BMS
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
Avalahalli, Doddaballapur Main Road, Bengaluru – 560064

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SUBJECT

CRYPTOGRAPHY, NETWORK SECURITY AND CYBER LAW

Subject Code: 15CS61

Semester: 6th Semester

CRYPTOGRAPHY, NETWORK SECURITY AND CYBER LAW
[As per Choice Based Credit System (CBCS) scheme]
(Effective from the academic year 2017 - 2018)
SEMESTER – VI

Subject Code	17CS61	IA Marks	40
Number of Lecture Hours/Week	4	Exam Marks	60
Total Number of Lecture Hours	50	Exam Hours	03

CREDITS – 04

Module – 1	**Teaching Hours**
Introduction - Cyber Attacks, Defence Strategies and Techniques, Guiding Principles, Mathematical Background for Cryptography - Modulo Arithmetic's, The Greatest Comma Divisor, Useful Algebraic Structures, Chinese Remainder Theorem, Basics of Cryptography - Preliminaries, Elementary Substitution Ciphers, Elementary Transport Ciphers, Other Cipher Properties, Secret Key Cryptography – Product Ciphers, DES Construction.	**10 Hours**
Module – 2	
Public Key Cryptography and RSA – RSA Operations, Why Does RSA Work?, Performance, Applications, Practical Issues, Public Key Cryptography Standard (PKCS), Cryptographic Hash - Introduction, Properties, Construction, Applications and Performance, The Birthday Attack, Discrete Logarithm and its Applications - Introduction, Diffie-Hellman Key Exchange, Other Applications.	**10 Hours**
Module – 3	
Key Management - Introduction, Digital Certificates, Public Key Infrastructure, Identity–based Encryption, Authentication–I - One way Authentication, Mutual Authentication, Dictionary Attacks, Authentication – II – Centalised Authentication, The Needham-Schroeder Protocol, Kerberos, Biometrics, IPSec-Security at the Network Layer – Security at Different layers: Pros and Cons, IPSec in Action, Internet Key Exchange (IKE) Protocol, Security Policy and IPSEC, Virtual Private Networks, Security at the Transport Layer - Introduction, SSL Handshake Protocol, SSL Record Layer Protocol, OpenSSL.	**10 Hours**
Module – 4	
IEEE 802.11 Wireless LAN Security - Background, Authentication, Confidentiality and Integrity, Viruses, Worms, and Other Malware, Firewalls – Basics, Practical Issues, Intrusion Prevention and Detection - Introduction, Prevention Versus Detection, Types of Instruction Detection Systems, DDoS Attacks Prevention/Detection, Web Service Security – Motivation, Technologies for Web Services, WS- Security, SAML, Other Standards.	**10 Hours**
Module – 5	
IT act aim and objectives, Scope of the act, Major Concepts, Important provisions, Attribution, acknowledgement, and dispatch of electronic records, Secure electronic records and secure digital signatures, Regulation of certifying authorities: Appointment of Controller and Other officers, Digital Signature certificates, Duties of Subscribers, Penalties and adjudication, The cyber regulations appellate tribunal, Offences, Network service providers not to be liable in certain cases, Miscellaneous Provisions.	**10 Hours**

Course outcomes: The students should be able to:

- Discuss the cryptography and its need to various applications
- Design and Develop simple cryptography algorithms

- Understand the cyber security and need cyber Law

Question paper pattern:

The question paper will have TEN questions.
There will be TWO questions from each module.
Each question will have questions covering all the topics under a module.
The students will have to answer FIVE full questions, selecting ONE full question from each module.

