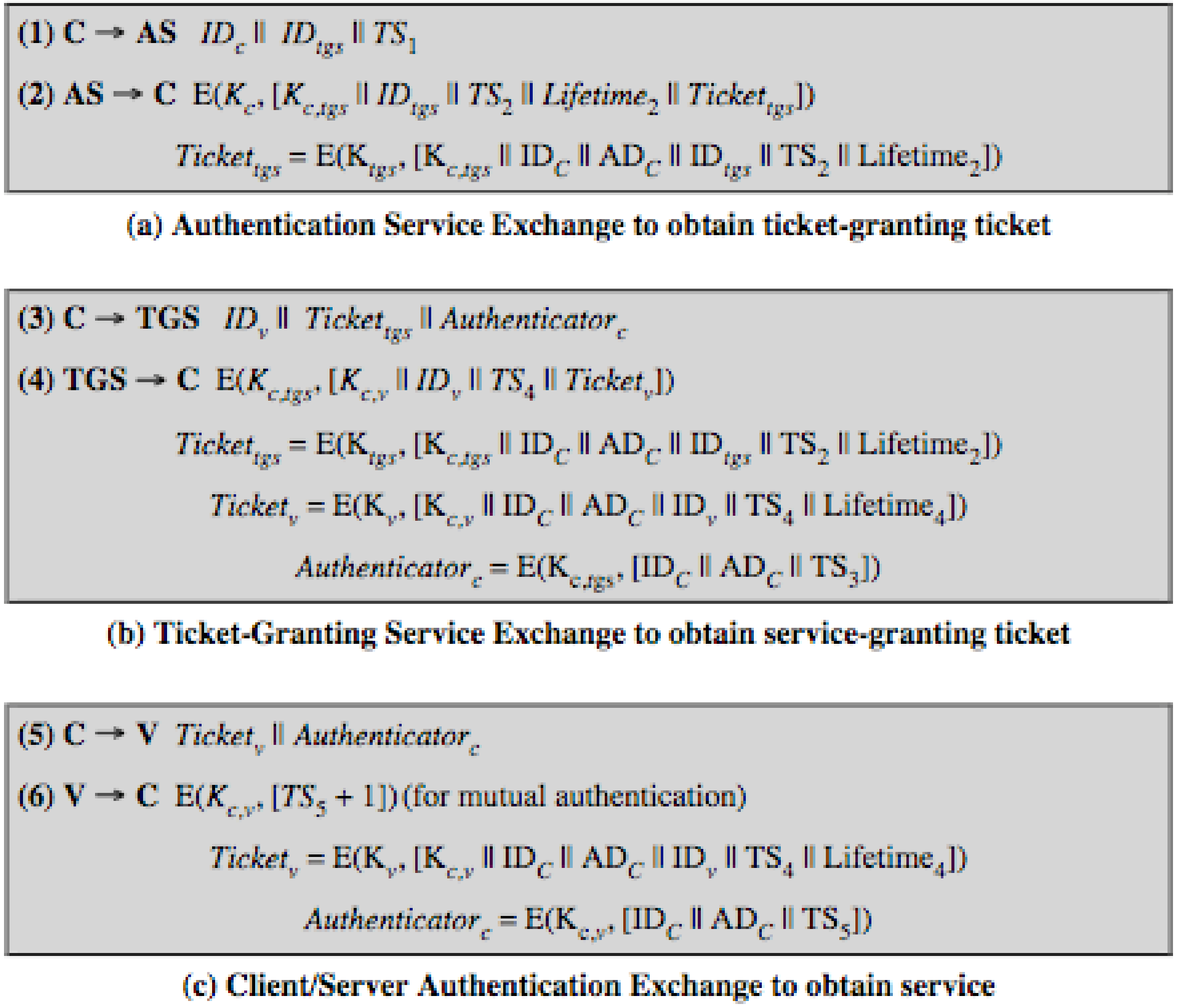
4. TGS decrypts ticket and authenticator, verifies request, then creates ticket for requested server.

5. Workstation sends ticket and authenticator to server.

6. Server verifies that ticket and authenticator match, then grants access to service. If mutual authentication is required, server returns an authenticator.

Kerberos
Authentication Server (AS)
Ticket-granting Server (TGS)
once per user logon session
request ticket-granting ticket
ticket + session key
request service-granting ticket
ticket + session key
once per type of service
request service
provide server authenticator
once per service session

Kerberos v4 Dialogue



(1) C → AS $ID_c \parallel ID_{tgs} \parallel TS_1$

(2) AS → C $\mathrm{E}(K_c, [K_{c,tgs} \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2 \parallel Ticket_{tgs}])$

$$Ticket_{tgs} = \mathrm{E}(\mathrm{K}_{tgs}, [\mathrm{K}_{c,tgs} \parallel \mathrm{ID}_C \parallel \mathrm{AD}_C \parallel \mathrm{ID}_{tgs} \parallel \mathrm{TS}_2 \parallel \mathrm{Lifetime}_2])$$

(a) Authentication Service Exchange to obtain ticket-granting ticket

(3) C → TGS $ID_v \parallel Ticket_{tgs} \parallel Authenticator_c$

(4) TGS → C $\mathrm{E}(K_{c,tgs}, [K_{c,v} \parallel ID_v \parallel TS_4 \parallel Ticket_v])$

$$Ticket_{tgs} = \mathrm{E}(\mathrm{K}_{tgs}, [\mathrm{K}_{c,tgs} \parallel \mathrm{ID}_C \parallel \mathrm{AD}_C \parallel \mathrm{ID}_{tgs} \parallel \mathrm{TS}_2 \parallel \mathrm{Lifetime}_2])$$

$$Ticket_v = \mathrm{E}(\mathrm{K}_v, [\mathrm{K}_{c,v} \parallel \mathrm{ID}_C \parallel \mathrm{AD}_C \parallel \mathrm{ID}_v \parallel \mathrm{TS}_4 \parallel \mathrm{Lifetime}_4])$$

$$Authenticator_c = \mathrm{E}(\mathrm{K}_{c,tgs}, [\mathrm{ID}_C \parallel \mathrm{AD}_C \parallel \mathrm{TS}_3])$$

(b) Ticket-Granting Service Exchange to obtain service-granting ticket

(5) C → V $Ticket_v \parallel Authenticator_c$

(6) V → C $\mathrm{E}(K_{c,v}, [TS_5 + 1])$ (for mutual authentication)

$$Ticket_v = \mathrm{E}(\mathrm{K}_v, [\mathrm{K}_{c,v} \parallel \mathrm{ID}_C \parallel \mathrm{AD}_C \parallel \mathrm{ID}_v \parallel \mathrm{TS}_4 \parallel \mathrm{Lifetime}_4])$$

$$Authenticator_c = \mathrm{E}(\mathrm{K}_{c,v}, [\mathrm{ID}_C \parallel \mathrm{AD}_C \parallel \mathrm{TS}_5])$$

(c) Client/Server Authentication Exchange to obtain service

Kerberos Realms

- a Kerberos environment consists of:
  - a Kerberos server
  - a number of clients, all registered with server
  - application servers, sharing keys with server
- this is termed a realm
  - typically a single administrative domain
- if have multiple realms, their Kerberos servers must share keys and trust

LAN Security

Modified By: Dr. Ramzi Saifan

LAN

- Many data traffic is available to every node in the LAN zone
- NIC provides physical and logical conversions.
- Every NIC has an address.
- When messages are inserted in the network, the address of the destination NIC is part of the message header.
- As messages flow through an NIC, the destination address is examined.
- If the destination address matches the NIC doing the examining, the message is transmitted to upper layers.
- It is also easy to provide broadcast communication to all NICs by using a special address such as the binary value of all ones.

LAN simplicity-security tradeoff

- There are many reasons why LANs have become popular,
  - the most important is flexibility and cost.
  - New NICs may be added to the net or activated,
  - or NICs may be removed or deactivated without making a significant change.
  - This dynamic flexibility happens without notification and coordination with a central authority.
- A PC can record all the communications traffic. Address filtering can be turned off.
  - The NIC can operate in “promiscuous” or “snooper” mode, passing all traffic to the PC, which in turn can record it for some future use.

Wiretapping

- Wiretapping is conventionally subdivided into passive and active categories.
- In passive, the message traffic is observed but not modified.
  - The most obvious objective of passive wiretapping is to learn the contents of messages;
  - traffic analysis may provide the adversary with information when message content is not available.
  - E.g., sudden change in traffic volume between national central banks, might signal a change in the rate of exchange or some other financial activity that could be turned into a profit by someone.
- In active wiretapping: Messages can be completely deleted, they can be inserted, or their contents can be modified.
  - Delay, reordering, duplication, and retransmission are also possible.

Packet Sniffing

- This works for wireless too!
- In fact, it works for any broadcast-based medium

Packet Sniffing Countermeasures

- How can we protect ourselves?
- SSH, not Telnet
  - Many people are still using Telnet and send their password in the clear (use PuTTY instead!)
  - Now that I have told you this, please do not exploit this information
  - Packet sniffing is, by the way, prohibited by Computing Services
- HTTP over SSL
  - Especially when making purchases with credit cards!
- SFTP, not FTP
  - Unless you ***really*** don’t care about the password or data
  - Can also use KerbFTP (download from MyAndrew)
- IPSec
  - Provides network-layer confidentiality

Switch Learning Attacks

- Switch learning is what makes Ethernet scale
- Two key attacks: MAC flooding and spoofing
  - Extremely simple to carry out, yet very potent
  - Can help attacker collects usernames/passwords, prevent proper operation of LAN, etc
  - Can turn a $50,000 switch into a $12 hub

Limitations on switch memory

- High end switches can store hundreds of thousands of learning table entries
- What happens if learning table fills up?
- Depends on vendor –
  - Most Cisco switches do not replace older entries with new ones.
    - Need to “age out” entries (wait for them to time out)
  - Other switches circular buffer
    - Existing entries get overwritten

MAC Flooding Attack

- Problem: attacker can cause learning table to fill – Generate many packets to varied (perhaps nonexistant) MAC addresses
- This harms efficiency
  - Effectively transforms switch into hub
  - Wastes bandwidth, and end-host CPU
- This harms privacy
  - Attacker can eavesdrop by preventing switch from learning destination of a flow
  - Causes flow’s packet to be flooded throughout LAN

MAC Spoofing Attack

- Host pretends to own the MAC address of another host
  - Easy to do: most Ethernet adapters allow their address to be modified
  - Powerful: can immediately cause complete DoS to spoofed host
    - All learning table entries point to the attacker
    - All traffic redirected to attacker
    - Can enable attacker to evade ACLs set based on MAC information

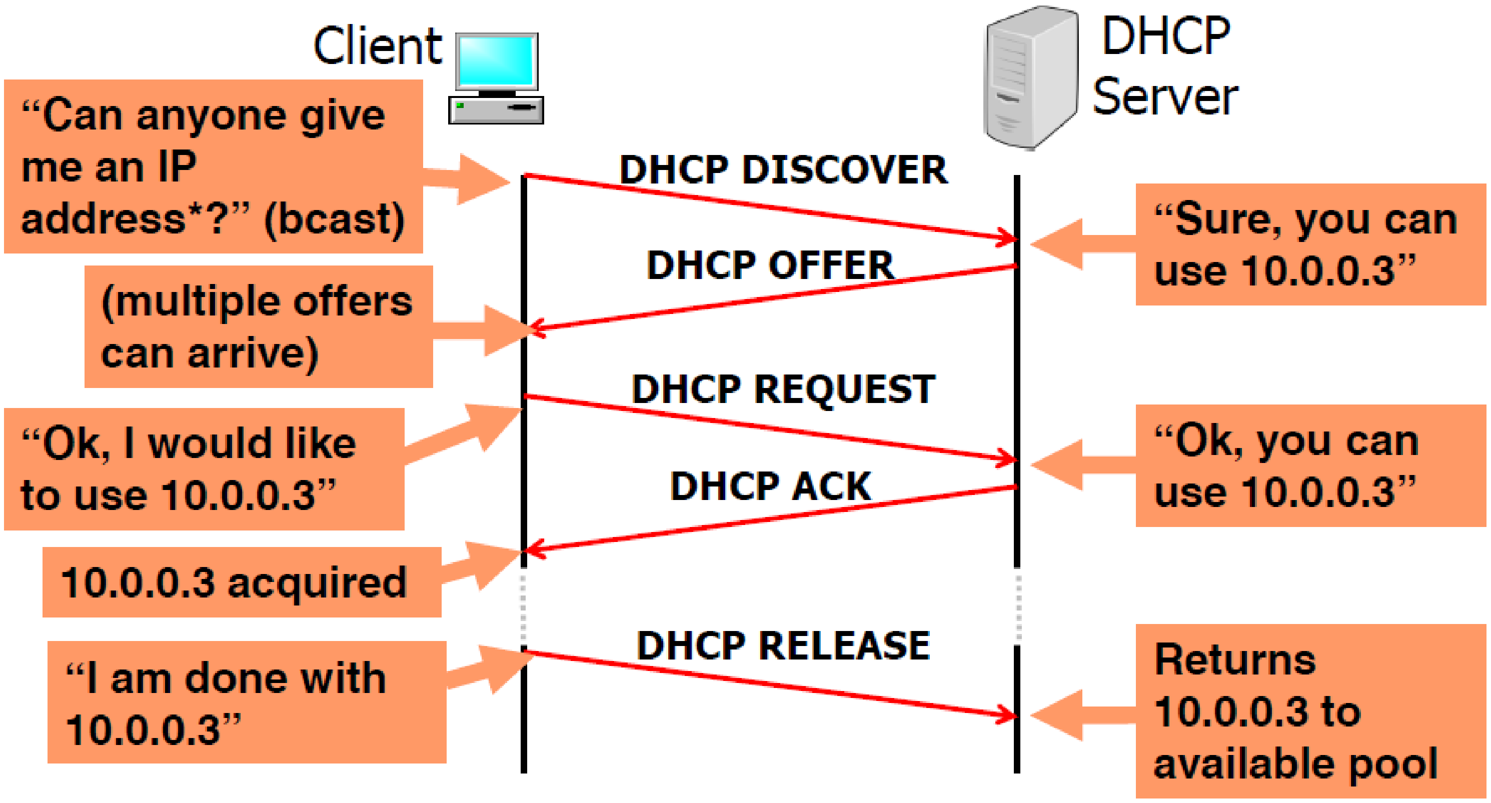
Switch Learning Attacks: Countermeasures

- ♦ Detecting MAC activity
  - – Many switches can be config'd to warn administrator about many sudden MAC address moves
- ♦ Port Security
  - – Ties a given MAC address to a port
  - – On violation, can drop frames, disable port for specified duration, signal alarm, increment violation counter

Switch Learning Attacks: Countermeasures

- Unicast Flooding Protection
  - Send alert when user-defined rate limit is exceeded
  - Can also filter traffic or shut down port generating excessive floods

DHCP



Client

DHCP Server

"Can anyone give me an IP address*?" (bcast)

DHCP DISCOVER

"Sure, you can use 10.0.0.3"

