CMP 661
Cryptography (Network and Data Security)

Spring 2018

Murat Aydos

maydos@hacettepe.edu.tr

What is this course about?

This course is to discuss

- security needs
- security services
- security mechanisms and protocols

for data stored in computers and transmitted across computer networks

What we will/won't cover?

- We will cover
  - security threats
  - some security attacks (practice in labs)
  - security protocols in use
  - security protocols not in use
  - securing computer systems
  - introductory cryptography
- We will not cover
  - advanced cryptography
  - computer networks
  - operating systems
  - computers in general
  - how to hack ☺

What security is about in general?

- Security is about protection of assets
  - D. Gollmann, Computer Security, Wiley
- Prevention
  - take measures that prevent your assets from being damaged (or stolen)
- Detection
  - take measures so that you can detect when, how, and by whom an asset has been damaged
- Reaction
  - take measures so that you can recover your assets

Real world example

- Prevention
  - locks at doors, window bars, secure the walls around the property, hire a guard
- Detection
  - missing items, burglar alarms, closed circuit TV
- Reaction
  - attack on burglar, call the police, replace stolen items, make an insurance claim

Internet shopping example

- Prevention
  - encrypt your order and card number, enforce merchants to do some extra checks, using PIN even for Internet transactions, don't send card number via Internet
- Detection
  - an unauthorized transaction appears on your credit card statement
- Reaction
  - complain, dispute, ask for a new card number, sue (if you can find of course ☺)
  - Or, pay and forget (a glass of cold water) ☺

Information security in past and present

- Traditional Information Security
  - keep the cabinets locked
  - put them in a secure room
  - human guards
  - electronic surveillance systems
  - in general:physical and administrative mechanisms
- Modern World
  - Data are in computers
  - Computers are interconnected

Computer and Network Security

Terminology

- Computer Security
  - 2 main focuses: Information and Computer itself
  - tools and mechanisms to protect data in a computer (actually an automated information system), even if the computers/system are connected to a network
  - tools and mechanisms to protect the information system itself (hardware, software, firmware, *ware ☺)
- Against?
  - against hackers (intrusion)
  - against viruses
  - against denial of service attacks
  - etc. (all types of malicious behavior)

Terminology

- (Inter)Network Security
  - measures to prevent, detect, and correct security violations that involve the transmission of information in a network or interconnected network
  - Our text (Stallings) makes a distinction between network and internetwork security, but that distinction is not so clear

A note on security terminology

- No single and consistent terminology in the literature!
- Be careful not to confuse while reading papers and books

- See the next slide for some terminology taken from Stallings and Brown, Computer Security who took from RFC2828, Internet Security Glossary

Adversary (threat agent)
An entity that attacks, or is a threat to, a system.

Attack
An assault on system security that derives from an intelligent threat; that is, an intelligent act that is a deliberate attempt (especially in the sense of a method or technique) to evade security services and violate the security policy of a system.

Countermeasure
An action, device, procedure, or technique that reduces a threat, a vulnerability, or an attack by eliminating or preventing it, by minimizing the harm it can cause, or by discovering and reporting it so that corrective action can be taken.

Risk
An expectation of loss expressed as the probability that a particular threat will exploit a particular vulnerability with a particular harmful result.

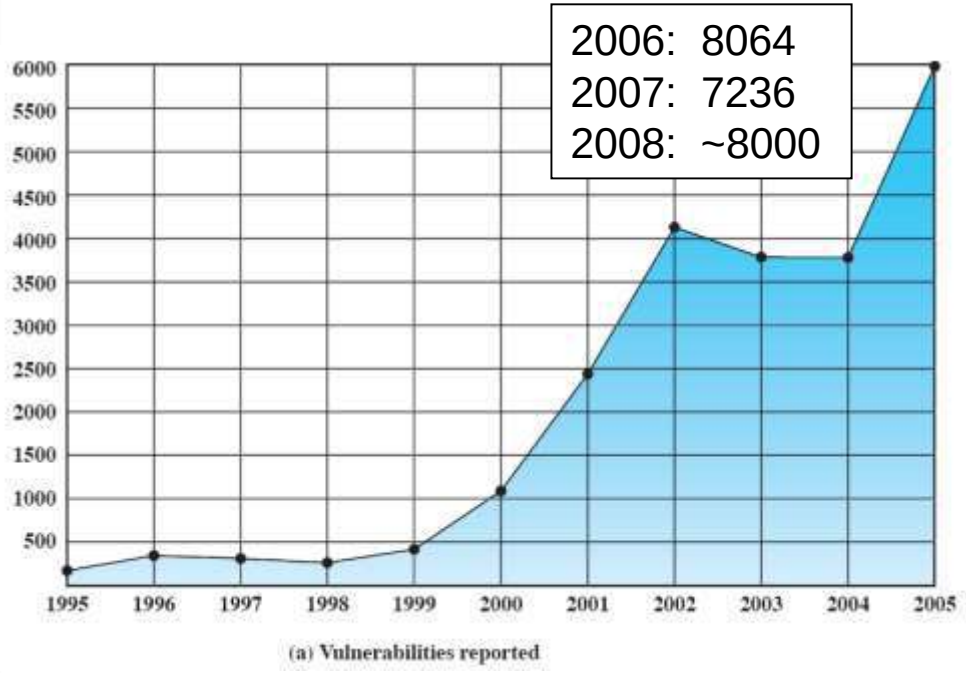
Security Policy
A set of rules and practices that specify or regulate how a system or organization provides security services to protect sensitive and critical system resources.

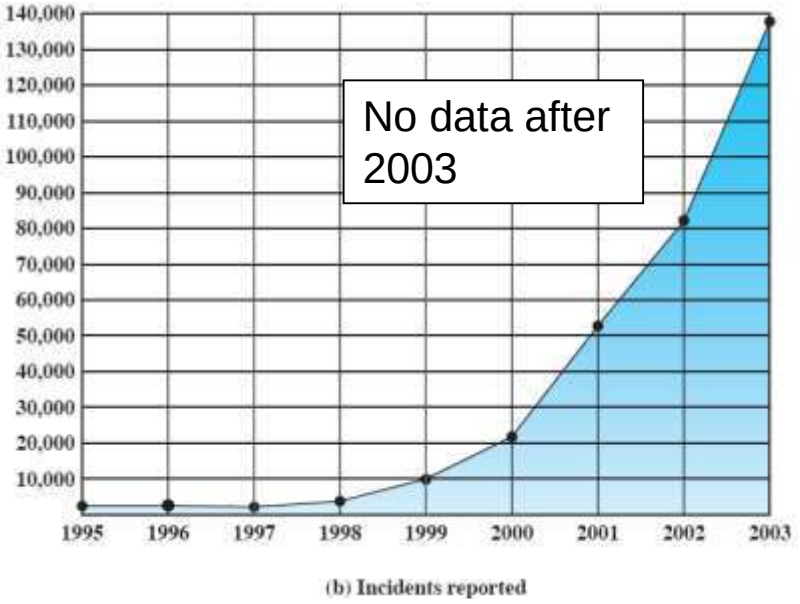
System Resource (Asset)
Data contained in an information system; or a service provided by a system; or a system capability, such as processing power or communication bandwidth; or an item of system equipment (i.e., a system component--hardware, firmware, software, or documentation); or a facility that houses system operations and equipment.

Threat
A potential for violation of security, which exists when there is a circumstance, capability, action, or event that could breach security and cause harm. That is, a threat is a possible danger that might exploit a vulnerability.

Vulnerability
A flaw or weakness in a system's design, implementation, or operation and management that could be exploited to violate the system's security policy.

Why Security is Important?



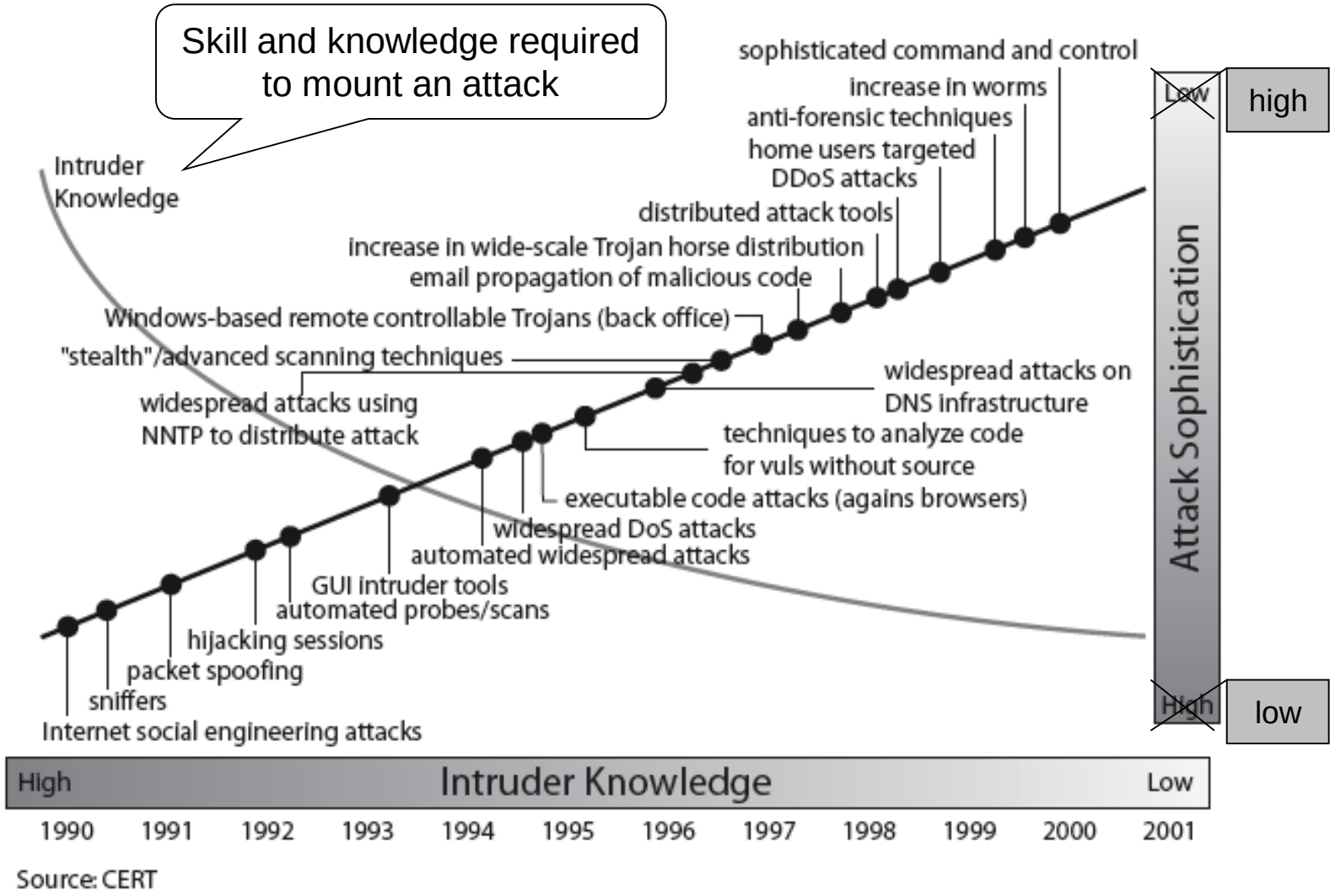


CERT Statistics - http://www.cert.org/stats/

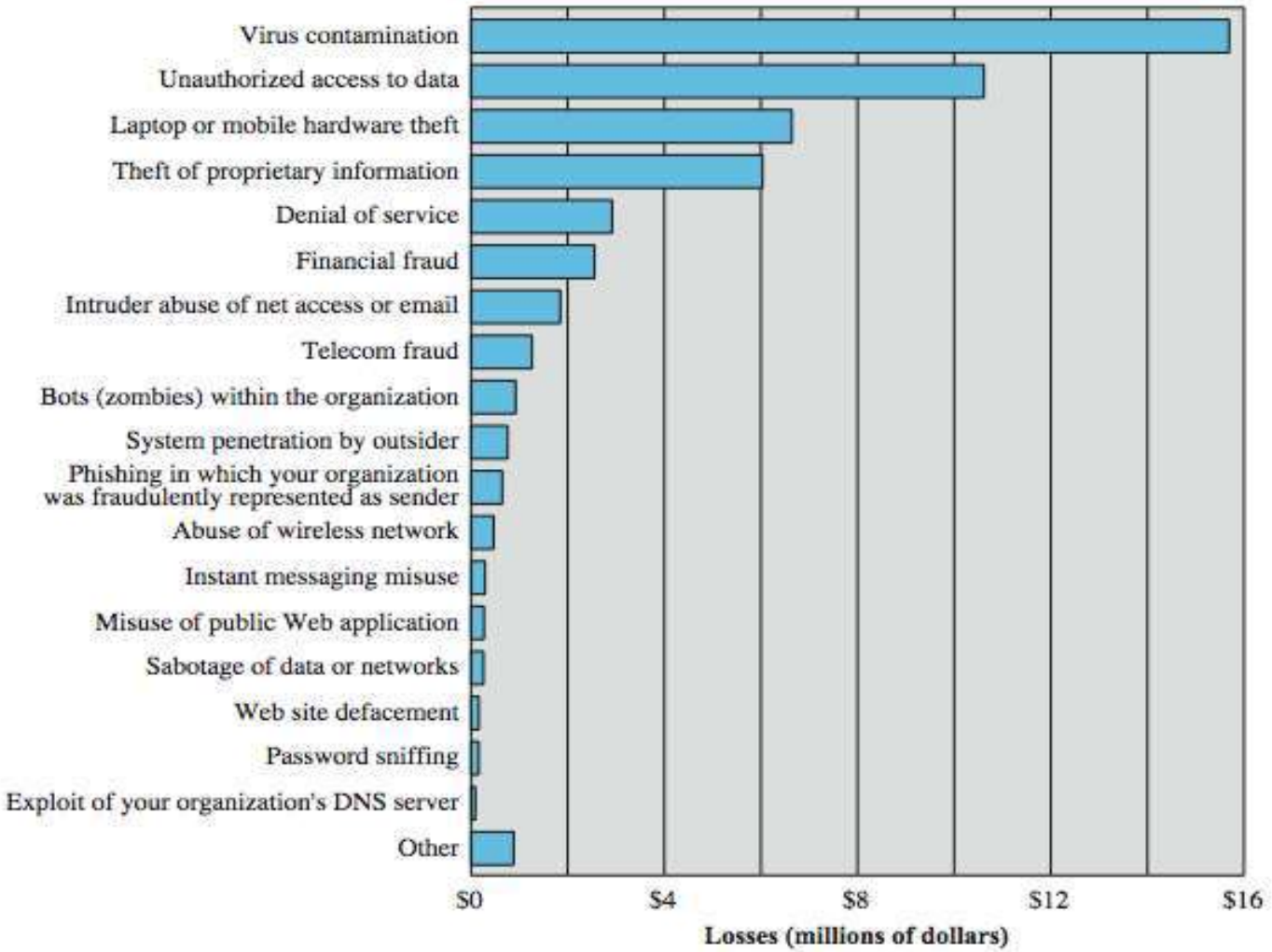
Vulnerabilities of OS and networking devices

Examples to incidents: DoS attacks, IP spoofing, attacks based on sniffing

Security Trends

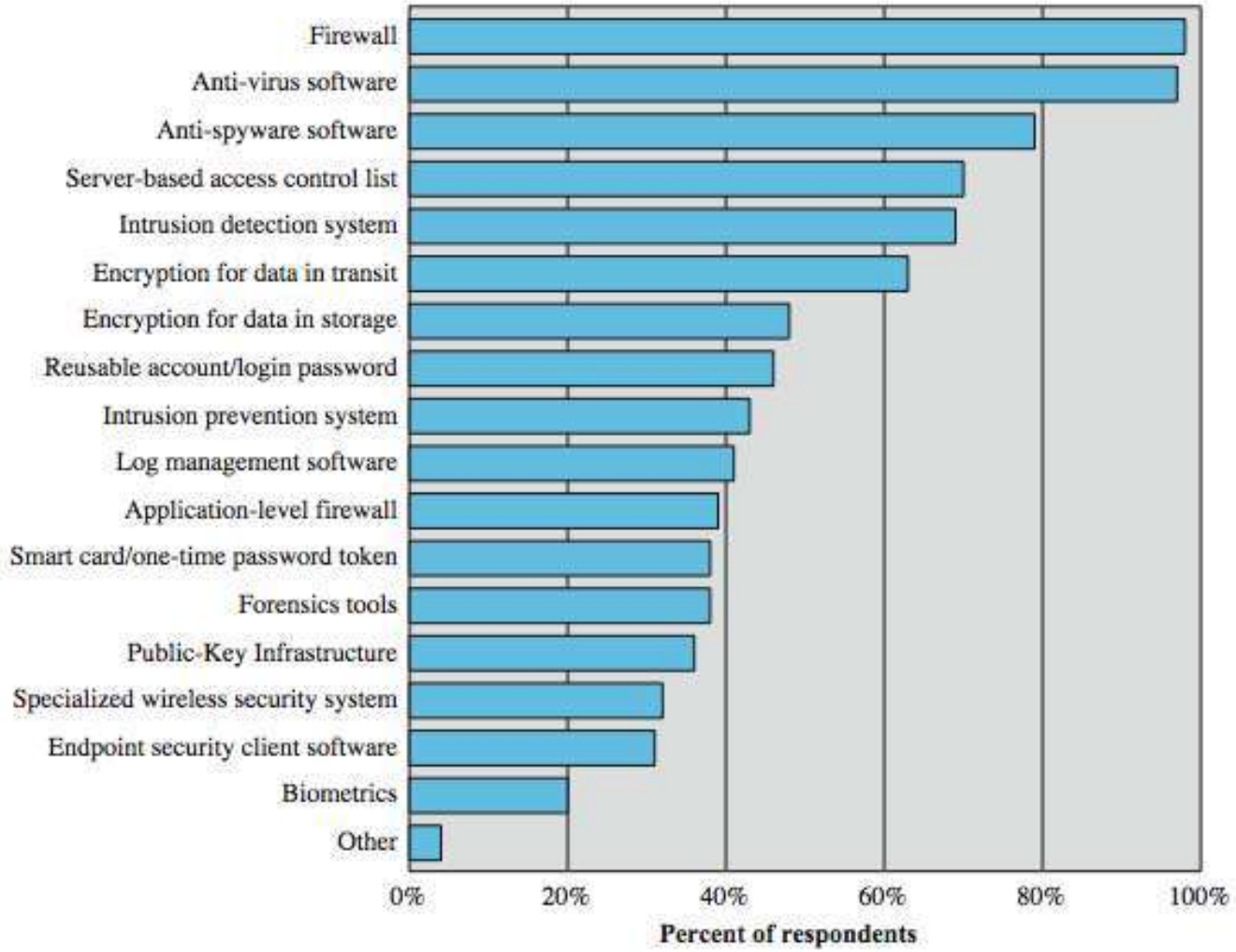


Computer Security Losses



Source: Computer Security Institute/FBI 2006 Computer Crime and Security Survey

Security Technologies Used



Source: Computer Security Institute/FBI 2006 Computer Crime and Security Survey

Services, Mechanisms, Attacks

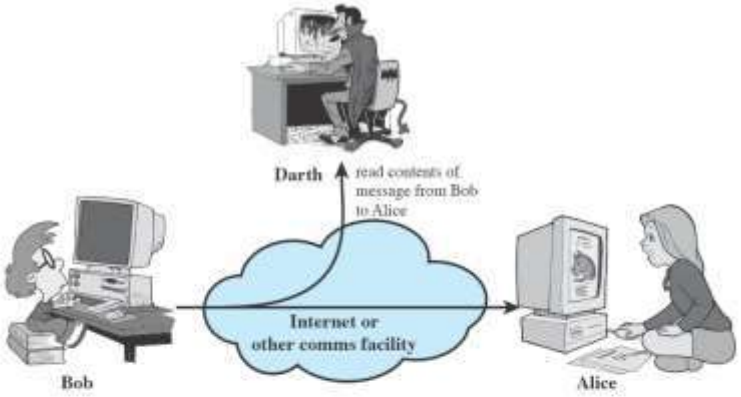
- 3 aspects of information security:
  - security attacks (and threats)
    - actions that (may) compromise security
  - security services
    - services counter to attacks
  - security mechanisms
    - used by services
    - e.g. secrecy is a service, encryption (a.k.a. encipherment) is a mechanism

Attacks

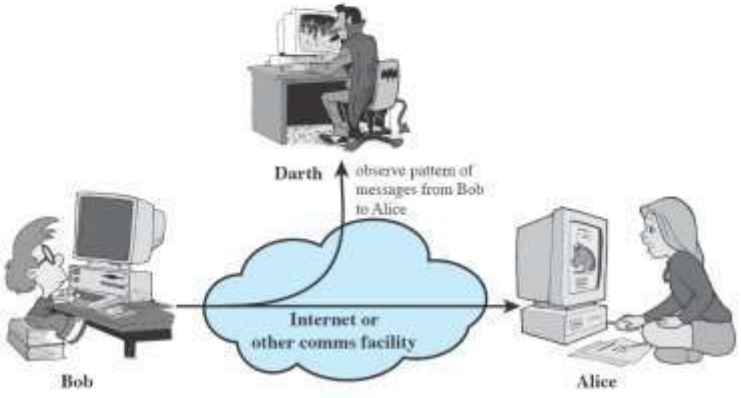
- Attacks on computer systems
  - break-in to destroy information
  - break-in to steal information
  - blocking to operate properly
  - malicious software
    - wide spectrum of problems
- Source of attacks
  - Insiders
  - Outsiders

Attacks

- Network Security
  - Active attacks
  - Passive attacks
- Passive attacks
  - interception of the messages
  - What can the attacker do?
    - use information internally
      - hard to understand
    - release the content
      - can be understood
    - traffic analysis
      - hard to avoid
  - Hard to detect, try to prevent



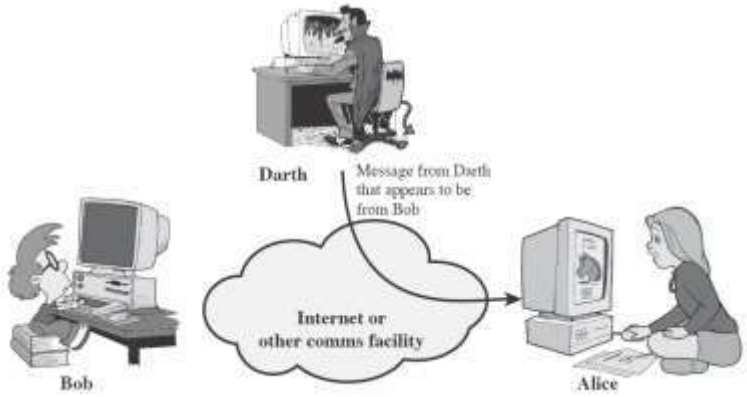
(a) Release of message contents

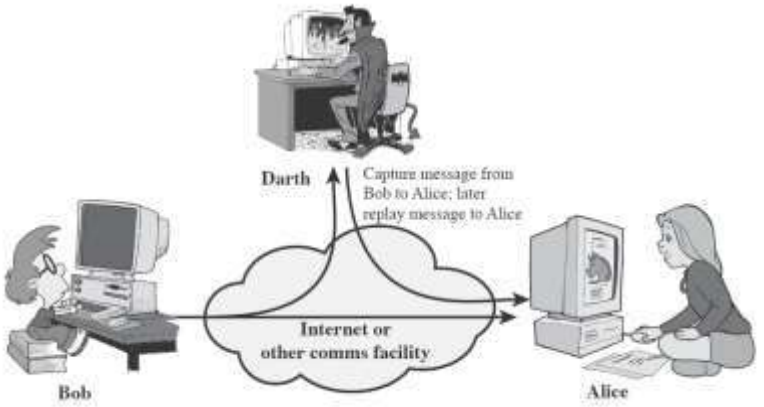


(b) Traffic analysis

Attacks

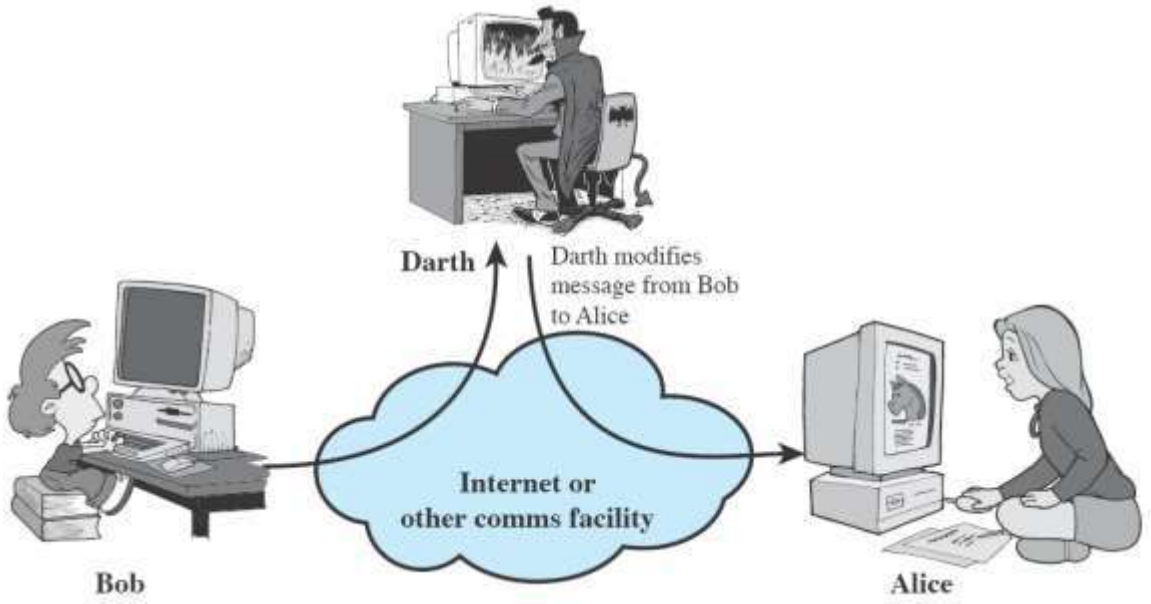
- Active attacks
  - Attacker actively manipulate the communication
  - Masquerade
    - pretend as someone else
    - possible to get more privileges
  - Fabrication
    - create a bogus message
  - Replay
    - passively capture data and send later
  - Denial-of-service
    - prevention the normal use of servers, end users, or network itself





Attacks

- Active attacks (cont'd)
  - deny
    - repudiate sending/receiving a message later
  - modification
    - change the content of a message



Security Services

- to prevent or detect attacks
- to enhance the security
- replicate functions of physical documents
  - e.g.
    - have signatures, dates
    - need protection from disclosure, tampering, or destruction
    - notarize
    - record