Text Books:

1. Cryptography, Network Security and Cyber Laws – Bernard Menezes, Cengage Learning, 2010 edition (Chapters-1,3,4,5,6,7,8,9,10,11,12,13,14,15,19(19.1-19.5),21(21.1-21.2),22(22.1-22.4),25

Reference Books:

1. Cryptography and Network Security- Behrouz A Forouzan, DebdeepMukhopadhyay, Mc-GrawHill, 3rd Edition, 2015
2. Cryptography and Network Security- William Stallings, Pearson Education, 7th Edition
3. Cyber Law simplified- VivekSood, Mc-GrawHill, 11th reprint , 2013
4. Cyber security and Cyber Laws, Alfred Basta, Nadine Basta, Mary brown, ravindrakumar, Cengage learning

BMS
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
Avalahalli, Doddaballapur Main Road, Bengaluru – 560064
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Course Name	Cryptography Network Security and Cyber Law
Course Code	17CS61

Module-1

- Introduction - Cyber Attacks
- Defence Strategies and Techniques
- Guiding Principles
- Mathematical Background for Cryptography
- Modulo Arithmetic's,
- The Greatest Comma Divisor
- Useful Algebraic Structures
- Chinese Remainder Theorem
- Basics of Cryptography – Preliminaries
- Elementary Substitution Ciphers
- Elementary Transport Ciphers
- Other Cipher Properties
- Secret Key Cryptography – Product Ciphers
- DES Construction.

Cyber Attack

Cyber attack :

- A **cyber attack** is any type of offensive action that targets computer information systems, computer networks or personal computer devices, using various methods to steal, alter or destroy data or information systems.

Cyber Attack

How often do cyber attacks occur?

Cyber attacks hit businesses every day. **Former Cisco CEO John Chambers once said,**

"There are two types of companies: those that have been hacked, and those who don't yet know they have been hacked."

Cyber Attack

- Cyber-attacks are hitting the headlines on a daily basis and a lot of effort goes into both preventing them and dealing with the consequences when they have happened.

- Understanding the motivation behind attacks can help organisations understand more about the risks they face so that they can tackle them.

Motives for Cyber Attacks

Why do cyber-attacks happen?