DHCP OFFER

(multiple offers can arrive)

DHCP REQUEST

"Ok, I would like to use 10.0.0.3"

"Ok, you can use 10.0.0.3"

DHCP ACK

10.0.0.3 acquired

DHCP RELEASE

"I am done with 10.0.0.3"

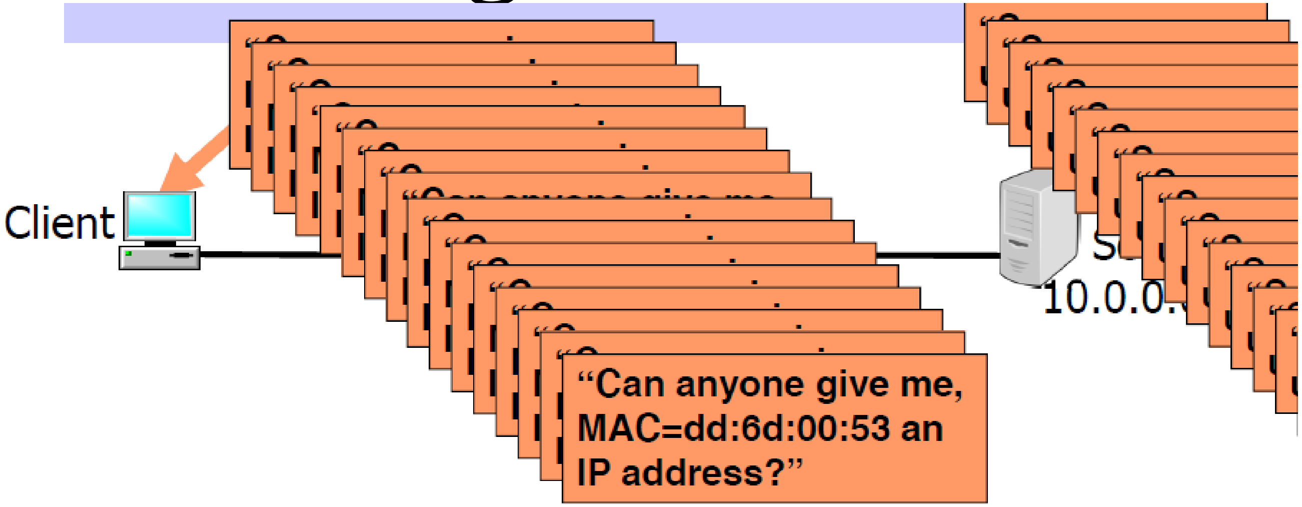
Returns 10.0.0.3 to available pool

*and other config information

Attacks on DHCP

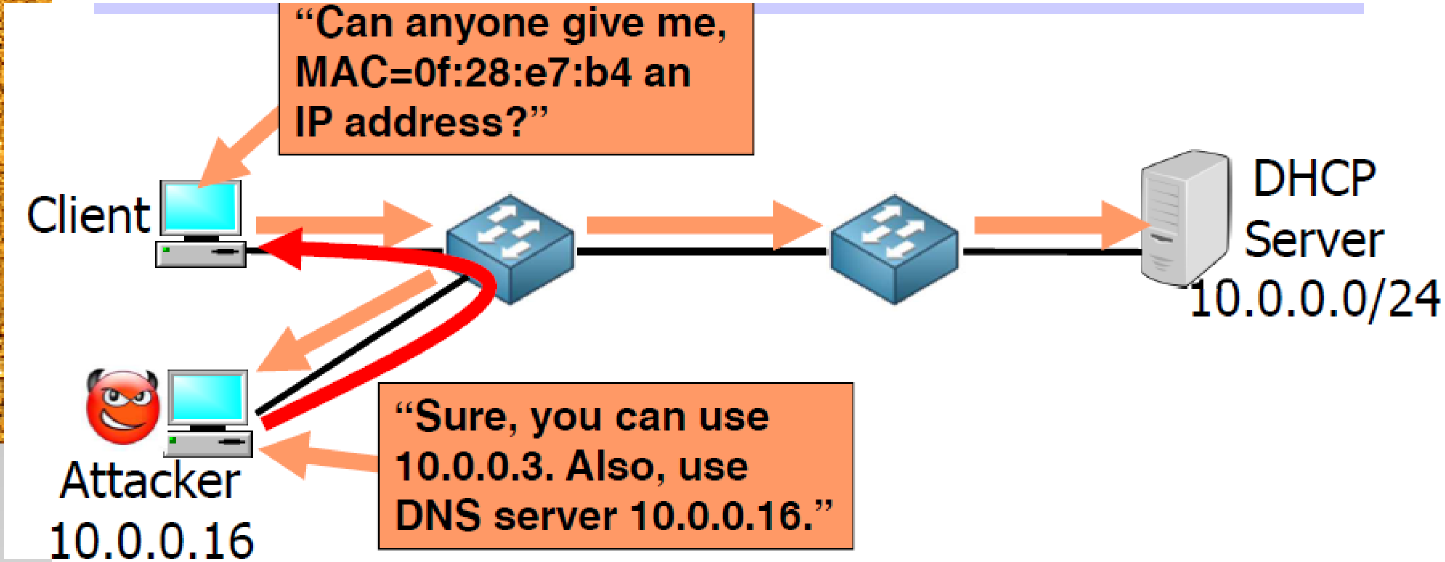
- Unfortunately, DHCP was designed without security in mind
  - Whoever requests an address is free to receive one
  - No authentication fields or any other security-inclined information in protocol

Attacks against DHCP



- DHCP Scope Exhaustion
  - Malicious client attempts to seize entire range of IP addresses
    - When legitimate client tries, it is abandoned with no IP connectivity

Attack: Rogue DHCP Server



“Can anyone give me, MAC=0f:28:e7:b4 an IP address?”

Client

DHCP Server 10.0.0.0/24

Attacker 10.0.0.16

“Sure, you can use 10.0.0.3. Also, use DNS server 10.0.0.16.”

- Installation of a Rogue DHCP Server
  - Client uses offer or of previously-used IP address, if none then uses first-received response:
    - Rogue can compromise all clients “near” itself

Countermeasures to DHCP Attacks

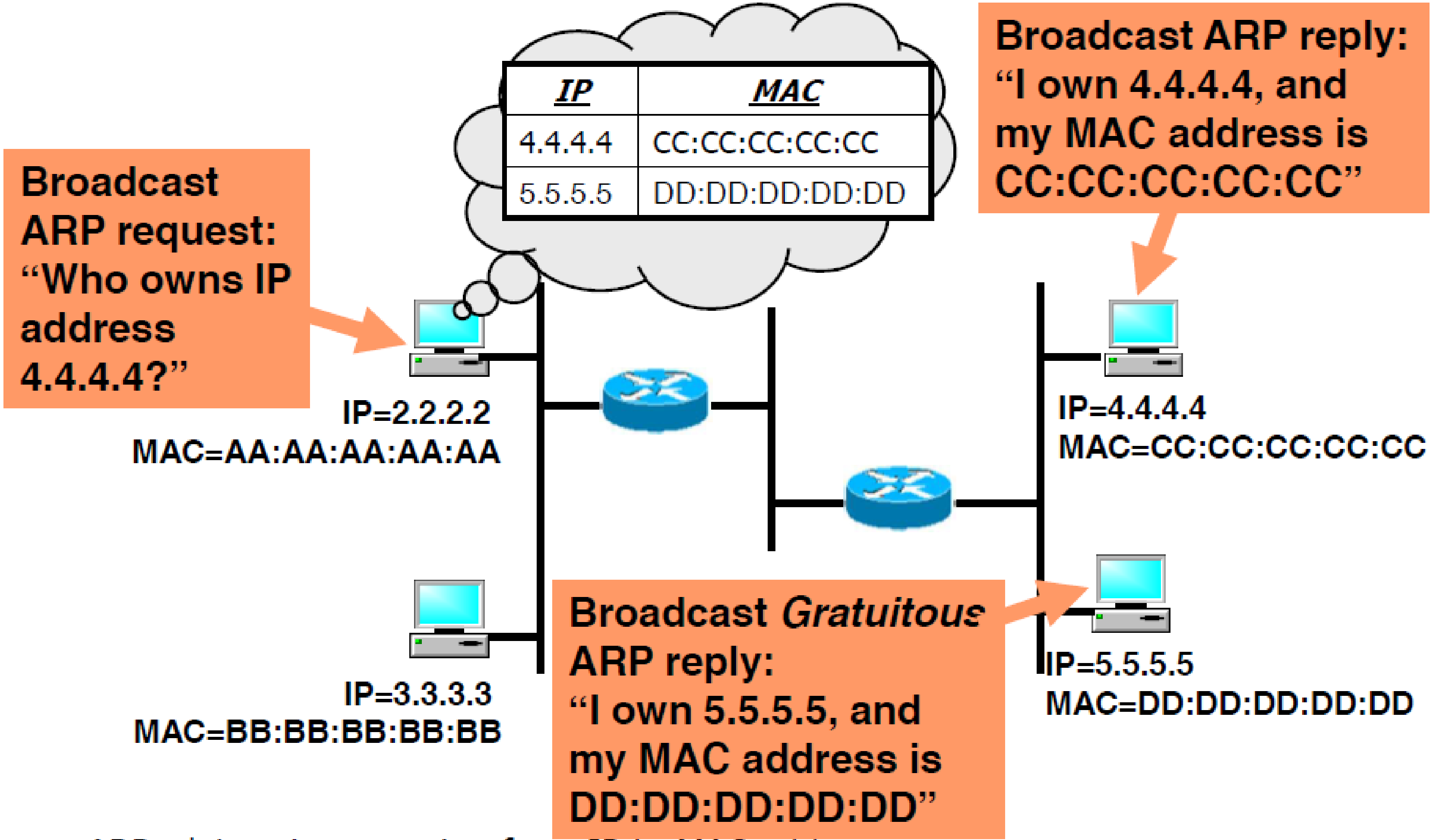
- Limit number or set of MAC addresses per port
  - This is called Port Security
  - Limit can be set manually or switch can be instructed to lock down on first dynamically learned address
- Limitations
  - DHCP lets you request multiple IP addresses for a single MAC address

Countermeasures to DHCP Attacks

- Prevent hosts from generating certain DHCP messages (DHCP Snooping)
  - Like a stateful firewall for DHCP
  - Runs on router's central management processor, to do deep packet inspection
  - Learns IP-to-MAC bindings by snooping on DHCP packets
  - Rules:
    - If port is connected to host, don't allow DHCPOFFER and DHCPACK packets
    - Don't allow DHCP packets that don't match learned bindings
    - Can also rate-limit DHCP messages per port, etc

Address Resolution Protocol (ARP)

- Networked applications are programmed to deal with IP addresses
- But Ethernet forwards to MAC address
- How can OS know the MAC address corresponding to a given IP address?
- Solution: Address Resolution Protocol
  - Broadcasts ARP request for MAC address owning a given IP address



IP	*MAC*
4.4.4.4	CC:CC:CC:CC:CC
5.5.5.5	DD:DD:DD:DD:DD

Broadcast ARP request: "Who owns IP address 4.4.4.4?"

Broadcast ARP reply: "I own 4.4.4.4, and my MAC address is CC:CC:CC:CC:CC"

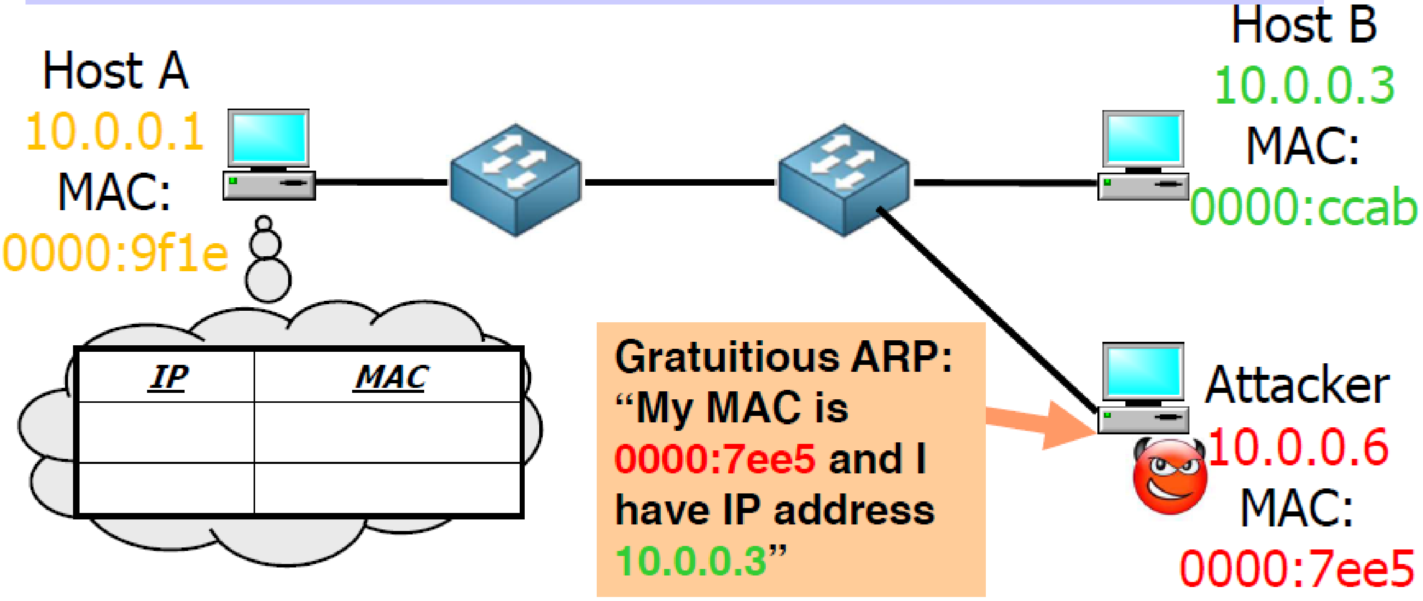
Broadcast *Gratuitous* ARP reply: "I own 5.5.5.5, and my MAC address is DD:DD:DD:DD:DD"

- ARP: determine mapping from IP to MAC address
- What if IP address not on subnet?
  - Each host configured with "default gateway", use ARP to resolve its IP address
- Gratuitous ARP: tell network your IP to MAC mapping
  - Used to detect IP conflicts, IP address changes; update other machines' ARP tables, update bridges' learned information

Risk Analysis for ARP

- No authentication
  - Hosts do not sign ARP replies
- Information leak
  - All hosts in same VLAN learn the advertised <IP,MAC> mapping
  - All hosts discover querying host wishes to communicate with replying host
- Availability
  - All hosts on same LAN receive ARP request, must process it in software
  - Attacker could send high rate of spurious ARP requests, overloading other hosts

ARP Spoofing Attack



IP	MAC

Gratuitious ARP: "My MAC is 0000:7ee5 and I have IP address 10.0.0.3"

- Attacker sends fake unsolicited ARP replies
  - Attacker can intercept forward-path traffic
  - Can intercept reverse-path traffic by repeating attack for source
  - Gratuitious ARPs make this easy
  - Only works within same subnet/VLAN

Countermeasures to ARP Spoofing

- Ignore Gratuitious ARP
  - Problems: gratuitious ARP is useful, doesn't completely solve the problem
- Dynamic ARP Inspection (DAI)
  - Switches record <IP,MAC> mappings learned from DHCP messages, drop all mismatching ARP replies
- Intrusion detection systems (IDS)
  - Monitor all <IP,MAC> mappings, signal alarms

SSL and IPSec

Modified by: Dr. Ramzi Saifan

Network layers

- Application
- Transport
- Network
- Data link
- Physical

Example security protocols

- Application layer: PGP
- Transport layer: SSL/TLS
- Network layer: IPsec
- Data link layer: IEEE 802.11
- Security at the physical layer?