Basic Security Services

- Authentication
  - assurance that the communicating entity is the one it claims to be
  - peer entity authentication
    - mutual confidence in the identities of the parties involved in a connection
  - Data-origin authentication
    - assurance about the source of the received data
- Access Control
  - prevention of the unauthorized use of a resource

Basic Security Services

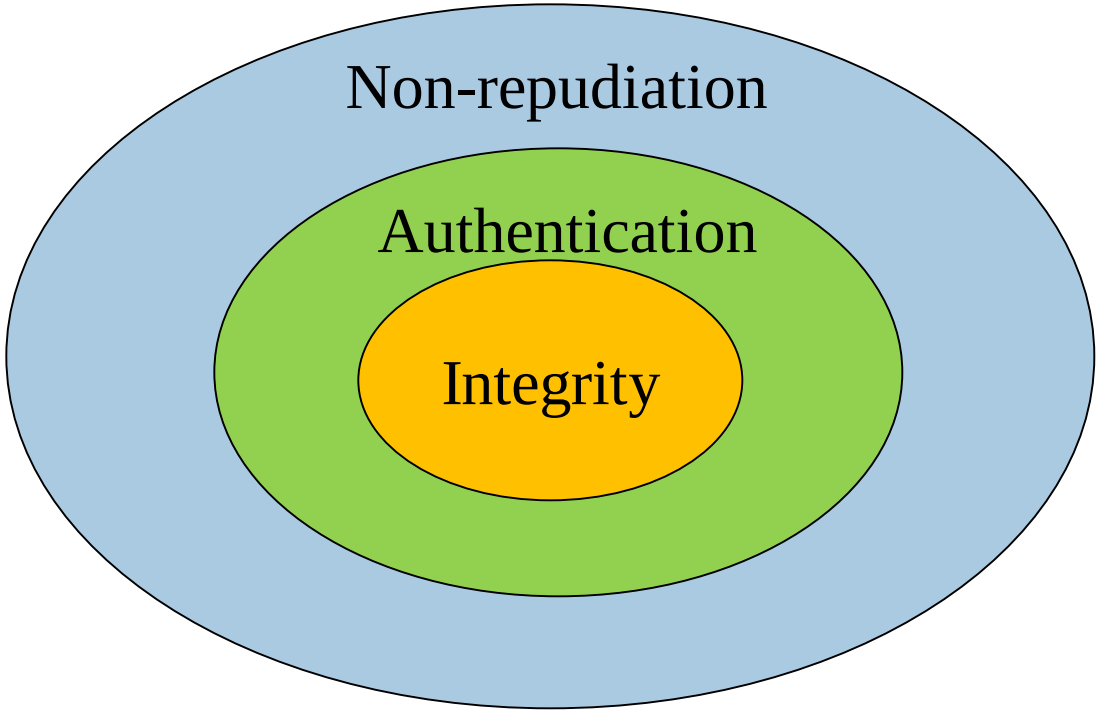
- Data Confidentiality
  - protection of data from unauthorized disclosure (against eavesdropping)
  - traffic flow confidentiality is one step ahead
- Data Integrity
  - assurance that data received are exactly as sent by an authorized sender
  - i.e. no modification, insertion, deletion, or replay

Basic Security Services

- Non-Repudiation
  - protection against denial by one of the parties in a communication
  - Origin non-repudiation
    - proof that the message was sent by the specified party
  - Destination non-repudiation
    - proof that the message was received by the specified party

Relationships

- among integrity, data-origin authentication and non-repudiation



Security Mechanisms

- Cryptographic Techniques
  - will see next
- Software and hardware for access limitations
  - Firewalls
- Intrusion Detection Systems
- Traffic Padding
  - against traffic analysis
- Hardware for authentication
  - smartcards
- Security Policies
  - define who has access to what resources.
- Physical security
  - Keep it in a safe place with limited and authorized physical access

Cryptographic Security Mechanisms

- Encryption (a.k.a. Encipherment)
  - use of mathematical algorithms to transform data into a form that is not readily intelligible
    - keys are involved

Cryptographic Security Mechanisms

- Message Digest
  - similar to encryption, but one-way (recovery not possible)
  - generally no keys are used
- Digital Signatures and Message Authentication Codes
  - Data appended to, or a cryptographic transformation of, a data unit to prove the source and the integrity of the data
- Authentication Exchange
  - ensure the identity of an entity by exchanging some information

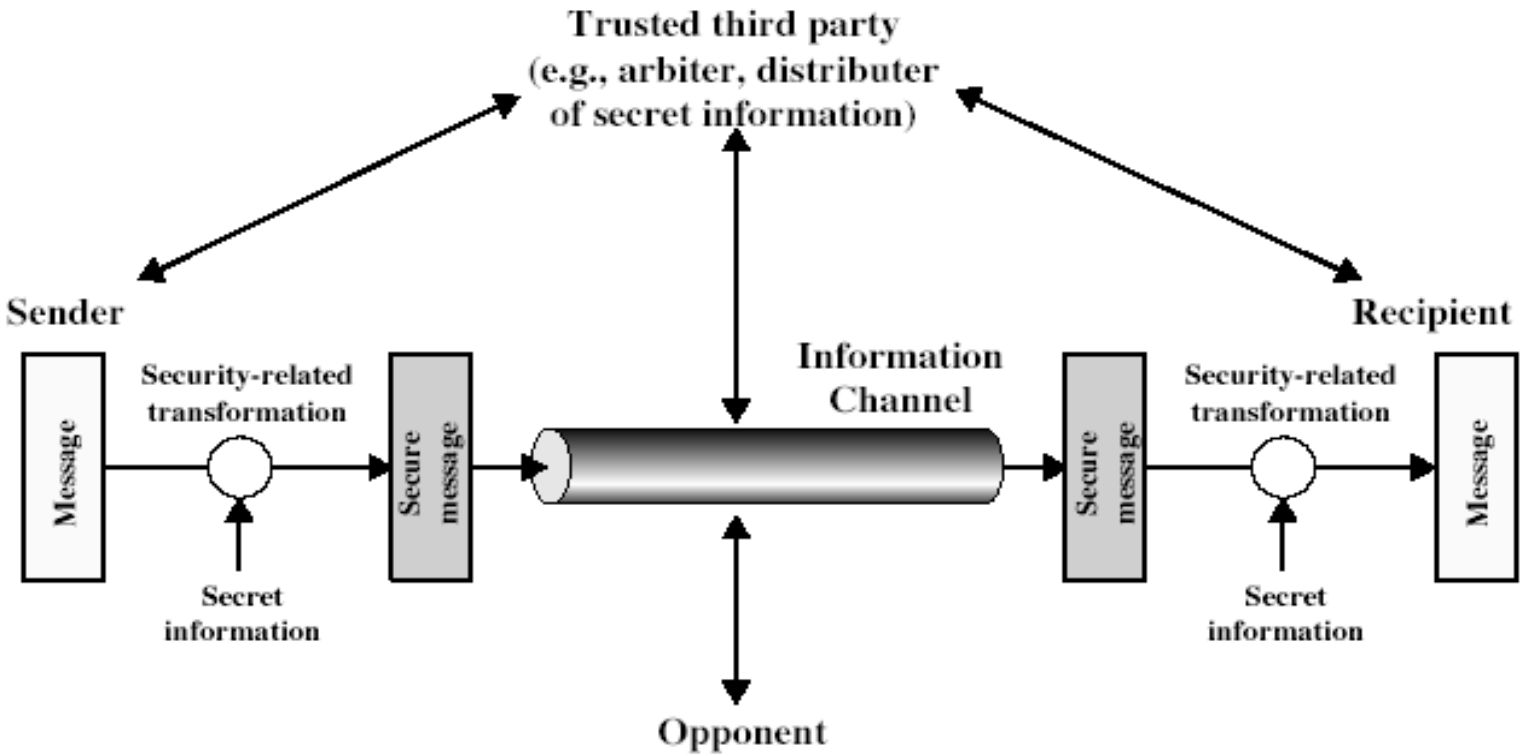
Security Mechanisms

- Notarization
  - use of a trusted third party to assure certain properties of a data exchange
- Timestamping
  - inclusion of correct date and time within messages

And the Oscar goes to …

- On top of everything, the most fundamental problem in security is
  - SECURE KEY EXCHANGE
    - mostly over an insecure channel

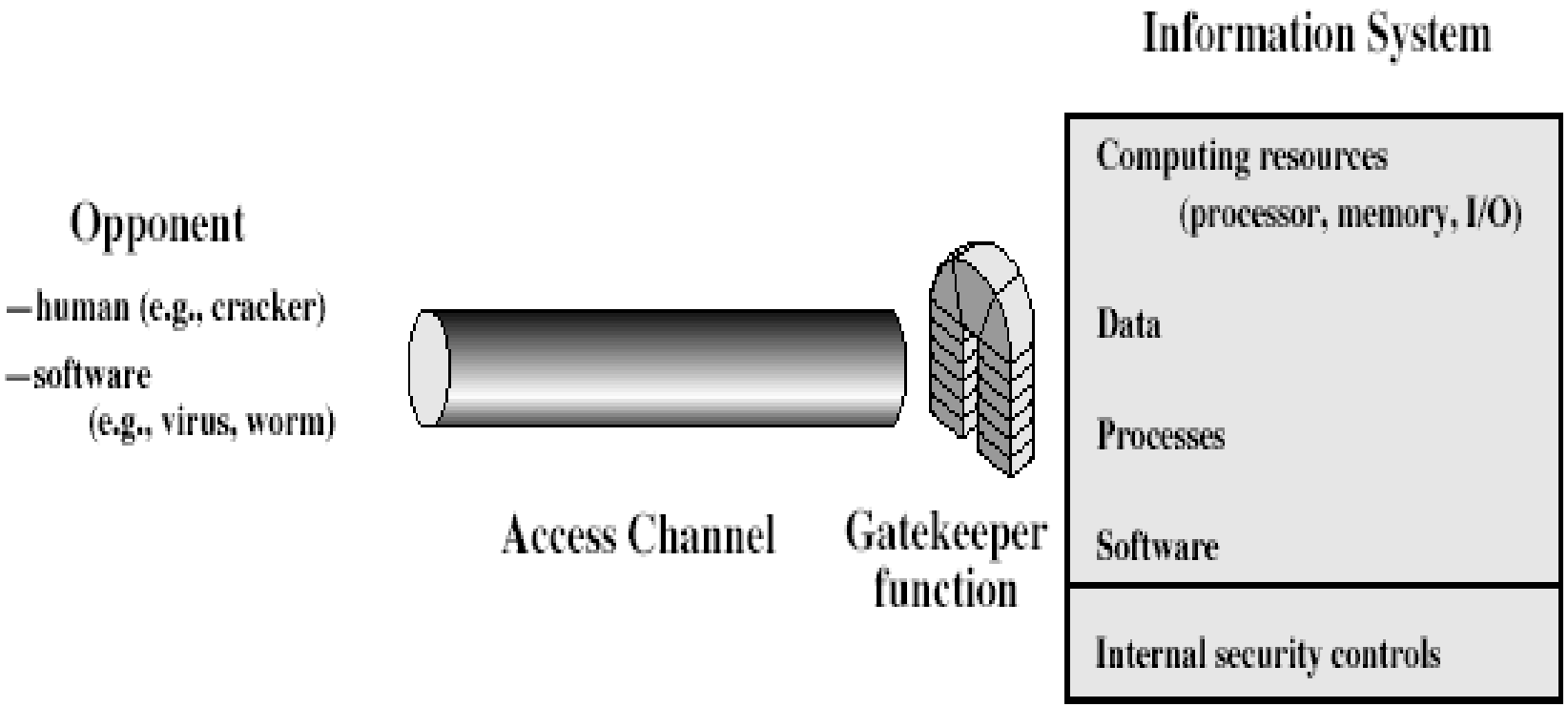
A General Model for Network Security



Model for Network Security

- using this model requires us to:
  - design a suitable algorithm for the security transformation
  - generate the secret information (keys) used by the algorithm
  - develop methods to distribute and share the secret information
  - specify a protocol enabling the principals to use the transformation and secret information for a security service

Model for Network Access Security



Model for Network Access Security

- using this model requires us to:
  - select appropriate gatekeeper functions to identify users and processes and ensure only authorized users and processes access designated information or resources
  - Internal control to monitor the activity and analyze information to detect unwanted intruders

More on Computer System Security

- Based on “Security Policies”
  - Set of rules that specify
    - How resources are managed to satisfy the security requirements
    - Which actions are permitted, which are not
  - Ultimate aim
    - Prevent security violations such as unauthorized access, data loss, service interruptions, etc.
  - Scope
    - Organizational or Individual
  - Implementation
    - Partially automated, but mostly humans are involved
  - Assurance and Evaluation
    - Assurance: degree of confidence to a system
    - Security products and systems must be evaluated using certain criteria in order to decide whether they assure security or not

Aspects of Computer Security

- Mostly related to Operating Systems
- Similar to those discussed for Network Security
  - Confidentiality
  - Integrity
  - Availability
  - Authenticity
  - Accountability
  - Dependability

Aspects of Computer Security

- Confidentiality
  - Prevent unauthorised disclosure of information
  - Synonyms: Privacy and Secrecy
    - any differences? Let's discuss
- Integrity
  - two types: data integrity and system integrity
  - In general, "make sure that everything is as it is supposed to be"
  - More specifically, "no unauthorized modification, deletion" on data (data integrity)
  - System performs as intended without any unauthorized manipulations (system integrity)

Aspects of Computer Security

- Availability
  - services should be accessible when needed and without extra delay
- Accountability
  - audit information must be selectively kept and protected so that actions affecting security can be traced to the responsible party
  - How can we do that?
    - Users have to be **identified** and **authenticated** to have a basis for access control decisions and to find out responsible party in case of a violation.
    - The security system keeps an **audit log (audit trail)** of security relevant events to detect and investigate intrusions.
- Dependability
  - Can we trust the system as a whole?

Fundamental Dilemma of Security

- **“Security unaware users have specific security requirements but no security expertise.”**
  - from D. Gollmann
  - Solution: level of security is given in predefined classes specified in some common criteria
    - Orange book (Trusted Computer System Evaluation Criteria) is such a criteria

Fundamental Tradeoff

- Between security and ease-of-use
- Security may require clumsy and inconvenient restrictions on users and processes

"If security is an add-on that people have to do something special to get, then most of the time they will not get it"

Martin Hellman,

co-inventor of Public Key Cryptography

Good Enough Security

"Everything should be as secure as necessary, but not securer"

Ravi Sandhu, "Good Enough Security", IEEE Internet Computing, January/February 2003, pp. 66- 68.

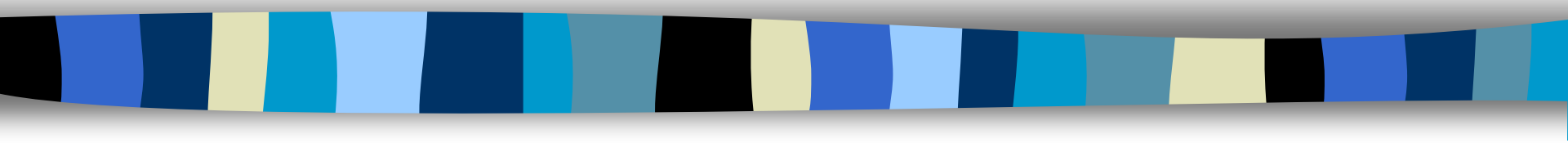
- Read the full article at

  http://dx.doi.org/10.1109/MIC.2003.1167341

Some Other Security Facts

- Not as simple as it might first appear to the novice
- Must consider all potential attacks when designing a system
- Generally yields complex and counterintuitive systems
- Battle of intelligent strategies between attacker and admin
- Requires regular monitoring
- Not considered as a beneficial investment until a security failure occurs
  - Actually security investments must be considered as insurance against attacks
- too often an afterthought
  - Not only from investment point of view, but also from design point of view

Overview of Cryptography



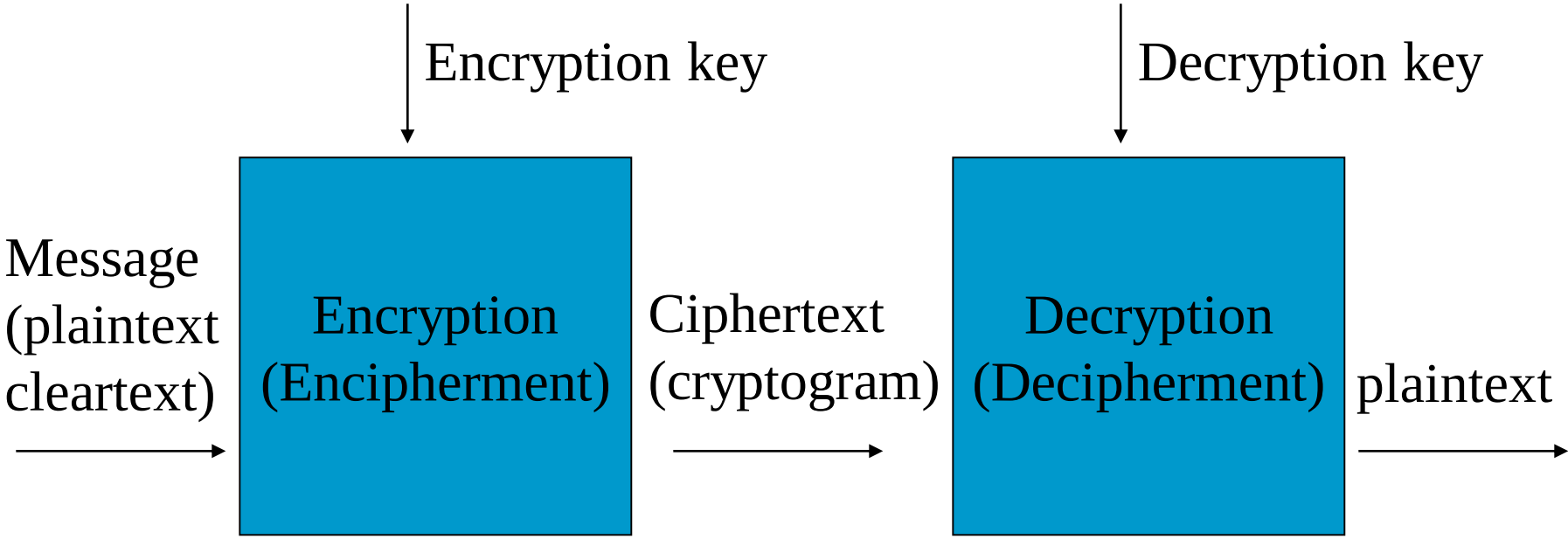
Part 1: Concepts and Principles

Part 2: Symmetric Cryptography

Meaning of Cryptography

- from Greek
  - Cryptos: secret, hidden
  - graphos: writing
  - cryptography: study (some calls *science* or *art* too) of secret writing

Basics


Encryption key
Decryption key
Message
(plaintext
cleartext)
Encryption
(Encipherment)
Ciphertext
(cryptogram)
Decryption
(Decipherment)
plaintext

Basic Terminology

- **plaintext** - the original message
- **ciphertext -** the coded message
- **cipher** - algorithm for transforming plaintext to ciphertext
- **key -** info used in cipher known only to sender/receiver
- **encipher (encrypt)** - converting plaintext to ciphertext
- **decipher (decrypt)** - recovering plaintext from ciphertext
- **cryptography -** study of encryption principles/methods
- **cryptanalysis (codebreaking)** - the study of principles/ methods of deciphering ciphertext *without* knowing key
- **cryptology** - the field of both cryptography and cryptanalysis

Kerckhoffs' principles

"The security of a cipher must not depend on anything that cannot be easily changed"

"The opponent is not to be underestimated. In particular, the opponent knows the encryption and decryption algorithms. So the strength of a cipher system depends on keeping the key information secret, not the algorithm"

Auguste Kerckhoff, 1883

Open discussion

- Published algorithm vs. unpublished algorithm

Characteristics of Cryptosystems

- types of operations for transformation into ciphertext
  - substitution
  - transposition
  - product
    - multiple stages of substitutions and transpositions
- number of keys used
  - single-key or *private key*
  - two-key or *public key*
- the way in which plaintext is processed
  - block
  - stream

Attacks on Ciphers

- Brute-force
  - try all possible keys until solved
- Cryptanalytic attacks
  - use
    - nature of algorithms
    - knowledge about general characteristics of plaintext
    - some sample plaintext-ciphertext pairs
    - Generally statistical techniques
  - aim
    - learn a specific plaintext
    - learn the key (that makes all past and future communication vulnerable)

Types of Cryptanalytic Attacks

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

A good algorithm…

- resists ciphertext-only and known-plaintext attacks
- Actually, no algorithm, but one is proven to be **unconditionally secure**
  - only one-time pad

Unconditionally Secure Encryption Scheme

- No matter
  - how much ciphertext is available to opponent
  - how much time and computing power that opponent has
- it is impossible for the opponent to decrypt the ciphertext
  - because there is no statistical relationship between the ciphertext and plaintext
- Only one-time pad is unconditionally secure

A Practical Encryption Scheme

- should be **computationally secure**
  - the cost of breaking the cipher exceeds the value of encrypted information
  - the time required to break the cipher exceeds the useful lifetime of the information
  - assumes the processing powers are limited and estimated breaking time is impractically long (millions of years!)

Brute Force Search

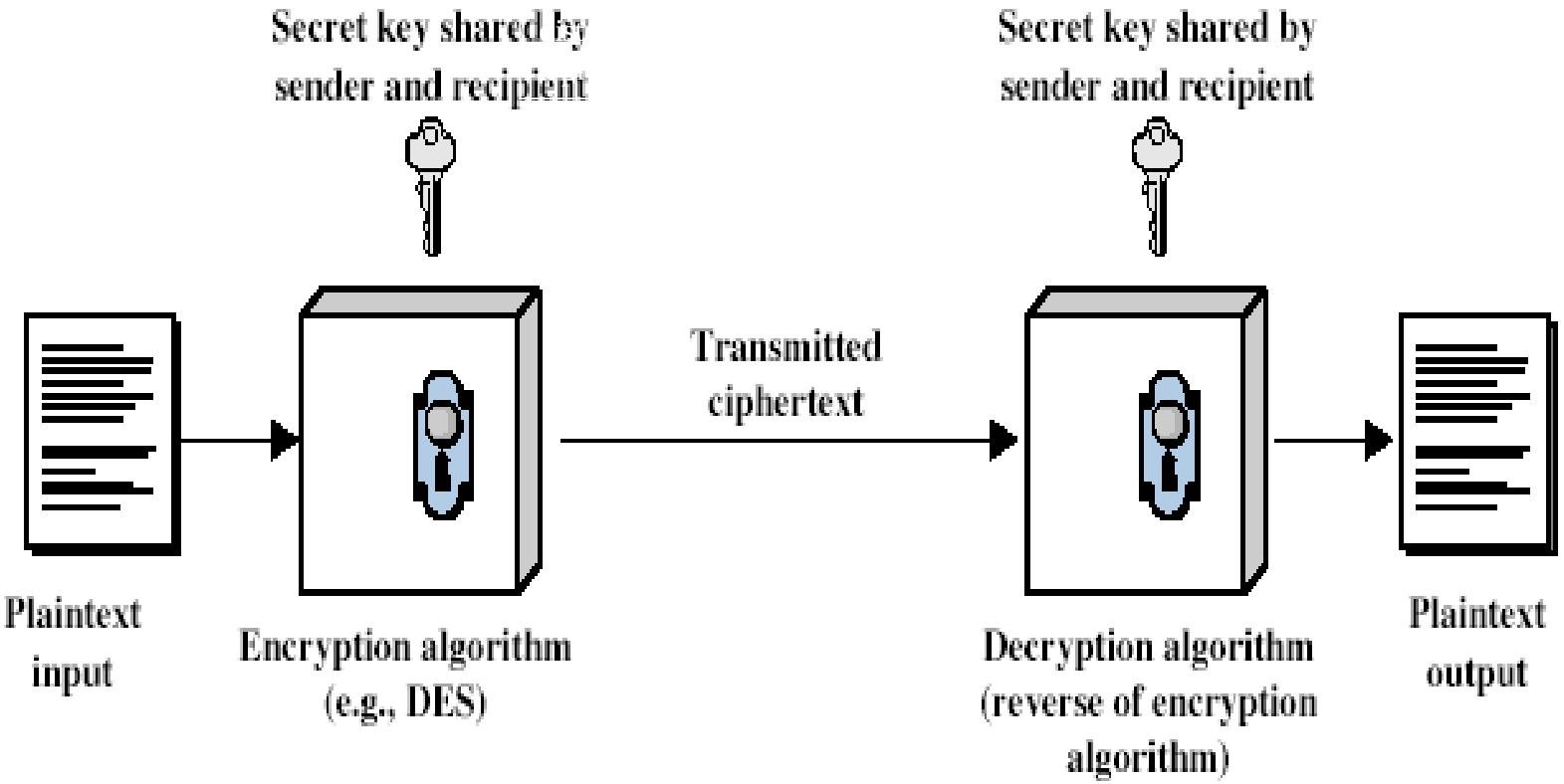
- simply try every key
- On average, half of the key space is searched until an intelligible translation is found

Key Size (bits)	Number of Alternative Keys	Time required at 1 encryption/μs	Time required at 10^6 encryptions/μs
32	$2^{32} = 4.3 \times 10^9$	$2^{31}\ \mu s = 35.8$ minutes	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	$2^{55}\ \mu s = 1142$ years	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	$2^{127}\ \mu s = 5.4 \times 10^{24}$ years	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	$2^{167}\ \mu s = 5.9 \times 10^{36}$ years	5.9×10^{30} years
26 characters (permutation)	$26! = 4 \times 10^{26}$	$2 \times 10^{26}\ \mu s = 6.4 \times 10^{12}$ years	6.4×10^{6} years

Symmetric Encryption

- also known as
  - classical
  - conventional
  - private-key
  - single-key
- sender and recipient share a common key
- was only type prior to invention of public-key cryptography
  - until second half of 1970’s

Symmetric Cipher Model



Requirements

- two requirements for secure use of symmetric encryption:
  - a strong encryption algorithm
  - a secret key, *K*, known only to sender and receiver

    $Y = E_K(X)$ //another notation $E(K, X)$

    $X = D_K(Y)$ //another notation $D(K, Y)$
- assumes encryption algorithm is known
- implies a secure channel to distribute key