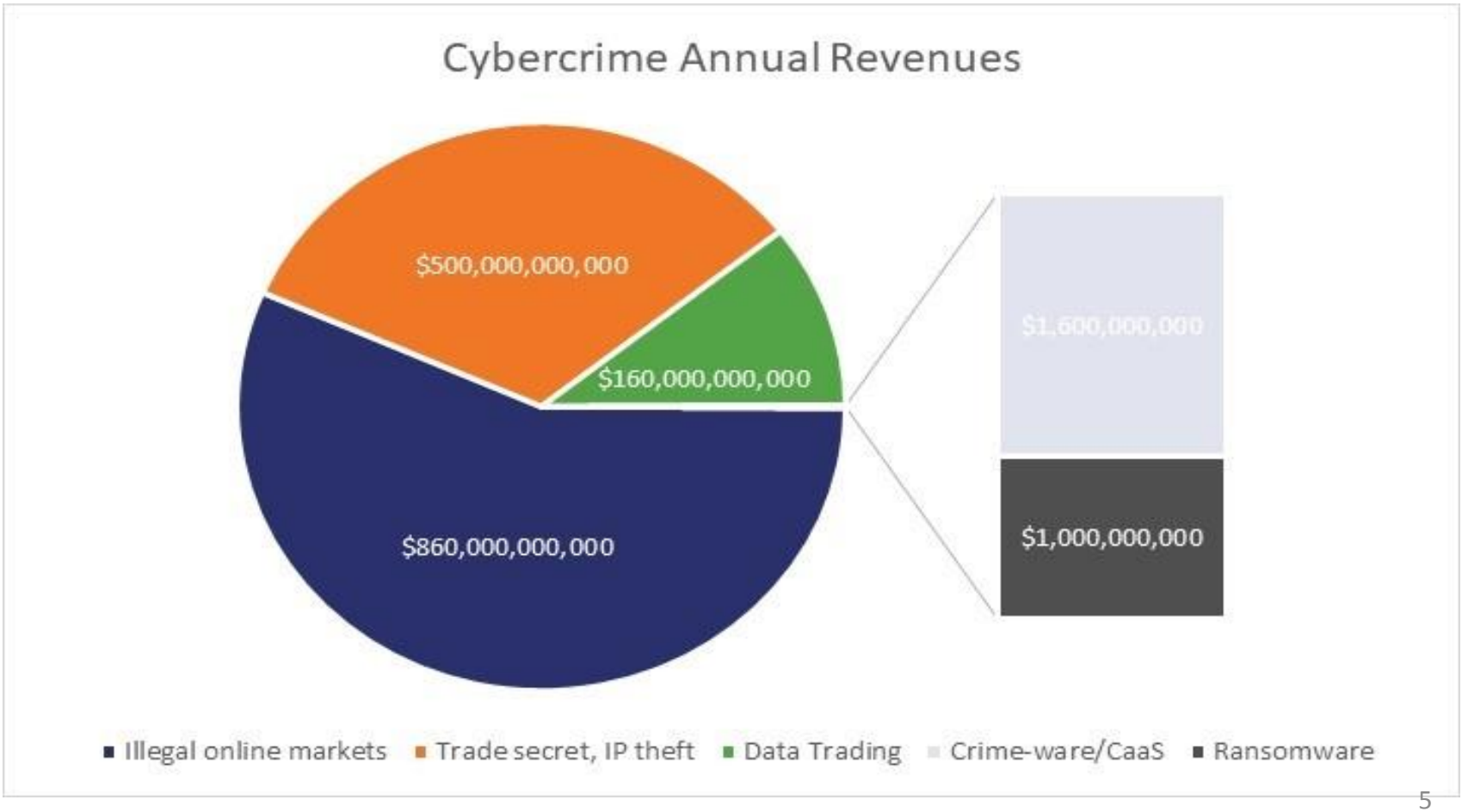
1. For financial gain

Financial gain is the biggest motive behind most of the cyber attacks.

Financial gains: One of the biggest reasons for the popularity of **cyber**-**attacks** is **financial gains**. It is estimated that by the year 2021, the global cost of **cyber** crimes will reach $6 trillion. This market has been expanding so fast that on an average, **cyber**-attackers are earning $1.5 trillion in **profit** annually.

Motives for Cyber Attacks

1. For financial gain



Motives for Cyber Attacks

1. For Financial Gain

Crime	Annual Revenues
Illegal online markets	$860 Billion
Trade secret, IP theft	$500 Billion
Data Trading	$160 Billion
Crime-ware	$1.6 Billion
Ransomware	$1 Billion
Total Cybercrime Revenues	$1.5 Trillion

FACT: *Over 50% of cybercrime revenues are generated in online markets.*

How much money do cybercriminals earn?

- For an individual with the right skillset, cybercrime can be incredibly lucrative. An individual cybercriminal can make upwards of half a million dollars in a year simply by trafficking in stolen data.
- Like real criminality, cyber criminals can generally be broken down into levels. Some, like **low-level criminals**, are content to execute petty crimes that don't pay all that well. Others, are **highly specialized** and only work when the money is good.

- A cyber attack on Union bank of India last July. The opening of the email attachment, which looked like it had come from India's central bank. After an employee opened email attachment releasing malware that allowed hackers to steal the banks data.

How much money do cybercriminals earn?

WHAT CYBERCRIMINALS EARN

- **Individual hackers** may earn around **$30,000** for one or several jobs; but **platform managers** offering multiple card data forums can **earn up to $2 million**.

- **Individual earnings** from **cybercrime** are on **average 10-15% higher** than most traditional crimes.