Security in what layer?

- Depends on the purpose…
  - What information needs to be protected?
  - Who shares keys in advance?
  - Should the user be involved?
- E.g., a network-layer protocol cannot authenticate two end-users to each other
- An application-layer protocol cannot protect IP header information
- Also affects efficiency, ease of deployment, etc.

Generally…

- When security is placed at lower levels, it can provide automatic, “blanket” coverage…
  - …but it can take a long time before it is widely adopted
- When security is placed at higher levels, individual users can choose when to use it…
  - …but users who are not security-conscious may not take advantage of it

Note…

- The “best” solution is *not* necessarily to use PGP over IPsec!
  - Would have been better to design the Internet with security in mind from the beginning…

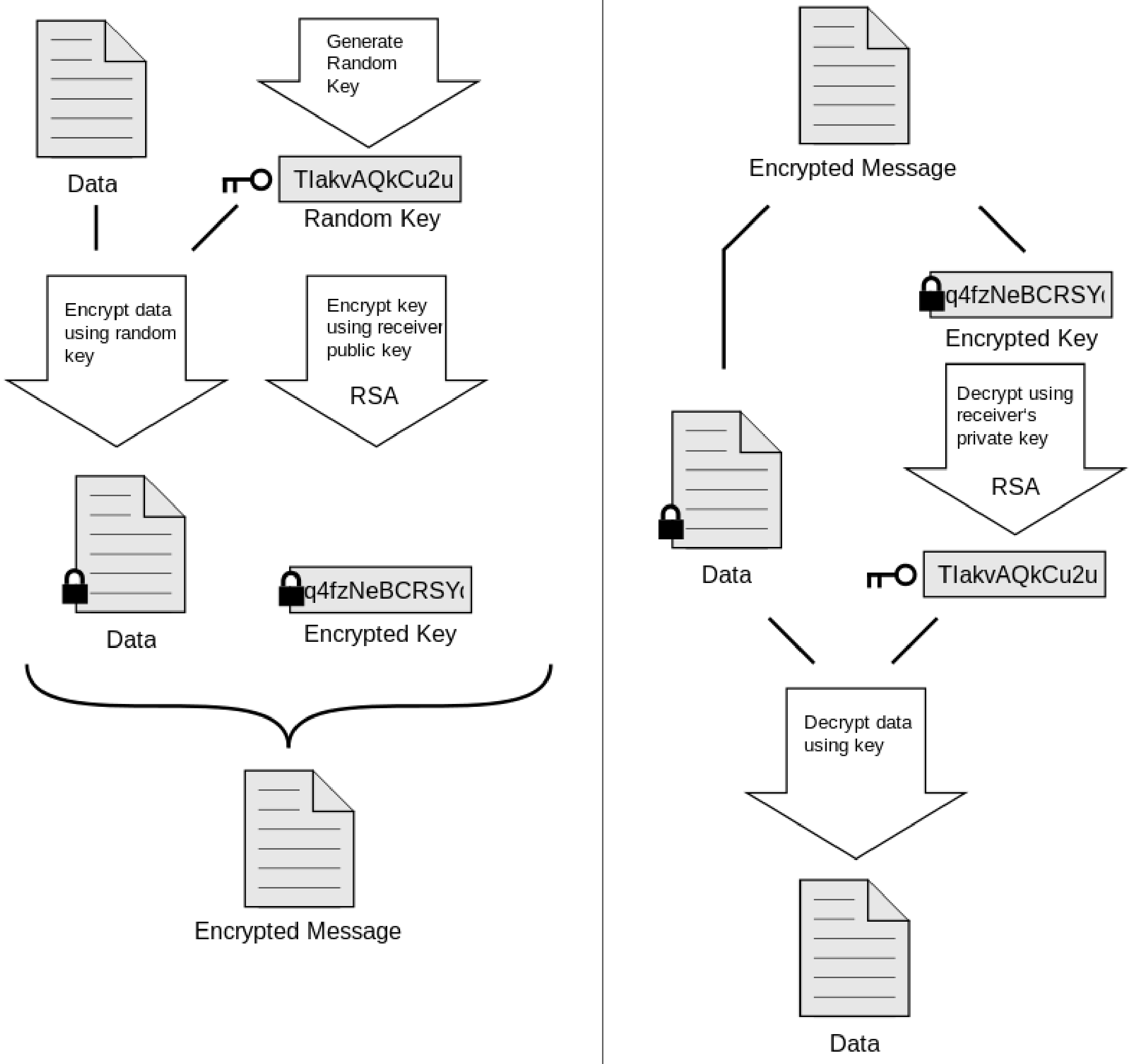
Example: PGP vs. SSL vs. IPsec

- PGP is an application-level protocol for “secure email”
  - Can provide security on “insecure” systems
  - Users choose when to use PGP; user must be involved
  - Alice’s signature on an email proves that Alice actually generated the message, and it was received unaltered; also non-repudiation
- In contrast, SSL would secure “the connection” from Alice’s computer;
  - would need an additional mechanism to authenticate the user
- IPsec is between every two hops in the network

PGP

Encrypt

Decrypt



Data

Generate Random Key

TlakvAQkCu2u

Random Key

Encrypt data using random key

Encrypt key using receiver public key

RSA

Data

q4fzNeBCRSY(

Encrypted Key

Encrypted Message

Encrypted Message

q4fzNeBCRSY(

Encrypted Key

Decrypt using receiver's private key

RSA

Data

TlakvAQkCu2u

Decrypt data using key

Data

Example: PGP vs. SSL vs. IPsec

- SSL sits at the transport layer, “above” TCP
  - Packet stream authenticated/encrypted
  - End-to-end security, best for connection-oriented sessions (e.g., http traffic)
  - User does not need to be involved
  - The OS does not have to change, but applications do if they want to communicate securely

Example: PGP vs. SSL vs. IPsec

- IPsec sits at the network layer
  - Individual packets authenticated/encrypted
  - End-to-end or hop-by-hop security
    - Best for connectionless channels
  - Need to modify OS
  - All applications are “protected” by default, without requiring any change to applications or actions on behalf of users
  - Only authenticates hosts, not users
  - User completely unaware that IPsec is running

SSL/TLS

Brief history…

- SSLv2 deployed in Netscape 1.1 (1995)
- Modified version of SSLv3 standardized at TLS

Broad overview

- SSL runs on top of TCP
  - Provides an API similar to that of TCP
- Technically, SSL runs in the application layer
  - Advantage: does not require changes to TCP
- From the programmer’s point of view, it is in the transport layer
  - Same API as for TCP
  - Runs only with TCP, not UDP
- Primarily used for HTTP traffic

SSL overview

- Three phases
  - Handshake
  - Key derivation
  - Data transfer

Handshake phase

- Client:
  - Establishes TCP connection with server;
  - Verifies server’s identity
    - Obtains server’s public key and certificate; verifies certificate
  - Sends server a master secret key K
    - Encrypted using server’s public key

Key derivation

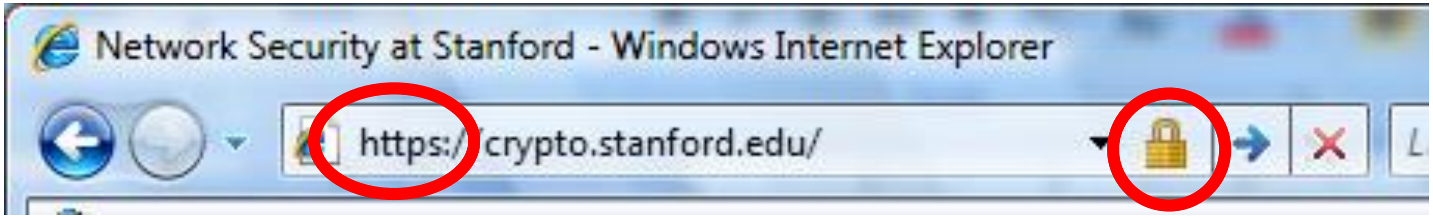
- Client and server use K to establish four keys: encryption and authentication, for each direction

Data transfer

- SSL breaks data stream into *records*; appends a MAC to each record; and then encrypts the result
  - Mac-then-encrypt…
- The MAC is computed over the record plus a sequence number
  - Prevents replay, re-ordering, or dropping packets

Note…

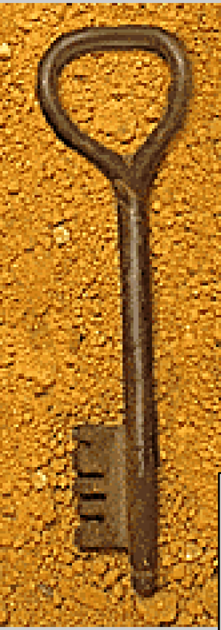
- As described, SSL only provides one-way authentication (server-to-client)
  - Not generally common for clients to have public keys
- Can do mutual authentication over SSL using, e.g., a password
  - SSL also allows for clients to have public keys



Network Security at Stanford - Windows Internet Explorer

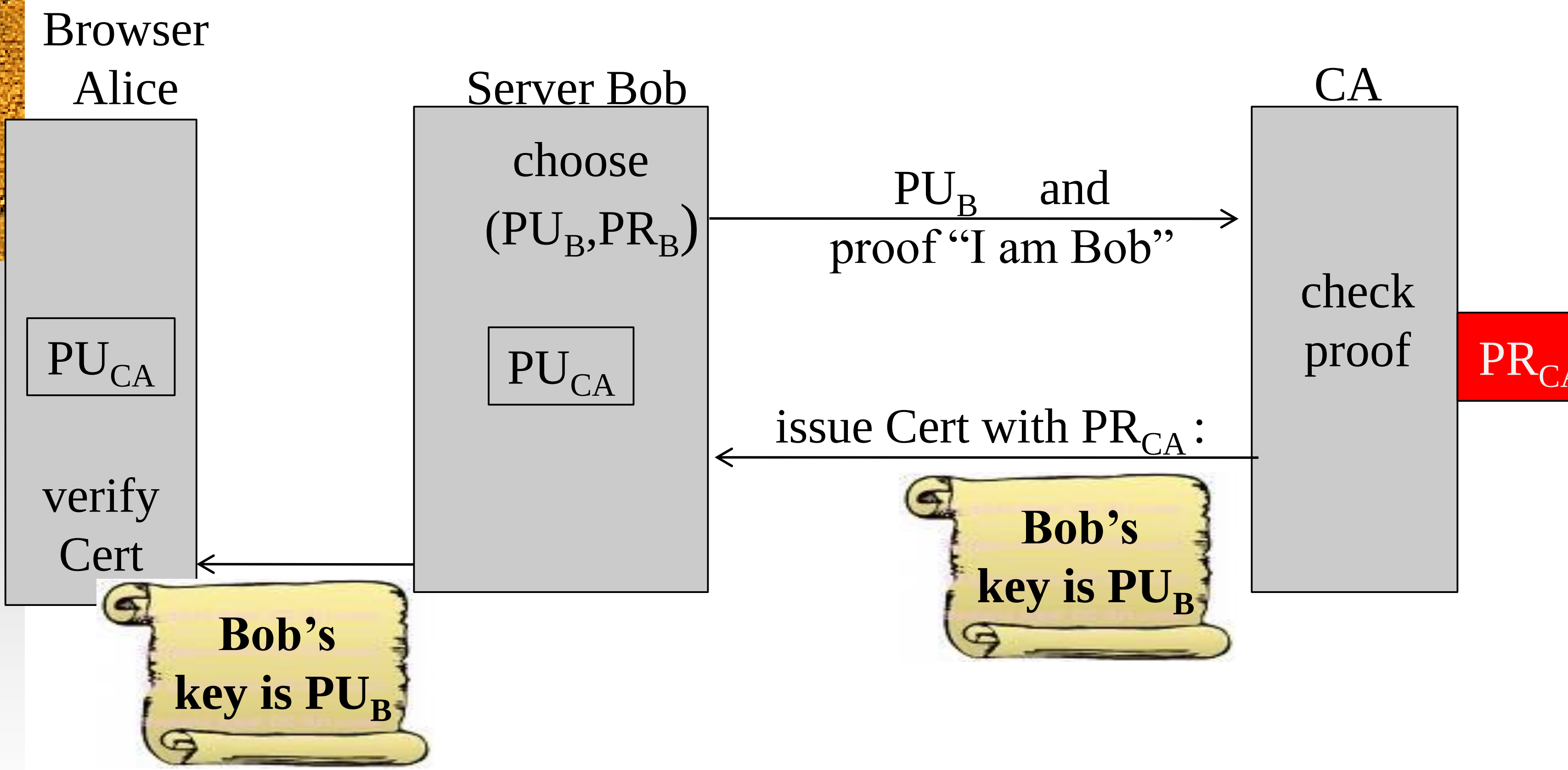
https://crypto.stanford.edu/

HTTPS and the Lock Icon



Certificates

- How does Alice (browser) obtain PK_{Bob} ?