Historical secret key cryptography - 1

- Pre-DES (before mid-70's)
  - Substitution and Permutation techniques
    - Substitution: each letter/symbol is replaced by another one
    - Permutation: same letters/symbols, but their orders are mixed
  - inspired DES and other modern block ciphers. Now, only has a theoretical value!
- earliest known is Caesar's cipher
  - replace each letter by the one with 3 letters (circularly) down in the alphabet
  - a becomes d, b becomes e, …, y becomes b, z becomes c
  - no key
  - Substitution technique

Historical secret key cryptography - 2

- Make the offset the key
  - 25 keys
  - easy to try
- Monoalphabetic ciphers
  - shuffle the letters arbitrarily based on a 26 letters long key

```
Plain:  abcdefghijklmnopqrstuvwxyz
Cipher: DKVQFIBJWPESCXHTMYAUOLRGZN

Plaintext:  ifwewishtoreplaceletters
Ciphertext: WIRFRWAJUHYFTSDVFSFUUFYA
```

Historical secret key cryptography - 3

- Security of Monoalphabetic ciphers
  - 26! = $4*10^{26}$ different keys
  - but still insecure due to redundancies in the natural languages
    - some letters or letter pairs/triples occur more than others
    - ciphertext reflects those characteristics
    - cryptanalysis is based on this fact and it really works
    - see the example on pages 38, 39 and 40 of the textbook

Historical secret key cryptography - 4

- Playfair cipher
  - improves security by encrypting the letters 2 by 2 (called digrams)
    - e.g. *hs* encrypts to *BP*
    - 26*26 = 676 digrams
    - cryptanalysis should be based on the frequency of the digrams which is more difficult than monoalphabetic crypto
  - invented by Charles Wheatstone in 1854, but named after his friend Baron Playfair
  - widely used for many years
    - by British army in WW1 as a standard system
    - also used (among other systems) in WW2 by the US Army and other allied forces

Historical secret key cryptography - 5

- Polyalphabetic substitution ciphers
  - different monoalphabetic substitutions as proceeding through the plaintext
  - key determines which monoalphabetic substitution rule to be applied to each letter
- Famous example is Vigenère cipher

```
key:        deceptivedeceptivedeceptive
plaintext:  wearediscoveredsaveyourself
ciphertext:ZICVTWQNGRZGVTWAVZHCQYGLMGJ
```

- makes cryptanalysis harder
  - multiple ciphertext letters for each plaintext letter
  - frequency distribution is kind of obscured, but cryptanalysis is still possible

Vigenere Table

Table 2.3 The Modern Vigenère Tableau

Key	Plaintext																									
	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
b	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
c	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
d	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
e	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
f	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
g	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
h	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
i	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
j	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
k	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
l	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
m	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
n	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
o	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
p	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
r	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
s	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
t	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
u	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
v	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
w	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
x	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Historical secret key cryptography - 6

- *Transposition* (or *permutation*) ciphers
  - hide the message by rearranging the letter order without altering the actual letters
  - same frequency distribution as the original text
    - cryptanalysis is possible
- Example scheme: write letters of message out in rows over a specified number of columns
  - then reorder the columns according to some key before reading off the rows

```
Key:        4 3 1 2 5 6 7
Plaintext: a t t a c k p
           o s t p o n e
           d u n t i l t
           w o a m x y z
Ciphertext: TTNAAPTMTSUOAODWCOIXKNLYPETZ
```

Towards modern cryptography - 1

- Vernam cipher
  - AT&T's Gilbert Vernam invented in 1918
  - treats the messages as binary data
  - XOR the plaintext with the key
    - reversible
  - very long key in tapes
    - repetitions possible for long messages
    - cryptanalysis is hard but possible with sufficient amount of ciphertext

Towards modern cryptography - 2

- One-time pad
  - key is random and as long as the plaintext
  - key is not re-used
  - unconditionally secure
    - ciphertext bears no statistical relationship to the plaintext
    - for a given ciphertext, there exists several intelligible decryptions that use different keys
    - even brute-force does not work, since it is not possible to understand which decryption is the correct one
  - generally, data and key are represented in binary and they are bitwise XORed
- Problems of one time pad in practice
  - large amount of random number generation
  - protection and safe distribution of those keys

Towards modern cryptography - 3

- Rotor machines
  - basic idea: multiple stages of substitutions
  - were widely used in WW2
    - German (Enigma), Japan (Purple)
  - implemented as a series of cylinders that move after each letter is encrypted
    - each cylinder represents a substitution alphabet
  - 3 cylinders = 26*26*26 = 17576 different substitution alphabets
    - this number is even bigger for 4 and 5 cylinders

Towards modern cryptography - 4

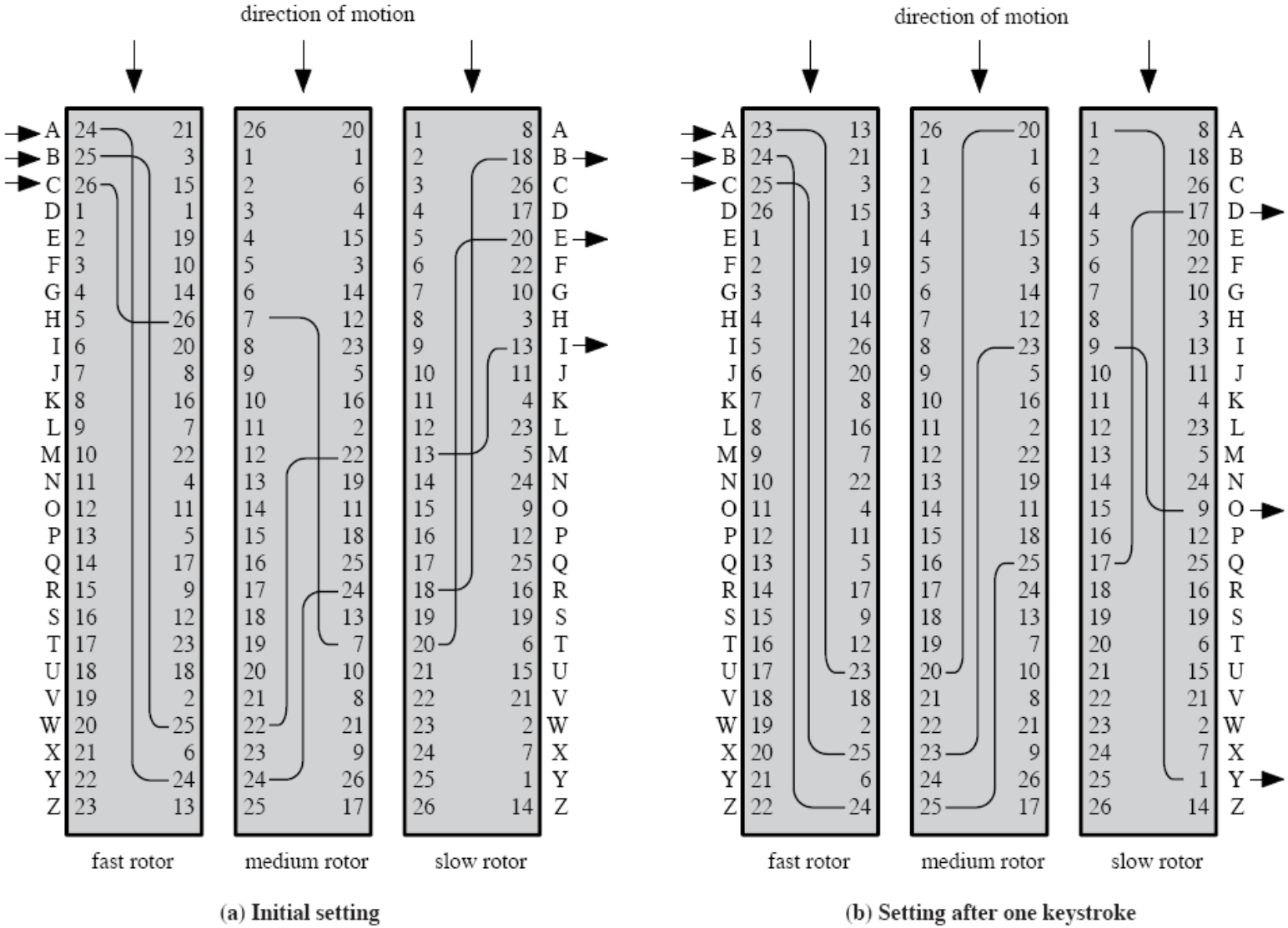


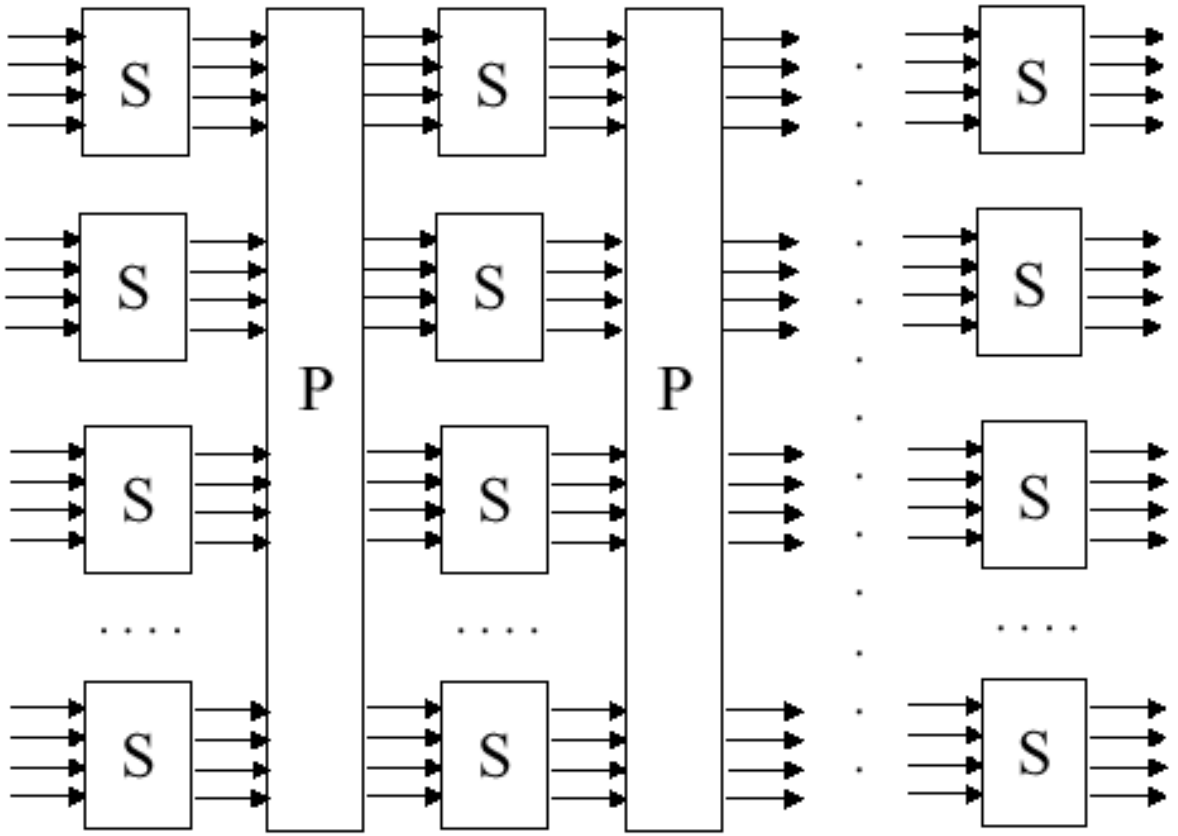
Figure 2.7 Three-Rotor Machine With Wiring Represented by Numbered Contacts

Towards modern cryptography - 5

- Product ciphers
  - general name for having multiple stages of substitutions, permutations or both
  - aim: to make cryptanalysis difficult by having irregularities in the cipher
  - rotor machine is an example
  - this idea led to Fiestel cipher and DES (Data Encryption Standard)
    - bridge between classical and modern ciphers

Towards modern cryptography - 5

- Product ciphers



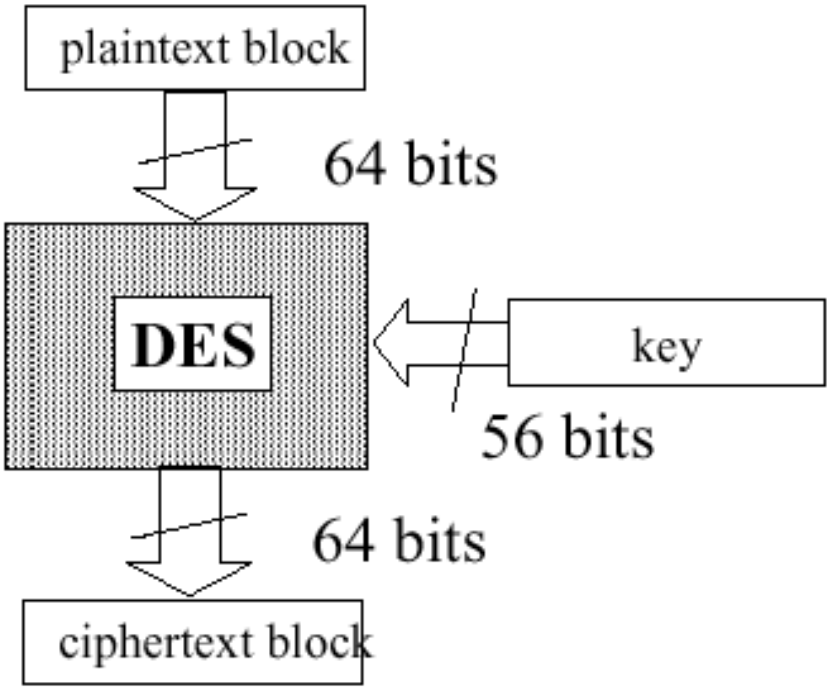
Modern Ciphers

- Block ciphers vs. Stream Ciphers
- Block ciphers operate on a block of data
  - entire block must be available before processing
- Stream ciphers process messages one bit or byte at a time when en/decrypting
  - need not wait the entire block
- Most ciphers are block ciphers
  - but it is possible to use a block cipher as a stream cipher (in some modes of operations that we will see later)

DES (Data Encryption Standard)

- most widely used block cipher in world
- adopted in 1977 by NBS (now NIST)
  - as FIPS PUB 46
- encrypts 64-bit data using 56-bit key
- has widespread use
- There has been considerable controversy over its security

DES – Black box view



DES History

- IBM developed Lucifer cipher
  - by team led by Horst Feistel (1971)
  - used 64-bit data blocks with 128-bit key
- then redeveloped as a commercial cipher with input from NSA and others
- in 1973 NBS issued request for proposals for a national cipher standard
- IBM submitted their revised Lucifer which was eventually accepted as the DES
  - 56-bit key size!
- recertified in 1983, 1987 and 1993
- 3-DES (triple DES) has been issued as a new standard in 1999

DES Controversy

- Controversy over design
  - in choice of 56-bit key (vs Lucifer 128-bit)
  - design criteria (of the S-boxes) were classified
- S-boxes were fine
- but 56-bits became problem for DES as time goes by
  - Due to advances in cryptanalysis and electronics
  - in 1998 a project funded ($220K) by EFF (Electronic Frontier Foundation) broke DES in less than three days

Design of DES

- is not our concern in this course
- neither the details of cryptanalysis of DES
- will give only basic characteristics of DES in the next few slides

DES Characteristics

- DES is basically a product cipher
  - several rounds of substitutions and permutations
  - actually not that simple ☺
- originally designed for hardware implementation
  - software implementations validated in 1993
  - but software DES is slow

DES Characteristics

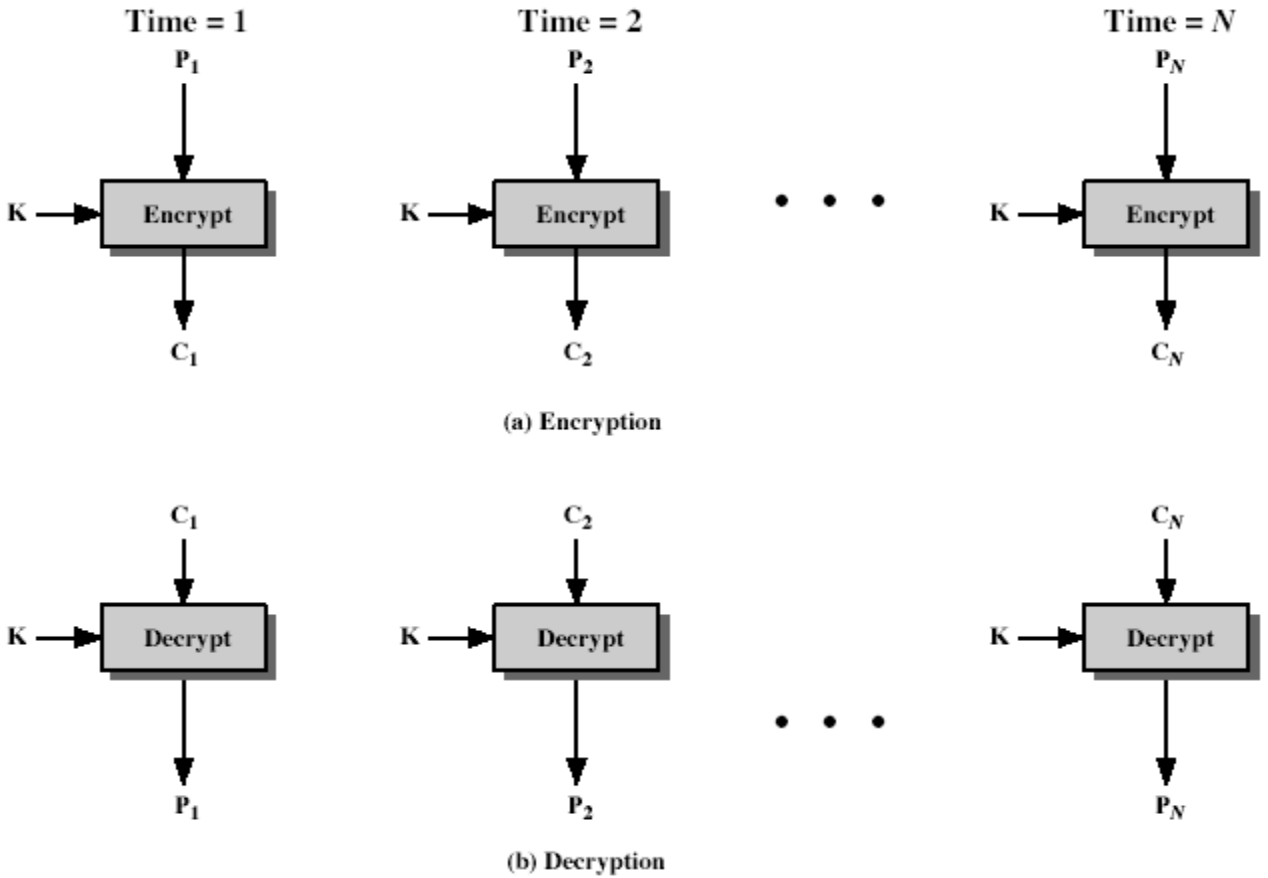
- DES shows strong avalanche effect
  - one bit change in the input affects on average half of the output bits
  - to make attacks based on guessing difficult
- S-boxes are non-linear
  - provides confusion
    - i.e. makes relationship between ciphertext and key as complex as possible

DES Mode of Operations

- block ciphers encrypt fixed size blocks
  - DES encrypts 64-bit blocks, with 56-bit key
- in practise, we have arbitrary amount of information to encrypt
  - we use DES (and other symmetric ciphers too) in different modes
  - four were defined for DES in ANSI standard ANSI X3.106-1983
- then a fifth one is added
- Similar modes exists for AES and other block ciphers as well

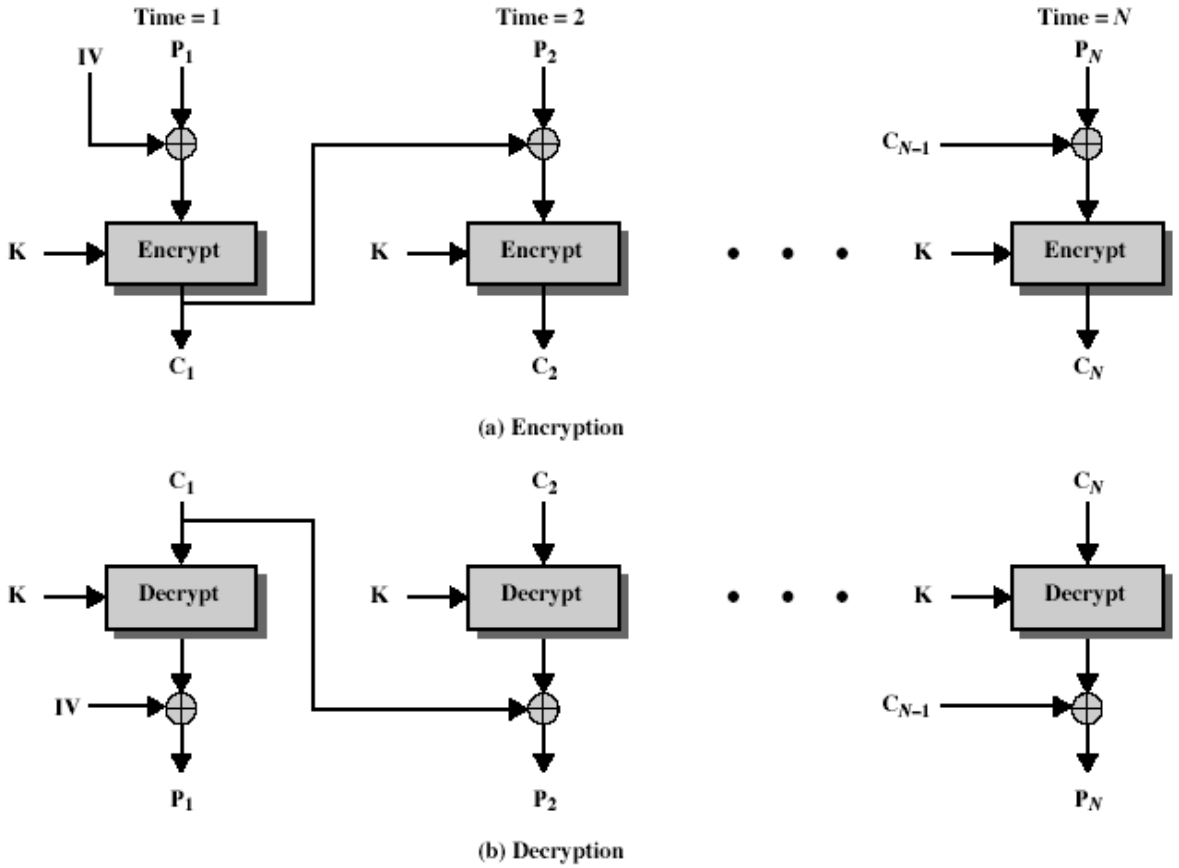
Electronic Codebook Book (ECB) Mode

- each block is encrypted independent of the other blocks
  - using the same key
- not so secure for long messages due to repetitions in code



Cipher Block Chaining (CBC)

- each previous cipher blocks is XORed with current plaintext
- each ciphertext block depends on other message blocks
- need Initialization Vector (IV) known to sender & receiver



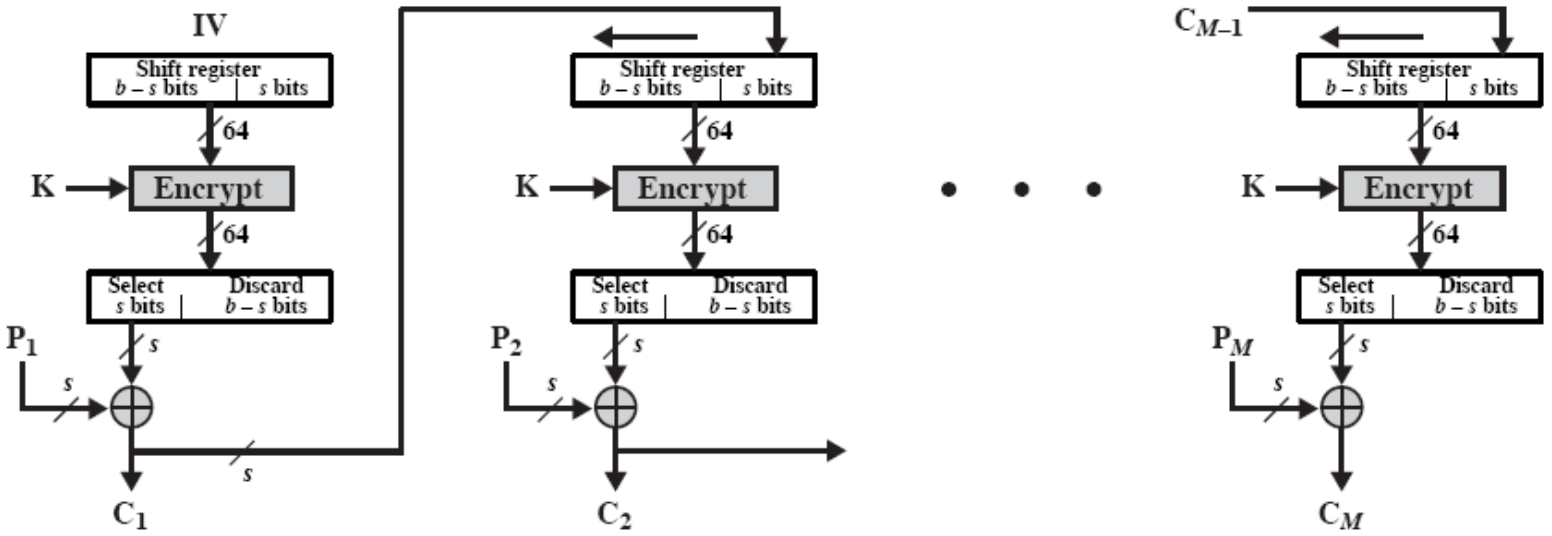
Cipher Block Chaining (CBC)

- Initialization Vector (IV)
  - both parties should agree on an IV
  - for maximum security, IV should be protected for unauthorized changes
  - Otherwise, attacker's change in IV also changes the decrypted plaintext
    - let's see this on board