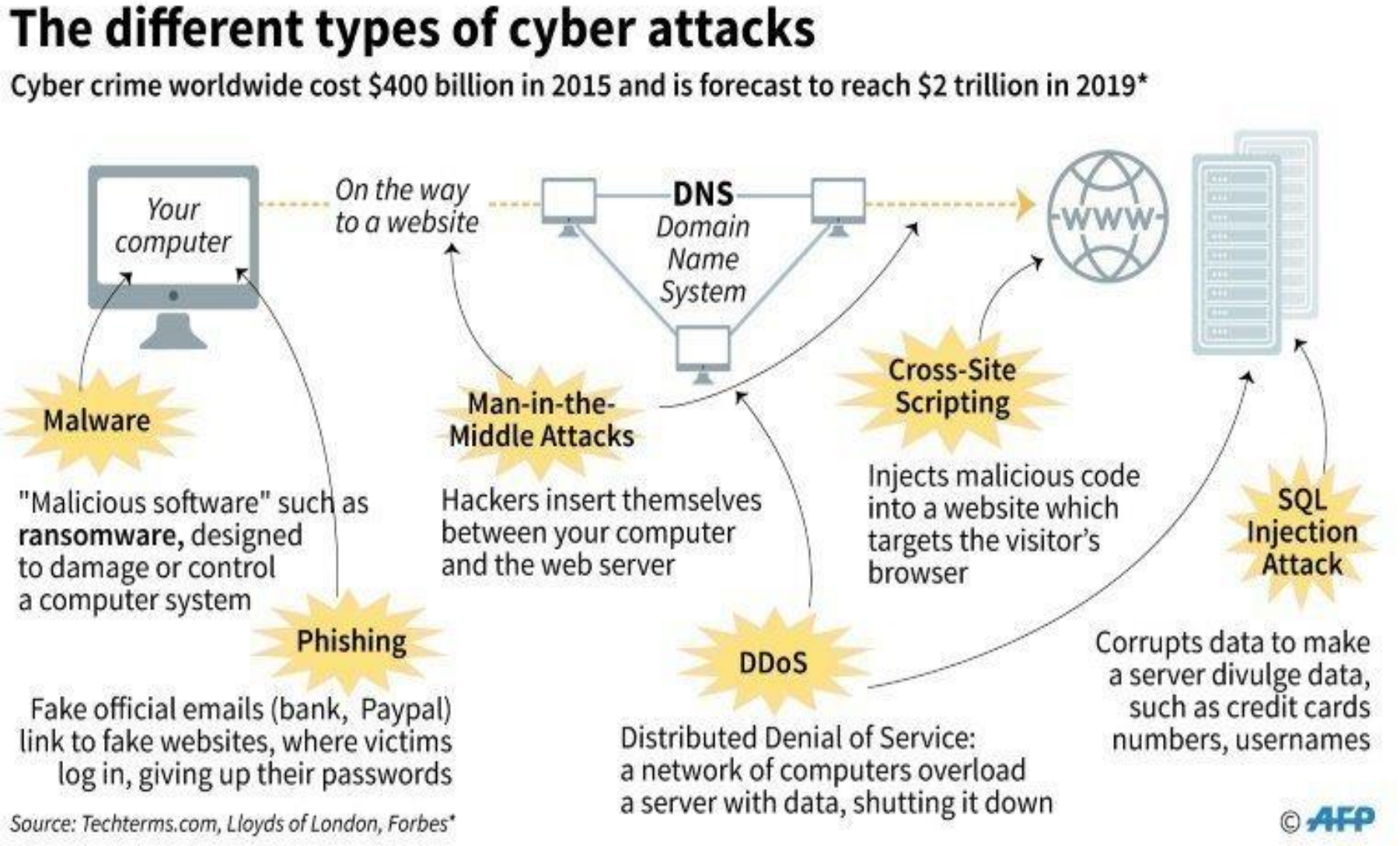
- **Earning levels:**
  High earners: $166K+
  Middle earners: $75K+
  Low earners: $3.5K

Motives for Cyber Attacks

This is the most likely reason an organization get attacked.

- Business' financial details
- Customers' financial details (eg credit card data)
- Sensitive personal data
- Customers' or staff email addresses and login credentials
- Customer Databases
- Clients Lists
- IT infrastructure
- IT services (eg the ability to accept online payments)
- Intellectual Property (eg trade secrets or product designs)

Common Cyber Attacks



Types of cyber attack

Types of cyber attack

To achieve those goals of gaining access or disabling operations, a number of different technical methods are deployed by cybercriminals.

- Phishing
- Malware
- Denial of service
- Man in the middle
- Cryptojacking
- SQL injection
- Zero-day exploits

Phishing

Phishing is the fraudulent attempt to obtain sensitive information such as usernames, passwords and credit card details by disguising oneself as a trustworthy entity in an electronic communication. Typically carried out by **email spoofing or instant messaging** it often directs users to enter personal information at a fake website which matches the look and feel of the legitimate site.

Email spoofing is one of the easiest types of phishing used to get data from users without their knowledge.

It can be done in different ways:

- Sending an email through a familiar username,
- Impersonating the identity of an organization and asking employees to share internal data.