Browser Alice

Server Bob

CA

choose (PU_B,PR_B)

PU_B and proof "I am Bob"

check proof

PR_{CA}

PU_{CA}

PU_{CA}

issue Cert with PR_{CA} :

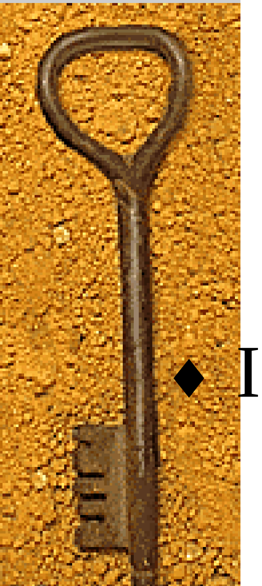
Bob's key is PU_B

verify Cert

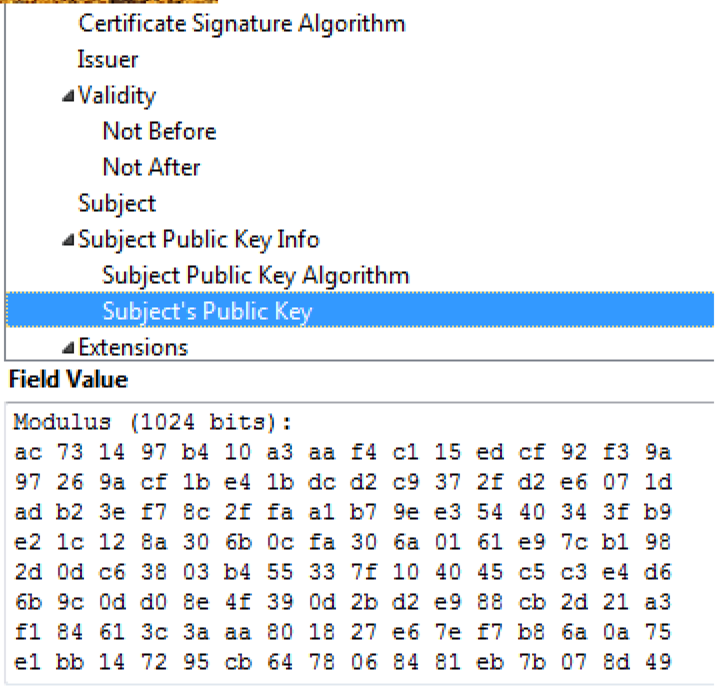
Bob's key is PU_B

Bob uses Cert for an extended period (e.g. one year)

Certificates: example



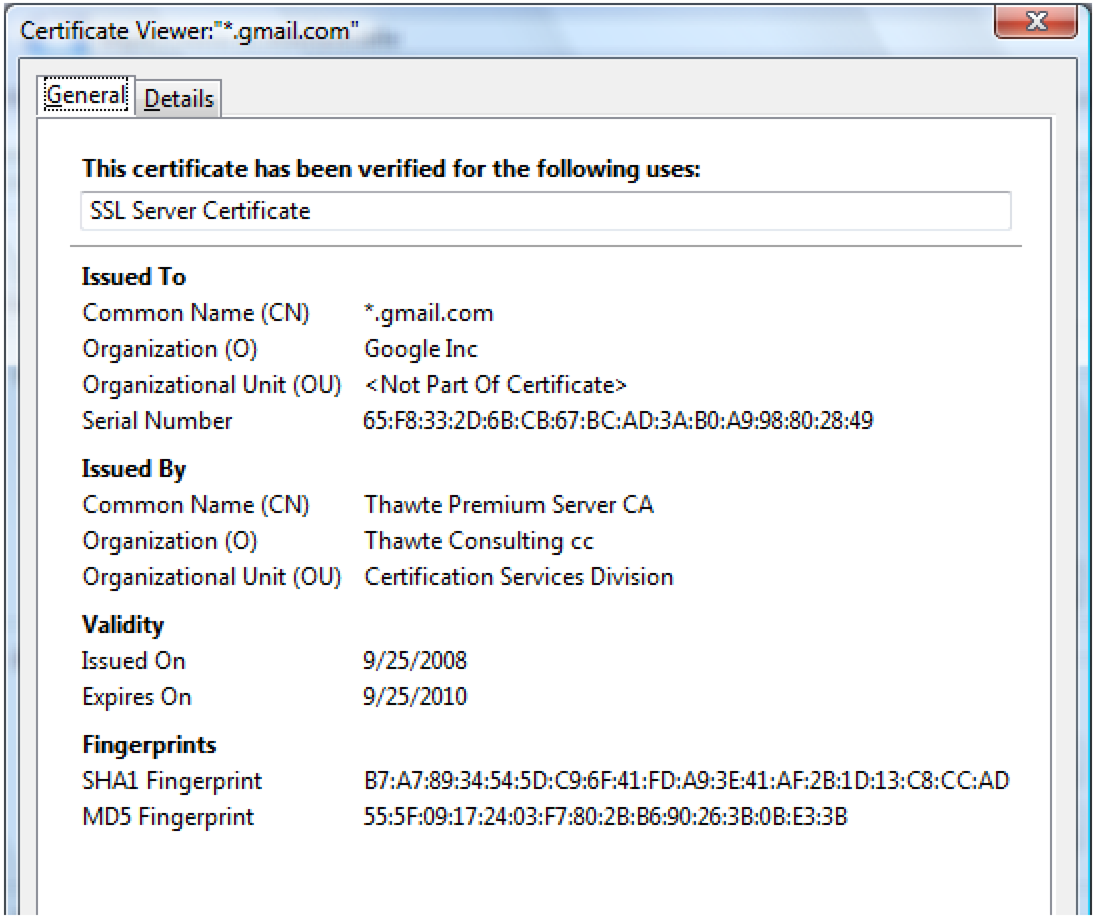
- Important fields:



Certificate Signature Algorithm
Issuer
◢Validity
  Not Before
  Not After
Subject
◢Subject Public Key Info
  Subject Public Key Algorithm
  Subject's Public Key
◢Extensions

Field Value

```
Modulus (1024 bits):
ac 73 14 97 b4 10 a3 aa f4 c1 15 ed cf 92 f3 9a
97 26 9a cf 1b e4 1b dc d2 c9 37 2f d2 e6 07 1d
ad b2 3e f7 8c 2f fa a1 b7 9e e3 54 40 34 3f b9
e2 1c 12 8a 30 6b 0c fa 30 6a 01 61 e9 7c b1 98
2d 0d c6 38 03 b4 55 33 7f 10 40 45 c5 c3 e4 d6
6b 9c 0d d0 8e 4f 39 0d 2b d2 e9 88 cb 2d 21 a3
f1 84 61 3c 3a aa 80 18 27 e6 7e f7 b8 6a 0a 75
e1 bb 14 72 95 cb 64 78 06 84 81 eb 7b 07 8d 49
```



Certificate Viewer:"*.gmail.com"

This certificate has been verified for the following uses:

SSL Server Certificate

Issued To

Common Name (CN)	*.gmail.com
Organization (O)	Google Inc
Organizational Unit (OU)	<Not Part Of Certificate>
Serial Number	65:F8:33:2D:6B:CB:67:BC:AD:3A:B0:A9:98:80:28:49

Issued By

Common Name (CN)	Thawte Premium Server CA
Organization (O)	Thawte Consulting cc
Organizational Unit (OU)	Certification Services Division

Validity

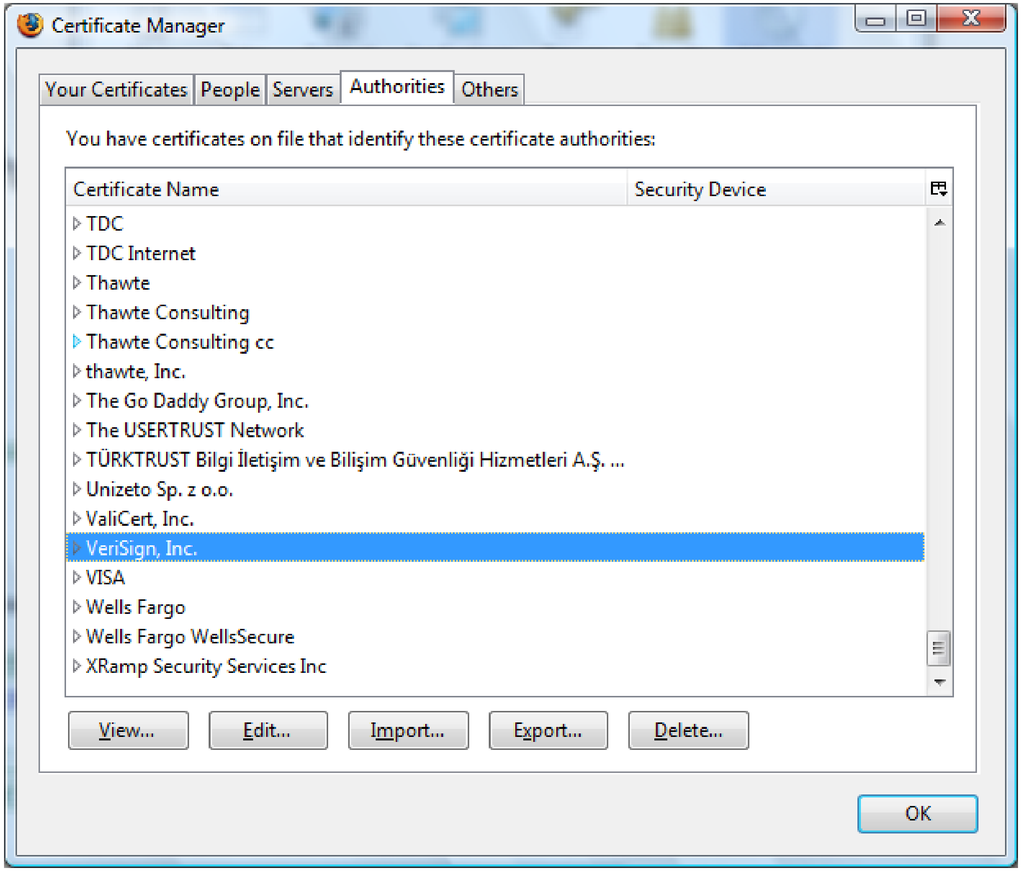
Issued On	9/25/2008
Expires On	9/25/2010

Fingerprints

SHA1 Fingerprint	B7:A7:89:34:54:5D:C9:6F:41:FD:A9:3E:41:AF:2B:1D:13:C8:CC:AD
MD5 Fingerprint	55:5F:09:17:24:03:F7:80:2B:B6:90:26:3B:0B:E3:3B

Certificate Authorities

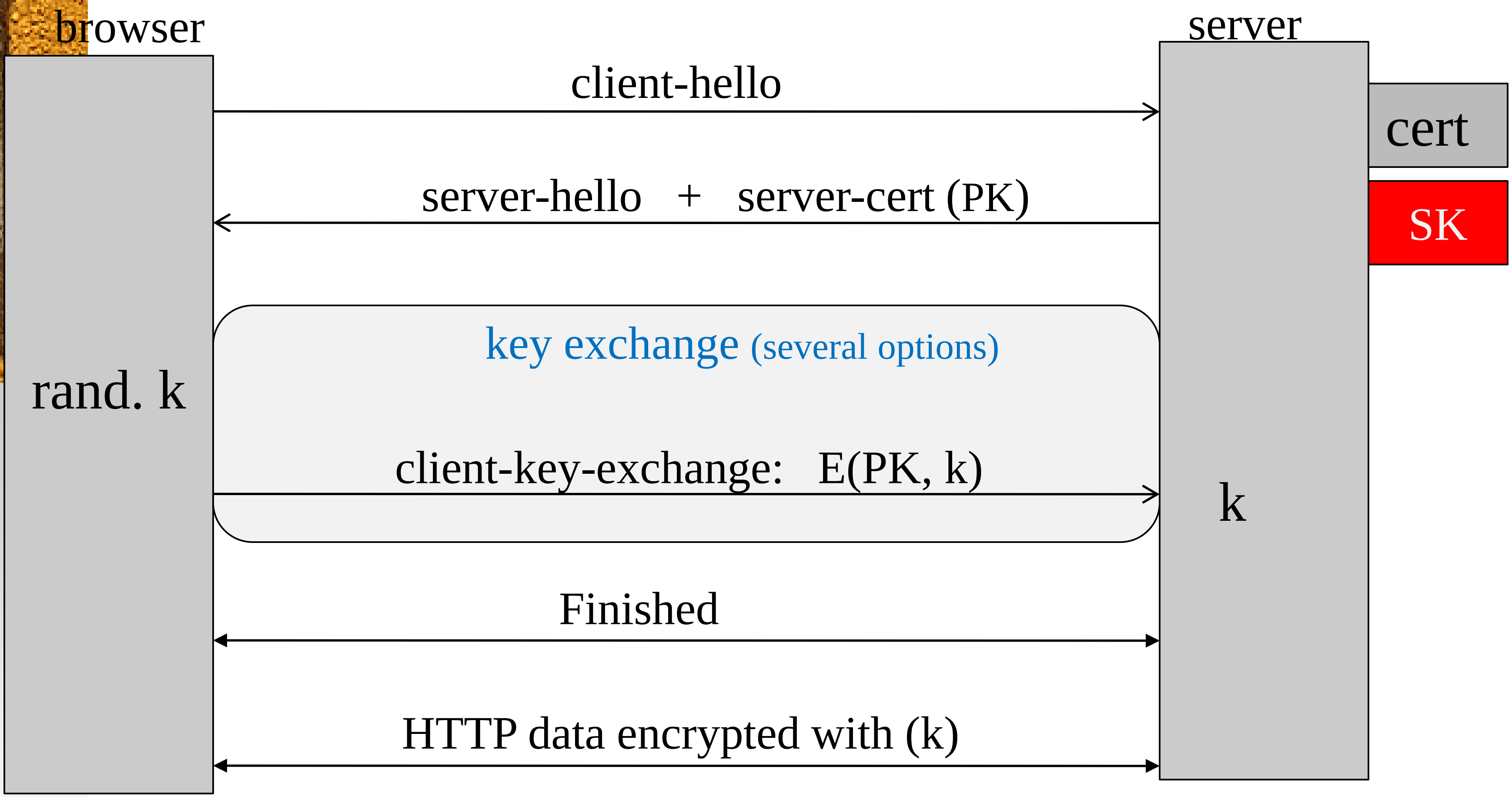
Browsers accept certificates from a large number of CAs



You have certificates on file that identify these certificate authorities:

Certificate Name	Security Device
TDC	
TDC Internet	
Thawte	
Thawte Consulting	
Thawte Consulting cc	
thawte, Inc.	
The Go Daddy Group, Inc.	
The USERTRUST Network	
TÜRKTRUST Bilgi İletişim ve Bilişim Güvenliği Hizmetleri A.Ş. ...	
Unizeto Sp. z o.o.	
ValiCert, Inc.	
VeriSign, Inc.	
VISA	
Wells Fargo	
Wells Fargo WellsSecure	
XRamp Security Services Inc	

Brief overview of SSL/TLS



browser

server

cert

SK

client-hello

server-hello + server-cert (PK)

key exchange (several options)

rand. k

client-key-exchange: E(PK, k)

k

Finished

HTTP data encrypted with (k)

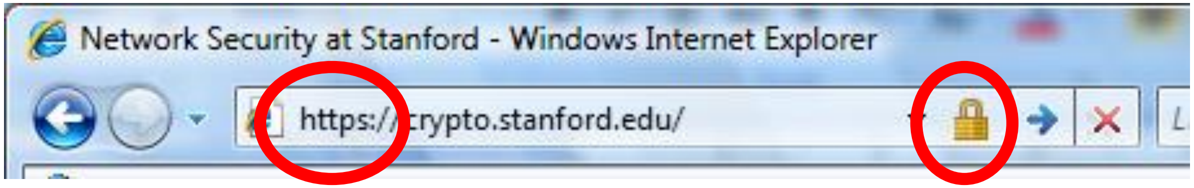
Most common: server authentication only

Why is HTTPS not used for all web traffic?