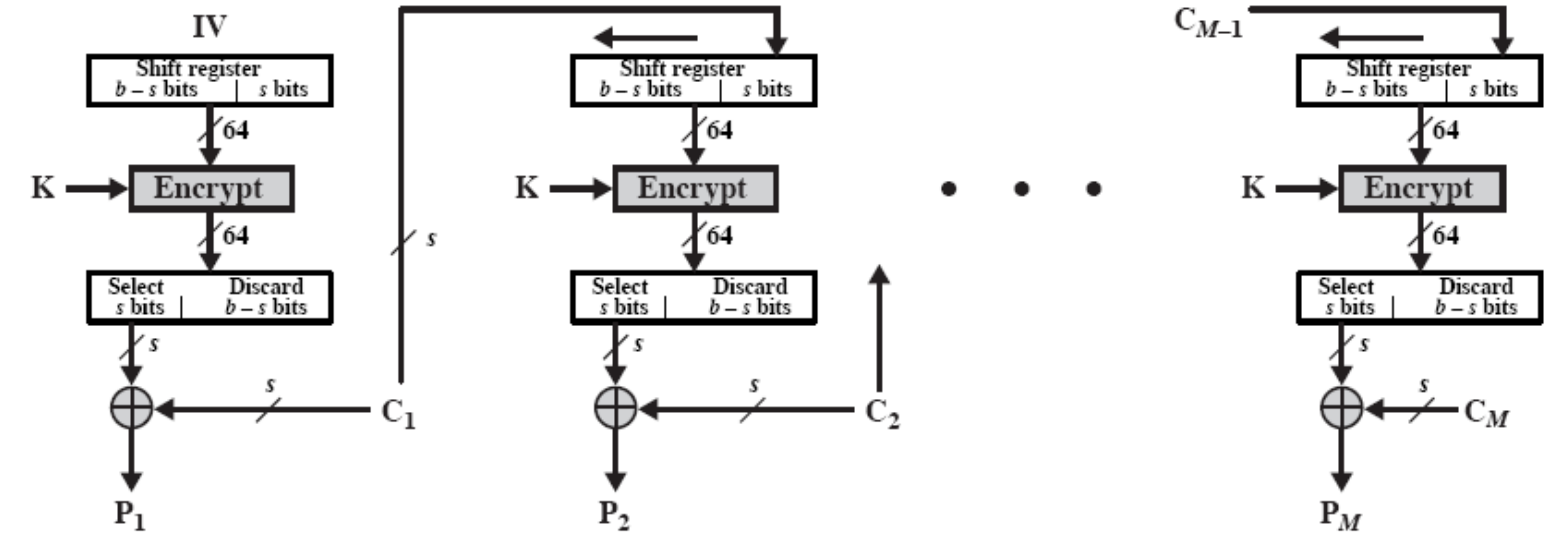
Cipher FeedBack (CFB)

- Message is treated as a stream of bits
  - DES (or any other block cipher) is used as a stream cipher
- standard allows any number of bit, *s*, (1,8 or 64) as the unit of encryption/decryption
- uses IV
  - as all other stream ciphers
- transmission errors propagate

Cipher FeedBack (CFB)



(a) Encryption

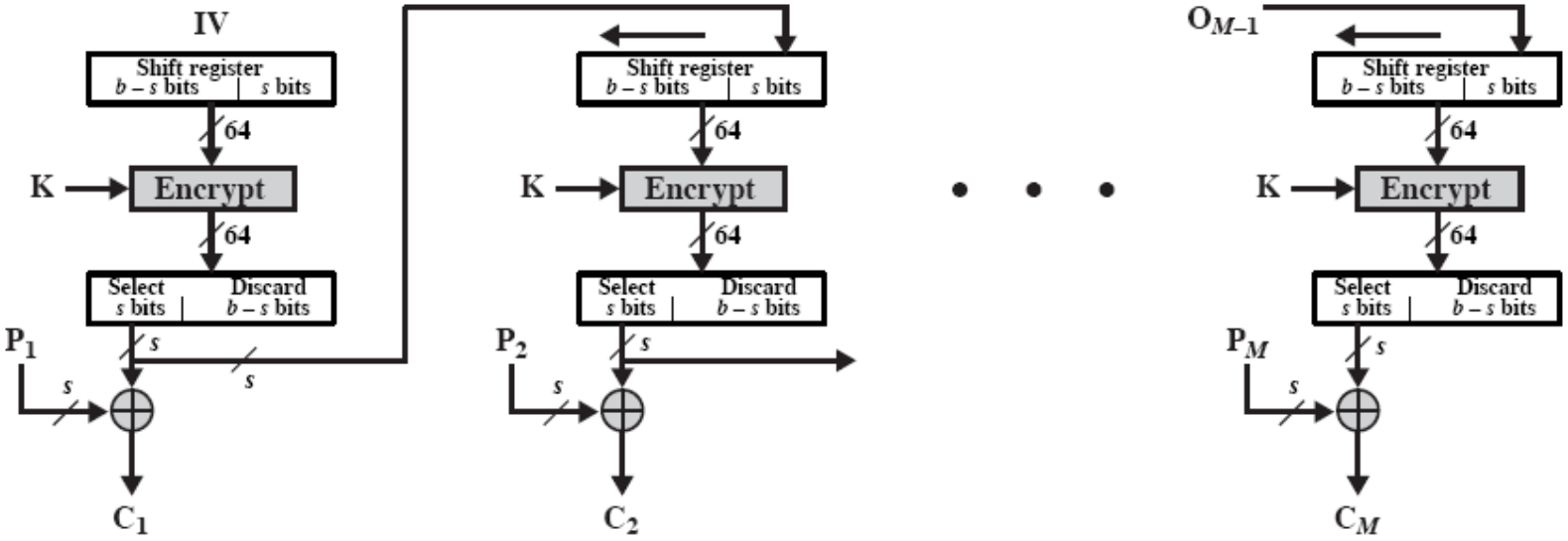


(b) Decryption

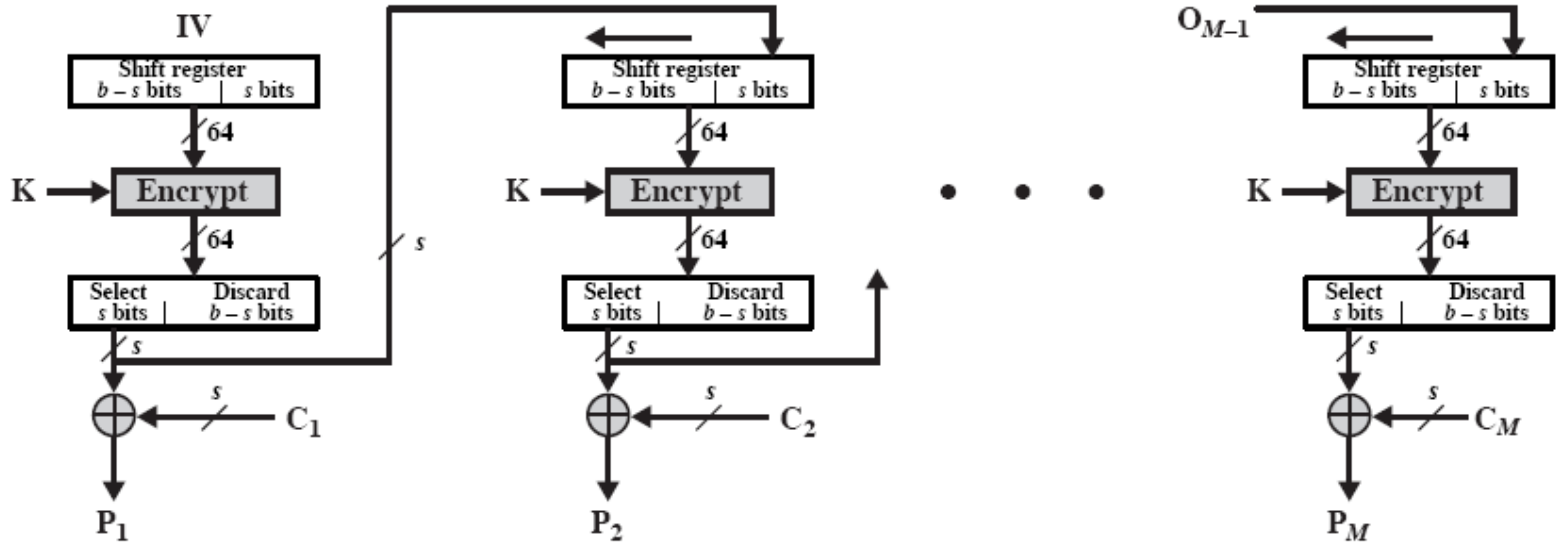
Output FeedBack (OFB)

- another stream mode
- output of cipher is
  - XORed with the message
  - it is also the feedback
- feedback is independent of transmission, so transmission errors do not propagate
- same IV should not be used twice for the same key (general problem of using IV)
  - otherwise, when two ciphertext blocks are XORed the random sequence is cancelled and the attacker has obtains XOR of two plaintexts

Output FeedBack (OFB)



(a) Encryption

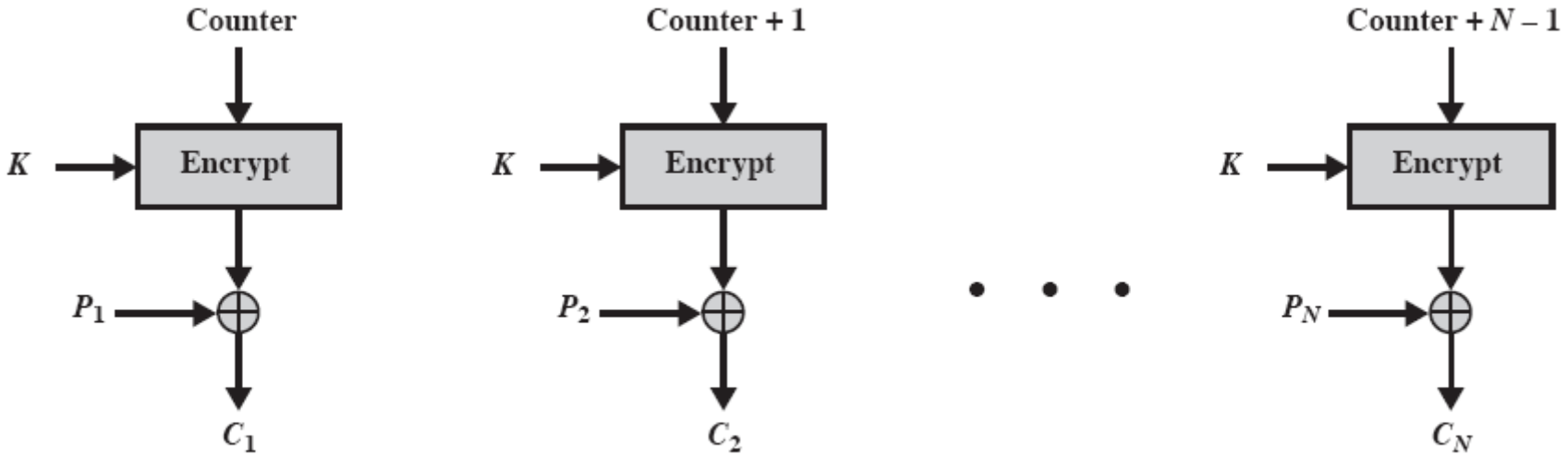


(b) Decryption

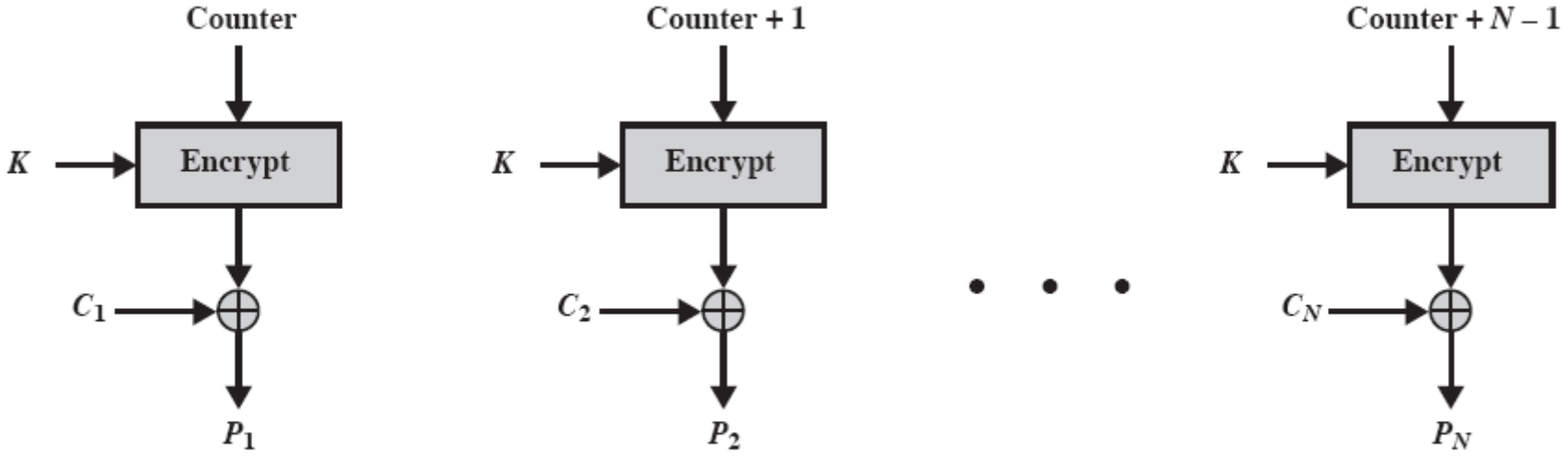
Counter (CTR)

- similar to OFB but encrypts counter value rather than any feedback value
- For the same key, the counter value should not repeat
  - same problem as in OFB
- efficient
  - can do parallel encryptions
  - encryption/decryption process is performed in advance of need
  - good for bursty high speed links

Counter (CTR)



(a) Encryption



(b) Decryption

Other Important Symmetric Ciphers

- AES (Rjindael)
- 3DES (Triple DES)
- Blowfish
- RC5
- IDEA
- RC4

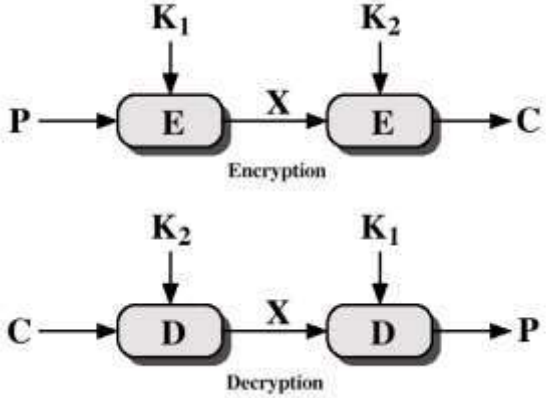
What happened after DES

- Replacement for DES was needed
  - vulnerability to cryptanalysis and practical brute-force attacks
- AES is the new standard (will see)
  - But took some time to standardize and deploy
- Meanwhile, some other ciphers are also used in practice (will briefly discuss too)
- But we still needed an immediate replacement of DES that can be standardized and deployed easily
  - This was 3DES

3DES (Triple DES)

- Another method for a strong cipher
- use multiple encryption with DES with different keys
  - to preserve the investment in DES
  - for quicker deployment
- Triple DES is chosen as a standard method
  - Standardized by ANSI, ISO and NIST

Why not double DES?

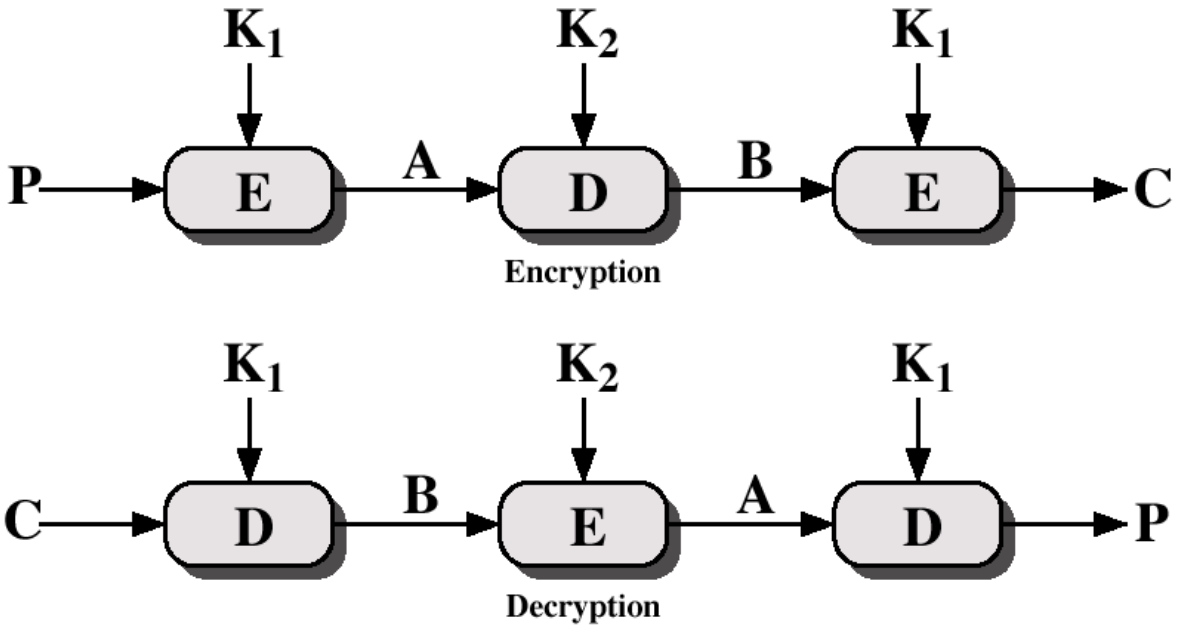


- Double DES
  - use DES two times with two different keys
  - Does not work due to meet-in-the-middle attack (which is a known-plaintext attack)
    - $X = E_{K1}[P] = D_{K2}[C]$
    - Try all possible K1's on P to create all possible X's and store them sorted
    - Try all possible K2's on C and match with above table
    - may create some false-alarms, so do the same attack for another plaintext-ciphertext pair
    - If the same K1-K2 pairs match for the second plaintext-ciphertext pair, then the correct keys are most probably found
    - complexity of this attack is close to the complexity of the single-DES brute-force attack, so double-DES is useless

Triple-DES

- Three stages of DES
  - with two different keys
    - some attacks are possible but impractical
    - Merkle and Hellman , 1981
      - 2^{56} trials, but requires 2^{56} plaintext-ciphertext pairs
    - Oorschot and Wiener, 1990
      - $2^{120}/n$ trials, where n is the number of plaintext-ciphertext pairs
  - with three different keys
    - complexity of meet-in-the-middle increases and becomes impractical

Triple-Des with two keys



- E-D-E sequence
  - use of decryption at the second stage does not reduce/increase the security
  - Why decryption in the middle stage?

Triple-DES with three keys

- For those who feel some concern about the attacks on two-key 3-DES
- E-D-E sequence

$$C = E_{K3}[D_{K2}[E_{K1}[P]]]$$

- has been adopted by some Internet applications, eg PGP, S/MIME

Blowfish

- Developed by Bruce Schneier
  - author of the book *Applied Cryptography*
- 64-bit of block size
- Key size is variable
  - one to fourteen 32-bit blocks
    - 32 to 448 bits
    - provides a good trade-off between security and performance
- Fast and compact
- Has been implemented in numerous products
  - including GnuPG, SSH
  - see http://www.schneier.com/blowfish-products.html
- no known practical security problems

RC5

- Ron's Code 5
  - developed by Ron Rivest who is also co-inventor of RSA cryptosystem
- owned and extensively used by RSA Inc.
- highly parametric
- word oriented processing that uses primitive operations that can be found in instruction sets of almost all microprocessors

RC5-w/r/b

- RC5 is actually a family of algorithms
- Parameters: w, r, b
  - w: Word size
    - 16, 32 or 64 bits
    - block size is 2*w
  - r: Number of rounds
    - 0 .. 255
  - b: key size in octets
    - 0 .. 255
- RC5 as suggested by Rivest is
  - RC5-32/12/16
  - 32-bit words (i.e. 64 bit blocks), 12 rounds, 128-bit key size

IDEA

- International Data Encryption Algorithm
- Lai and Massey of ETH Zurich (Swiss Federal Institute of Technology), 1990/91
- 64-bit blocks, 128-bit key size
- one of the early 128-bit algorithms
  - not US originated, so no export restrictions
  - used widely in PGP

AES (Advanced Encryption Standard)

- Replacement needed for DES
  - reasons discussed before
- 3DES is a solution, but temporary
  - 3DES is slow in software
  - 3DES uses small blocks that makes even slower
- Need a new standard cipher

AES Events in Chronological Order

- NIST issued call for a standard cipher in 1997
  - international
- 15 candidates (out of 21) accepted in June 98
- A shortlist of 5 selected in August 99
- Rijndael (from Belgium) was selected as the AES in October 2000
- issued as FIPS PUB 197 standard in November 2001

AES Requirements

- private key symmetric block cipher
- 128-bit data (block size)
- 128/192/256-bit keys
- stronger & faster than Triple-DES
- active life of 20-30 years
- provide full specification and design details

5 AES candidates

- MARS (IBM)
- RC6 (USA)
- Rijndael (Belgium)
- Serpent (Europe)
- Twofish (USA)

- Europe vs. USA
- commercial vs. academic
  - US based ones were all of commercial origin

AES Evaluation Criteria

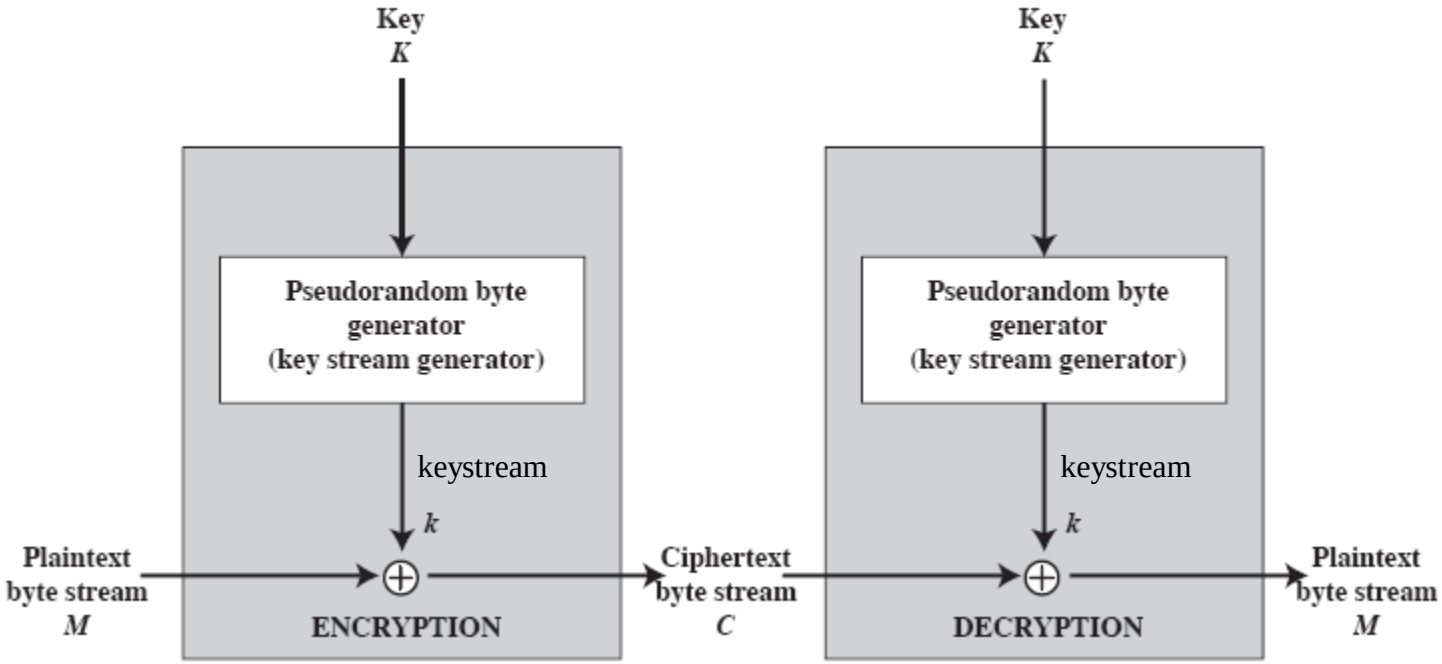
- final criteria (used to select the winner)
  - general security
    - NIST relied on evaluation done by cryptographic community
  - software implementation performance
    - execution speed, performance across different platforms (8 to 64 bit platforms)
  - hardware implementation
    - not only timings, but also cost is important
    - especially for restricted space environments (such as smartcards)
  - implementation (timing and power) attacks

The AES Cipher - Rijndael

- designed by Vincent Rijmen and Joan Daemen in Belgium (UCL)
- has 128/192/256 bit keys, 128 bit data
- Characteristics
  - resistant against known attacks
  - speed and code compactness on many platforms
  - design simplicity

Stream Ciphers

- process the message bit by bit
- Simply stating
  - a key and a Pseudo Random Number Generator (PRNG) is used to create a (pseudo) random key stream
  - keystream and the plaintext bitwise XORed to create the ciphertext
  - ciphertext is XORed with the same keystream to restore the plaintext



Some Stream Cipher Design Considerations

- A PRNG should eventually repeat
  - long period makes cryptanalysis difficult
- statistically randomness
  - e.g. approx. equal number of 0's and 1's
- large enough key (128-bit would be good to guard against brute-force attacks)

Stream Ciphers

- randomness of keystream destroys any statistical properties in the message
  - as in Vernam cipher and one-time pads
- Better than block ciphers in terms of
  - code space (implementations are simple)
  - throughput (faster per bit en/decryption)
- but must never use the same keystream more than once
  - otherwise the cryptanalyst can XOR two ciphertext streams and find out XOR of two plaintext streams
    - not so difficult to crack

Stream Ciphers

- are useful if data are transferred as a stream
  - web browser
  - voice
  - video
- actually any block cipher can be used as a stream cipher
  - OFB, CFB modes of operations

RC4

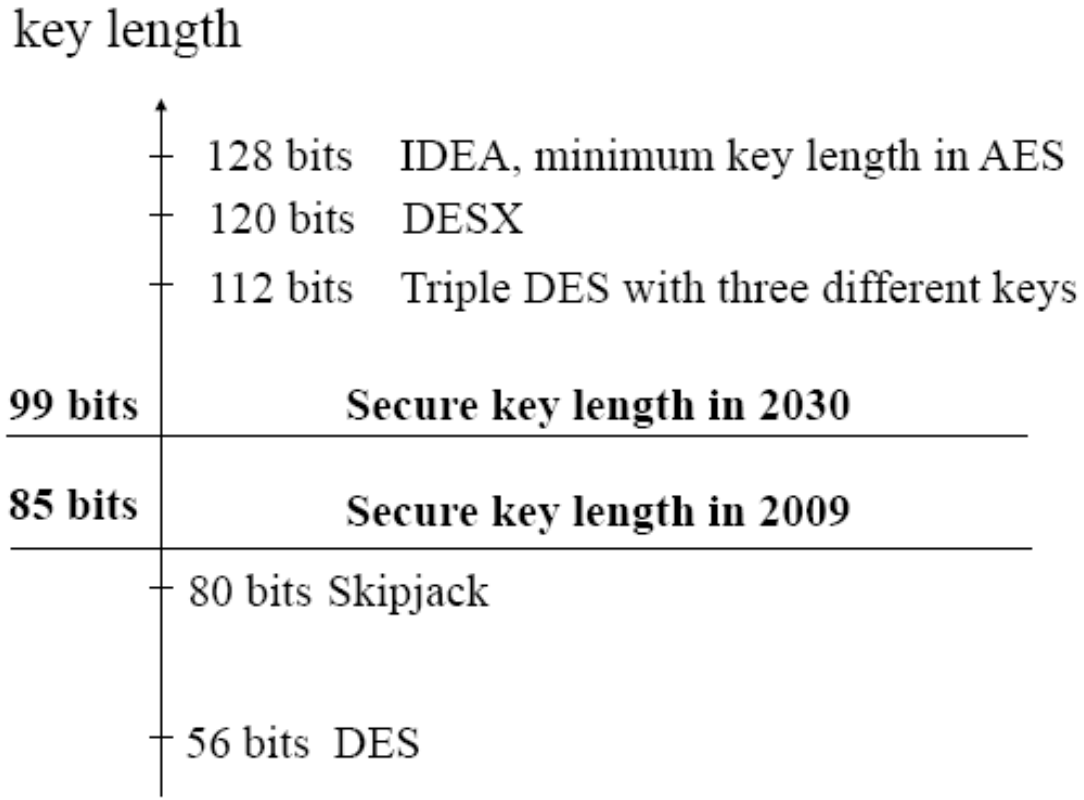
- Ron's Code 4
- Yet another cipher designed by Ron Rivest
  - owned by RSA Inc.
  - was kept as a trade secret, but in 1994 anonymously posted on the Internet
- variable key size, byte-oriented stream cipher
- simple but effective
  - 8 to 16 machine operations per output byte
- widely used (SSL/TLS, WEP/WPA)
- Some attacks reported, but not practical for key size greater than 128-bit
- However, WEP has a problem due to RC4 key generation
  - not a problem of RC4 in particular

and other symmetric ciphers

- CAST
- Skipjack
- Serpent
- Twofish
- RC6
- Mars
- SAFER+

Discussion

Secure key length today and in 20 years
(against an intelligence agency with the budget of $300M)



Courtesy of Kris Gaj

Discussion

- Assuming 80-bit is secure enough for today and Moore's Law continues
  - 1 bit per 18 months to be added
    - 2020's: 92-bit (approx.)
    - 2040's: 106-bit (approx.)
  - with 128-bit, AES we will be secure for a long time
- unless a new efficient cryptanalysis method is found
  - known cryptanalysis methods are not practical for secure key sizes for 3DES, AES, IDEA, etc. (except DES of course)

Overview of Cryptography

Part III: Public-key cryptography

Part IV: Other Cryptographic Primitives

Public-Key Cryptography – General Characteristics

- public-key/two-key/asymmetric cryptography
  - A concept, there are several such cryptosystems
- probably the only revolution in the history of cryptography
- uses 2 keys
  - public-key
    - may be known by anybody, and can be used to encrypt messages, and verify signatures
  - private-key
    - known only to the owner, used to decrypt messages, and sign (create) signatures

Public-Key Cryptography – General Characteristics

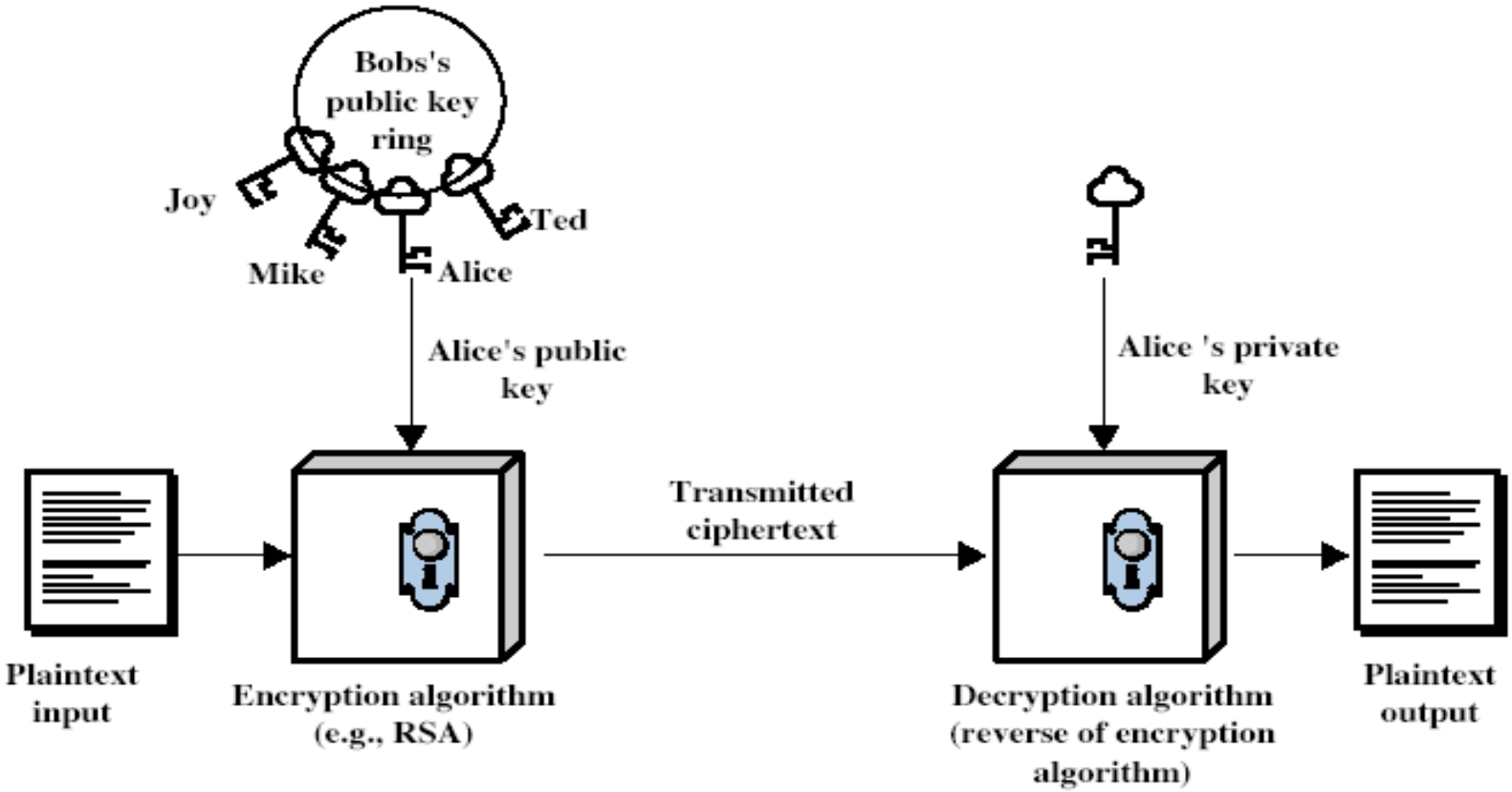
- Keys are related to each other but it is not feasible to find out private key from the public one
- It is computationally easy to en/decrypt messages when the relevant keys are known

$Y = f_{ku}(X)$ easy, if ku and X are known

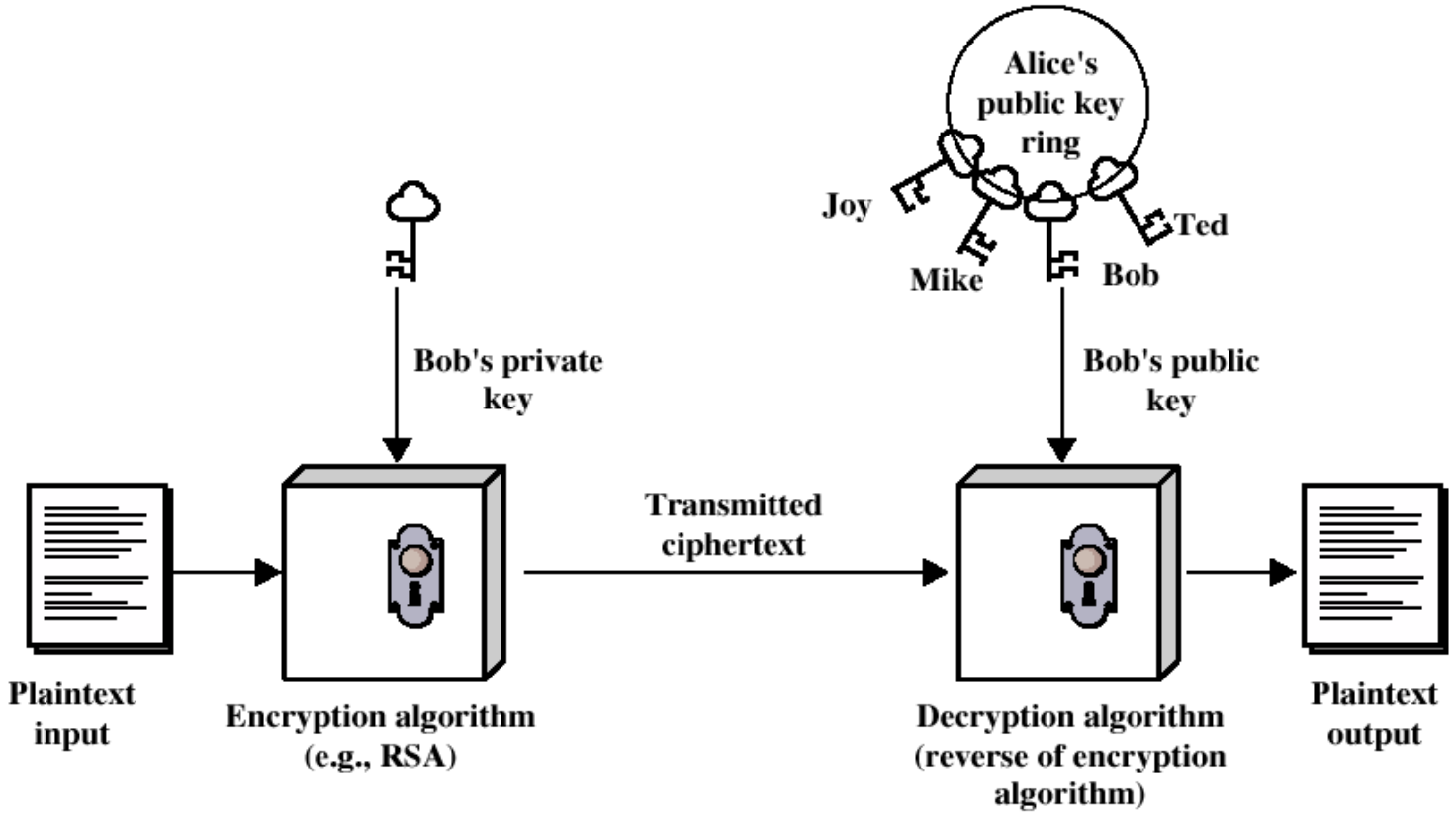
$X = f_{kr}^{-1}(Y)$ easy, if kr and Y are known, but infeasible if Y is known but kr is not known

  - ku: public-key, kr: private key

Public-Key Cryptography - Encryption



Public-Key Cryptography - Authentication



Public-Key Cryptography – General Characteristics

- based on **number theoretic hard problems**
  - rather than substitutions and permutations
- 3 misconceptions about PKC
  - it replaces symmetric crypto
    - PKC rather complements private key crypto
  - PKC is more secure
    - no evidence for that, security mostly depends on the key size in both schemes
  - key distribution is trivial in PKC since public keys are public
    - making something public is not easy. How can you make sure that a public key belongs to the intended person?
    - key distribution is easier, but not trivial

Invention of PKC

- PKC is invented by Whitfield Diffie and Martin Hellman in 1976
  - PhD student – advisor pair at Stanford Univ.
- Some gives credit to Ralph Merkle too
- NSA says that they knew PKC back in 60's
- First documented introduction of PKC is by James Ellis of UK's CESG (Communications-Electronics Security Group) in 1970
  - was a classified report
  - declassified in 1987

Why Public-Key Cryptography?

- Initially developed to address two key issues:
  - key distribution
    - symmetric crypto requires a trusted Key Distribution Center (KDC)
    - in PKC you do not need a KDC to distribute secret keys, but you still need trusted third parties
  - digital signatures (non-repudiation)
    - not possible with symmetric crypto

Public-Key Cryptosystems

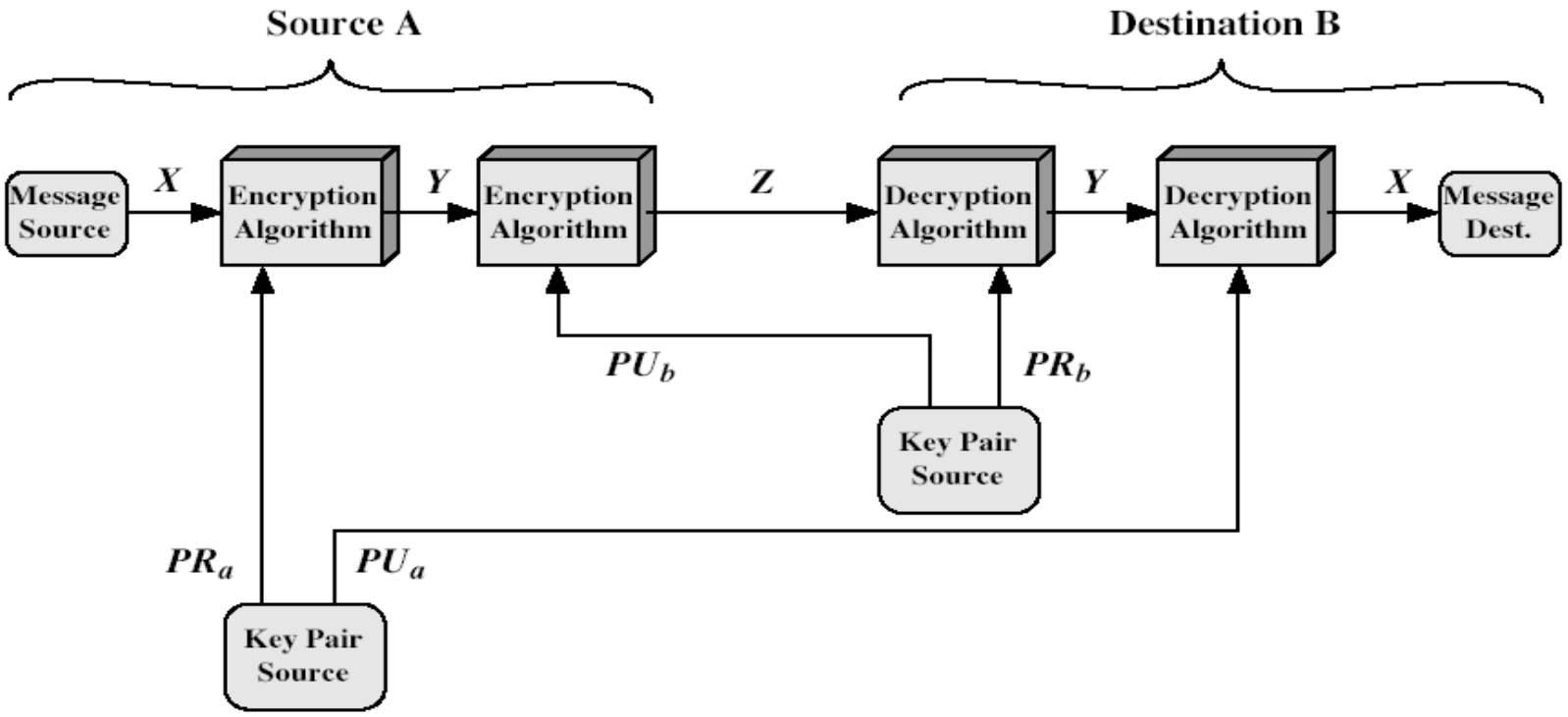


Figure 9.4 Public-Key Cryptosystem: Authentication and Secrecy

PU_a	A's Public Key	PU_b	B's Public Key
PR_a	A's Private Key	PR_b	B's Private Key

Applications of Public-Key Cryptosystems

- 3 categories
  - encryption/decryption
    - to provide secrecy
  - digital signatures
    - to provide authentication and non-repudiation
  - key exchange
    - to agree on a session key
- some algorithms are suitable for all uses, others are specific to one

Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No

Some Issues of Public Key Schemes

- like private key schemes brute force attack is always theoretically possible
  - use large keys
  - consider the security / performance tradeoff
- due to public key / private key relationships number of bits in the key should be much larger than symmetric crypto keys
  - to make the hard problem really hard
  - 80-bit symmetric key and 1024-bit RSA key has comparable resistance to cryptanalysis
- a consequence of use of large keys is having slower encryption and decryption as compared to private key schemes
  - thus, PKC is not a proper method for bulk encryption

RSA

- by Rivest, Shamir & Adleman of MIT in 1977
  - published in 1978
- best known and widely used public-key scheme
- was patented and patent was used by RSA Inc
  - however patent expired in 2000
- uses large integers
  - 1024+ bits
- security depends on the cost of factoring large numbers

RSA Key Setup

Key Generation

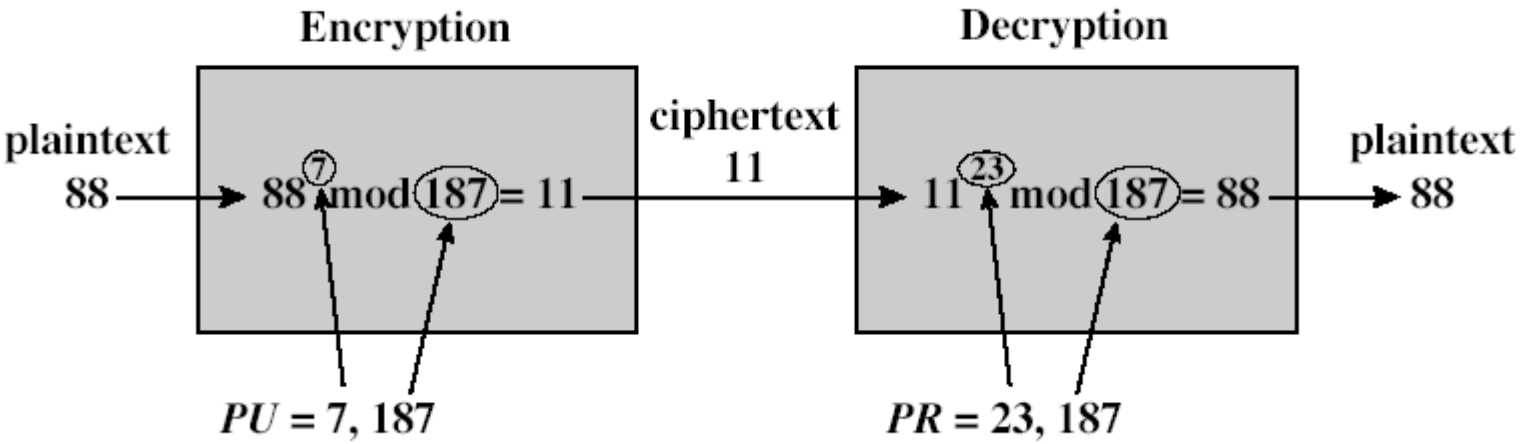
Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n) = (p - 1)(q - 1)$	
Select integer e	$\gcd(\phi(n), e) = 1;\ 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

e is usually a small number

RSA Use

- to encrypt a message M < n, the sender:
  - obtains public key of recipient $PU=\{e,n\}$
  - computes: $C=M^e \bmod n$, where $0 \leq M < n$
- to decrypt the ciphertext C the owner:
  - uses their private key $PR=\{d,n\}$
  - computes: $M=C^d \bmod n$
- note that the message M must be smaller than the modulus n
  - use several blocks if needed
- RSA works due to Euler's theorem given in Section 8 and explained in Section 9.2

RSA Example



p = 17, q = 11, n = p*q= 187

Φ(n) = 16*10 =160, pick e=7, d.e=1 mod Φ(n) ➔ d = 23

Computational Aspects

- An RSA implementation requires complex arithmetic
  - modular exponentiation for encryption and encryption
  - primality tests
  - finding inverse of e mod $\Phi(n)$
- There are acceptably fast solutions to those computational problems (see Stallings for details)

RSA Security

- 4 approaches of attacking on RSA
  - brute force key search
    - not feasible for large keys
    - actually nobody attacks on RSA in that way
  - mathematical attacks
    - based on difficulty of factorization for large numbers as we shall see in the next slide
  - side-channel attacks
    - based on running time and other implementation aspects of decryption
  - chosen-ciphertext attack
    - Some algorithmic characteristics of RSA can be exploited to get information for cryptanalysis

Factorization Problem

- 3 forms of mathematical attacks
  - factor $n=p.q$, hence find $\varnothing(n)$ and then d
  - determine $\varnothing(n)$ directly and find d
    - is equivalent of factoring n
  - find d directly
    - as difficult as factoring n
- so RSA cryptanalysis is focused on factorization of large n

Factorization Problem

Number of Decimal Digits	Approximate Number of Bits	Date Achieved	MIPS-years	Algorithm
100	332	April 1991	7	quadratic sieve
110	365	April 1992	75	quadratic sieve
120	398	June 1993	830	quadratic sieve
129	428	April 1994	5000	quadratic sieve
130	431	April 1996	1000	generalized number field sieve
140	465	February 1999	2000	generalized number field sieve
155	512	August 1999	8000	generalized number field sieve
160	530	April 2003	—	Lattice sieve
174	576	December 2003	—	Lattice sieve
200	663	May 2005	—	Lattice sieve

- RSA-129 was a challenge by RSA inventors
  - 1977, reward is $100
  - they estimated 40 quadrillion ($40*10^{15}$) years
  - solved in 1993/4 in 8 months (Atkins, Graff, Lenstra and Leyland + 600 volunteers worldwide)
  - A group of computers (1600) over the Internet used their spare time

Reasons of improvement in Factorization

- increase in computational power
- biggest improvement comes from improved algorithm
  - "Quadratic Sieve" to "Generalized Number Field Sieve"
  - Then to "Lattice Sieve"

(Latest-3) RSA challenge factored

- RSA-576 (174 decimal digits)
- Mostly German team
  - December 2003
- First of the RSA challenge numbers to be factored from the "new" challenge started in 2001
- ~13200 MIPS-years

(Latest-2) RSA challenge factored

- RSA-200
  - May 2005
  - One of the old challenges
  - Bit equivalent is 663
    - Was the largest RSA challenge number factored until December 2009
  - The team is F. Bahr, M. Boehm, J. Franke, and T. Kleinjung

http://www.rsa.com/rsalabs/node.asp?id=2879

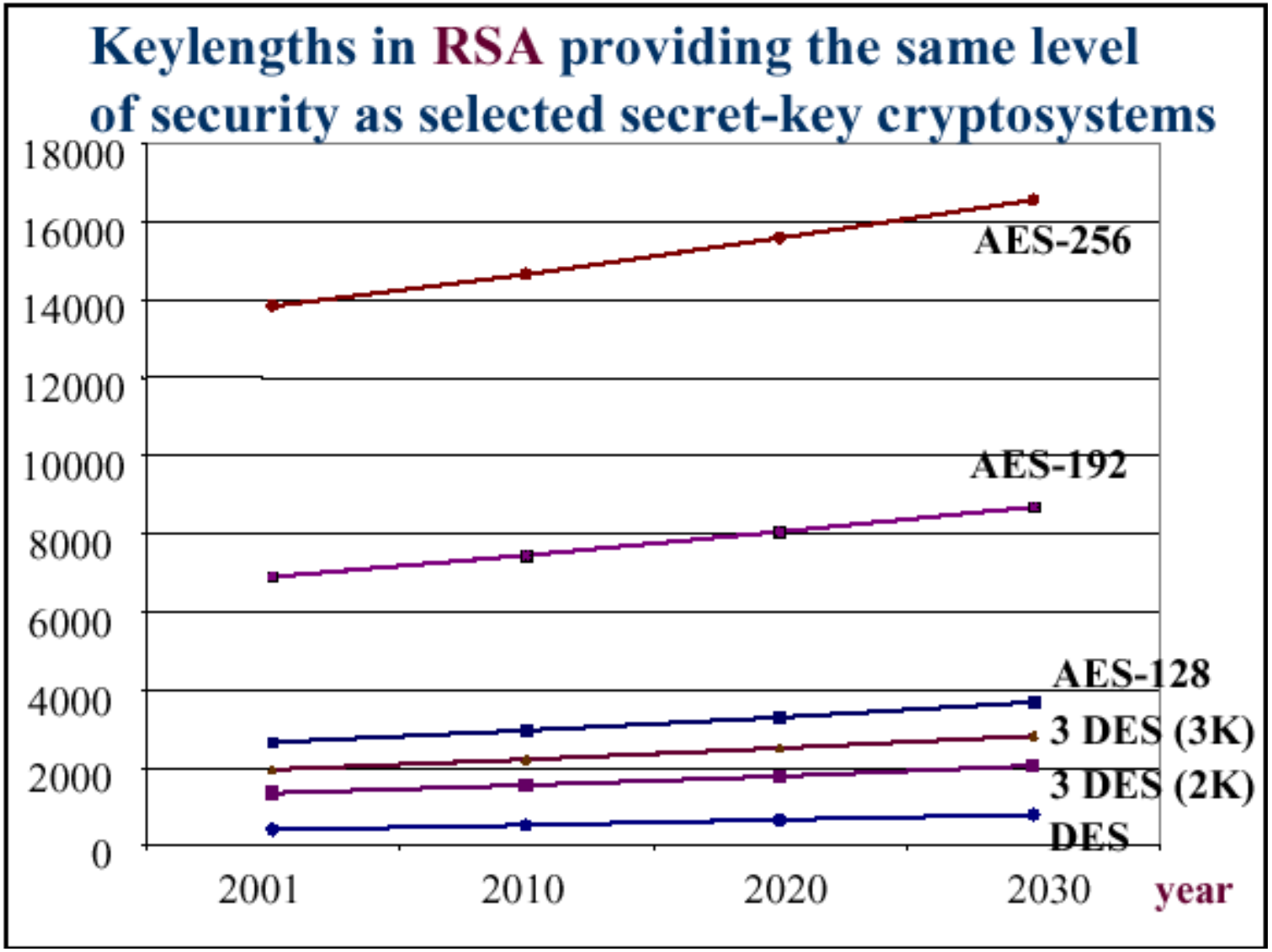
(Latest-1) RSA challenge factored

- RSA 640
  - November 2005
  - 2nd challenge of the new set
    - Prize USD 20K
  - Same team as RSA-200
  - Smaller number than RSA 200
  - Reported computation effort is half of the RSA-200

  http://www.rsa.com/rsalabs/node.asp?id=2964

Latest RSA challenge factored

- RSA 768
  - December 2009
  - 4th challenge of the new set
    - No prize since RSA discontinued RSA challenge (prize was $ 50,000)
    - 3rd challenge (RSA 704) is currently skipped
  - A multinational and multi-institutional team led by Thorsten Kleinjung
  - Largest RSA challenge factored so far
  - Reported computational effort is 2000 2.2GHz-Opteron-CPU years (~66 times more than RSA-640) http://www.rsa.com/rsalabs/node.asp?id=3723
- Next RSA challenge is 896-bit (prize $ 75,000)
  - RSA Labs discontinued RSA challenge in 2007, so if you factorize these numbers, you'll get no money!