- Slows down web servers

- Breaks Internet caching
  - ISPs cannot cache HTTPS traffic
  - Results in increased traffic at web site

The lock icon: SSL indicator

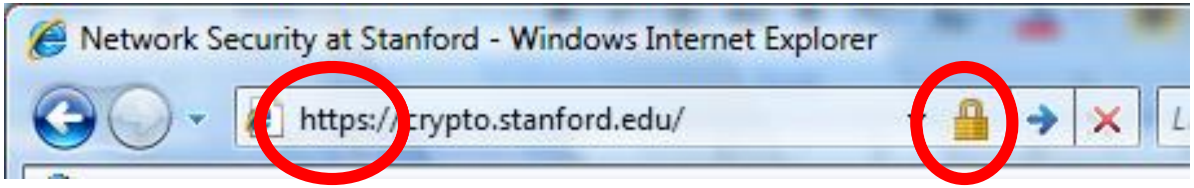


Network Security at Stanford - Windows Internet Explorer

https://crypto.stanford.edu/

- Intended goal:
  - Provide user with identity of page origin
  - Indicate to user that page contents were not viewed or modified by a **network attacker**

When is the (basic) lock icon displayed

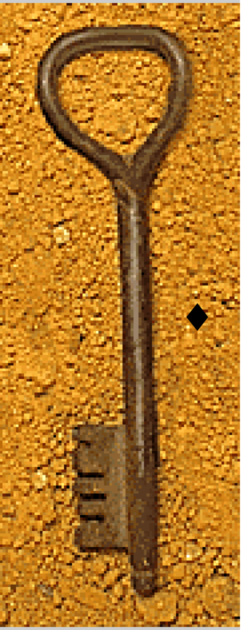


Network Security at Stanford - Windows Internet Explorer

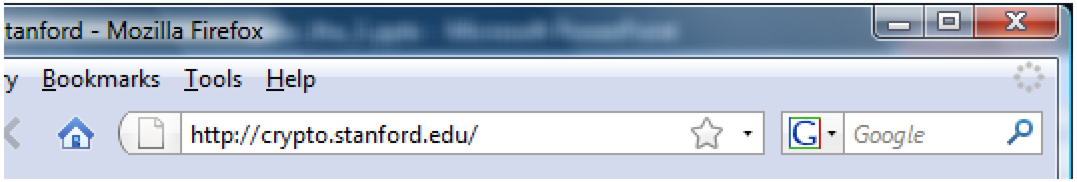
https://crypto.stanford.edu/

- All elements on the page fetched using HTTPS
- For all elements:
  - HTTPS cert issued by a CA trusted by browser
  - HTTPS cert is valid   (e.g. not expired)
  - CommonName in cert matches domain in URL

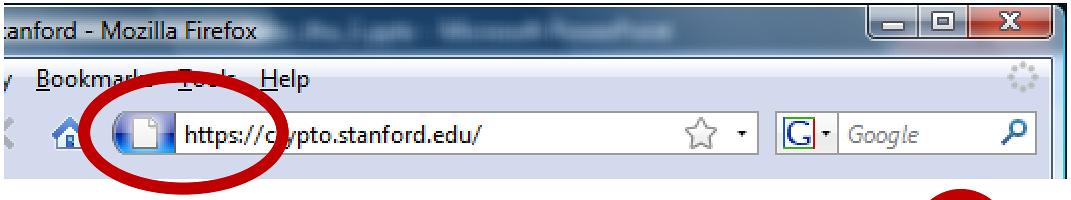
The lock UI: help users authenticate site

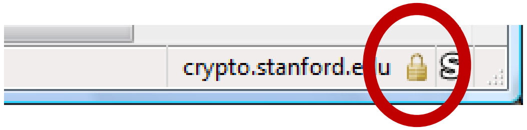


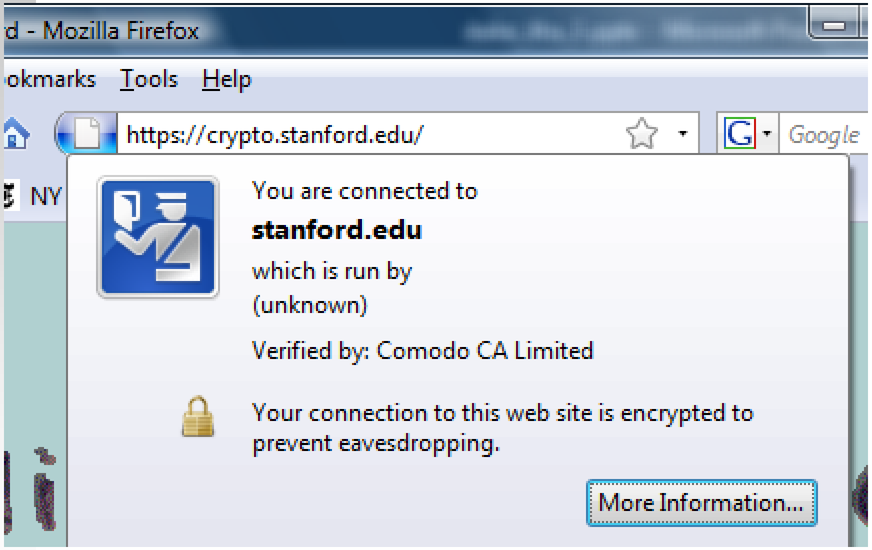
- Firefox 3: (no SSL)



(SSL)







You are connected to

stanford.edu

which is run by

(unknown)

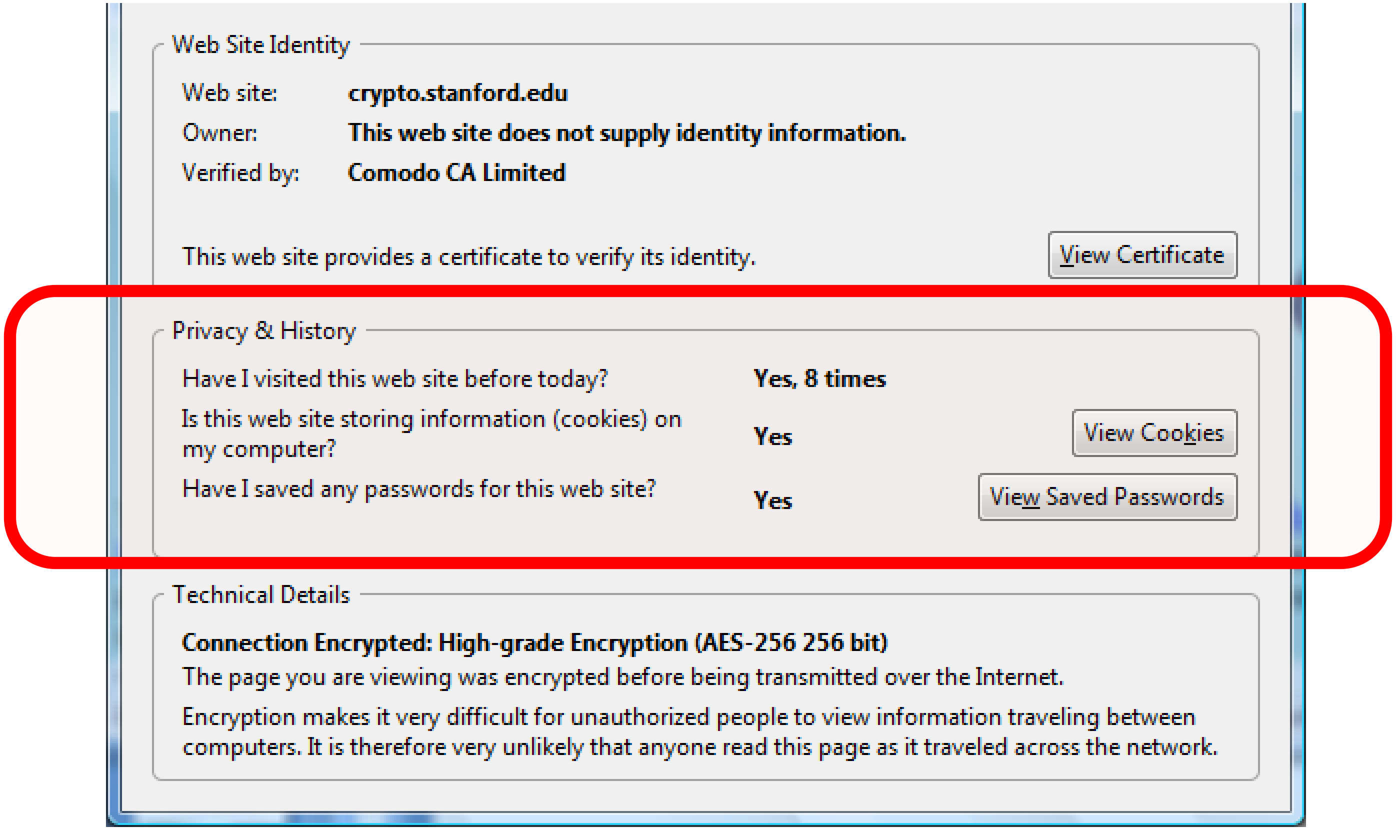
Verified by: Comodo CA Limited

Your connection to this web site is encrypted to prevent eavesdropping.

More Information...

The lock UI: help users authenticate site

- Firefox 3: clicking on bottom lock icon gives



Web Site Identity

Web site: **crypto.stanford.edu**

Owner: **This web site does not supply identity information.**

Verified by: **Comodo CA Limited**

This web site provides a certificate to verify its identity.

Privacy & History

Have I visited this web site before today? **Yes, 8 times**

Is this web site storing information (cookies) on my computer? **Yes**

Have I saved any passwords for this web site? **Yes**

Technical Details

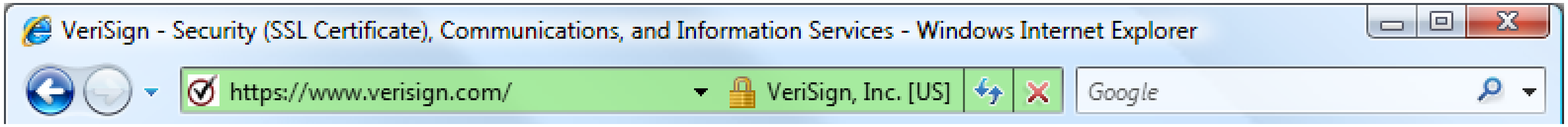
Connection Encrypted: High-grade Encryption (AES-256 256 bit)

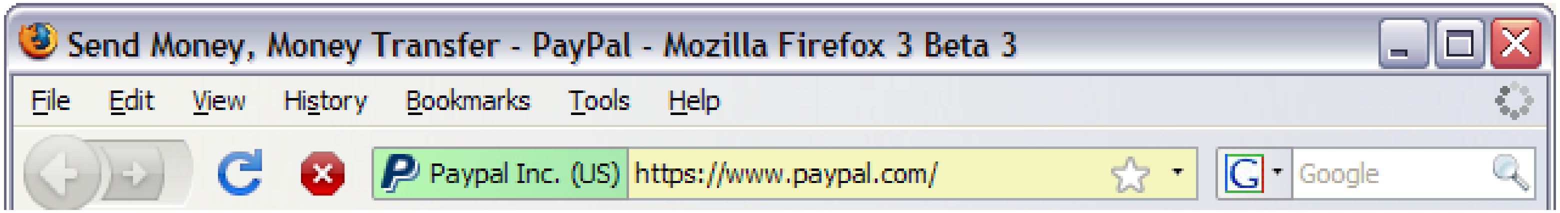
The page you are viewing was encrypted before being transmitted over the Internet.

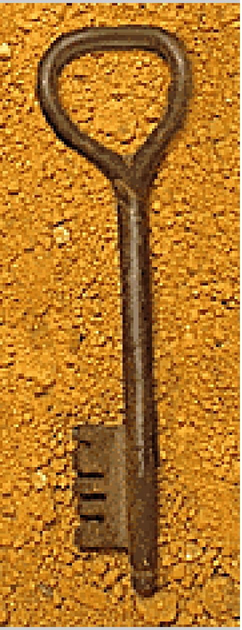
Encryption makes it very difficult for unauthorized people to view information traveling between computers. It is therefore very unlikely that anyone read this page as it traveled across the network.

The lock UI: Extended Validation (EV) Certs

- Harder to obtain than regular certs
  - requires human lawyer at CA to approve cert request
- Designed for banks and large e-commerce sites

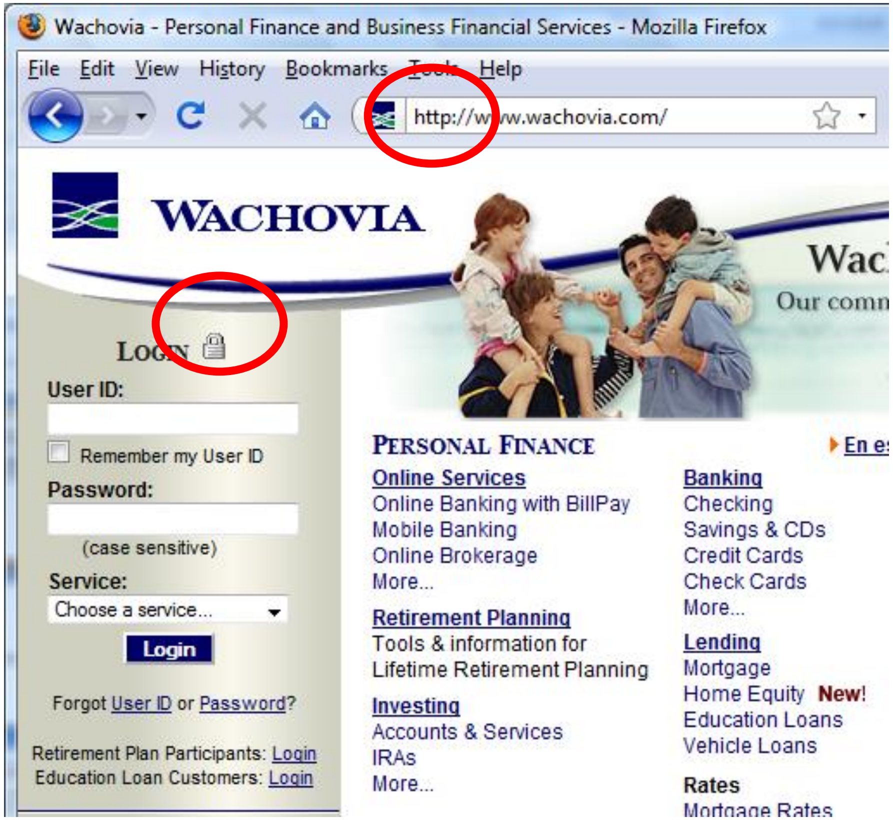

VeriSign - Security (SSL Certificate), Communications, and Information Services - Windows Internet Explorer
https://www.verisign.com/
VeriSign, Inc. [US]
Google





HTTPS and login pages: incorrect version

- ♦Users often land on login page over HTTP:
  - Type site’s HTTP URL into address bar, or
  - Google links to the HTTP page



LOGIN

User ID:

□ Remember my User ID

Password:

(case sensitive)

Service:

Choose a service...

Forgot User ID or Password?

Retirement Plan Participants: Login
Education Loan Customers: Login

PERSONAL FINANCE

En es

Online Services
Online Banking with BillPay
Mobile Banking
Online Brokerage
More...

Retirement Planning
Tools & information for Lifetime Retirement Planning

Investing
Accounts & Services
IRAs
More...

Banking
Checking
Savings & CDs
Credit Cards
Check Cards
More...

Lending
Mortgage
Home Equity New!
Education Loans
Vehicle Loans

Rates
Mortgage Rates

Invalid certs

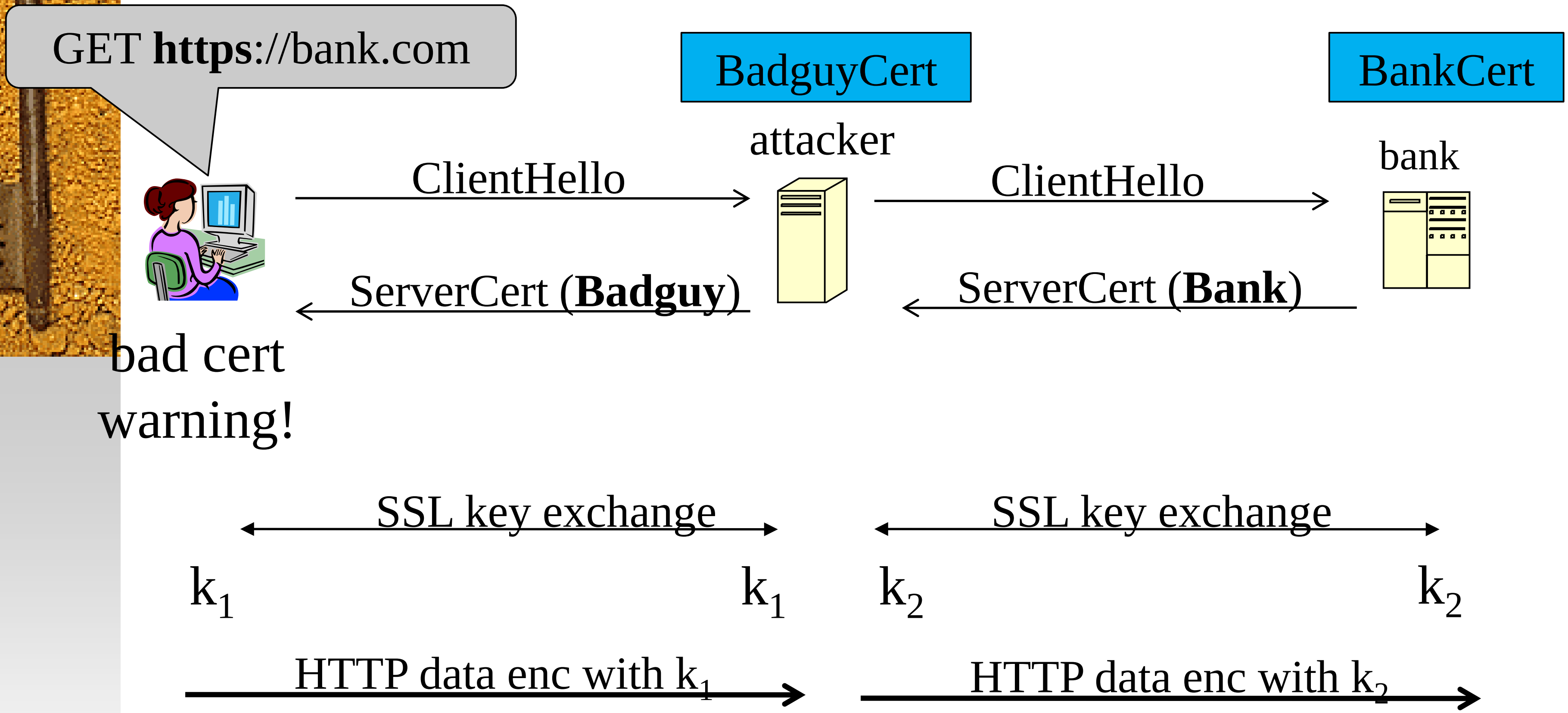
- Examples of invalid certificates:
  - expired: current-date > date-in-cert
  - CommonName in cert does not match domain in URL
  - unknown CA (e.g. self signed certs)
    - Small sites may not want to pay for cert

- Users often ignore warning:
  - Is it a miss-configuration or an attack? User can’t tell.

- Accepting invalid cert enables man-in-middle attacks

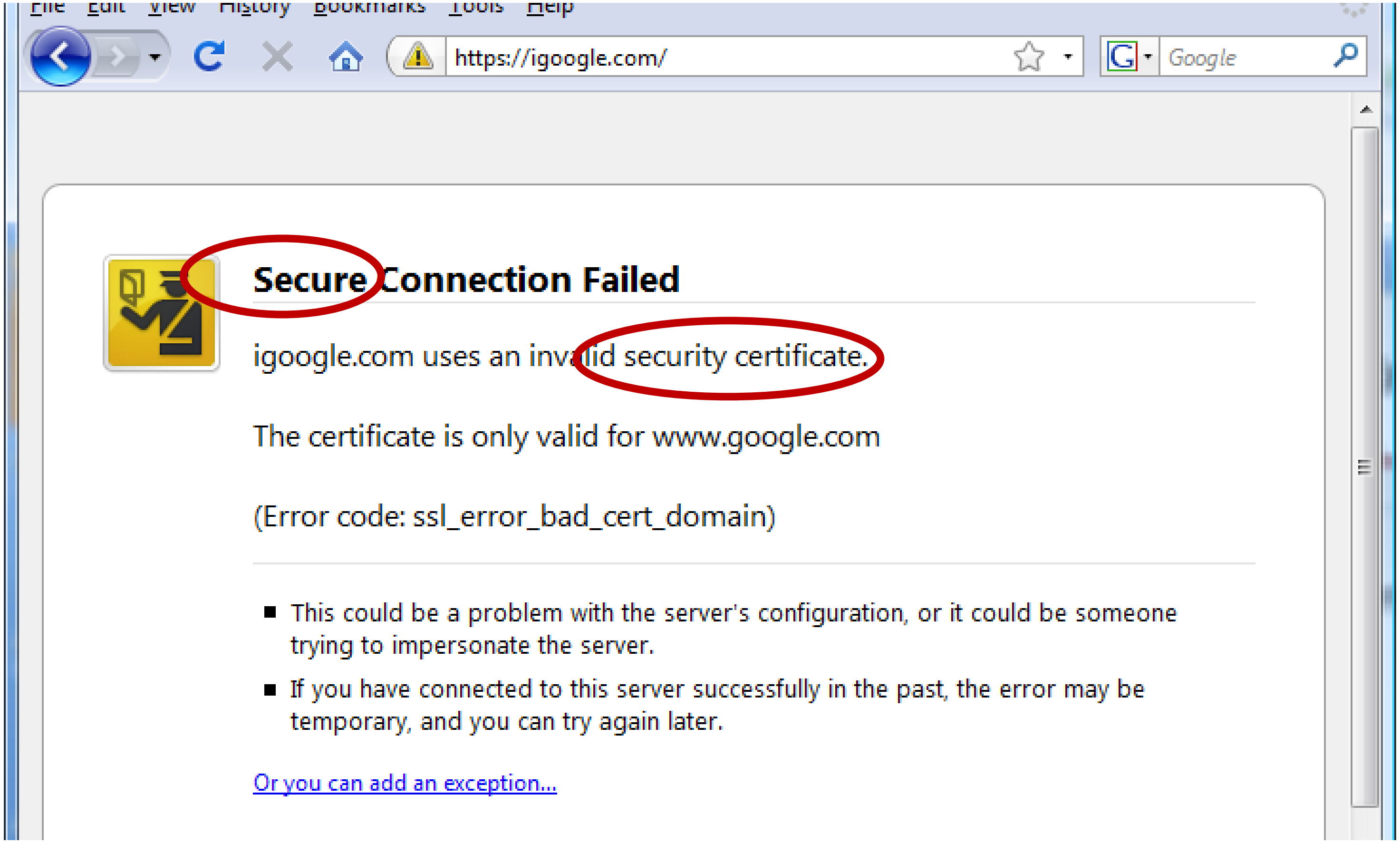
(see http://crypto.stanford.edu/ssl-mitm)

Man in the middle attack using invalid certs



- Attacker proxies data between user and bank. Sees all traffic and can modify data as will.

Firefox: Invalid cert dialog



Secure Connection Failed

igoogle.com uses an invalid security certificate.

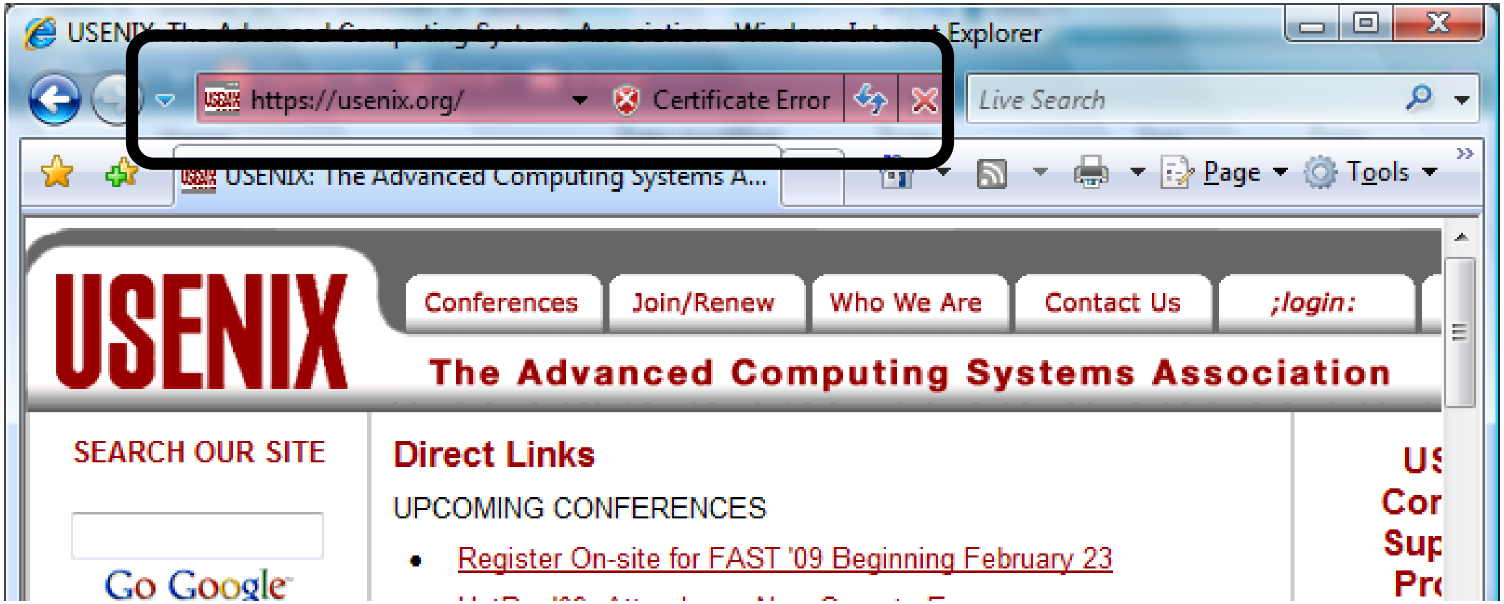
The certificate is only valid for www.google.com

(Error code: ssl_error_bad_cert_domain)

- This could be a problem with the server's configuration, or it could be someone trying to impersonate the server.
- If you have connected to this server successfully in the past, the error may be temporary, and you can try again later.

Or you can add an exception...

IE: invalid cert URL bar


USENIX: The Advanced Computing Systems Association - Windows Internet Explorer
https://usenix.org/
Certificate Error
Live Search
USENIX: The Advanced Computing Systems A...
Page
Tools
USENIX
Conferences
Join/Renew
Who We Are
Contact Us
;login:
The Advanced Computing Systems Association
SEARCH OUR SITE
Go Google
Direct Links
UPCOMING CONFERENCES
Register On-site for FAST '09 Beginning February 23
US
Cor
Sup
Pro

IPsec

Overview

- IPsec can provide security between any two network-layer entities
  - host-host, host-router, router-router
- Used widely to establish *VPNs*
- IPsec encrypts and/or authenticates network-layer traffic, and encapsulates it within a standard IP packet for routing over the Internet

Overview

- IPsec consists of two components
  - IKE --- Can be used to establish a key
  - AH/ESP --- Used to send data once a key is established (whether using IKE or out-of-band)
- AH
  - Data integrity, but no confidentiality
- ESP
  - Data integrity + confidentiality
  - (Other differences as well)

Security policy database

- Nodes maintain a table specifying what is required for each incoming packet
  - Drop
  - Forward/accept without IPsec protection
  - Require IPsec protection
    - Auth only
    - Enc only
    - Both
- As with firewalls, decisions can be based on any information in the packet

Security associations (SAs)

- When a node receives a packet, needs to know who it is from
  - May be receiving IPsec traffic from multiple senders at the same time -- possibly even with the same IP address
- An SA defines a network-layer <u>unidirectional</u> logical connection
  - For bidirectional communication, need two SAs
- The IPsec header indicates which *security association* to use

Security associations (SAs)

- A tremendous amount of information is kept in the SADB, and we can only touch on a few of them:
  - AH: authentication algorithm
  - AH: authentication secret
  - ESP: encryption algorithm
  - ESP: encryption secret key
  - ESP: authentication enabled yes/no
  - *Many* key-exchange parameters
  - Routing restrictions
  - IP filtering policy

Firewalls…

- Potential problem if upper-layer header data is used for decision-making; this information will be encrypted when using IPsec
- Arguments pro and con as to whether this data should be encrypted or not:
  - Pro: This data shouldn’t be divulged; get rid of firewalls
  - Con: administrators will likely keep firewalls and turn off encryption…

AH vs. ESP

- Two header types…
- Authentication header (AH)
  - Provides integrity only
- Encapsulating security payload (ESP)
  - Provides encryption + integrity
- Both provide cryptographic protection of everything beyond the IP headers
  - AH additionally provides integrity protection of some fields of the IP header

Transport vs. tunnel mode

- Transport mode: original IP header not touched; IPsec information added between IP header and packet body
  - IP header | IPsec | $\underbrace{\text{[packet]}}_{\text{protected}}$
  - Most logical when IPsec used end-to-end

Transport vs. tunnel mode

- Tunnel mode: keep original IP packet intact but protect it; add new header information outside
  - New IP header | $\underbrace{\text{IPsec} \mid \underbrace{\text{[old IP header | packet]}}_{\text{encrypted}}}_{\text{authenticated}}$
  - Can be used when IPSec is applied at intermediate point along path (e.g., for firewall-to-firewall traffic)
    - Treat the link as a secure tunnel
  - Results in slightly longer packet