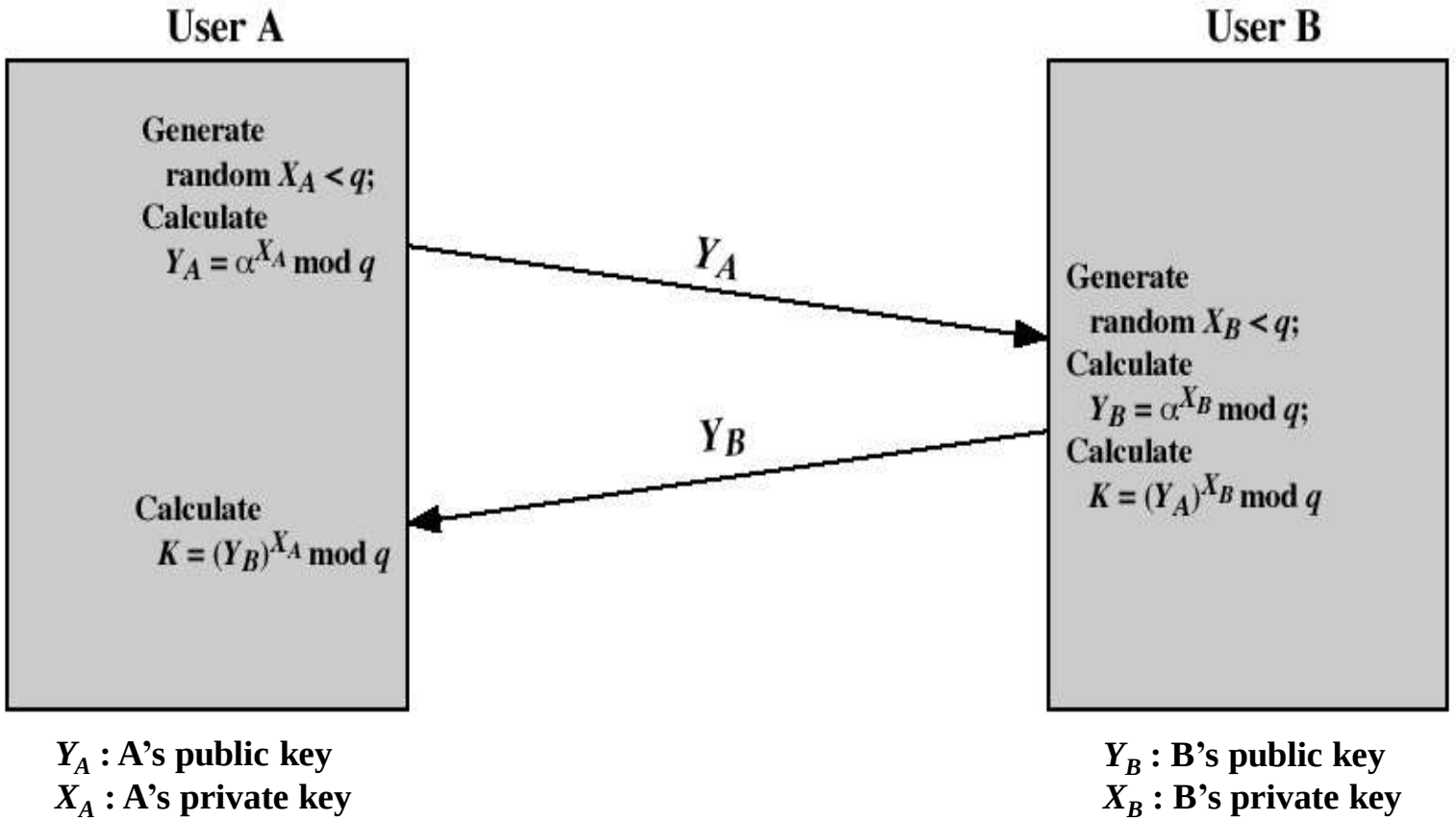


Thanks to Kris Gaj for this figure

Diffie-Hellman Key Exchange

- First PKC offered by Diffie and Hellman in 1976
- still in commercial use
- purpose is secure key-exchange
  - actually key “agreement”
  - both parties agree on a session key without releasing this key to a third party
    - to be used for further communication using symmetric crypto
- Security is in the hardness of the discrete logarithm problem
  - given $a^b \bmod n$, a and n, it is computationally infeasible to find out b if n is large enough prime number

D-H Key Exchange



q and α are known by both A and B beforehand. q is a prime number, α < q and α is a primitive root of q

D-H Key Exchange – PK Management

- Two issues
  - should we use global parameters (α and q) fixed for all public keys or unique?
  - do we need to make sure that a particular public key Y_i produced by *i*?
- In practice global parameters (α and q) are tied to Y values (public keys). However,
  1. both parties should use the same α and q, and
  2. there is no harm to use fixed α and q for all.
- If the D-H public values are anonymous, then a man-in-the-middle attack is possible

D-H Key Exchange – PK Management

- One PK management method
  - a closed group share common global parameters (α and q)
  - all users pick random secret values (X) and calculate corresponding public values (Y)
  - Y's are published at a trusted database
  - when B wants to create a key for A
    - B gets A's public value Y_A, and calculates the session key
    - A does the same when B sends an encrypted message to it
  - However this method is not practical for distributed applications

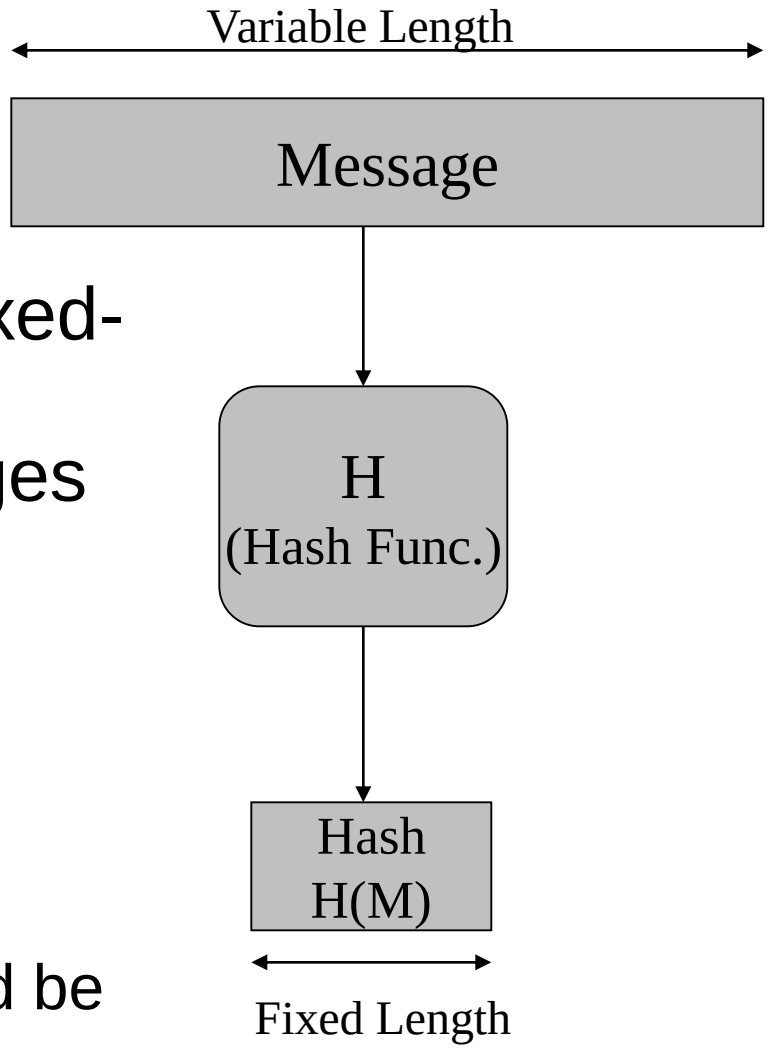
D-H Key Exchange – PK Management

- Anonymous public values are problematic
  - causes man-in-the-middle attacks
  - Attacker replaces the Y values with Y' values for which it knows the corresponding X' values
    - at the end A and B generates different sessions keys that are also known by the attacker
    - both A and B presume that other party has the same key, but this is not the case
  - Solution: public values and parameters should be either known or should be endorsed by a trusted entity
    - previous example of trusted database is one solution
    - public key certificates are the most common solution

PKC - Remained

- Implementation of RSA signatures
- DSA / DSS
  - Digital Signature Algorithm / Standard
- Elliptic Curve Cryptography (ECC)
  - ECDSA – Elliptic Curve DSA
  - ECDH – Elliptic Curve D-H
- First we will see hash functions
  - several application areas

Hash Functions



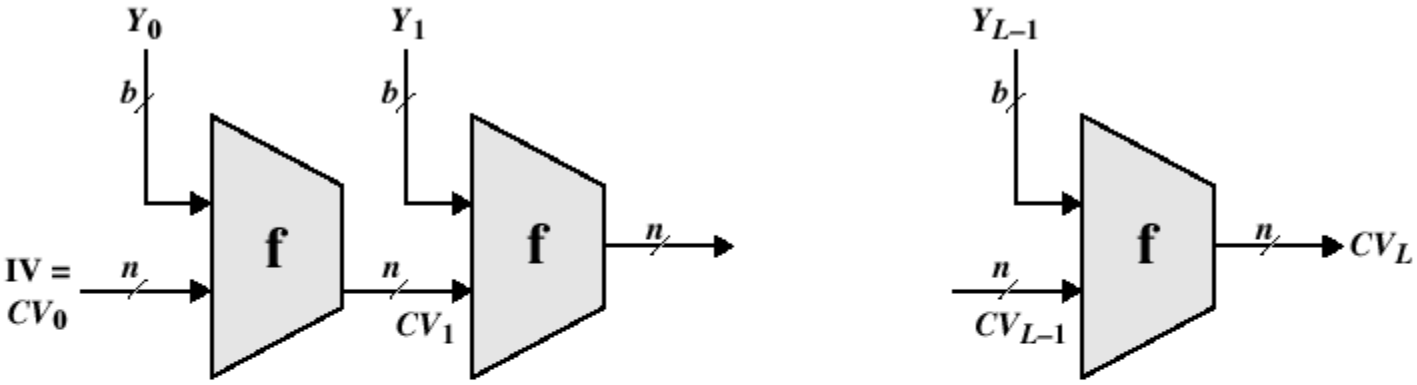
- are used to generate fixed-length fingerprints of arbitrarily large messages
- denoted as H(M)
  - M is a variable length message
  - H is the hash function
  - H(M) is of fixed length
  - H(M) calculations should be easy and fast
    - indeed they are even faster than symmetric ciphers

Hash functions – Requirements and Security

- Hash function should be a **one-way** function
  - given h, it is computationally infeasible to find x such that h = H(x)
  - complexity of finding x out of h is 2^n, where n is the number of bits in the hash output
- **Weak collision resistance**
  - given x, it is computationally infeasible to find y with H(x) = H(y)
  - complexity of attack is 2^n
- **Strong collision resistance**
  - It is computationally infeasible to find any pair x, y such that H(x) = H(y)
  - complexity is $2^{n/2}$

Hash function – General idea

- Iterated hash function idea by Ralph Merkle
  - a sequence of compressions
  - if the compression function is collision-free, so is the hash function
  - MD5, SHA-1 and others are based on that idea



IV = Initial value
CV = chaining variable
Y_i = ith input block
f = compression algorithm
L = number of input blocks
n = length of hash code
b = length of input block

Important Hash Functions

- MD5
  - Message Digest 5
  - another Ron Rivest contribution
  - arbitrarily long input message
    - block size is 512 bits
  - 128-bit hash value
- has been used extensively, but its importance is diminishing
  - brute force attacks
    - 2^{64} is not considered secure complexity any more
  - cryptanalytic attacks are reported

Important Hash Functions

- SHA-1
  - Secure Hash Algorithm – 1
  - NIST standard
    - FIPS PUB 180-1
  - input size < 2^{64} bits
  - hash value size 160 bits
    - brute force attacks are not so probable
      - 2^{80} is not-a-bad complexity
  - A Crypto 2005 paper explains an attack against strong collision with 2^69 complexity
    - have raised concerns on its use in future applications
  - Later several other attacks are reported
  - Final one is presented at rump session of Eurocrypt 2009 and reduces the attack complexity to 2^{52}
    - However, this attack is not yet confirmed

Important Hash Functions

- However, NIST had already (in 2002) published FIPS 180-2 to standardize (SHA-2 family)
  - SHA-256, SHA-384 and SHA-512
  - for compatible security with AES
  - structure & detail is similar to SHA-1
  - but security levels are rather higher

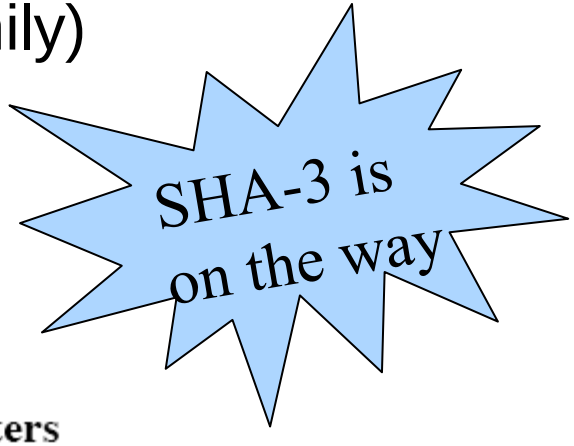


Table 12.1 Comparison of SHA Parameters

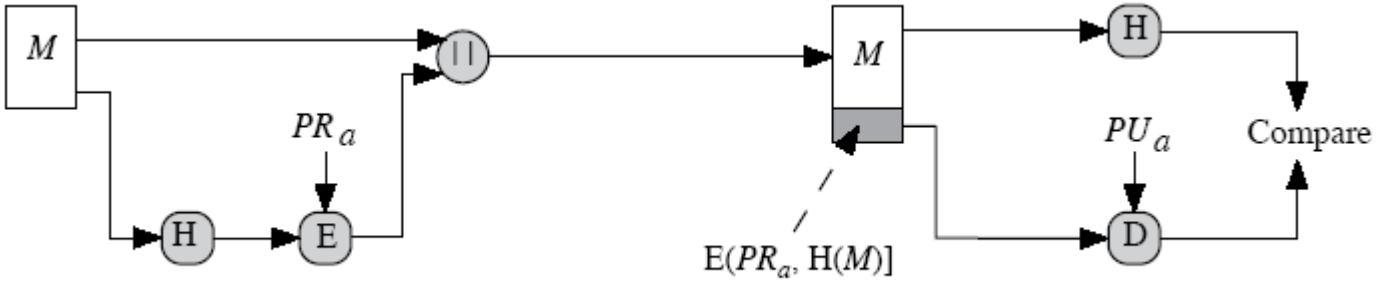
	SHA-1	SHA-256	SHA-384	SHA-512
Message digest size	160	256	384	512
Message size	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block size	512	512	1024	1024
Word size	32	32	64	64
Number of steps	80	64	80	80
Security	80	128	192	256

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 workfactor of approximately $2^{n/2}$.

Digital Signatures

- Mechanism for non-repudiation
- Basic idea
  - use private key on the message to generate a piece of information that can be generated only by yourself
    - because you are the only person who knows your private key
  - public key can be used to verify the signature
    - so everybody can verify
- Generally signatures are created and verified over the hash of the message
  - Why?

Digital Signature – RSA approach



M: message to be signed H: Hash function

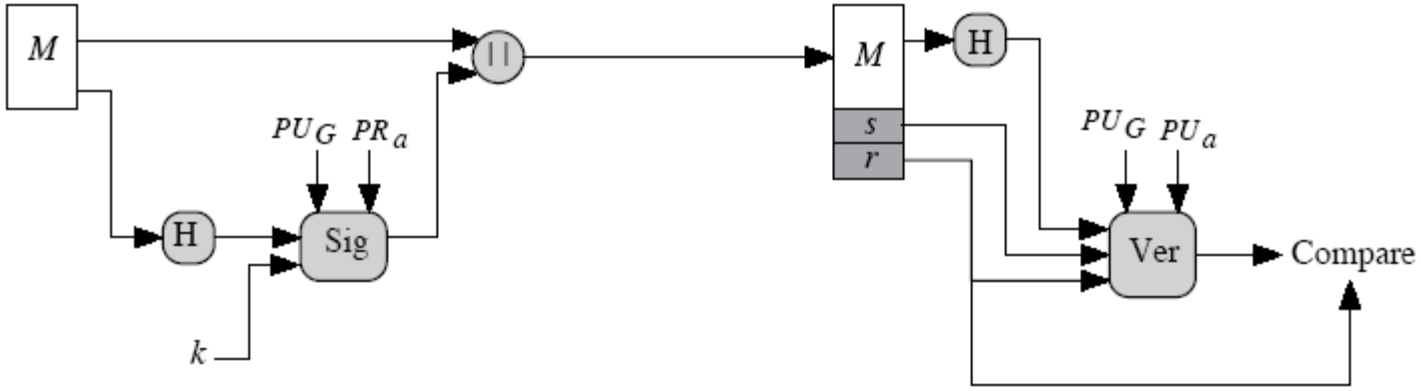
E: RSA Private Key Operation PR_a: Sender's Private Key

D: RSA Public Key Operation PU_a: Sender's Public Key

E $[PR_a, H(M)]$ Signature of A over M

Digital Signature – DSA approach

- DSA: Digital Signature Algorithm
  - NIST standard – FIPS 186
  - Key limit 512 – 1024 bits, only for signature, no encryption
  - based on discrete logarithm problem
  - Message hash is not restored for verification (difference from RSA)



M: message to be signed
Sig: DSA Signing Operation
Ver: DSA Verification Operation
s, r Sender's signature over M

H: Hash function
PR_a: Sender's Private Key
PU_a: Sender's Public Key
PU_G: Global Public Key components

Collision resistant hash functions and digital signatures

- Have you seen the reason why hash functions should be collision resistant?
  - because otherwise messages would be changed without changing the hash value used in signature and verification

Collision resistant hash functions and digital signatures

- Birthday attack
  - generate two messages
    - one with legitimate meaning
    - one fraudulent
  - create a set of messages from each of them that carries the same meaning
    - play with blanks, synonyms, punctuations
  - calculate the hashes of those two sets
  - you should have $2^{n/2}$ messages (and hashes) in each set for 0.63 probability of a match, where n is the hash size
  - if a match is found, then the fraudulent hash could be replaced with the legitimate one without affecting the signature

Elliptic Curve Cryptography

- Based on the difficulty of Elliptic Curve Discrete Logarithm problem
  - details are not in the scope of this course
  - a concise description is in Sections 10.3 and 10.4 of Stallings
- Actually a set of cryptosystems
  - each elliptic curve is one cryptosystem
    - 160-bit, 163-bit, 233-bit, … defined in IEEE P1363 standard
- Key size is smaller than RSA
  - 160-bit ECC is almost has the security as 1024 bit RSA
- Private Key operation is faster than RSA, public key operation is almost equal

Elliptic Curve Cryptography

- Key exchange
  - ECDH
    - Elliptic Curve Diffie-Hellman
- Digital Signatures
  - ECDSA
    - Elliptic Curve Digital Signature Algorithm
- ECDH and ECDSA are standard methods
- Encryption/Decryption with ECC is possible, but not common

Message Authentication

- Making sure of
  - message has been sent by the alleged sender
  - message has been received intact
    - no modification
    - no insertion
    - no deletion
  - i.e., Message Authentication also covers integrity
- Digital Signatures
  - provides integrity + authentication + nonrepudiation
- We will see mechanisms that provide authentication, but not non-repudiation

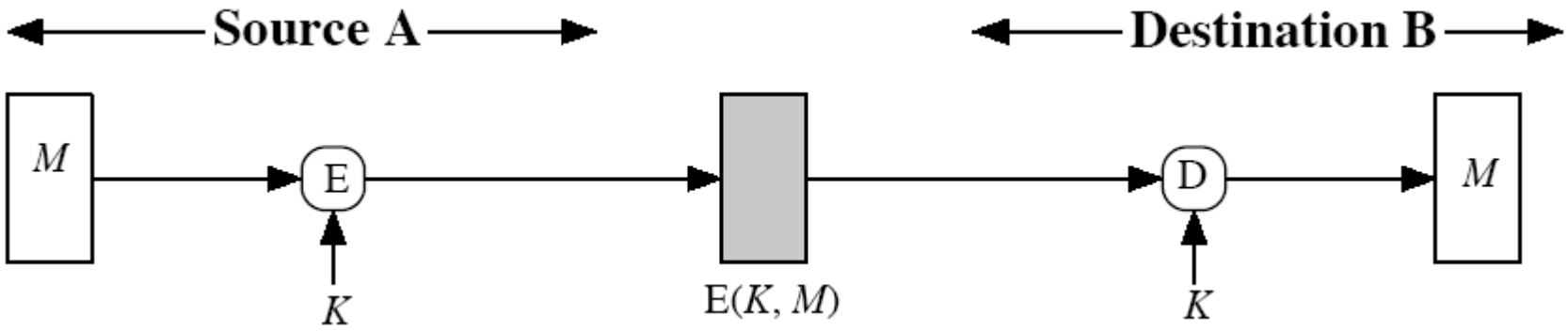
Mechanisms for Message Authentication

- General idea
  - receiver makes sure that the sender knows a secret shared between them
  - in other words, sender demonstrates knowledge of that shared secret
  - without revealing the shared secret to unauthorized parties of course
- We will see some mechanisms for this purpose

Mechanisms for Message Authentication

- Message Encryption
  - provides message authentication, but …
- Message Authentication Code Functions
  - similar to encryption functions, but not necessarily reversible
  - There is a standard method based on DES but not widely used (we will skip the details)
  - Generally Hash based MAC is used (will see)
- Actually hash functions are used for message authentication in several ways (will see)

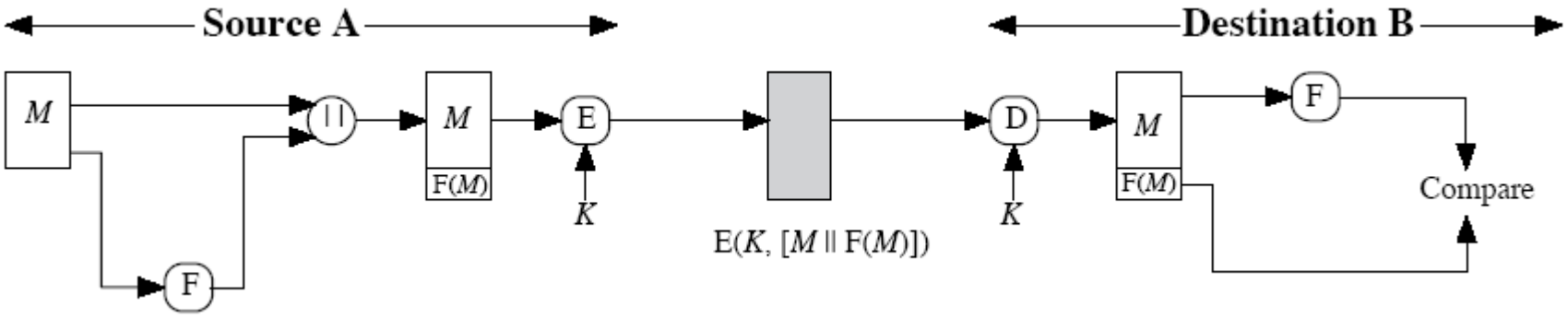
Using Message Encryption for Authentication



- Provides encryption. What about authentication?
  - yes, but there must be a mechanism to detect the restored M is the same as the sent M
    - intelligible restored plaintext (may be difficult)
    - error control codes (checksum), see next slide

Using Message Encryption for Authentication

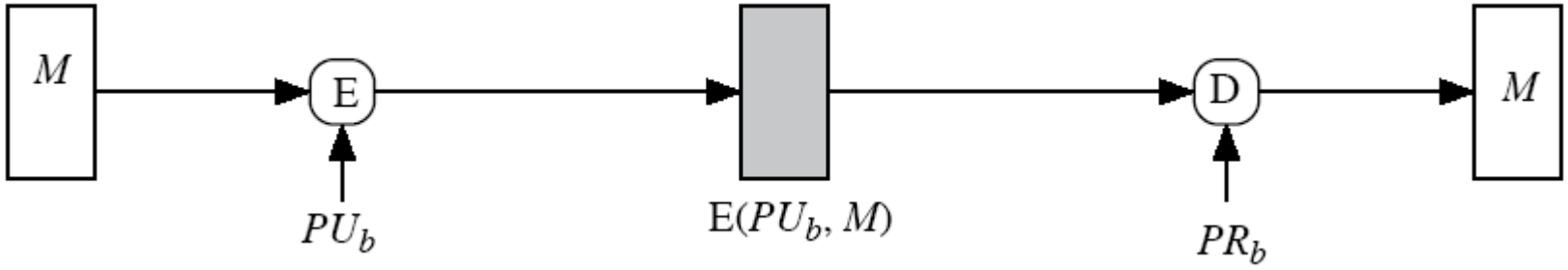
- Addition of FCS (frame check sequence) helps to detect if both M's are the same or not



F: FCS function

Using Message Encryption for Authentication

- What about public-key encryption?



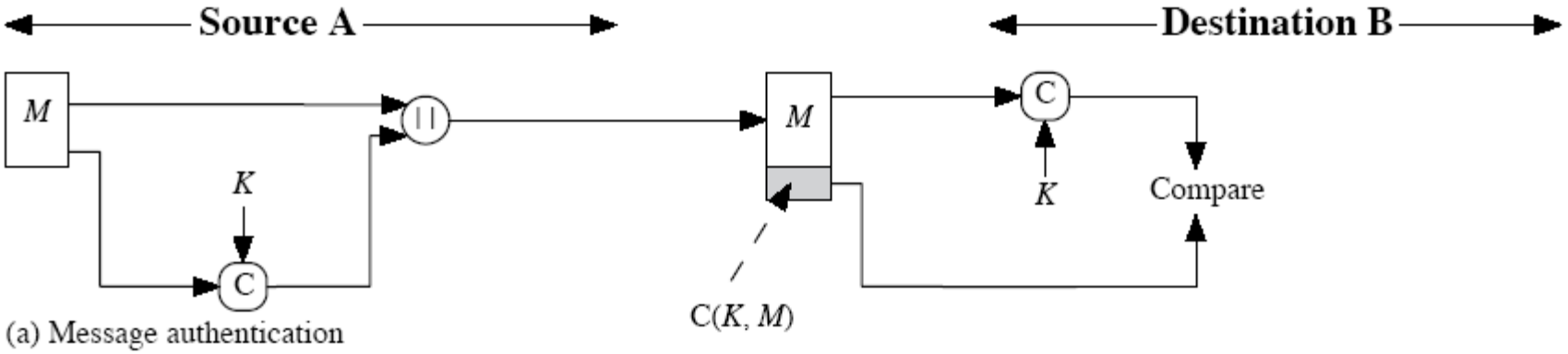
- Provides confidentiality, but not authentication
  - Why?
  - What should be done for authentication using public-key crypto?
  - we have seen the answer before.

Message Authentication Code (MAC) and MAC Functions

- An alternative technique that uses a secret key to generate a small fixed-size block of data
  - based on the message
  - not necessarily reversible
  - secret key is shared between sender and receiver
  - called *cryptographic checksum* or *MAC* (*message authentication code*)
- appended to message
- receiver performs same computation on message and checks if matches the received MAC
- provides assurance that message is unaltered and comes from sender

MAC

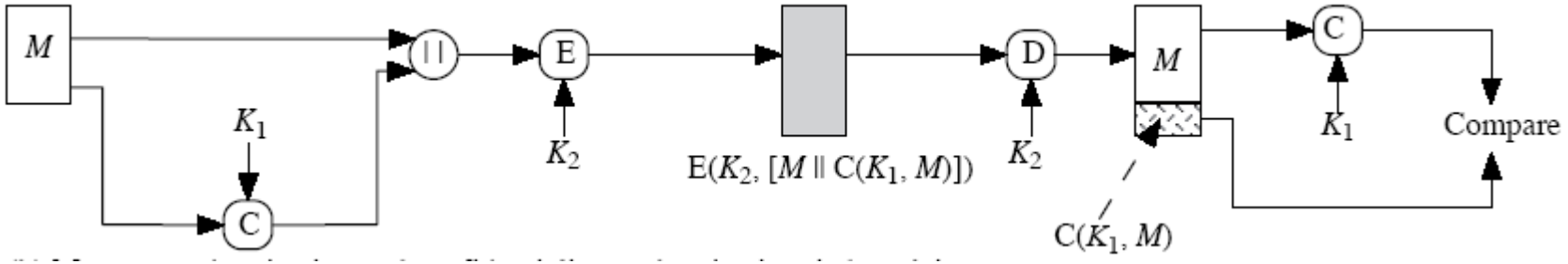
- Only authentication



(a) Message authentication

C: MAC function

- Authentication and confidentiality



MAC – The Basic Question

- Is MAC a signature?
  - No, because the receiver can also generate it

A MAC function based on DES

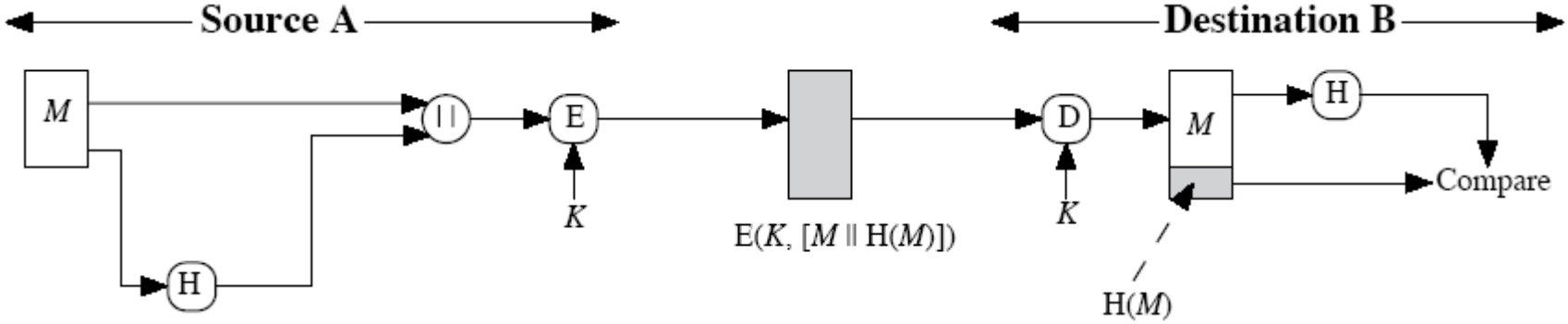
- DAA (Data Authentication Algorithm)
  - FIPS PUB 113 (NIST Standard), ANSI X9.17
  - based on DES-CBC
  - key (56 bits) and MAC (64 bits) sizes are too small to be considered secure
  - Not used in practice

Hash based Message Authentication

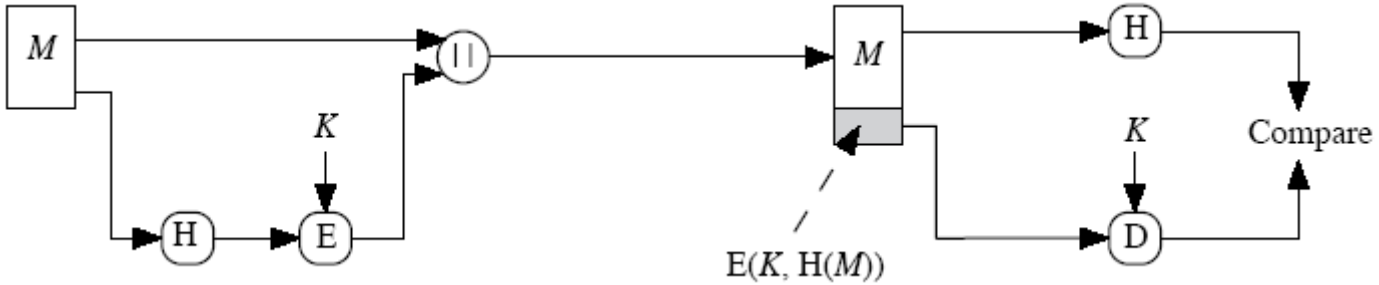
- Hash Functions
  - condenses arbitrary messages into fixed size
- We can use hash functions in authentication and digital signatures
  - with or without confidentiality

Hash based message authentication using symmetric encryption

- with confidentiality

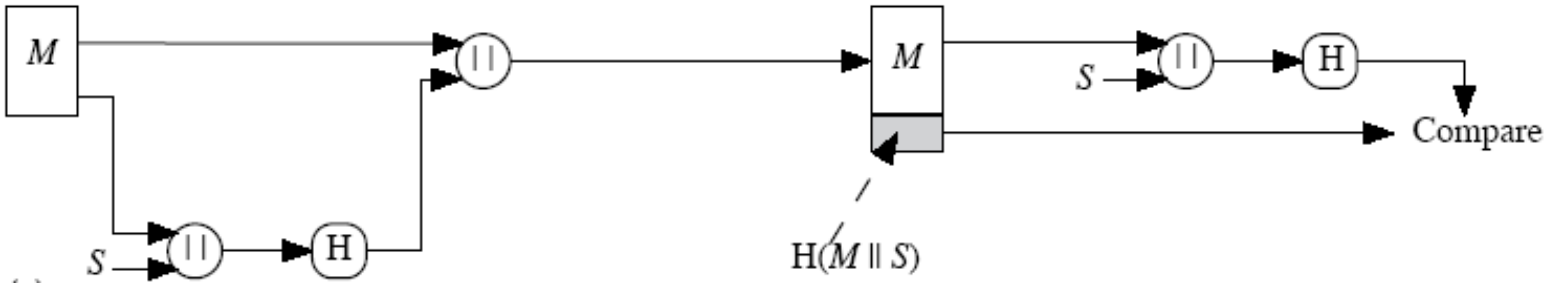


- without confidentiality



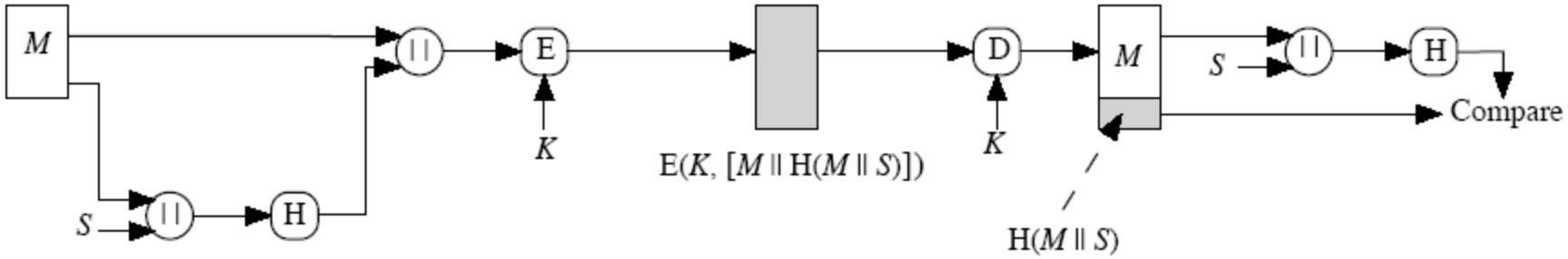
Other Hash based message authentication techniques

- Authentication is based on a shared-secret *s*, but no encryption function is employed



Other Hash based message authentication techniques

- Previous method + confidentiality
  - encryption is needed for confidentiality only



Keyed Hash Functions

- it is better to have a MAC using a hash function rather than a block cipher
  - because hash functions are generally faster
  - not limited by export controls unlike block ciphers
- hash functions are not designed to work with a key
- hash includes a key along with the message
- original proposal:

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

  - by Gene Tsudik (1992)
- eventually led to development of HMAC
  - by Bellare, Kanetti and Krawczyk

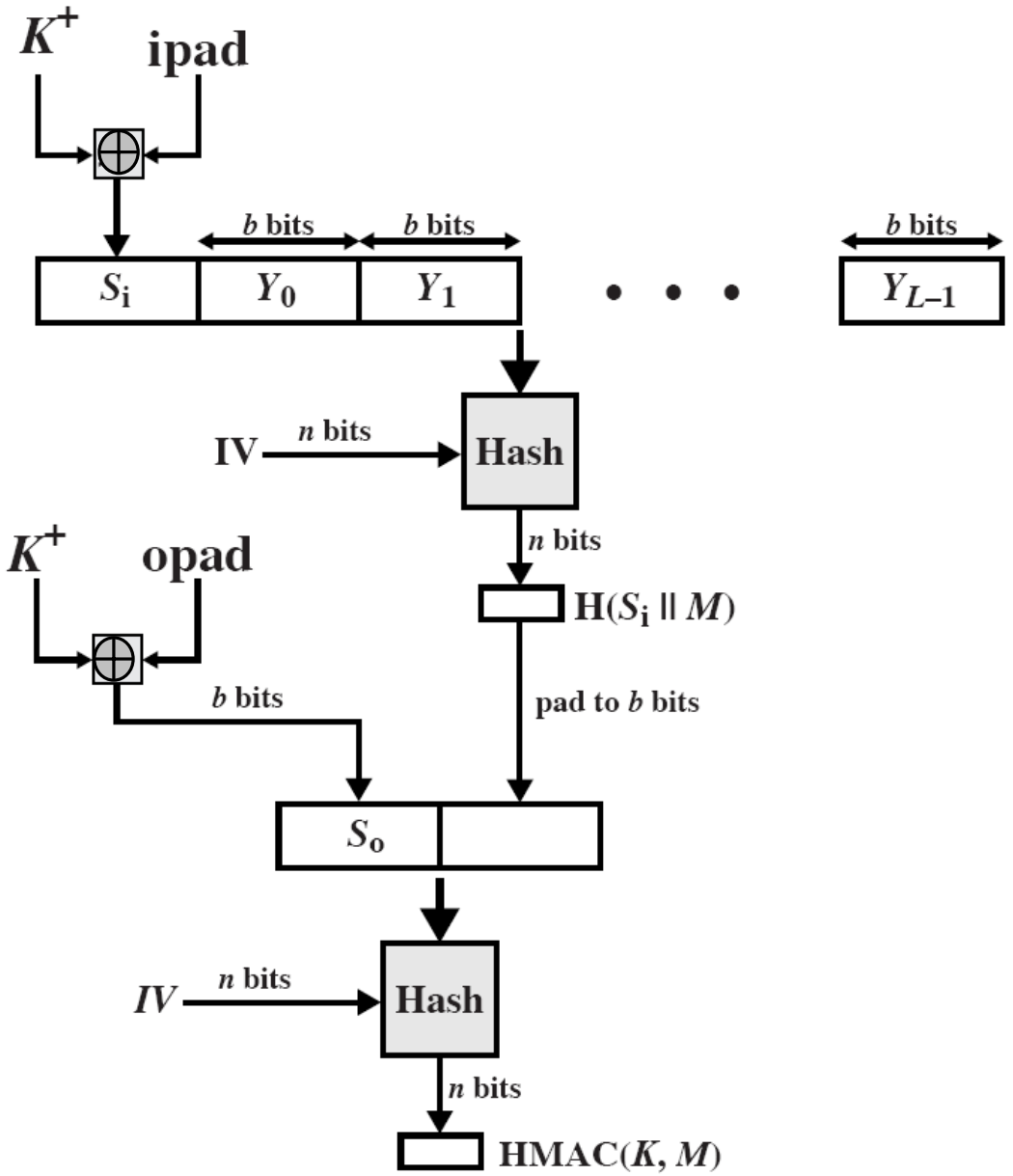
HMAC

- specified as Internet standard RFC2104
  - used in several products and standards including IPSec and SSL
- uses hash function on the message:

$$HMAC_K = Hash[(K^+ \text{ XOR } opad) \,||\, Hash[(K^+ \text{ XOR } ipad) \,||\, M)]]$$

- where K^+ is the key padded out to block size of the hash function
- and opad, ipad are some padding constants
- overhead is just 3 more blocks of hash calculations than the message needs alone
- any hash function (MD5, SHA-1, …) can be used

HMAC structure



HMAC Security

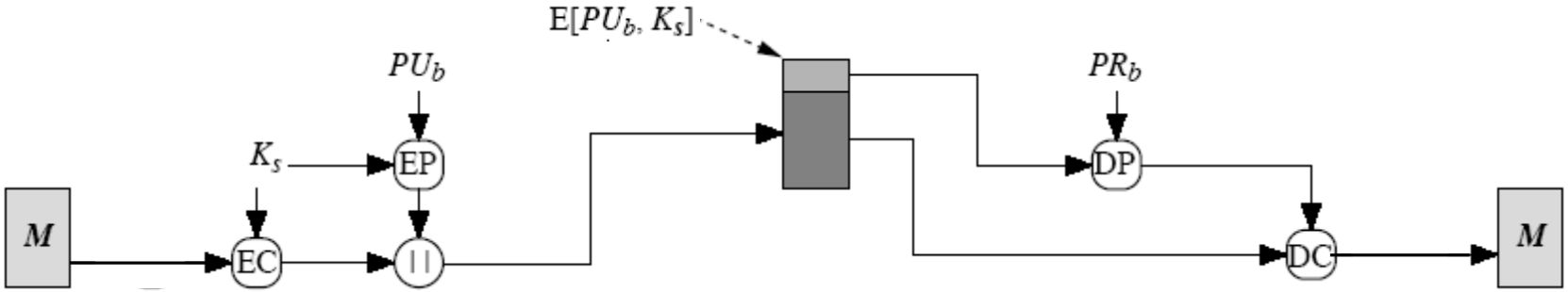
- HMAC assumes a secure hash function
  - as their creators said
    - “you cannot produce good wine using bad grapes”
- it has been proved that attacking HMAC is equivalent the following attacks on the underlying hash function
  - brute force attack on key used
  - birthday attack
    - find M and M’ such that their hashes are the same
    - since keyed, attacker would need to observe a very large ($2^{n/2}$ messages) number of messages that makes the attacks infeasible
    - Let’s see if MD5-based HMAC is secure.

Message Encryption

- Public key encryption for the bulk message is too costly
  - bulk encryption should be done using symmetric (conventional) crypto
- If a key is mutually known (e.g. if D-H is used)
  - use it to encrypt data
  - this method is useful for connection oriented data transfers where the same key is used for several data blocks
- If no key is established before
  - mostly for connectionless services (such as e-mail transfer)
  - best method is enveloping mechanism

Digital Envelopes

- A randomly chosen one-time symmetric encryption key is encrypted with public key of the recipient
- fast en/decryption without pre-establishment of keys



EC: Conventional Encryption

EP: Public-key Encryption

Ks: Session key (one-time)

DC: Conventional Decryption

DP: Public-key Decryption

What we have covered and will cover next?

- Symmetric Cryptography
- Asymmetric (Public-key) Cryptography
  - including D-H key agreement
- Hash functions
- Digital Signatures using PKC
- Message Authentication Mechanisms
  - MACs, HMAC
- After that we will continue with Key Distribution/Management and Authentication
  - they are closely related with each other

Key Distribution/Management and Authentication

- two closely related subjects
- why?

Key Distribution

- symmetric encryption schemes require both parties to share a common secret key
  - issue is how to securely distribute this key without revealing it to an adversary
- many attacks are based on poor key management and distribution
  - rather than breaking the codes
- This is, actually, the most difficult problem in developing secure systems

Key Distribution

various key distribution alternatives for parties A and B :

1. A can select key and physically deliver to B
   - does not scale for a large and distributed group
   - how many keys do we need for N users?
2. third party can select & physically deliver key to A & B
   - similar comment as 1
   - sometimes finding a “trusted” third party is another problem
3. if A & B have communicated previously, they can use previous key to encrypt a new key
   - good option but initially several keys to be distributed
4. if A & B have secure communications with a third party C, C can relay key between A & B on demand
   - only N master keys are enough

Session Key / Master Key

- The idea of having a key-encryption-key (**master key**) to generate random and temporary **session keys**
- can be implemented in several ways
  - Basic D-H is such an example
    - public/private keys are master keys, exchanged key is a session key
  - Kerberos is another example
  - SSL uses three layers
    - D-H for master key, master key for the session key

Session Key / Master Key

- Session key lifetime is a trade-off
  - if small
    - securer since attacker can obtain less ciphertext to analyze
    - But this creates more delay
  - If large
    - less secure, but less delay

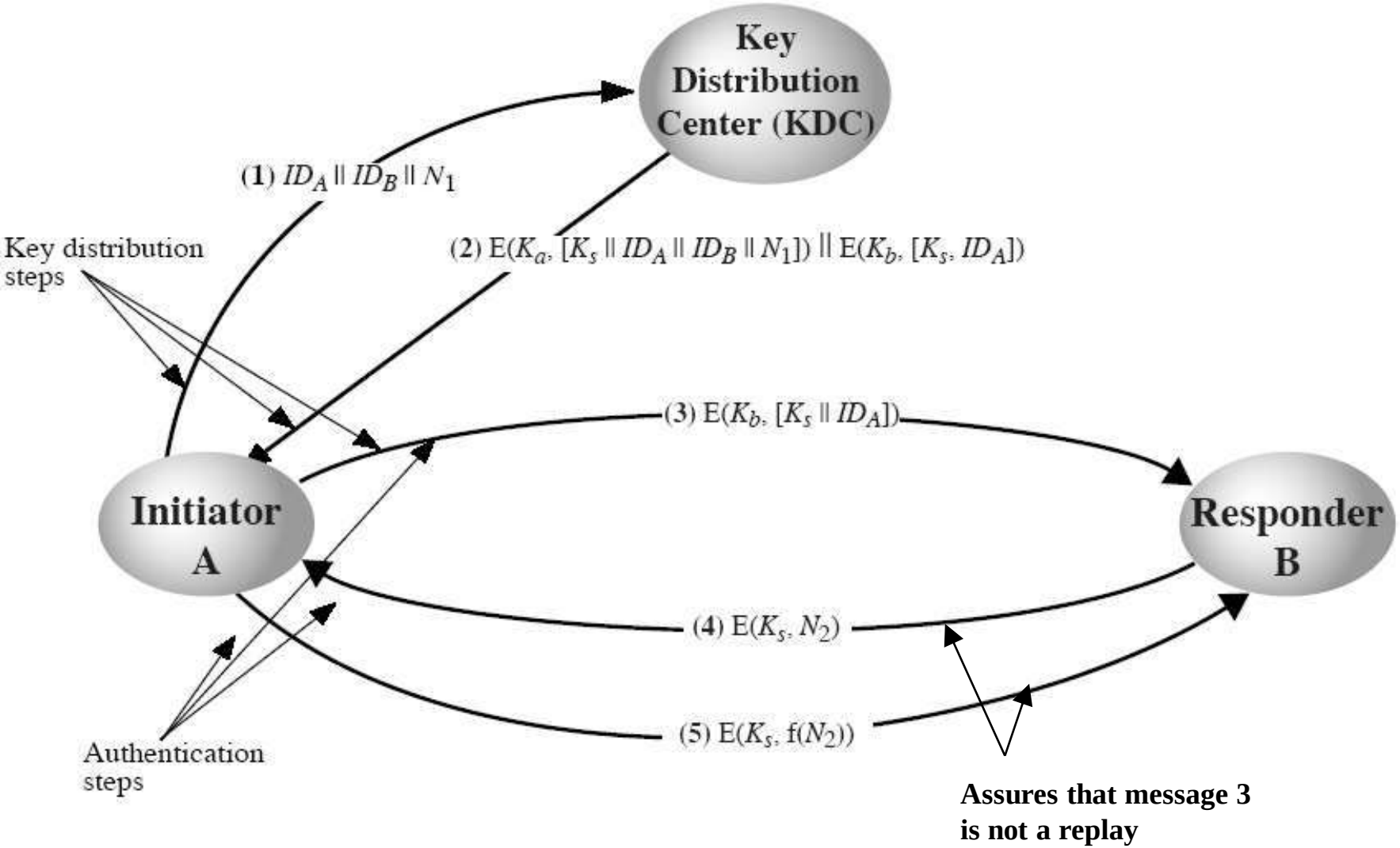
Key Distribution Facts

- Conservation of trust principle
  - a secure communication cannot be based on nothing; either there should be an initial direct contact or an indirect protocol to transfer trust
- Either physical delivery or a trusted third party
  - physical delivery is the only option to avoid a third party
    - most basic system is PIN entry
  - otherwise regardless of symmetric or asymmetric encryption, you need a trusted third party
    - even D-H does not work without a third party, why?

A Key Distribution Example

- Symmetric crypto based
- Each user shares a symmetric master key with the KDC (Key Distribution Center)
  - e.g. K_a, K_b, K_c, …
  - possibly physically distributed
- Basic idea:
  - whenever a user A wants to communicate with B, it requests a session key (K_s) from KDC
- Protocol is in the next slide

A Key Distribution Example



Key Management in PKC

- In other words
  - distribution of public keys
  - use of PKC to distribute secret keys
    - public/private key as a master key

Distribution of the Public Keys

- the most important barrier against the deployment of PKC in applications
- Basic question?
  - how can I make sure about the legitimacy of a public key?
  - how can I make sure that Bob's public key really belongs to Bob, not to Charlie?
- Why this is so important?
  - Name spoofing attacks become possible
    - remember the man-in-the-middle attacks in anonymous Diffie-Hellman

Distribution of the Public Keys

- Some methods
  - Public Announcement
  - Publicly available databases/directories
  - Centralized distribution
  - Certificates

Public Announcement

- Broadcast your public key to the public
  - via newsgroups, mailing lists, from personal website, etc.
  - major weakness is anyone can easily pretend as yourself
    - so attacks are possible

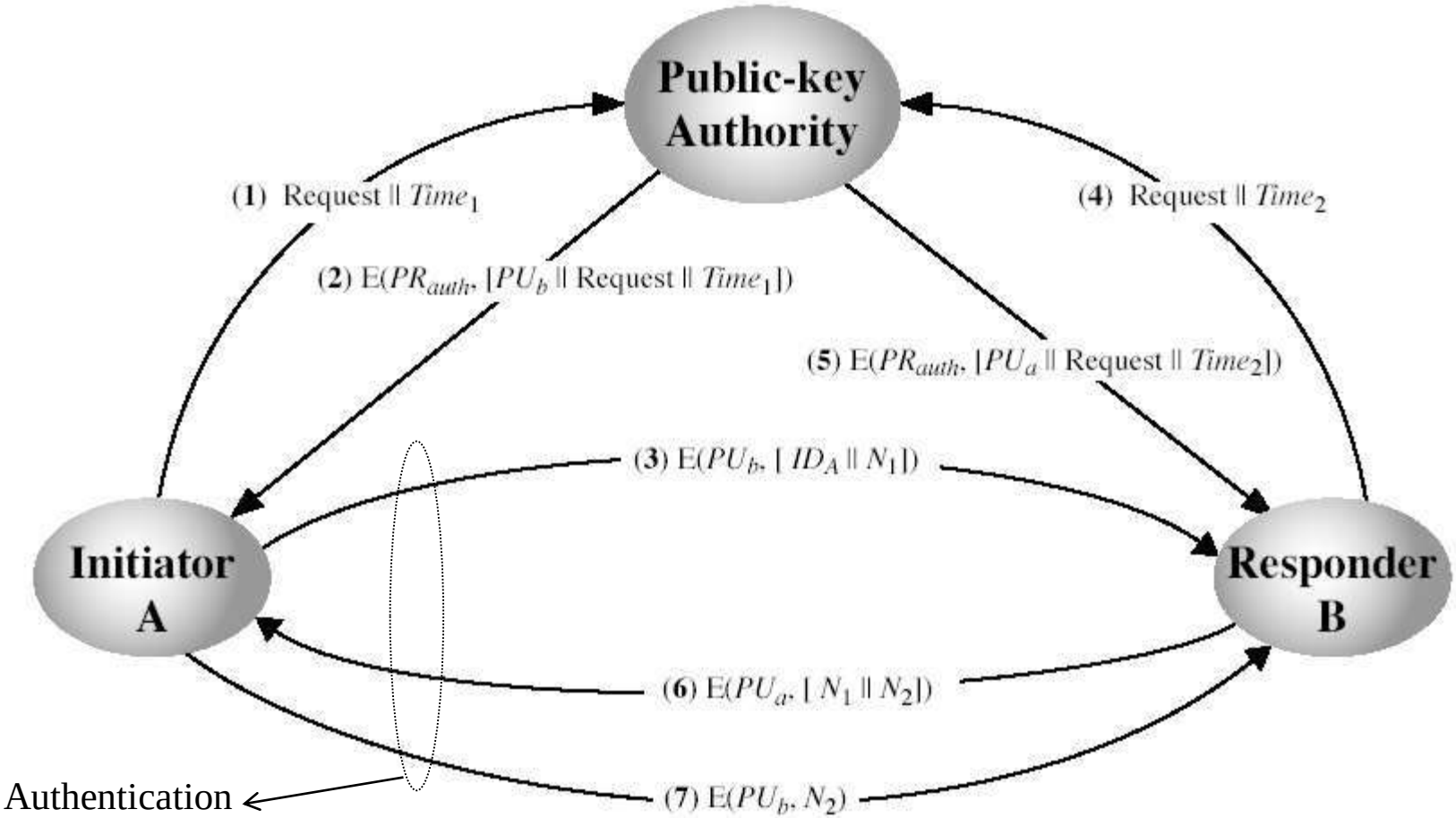
Publicly available directories/databases

- There exists a directory/database for {name, public key} pairs
- write controlled
  - a trusted administrator
- if administered thoroughly, good
  - but a proper administration is difficult
    - need secure mechanisms for registration, update, delete.