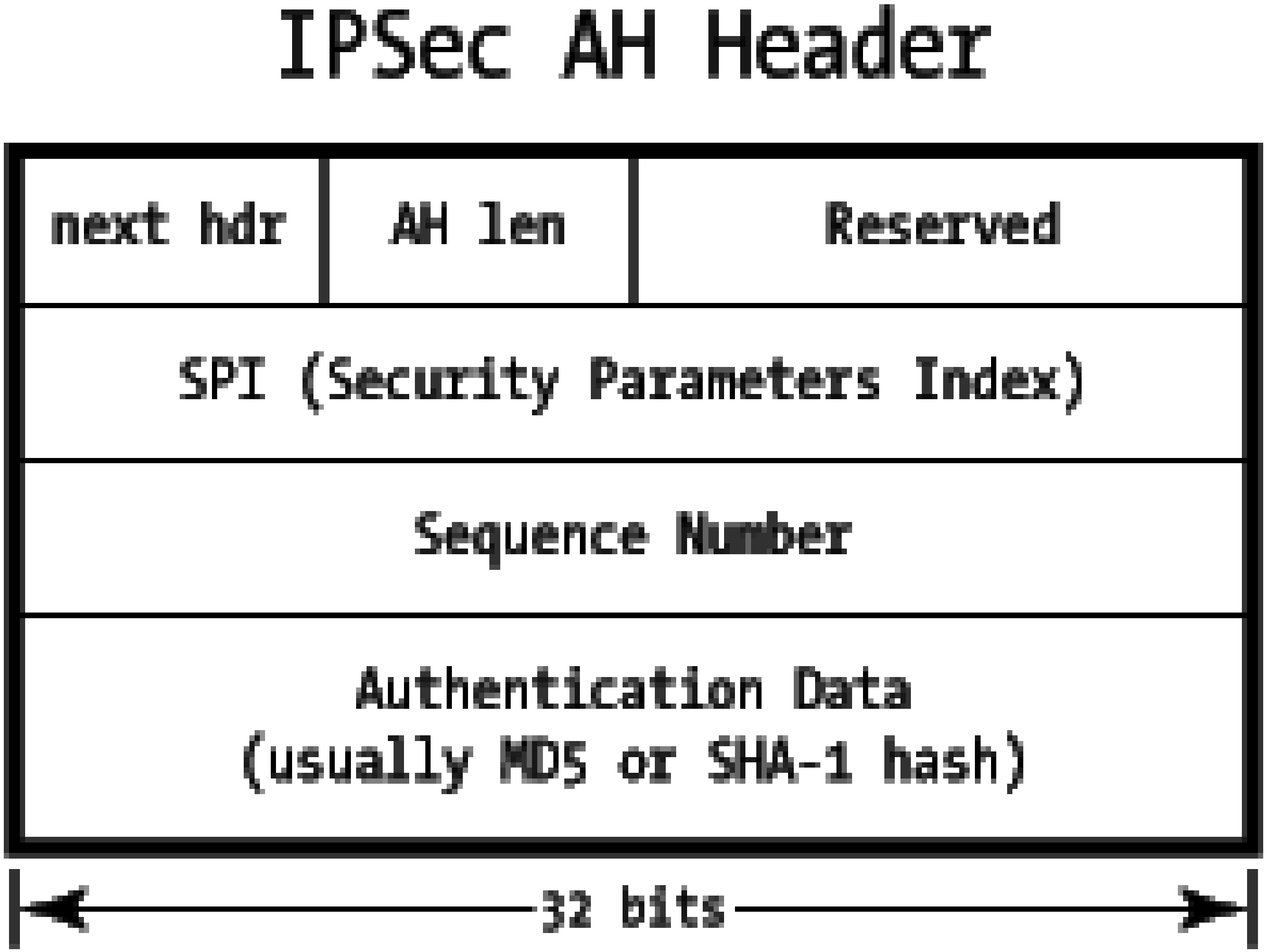
More on AH

- AH provides integrity protection on header
  - But some fields change *en route*!
- Immutable fields included in the integrity check
- Mutable but predictable fields are also included in the integrity check
  - The *final* value of the field is used

More on AH vs. ESP

- ESP can already provide encryption and/or authentication
- So why do we need AH?
  - AH also protects the IP header
  - Export restrictions
  - Firewalls need some high-level data to be unencrypted

AH Header

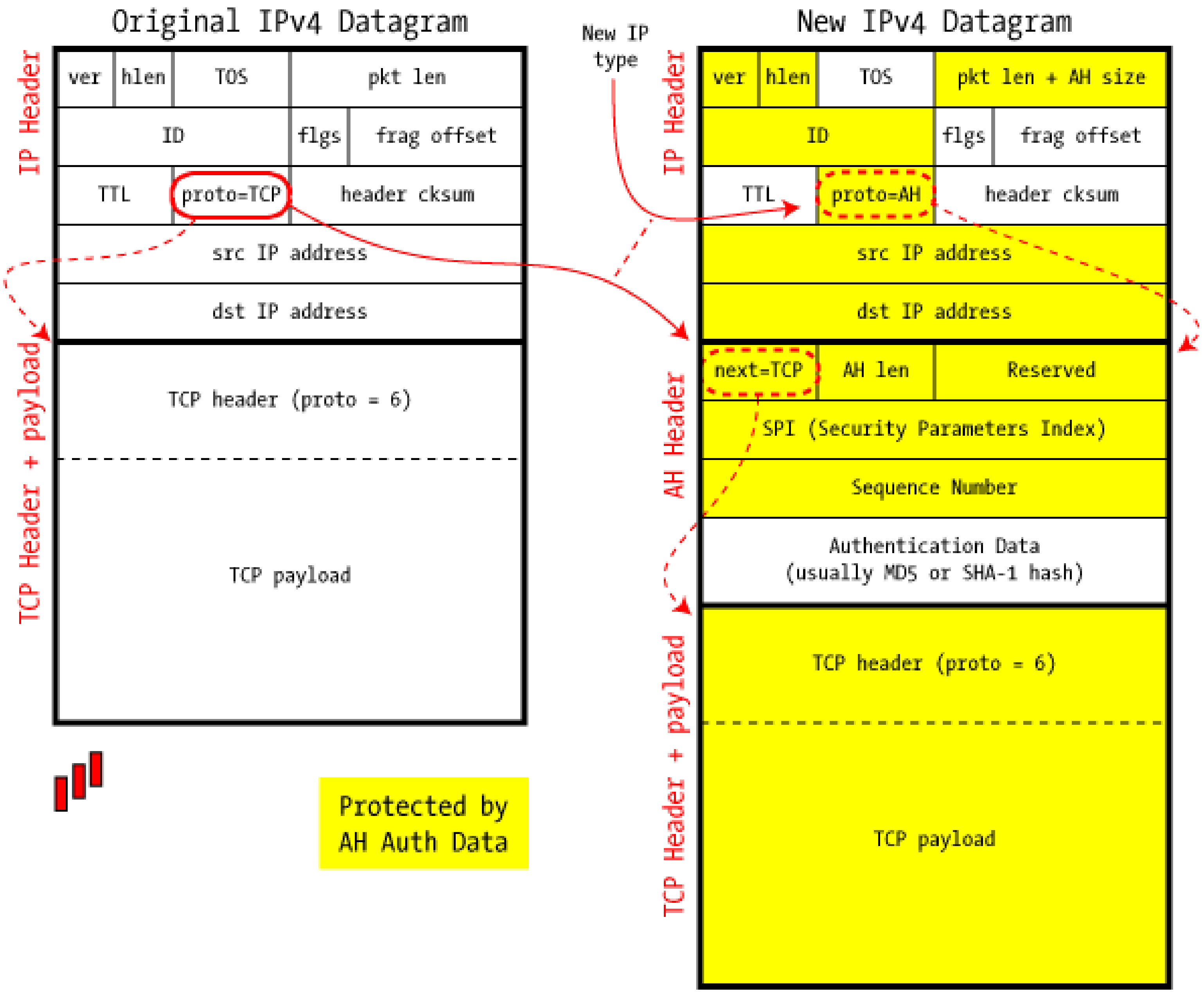


IPSec AH Header

next hdr	AH len	Reserved
SPI (Security Parameters Index)		
Sequence Number		
Authentication Data (usually MD5 or SHA-1 hash)		

32 bits

IPSec in AH Transport Mode



Original IPv4 Datagram

ver | hlen | TOS | pkt len

ID | flgs | frag offset

TTL | proto=TCP | header cksum

src IP address

dst IP address

TCP header (proto = 6)

TCP payload

New IPv4 Datagram

ver | hlen | TOS | pkt len + AH size

ID | flgs | frag offset

TTL | proto=AH | header cksum

src IP address

dst IP address

next=TCP | AH len | Reserved

SPI (Security Parameters Index)

Sequence Number

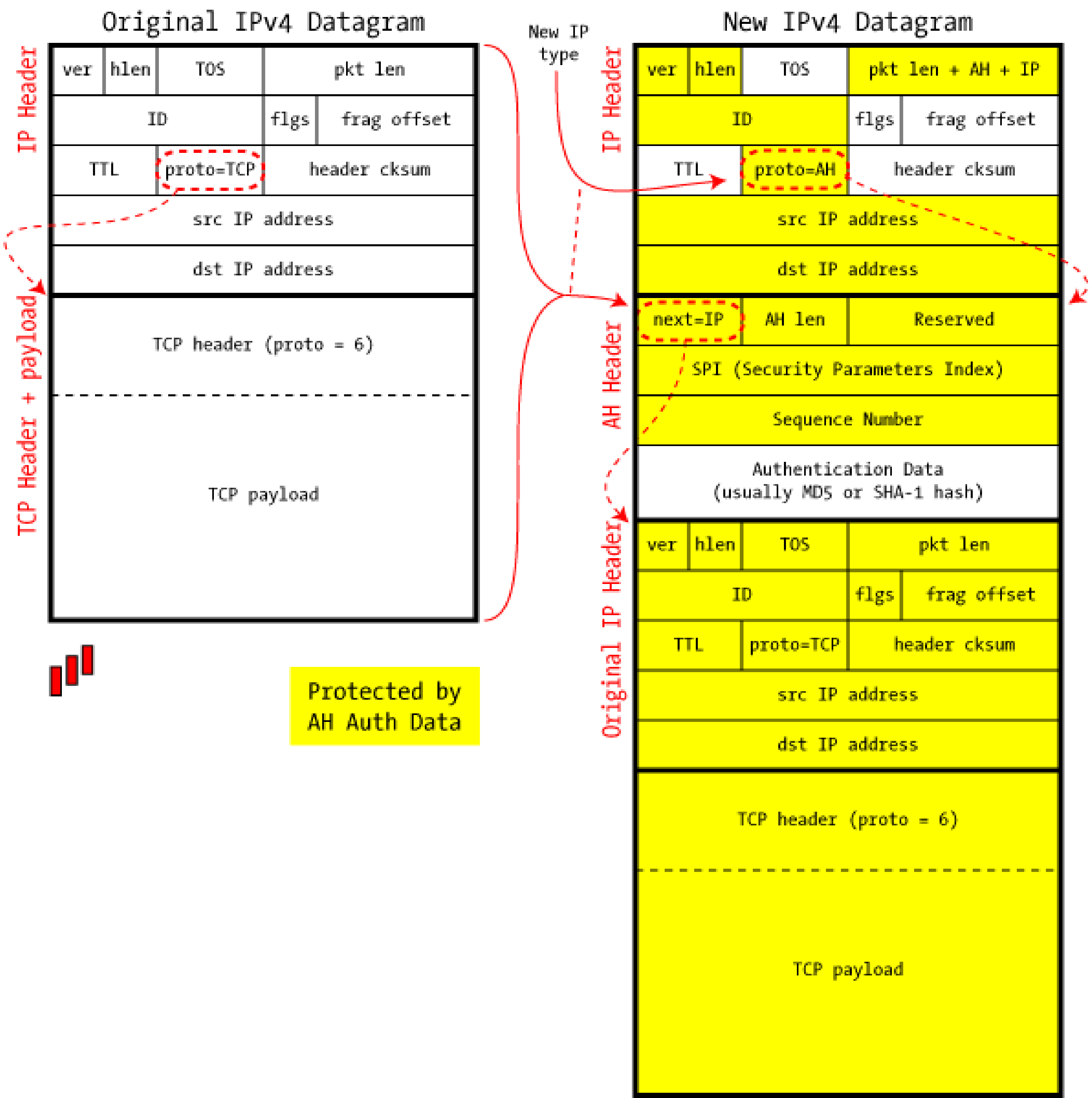
Authentication Data
(usually MD5 or SHA-1 hash)

TCP header (proto = 6)

TCP payload

Protected by
AH Auth Data

IPSec in AH Tunnel Mode



Original IPv4 Datagram

IP Header

ver	hlen	TOS	pkt len
ID		flgs	frag offset
TTL	proto=TCP	header cksum	
src IP address			
dst IP address			

TCP Header + payload

TCP header (proto = 6)

TCP payload

New IP type

New IPv4 Datagram

IP Header

ver	hlen	TOS	pkt len + AH + IP
ID		flgs	frag offset
TTL	proto=AH	header cksum	
src IP address			
dst IP address			

AH Header

next=IP	AH len	Reserved
SPI (Security Parameters Index)		
Sequence Number		
Authentication Data (usually MD5 or SHA-1 hash)		

Original IP Header

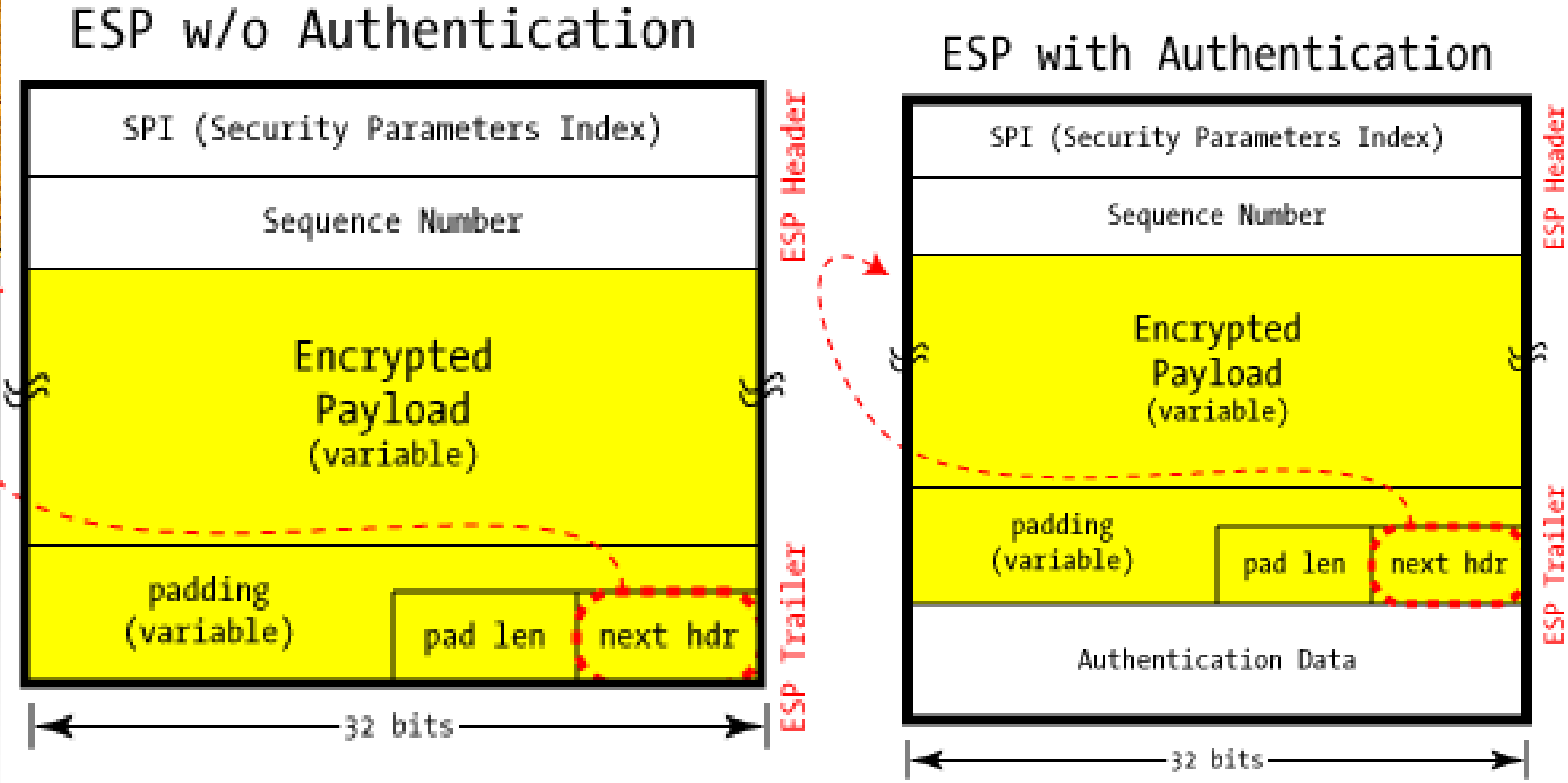
ver	hlen	TOS	pkt len
ID		flgs	frag offset
TTL	proto=TCP	header cksum	
src IP address			
dst IP address			

TCP header (proto = 6)

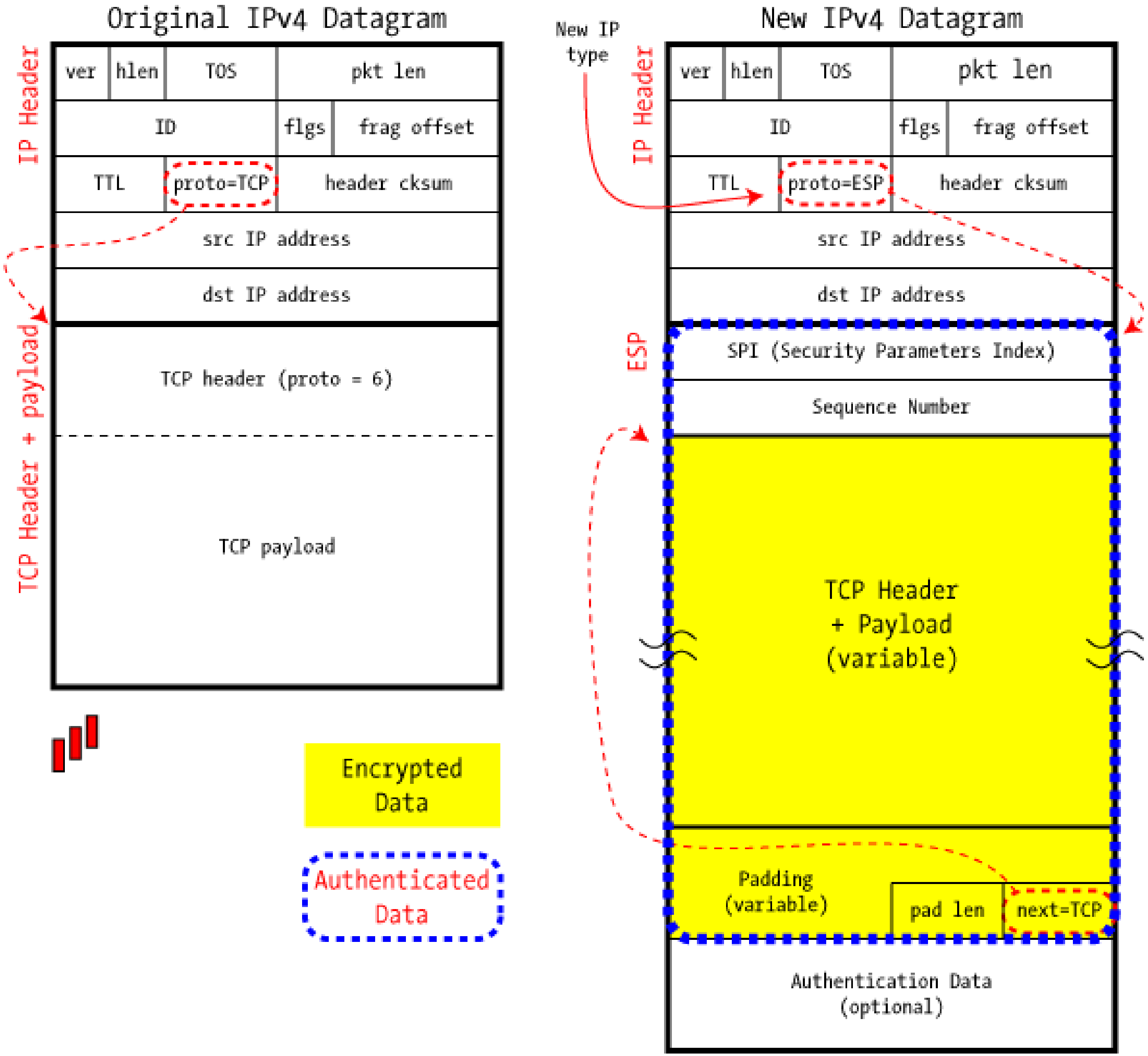
TCP payload

Protected by AH Auth Data

ESP Header


ESP w/o Authentication
SPI (Security Parameters Index)
Sequence Number
Encrypted Payload (variable)
padding (variable)
pad len
next hdr
ESP Header
ESP Trailer
32 bits
ESP with Authentication
SPI (Security Parameters Index)
Sequence Number
Encrypted Payload (variable)
padding (variable)
pad len
next hdr
Authentication Data
ESP Header
ESP Trailer
32 bits

IPSec in ESP Transport Mode



Original IPv4 Datagram

IP Header

ver | hlen | TOS | pkt len

ID | flgs | frag offset

TTL | proto=TCP | header cksum

src IP address

dst IP address

TCP Header + payload

TCP header (proto = 6)

TCP payload

New IPv4 Datagram

New IP type

IP Header

ver | hlen | TOS | pkt len

ID | flgs | frag offset

TTL | proto=ESP | header cksum

src IP address

dst IP address

ESP

SPI (Security Parameters Index)

Sequence Number

TCP Header + Payload (variable)

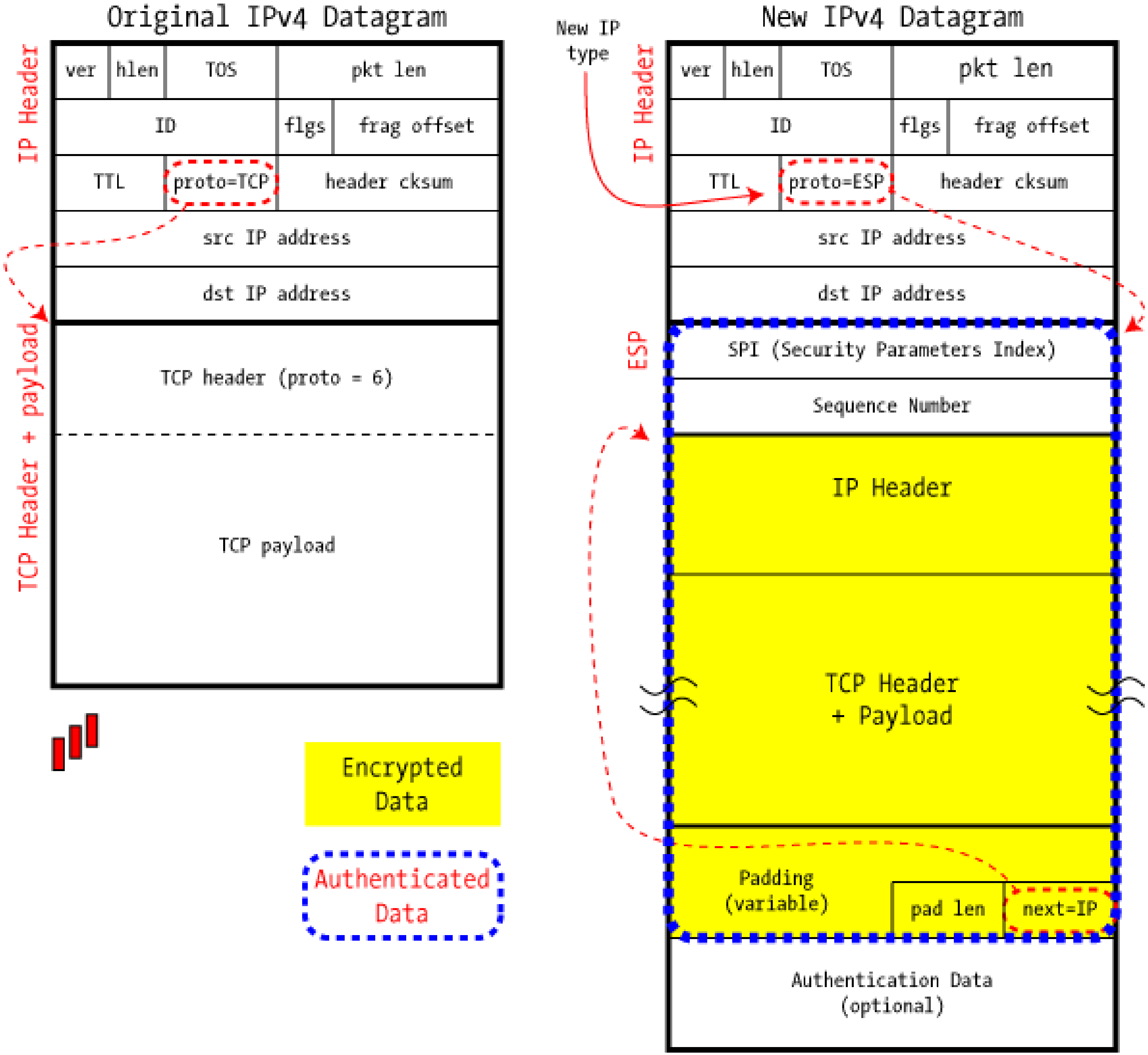
Padding (variable) | pad len | next=TCP

Authentication Data (optional)

Encrypted Data

Authenticated Data

IPSec in ESP Tunnel Mode



Original IPv4 Datagram

IP Header

ver | hlen | TOS | pkt len

ID | flgs | frag offset

TTL | proto=TCP | header cksum

src IP address

dst IP address

TCP Header + payload

TCP header (proto = 6)

TCP payload

New IP type

New IPv4 Datagram

IP Header

ver | hlen | TOS | pkt len

ID | flgs | frag offset

TTL | proto=ESP | header cksum

src IP address

dst IP address

ESP

SPI (Security Parameters Index)

Sequence Number

IP Header

TCP Header + Payload

Padding (variable) | pad len | next=IP

Authentication Data (optional)

Encrypted Data

Authenticated Data

The future of AH?

- In the long run, it seems that AH will become obsolete
  - Better to encrypt everything anyway
  - No real need for AH
  - Certain performance disadvantages
  - AH is complex…

IPsec: IKE

Overview of IKE

- IKE provides mutual authentication, establishes shared key, and creates SA
- Assumes a long-term shared key, and uses this to establish a session key (as well as to provide authentication)
- Supported key types
  - Public signature keys
  - Public encryption keys
  - Symmetric keys

IKE phases

- Phase 1: long-term keys used to derive a session key (and provide authentication)
- Phase 2: session key used to derive SAs
- Why…?
  - In theory, can run phase 1 once, followed by multiple executions of phase 2
    - E.g., different flows between same endpoints
    - Why not used same key for each? Is there any secure way to do this?
  - In practice, this anyway rarely happens

Key types

- Why are there two PK options?
  - Signature-based option
    - Efficiency (can start protocol knowing only your own public key, then get other side’s key from their certificate)
    - Legal reasons/export control
  - Encryption-based option
    - Can be used to provide anonymity in both directions
- Adds tremendously to the complexity of implementation

Anonymity

- Protocols can be designed so that identities of the parties are hidden from eavesdroppers
  - Even while providing authentication!
- Can also protect anonymity of one side against active attacks
  - Whom to protect?
    - Initiator: since responder’s identity is generally known…
    - Responder: since otherwise it is easy to get anyone’s identity

Phase 1 session keys

- Two session keys are defined in phase 1
  - One each for encryption/authentication
- These keys are used to protect the final phase 1 messages as well as all phase 2 messages
- These keys are derived from the DH key using hashing
  - Details in the book…

IKE phase 1

- Aggressive mode
  - 3 messages
- Main mode
  - 6 messages
  - Additional features:
    - Anonymity
    - Negotiation of crypto parameters

Aggressive mode

- Alice sends g^a, “Alice”, crypto algorithms
  - Note that choices are restricted by this message
- Bob sends g^b, choice of crypto algorithm, “proof” that he is really Bob
  - If Bob does not support any of the suggested algorithms, he simply does not reply
  - Note that there is no way to authenticate a refusal, since no session key yet established
- Alice sends “proof” that she is Alice

Main mode

- ♦ Negotiate crypto algorithms (2 rounds)
- ♦ Alice and Bob do regular Diffie-Hellman key exchange (2 rounds)
- ♦ Alice sends encryption of “Alice” plus a proof that she is Alice, using long-term secret keys plus [keys derived from] g^{ab}
- ♦ Bob does similarly…

Crypto parameters…

- Choice of:
  - Encryption method (DES, 3DES, …)
  - Hash function (MD5, SHA-1, …)
  - Authentication method (e.g., key type, etc.)
  - Diffie-Hellman group (e.g., (g, p), etc.)
- A complete set of protocols (a security *suite*) must be specified

Negotiating parameters

- Many protocols allow parties to negotiate cryptographic algorithms and parameters
  - Allows users to migrate to stronger crypto; increases inter-operability (somewhat)
- But, opens up a potential attack if not authenticated somehow…
- Also makes for more complicated implementations

“Proofs of identity”

- Depend on which type of long-term shared key is being used
- Similar (in spirit) to the authentication protocols discussed in class
  - Details in book…

Course wrap-up

What should you take away from this course (after the final)?

- Security mind-set
  - Not limited to computers/networks!
- Security is complex
  - Draws on many different disciplines
  - Need to know what you are doing
- Security is hard, still evolving
  - We did not cover some of the most important present-day attacks: spam, phishing, DDos, viruses, …
- Security is challenging…but fun!

Thank you!