Centralized Distribution - Public-Key Authority

- Similar to directory/database approach, but access to the directory is automated via a secure protocol
  - users interact with directory to obtain any desired public key securely
  - requires real-time access to directory when keys are needed
  - users should know public key for the directory
- the directory/database contains {name, public key} pairs
  - write permit is restricted

Centralized Distribution - Public-Key Authority PROTOCOL



Centralized Distribution - Public-Key Authority

- What about caching the public keys at the end users?
  - good for performance
    - saves some messages
  - but what happens if a public key is deleted form the database/directory?
    - fresh copies needed or another protocol for the validity check

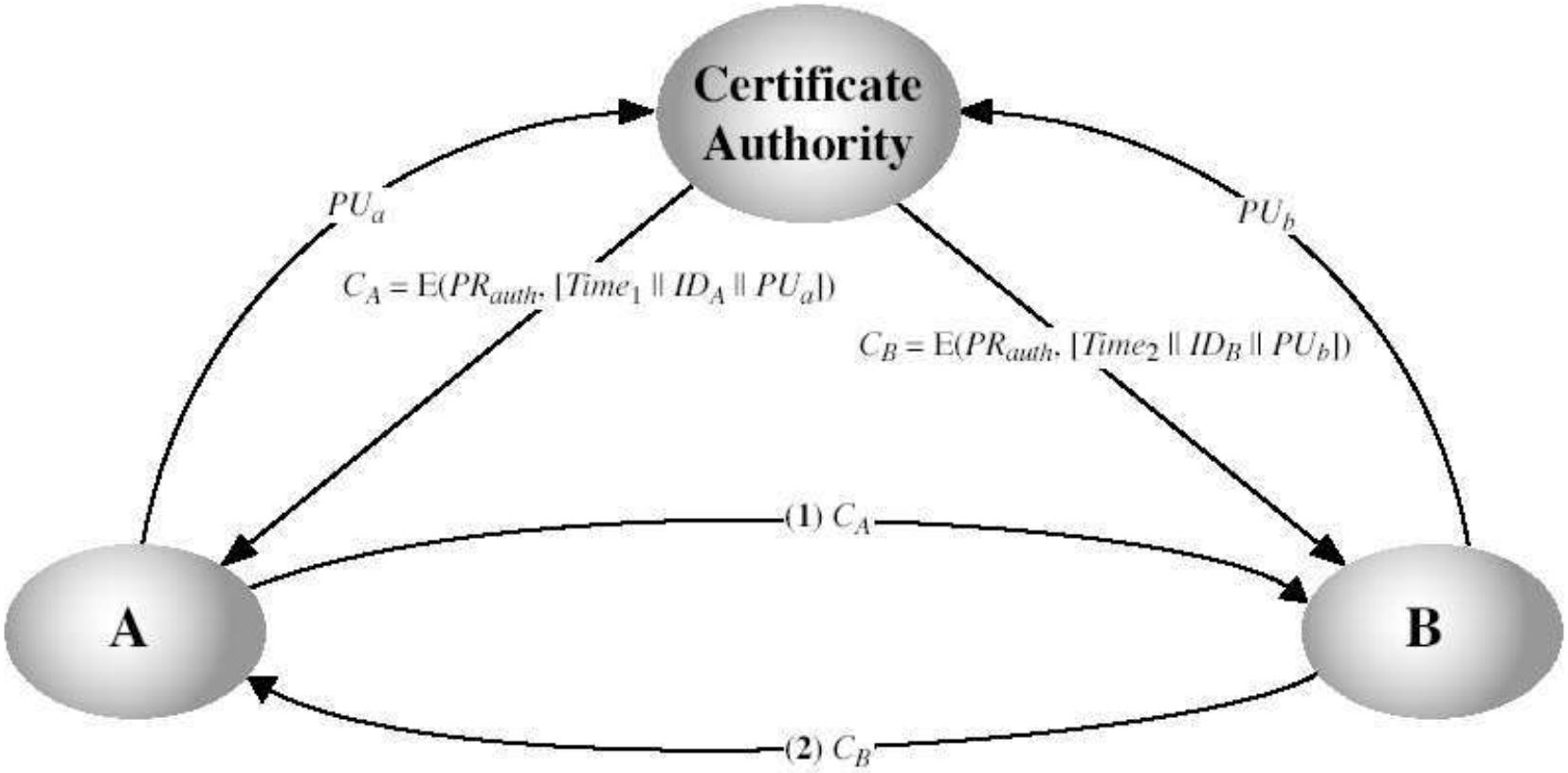
Centralized Distribution - Public-Key Authority

- Disadvantages
  - authority is an active entity and may create a performance bottleneck
  - database should be kept secure to prevent unauthorized modification
  - The problem of registration of public-keys is still unsolved!

Public-Key Certificates

- certificates allow key exchange without real-time access to public-key authority
- a certificate binds *identity* to *public key*
  - usually with other info such as period of validity, issuer's info, etc
- all contents signed by a trusted Certification Authority (CA)
- can be verified by anyone who knows the CA public-key
- CA must make sure about the identity of the certificate owner

Public-Key Certificates



Public-Key Certificates

- Certificates are widely used in practice
  - previous slides only show the idea
- need lots of polishing for practical use
  - is a single CA sufficient?
  - what happens if the CA's public key is not known?
    - how to distribute CA public keys?
  - what happens if a certificate is revoked?
  - How the users exchange their certificates in practical applications?
- We will discuss the use of certificates and state-of-the-art in this area later

What can you do with securely distributed public keys?

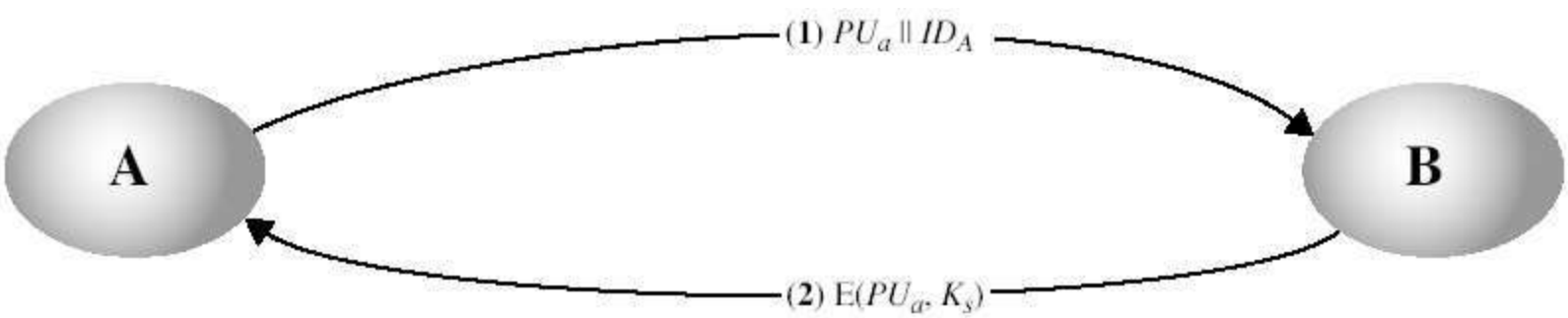
- Digital signatures
  - talked about them
- confidentiality
  - theoretically possible but slow
  - instead session keys can be distributed
    - those session keys are used for symmetric encryption

Distribution of Secret Keys using PKC

- Several methods exist
- Diffie-Hellman is one way
- we will see some alternatives

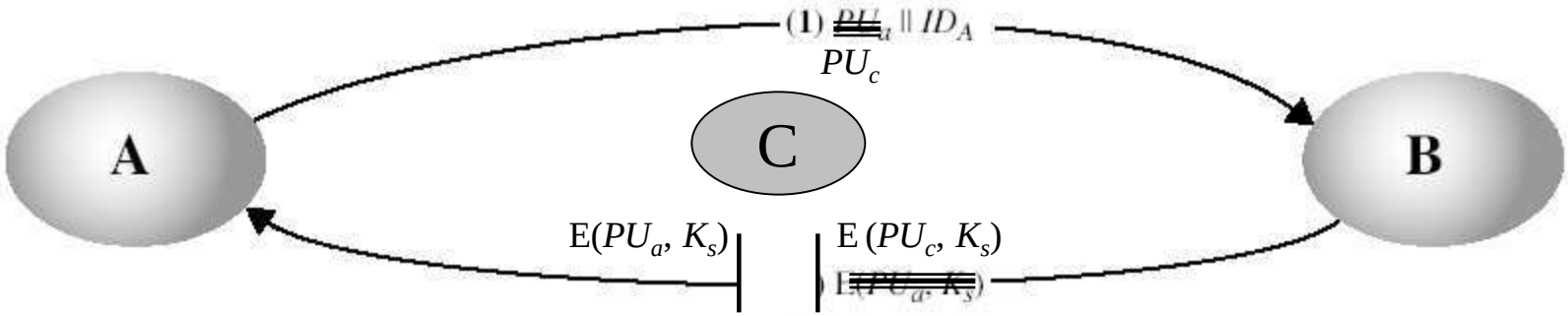
Simple Secret Key Distribution

- proposed by Merkle in 1979
  - A generates a new temporary PKC key pair
  - A sends B its public key and identity
  - B generates a session key and sends it to A encrypted using the supplied public key
  - A decrypts the session key and both use it



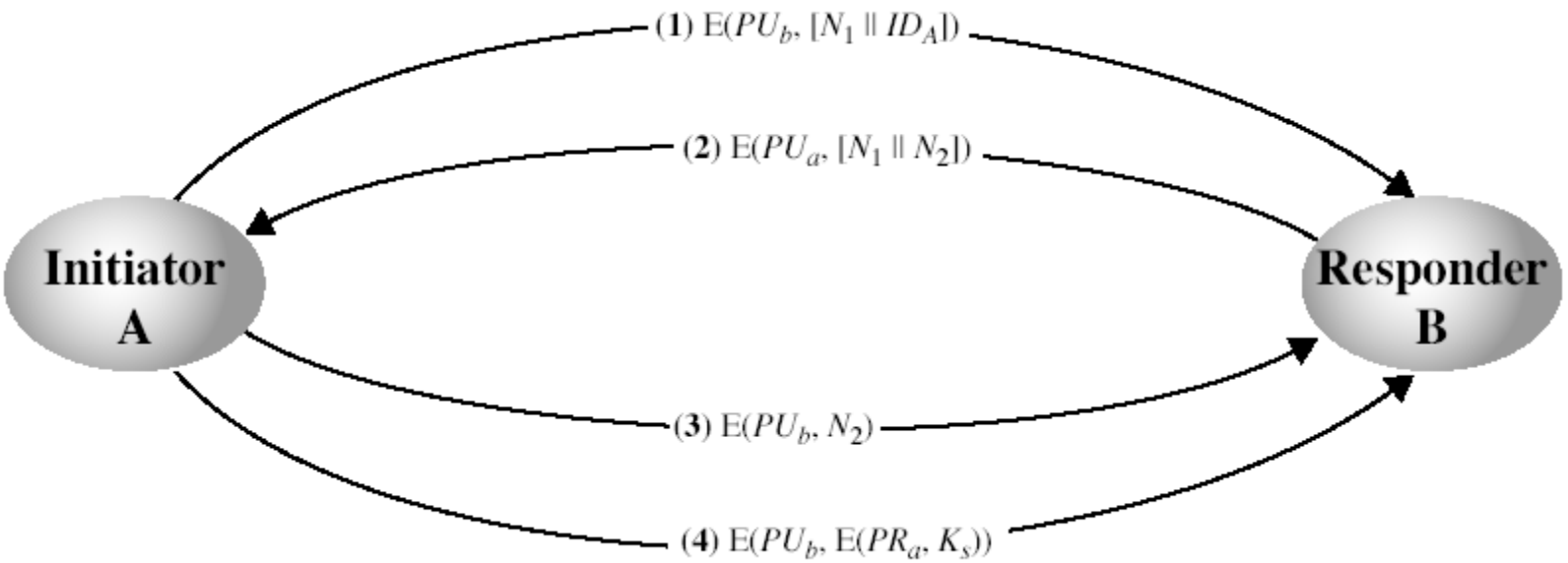
Simple Secret Key Distribution

- problem is that an opponent can intercept and impersonate both parties of protocol
  - man-in-the-middle attack
  - result: A and B think that they exchanged K_s securely but C also knows K_s and use it to eavesdrop the communication passively



Public-Key Distribution of Secret Keys

- assumption: public-keys are securely exchanged a priori
- First three steps are for authentication purposes
- Last step provides both the secrecy and authenticity of the session key



In practice

- Most systems offer a three-level approach
  - use of PKC to exchange master-key
  - use of master-key to exchange session keys
- most important advantage is at performance

A closer look to authentication

- making sure of peer entity's identity
  - mutual or one-way
  - non-repudiation is not an aim here
- generally implemented as a protocol
  - basic idea: making sure that other party knows a common secret
  - also used for session key distribution
- two types
  - message authentication
    - mostly one-way
  - peer entity authentication
    - connection oriented approach

Two key issues

- Protection of any secret information
- Timeliness
  - to prevent replay attacks
    - a valid message is copied and later resent
  - Why replays are bad?
    - at minimum, disrupt proper operation by presenting messages that appear genuine but actually are not
    - Think about a real world example!

Countermeasures - 1

- Sequence numbers
  - not a practical method
  - parties should keep track of the sequence numbers
  - and should take care of message losses, duplications in a secure manner
    - complicates the protocol

Countermeasures - 2

- Timestamps
  - messages contain a timestamp
  - accept only fresh messages based on this timestamp
  - sometimes used, but there are some practical problems
    - clocks must be synchronized in a secure manner
    - tolerance to network delays

Countermeasures - 3

- Challenge/response
  - Initiator sends a nonce (a challenge phrase or number used only one-time) and expects that nonce (or a transformation of it) in the response
  - easier to implement
  - but may require extra message transfer
  - and requires active parties
    - not suitable for connectionless services

Authentication using Symmetric Encryption

- We start with well-known Needham-Schroeder protocol
  - actually have seen it as a key distribution protocol
- There exists a Key Distribution Center (KDC)
  - each party shares own master key with KDC
  - KDC generates session keys used for connections between parties
  - master keys are used to distribute these session keys in a secure way

Needham-Schroeder Protocol

- original three-party key distribution protocol
  - for session between A and B mediated by a trusted KDC
  - KDC should be trusted since it knows the session key
- protocol overview
  1. A→KDC: $ID_A \parallel ID_B \parallel N_1$
  2. KDC→A: $E(K_a,[K_s \parallel ID_B \parallel N_1 \parallel E(K_b,[K_s \parallel ID_A])])$
  3. A→B: $E(K_b,[K_s \parallel ID_A])$
  4. B→A: $E(K_s,[N_2])$
  5. A→B: $E(K_s,[f(N_2)])$
- 4 and 5 prevent a kind of a replay attack
  - against replay of message 3 by an attacker

Needham-Schroeder (NS) Protocol

- but still vulnerable to a replay attack
  - if an old session key has been compromised, then message 3 can be resent to B by an attacker X impersonating A
  - after that X intercepts message 4 and sends a message 5 to B as if it is A
  - now X can impersonate A for the future communications with the session key
  - unlikely but a vulnerability
- modifications to address this problem
  - timestamps (Denning 81), B contacted at the beginning (Needham Schroeder 87)
- see http://www.lsv.ens-cachan.fr/spore/index.html for a repository of security protocols

NS Protocol with timestamps

- proposed by Dorothy Denning in 1981
- A and B can understand replays by checking the timestamp in the message
  - even if attacker knows K_s, he cannot generate message 3 with a fresh timestamp since he does not know K_b

1. A→KDC: $ID_A \,||\, ID_B$

2. KDC→A: $E\,(K_a,[K_s \,||\, ID_B \,||\, T \,||\, E\,(K_b,[K_s||ID_A||T])])$

3. A→B: $E\,(K_b,[K_s||ID_A||T])$

4. B→A: $E\,(K_s,[N_1])$

5. A→B: $E\,(K_s,[f(N_1)])$

Needham – Schroeder Protocol Revisited

- Amended by the creators themselves in 1987 by putting B in the loop early in the protocol

http://doi.acm.org/10.1145/24592.24593

1. A -> B : ID_A
2. B -> A : $E(K_b, [ID_A \| N_b])$
3. A -> KDC : $ID_A \| ID_b \| N_a \| E(K_b, [ID_A \| N_b])$
4. KDC -> A : $E(K_a, [N_a \| ID_b \| K_s \| E(K_b, [K_s \| N_b \| ID_A])])$
5. A -> B : $E(K_b, [K_s \| N_b \| ID_A])$
6. B -> A : $E(K_s, [N_b])$
7. A -> B : $E(K_s, [f(N_b)])$

Synchronization problem

- Proper use of timestamps requires synchronized clocks
- e.g. when sender's clock is ahead of recipients clock
  - attacker can intercept the message and replay later
- Neuman and Stubblebine in 1993 proposed use of nonces with timely session keys to constitute tickets
  - Nonces and timestamps are used together
    - Nonces are to avoid replays. Nonces are also used in challenges/response framework
    - timestamps are expiration dates of the session key established
  - attacker only has a limited time to break the session key
    - in original NS, attacker has unlimited time to break the session key

Neuman Protocol

1. A -> B : $ID_A \| N_a$
2. B -> KDC : $ID_B \| N_b \| E(K_b, [ID_A \| N_a \| T_b])$
3. KDC -> A : $E(K_a, [ID_B \| N_a \| K_s \| T_b]) \| E(K_b, [ID_A \| K_s \| T_b]) \| N_b$
4. A -> B : $E(K_b, [ID_A \| K_s \| T_b]) \| E(K_s, [N_b])$

- $E(K_b, [ID_A \| K_s \| T_b])$ is a ticket that can be used later within the limit of T_b

1. A -> B : $E(K_b, [ID_A \| K_s \| T_b]) \| N_{a'}$
2. B -> A : $N_{b'} \| E(K_s, [N_{a'}])$
3. A -> B : $E(K_s, [N_{b'}])$

- Important Question: Do we need synchronized clocks between A & B to ensure freshness?

Authentication using Public-Key Encryption

- We have given an example method that assumes public keys are known
- There exists protocols that also distributes public keys
  - using a central Authentication Server (AS) or KDC
  - using timestamps or nonces

A timestamp approach

- Denning (in 1981) presented the following:
  1. A→AS: ID_A || ID_B
  2. AS→A: E (PR$_{as}$, [ID_A||PU$_a$||T]) || E (PR$_{as}$, [ID_B||PU$_b$||T])
  3. A→B: E (PR$_{as}$, [ID_A||PU$_a$||T]) || E (PR$_{as}$, [ID_B||PU$_b$||T]) || E (PU$_b$, [E(PR$_a$, [K$_s$||T])])
- note that session key is chosen by A, hence AS need not be trusted to protect it
- timestamps prevent replay but require synchronized clocks

A nonce approach

- Proposed by Woo and Lam (1992)
- nonces are not sent in clear

- SEE THE PROTOCOL in Stallings page 387-388

One-Way Authentication

- required when sender & receiver are not online at same time
  - e-mail is a typical application
  - protocol should not rely on the processing of receiver
- A symmetric encryption approach

  1. A→KDC: $ID_A \| ID_B \| N_1$

  2. KDC→A: $E(K_a, [K_s \| ID_B \| N_1 \| E(K_b, [K_s \| ID_A])])$

  3. A→B: $E(K_b, [K_s \| ID_A]) \| E(K_s, [M])$
- Provides authentication that the sender is A
- does not protect against replays (of msg. 3)
  - could rely on timestamp in message, but delays in email make this problematic

Public-Key Approaches

- have seen some public-key approaches
- if confidentiality is major concern, can use:

  $A \rightarrow B: E(PU_b, [K_s]) \,||\, E(K_s, [M])$

  – digital envelopes
- if authentication needed, use a digital signature with a digital certificate:

  $A \rightarrow B: M \,||\, E(PR_a, [H(M)]) \,||\, E(PR_{as}, [T||ID_A||PU_a])$

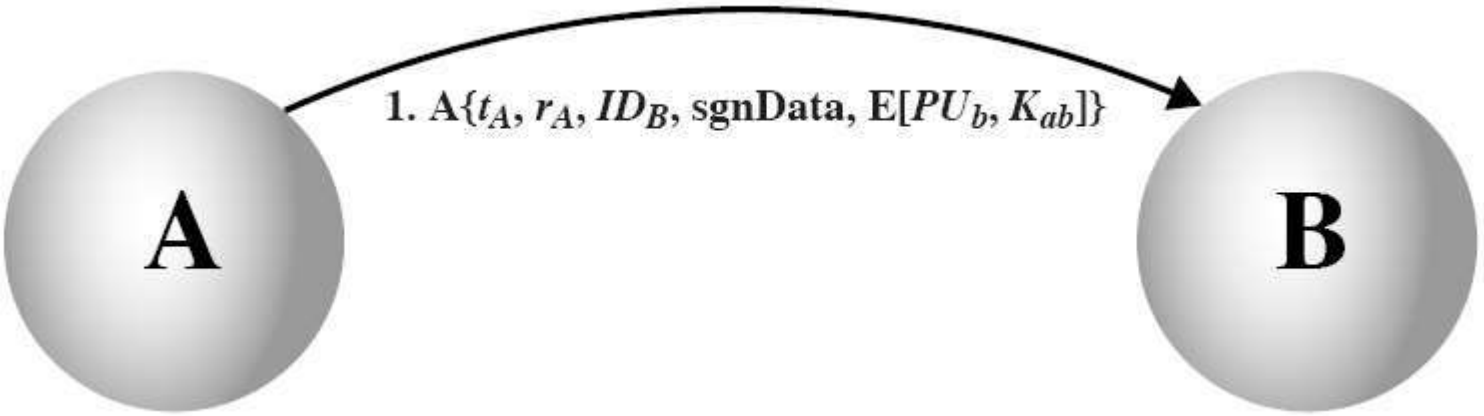
  – message, signature, certificate
- We will detail e-mail security issues later

X.509 Authentication Protocols

- X.509 is a ITU-T standard part of the "directory services"
  - mostly for certificates, but also propose three generic authentication protocols
  - one-way authentication
  - two-way authentication
  - three-way authentication
  - use both nonces and timestamps
    - nonces are unique only within the lifetime of timestamp
      - Timestamp includes expiration time

X.509 one-way authentication

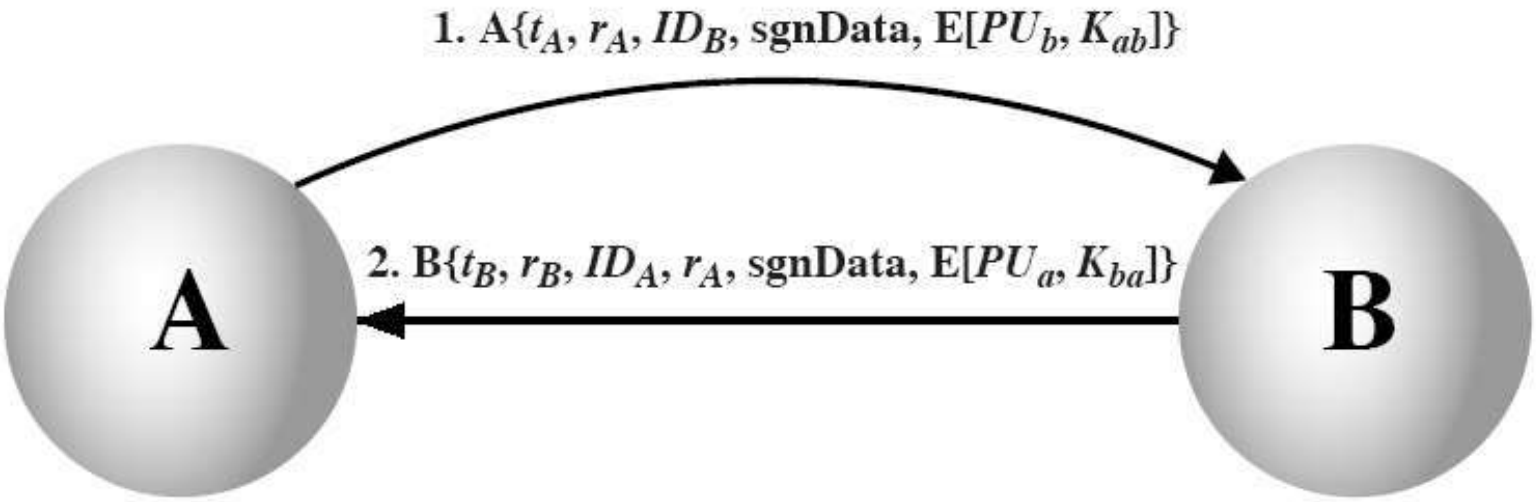
- Proves that the message generated by A and intended for B
  - also proves that message is not a replay
  - proves the identity of the sender, but not the recipient
  - optionally includes a session key and signed information



t_A: timestamp r_A: nonce sgnData: Data signed

X.509 two-way Authentication

- both parties verify identity of each other
- reply message
  - generated by B
  - not a replay (guaranteed by t_B and r_B)
  - includes r_A to match the messages
  - Question: Do we really need t_B and r_B to avoid replay of second message?
    - Answer is in the next protocol!



X.509 three-way Authentication

- Nonces are signed and echoed back
  - each side can check the replay by checking its own nonce
  - timestamps are not needed
    - synchronized clocks are not needed either
    - what is a potential risk in such a case?