Phishing

Here is an example

From: **avlor.hr@gmail.com**

To: bruce@avlorcloud.info

Subject: Message from human resources

Dear Bruce,

An information document has been sent to you by the Human Resources Department. **Click here to Login** to view the document.

Thank you

HR Department

Avlorcloud University of California

Just by seeing the company's name and the urgency of action, some users may click on the link.

Phishing

How to prevent email phishing?

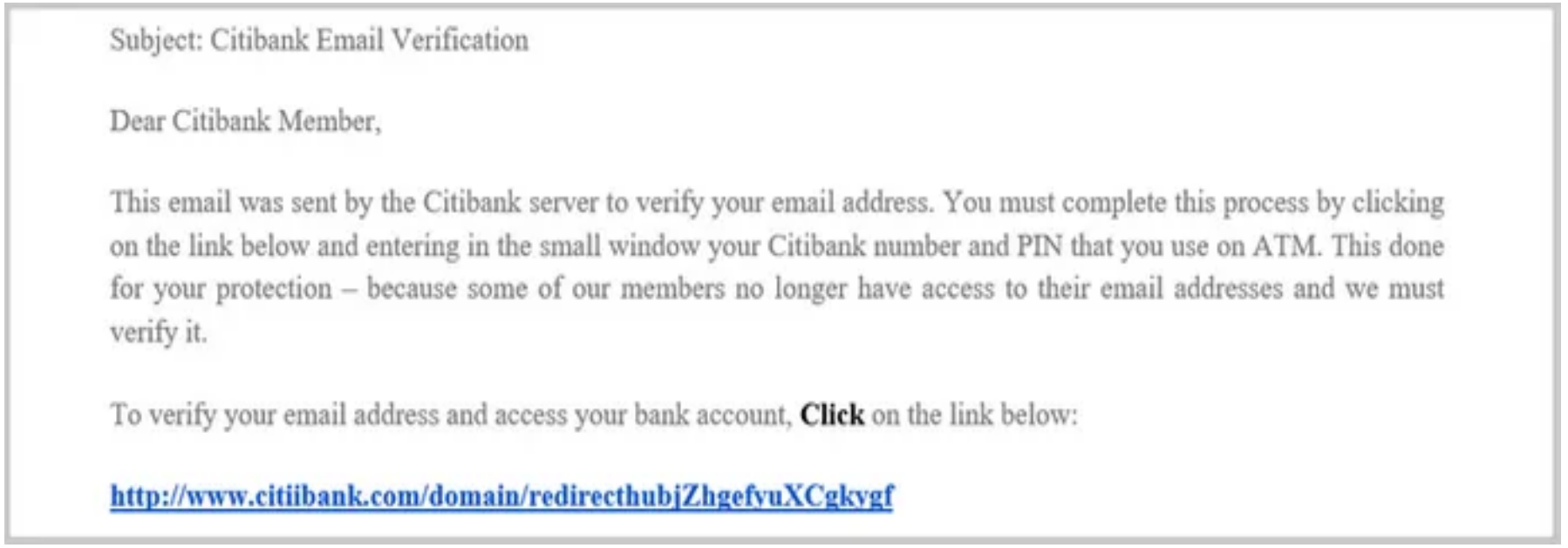
The best way to prevent these attacks is by carefully reading the sender's email address. If you are not sure about the characters in an email address, then copy and paste it in the notepad to check the use of numeric or special characters.

Phishing

Misspelled URL

Hackers buy domains that sound similar to popular websites. Then, they phish users by creating an identical website, where they ask targets to log in by submitting personal information.

In the example below, you can see that there's a typo in the link that people can easily miss: "www.citiibank.com…" instead of "www.citibank.com…"



Subject: Citibank Email Verification

Dear Citibank Member,

This email was sent by the Citibank server to verify your email address. You must complete this process by clicking on the link below and entering in the small window your Citibank number and PIN that you use on ATM. This done for your protection – because some of our members no longer have access to their email addresses and we must verify it.

To verify your email address and access your bank account, **Click** on the link below:

http://www.citiibank.com/domain/redirecthubjZhgefyuXCgkvgf

Phishing

Pop-Up Messages: In-Session Phishing

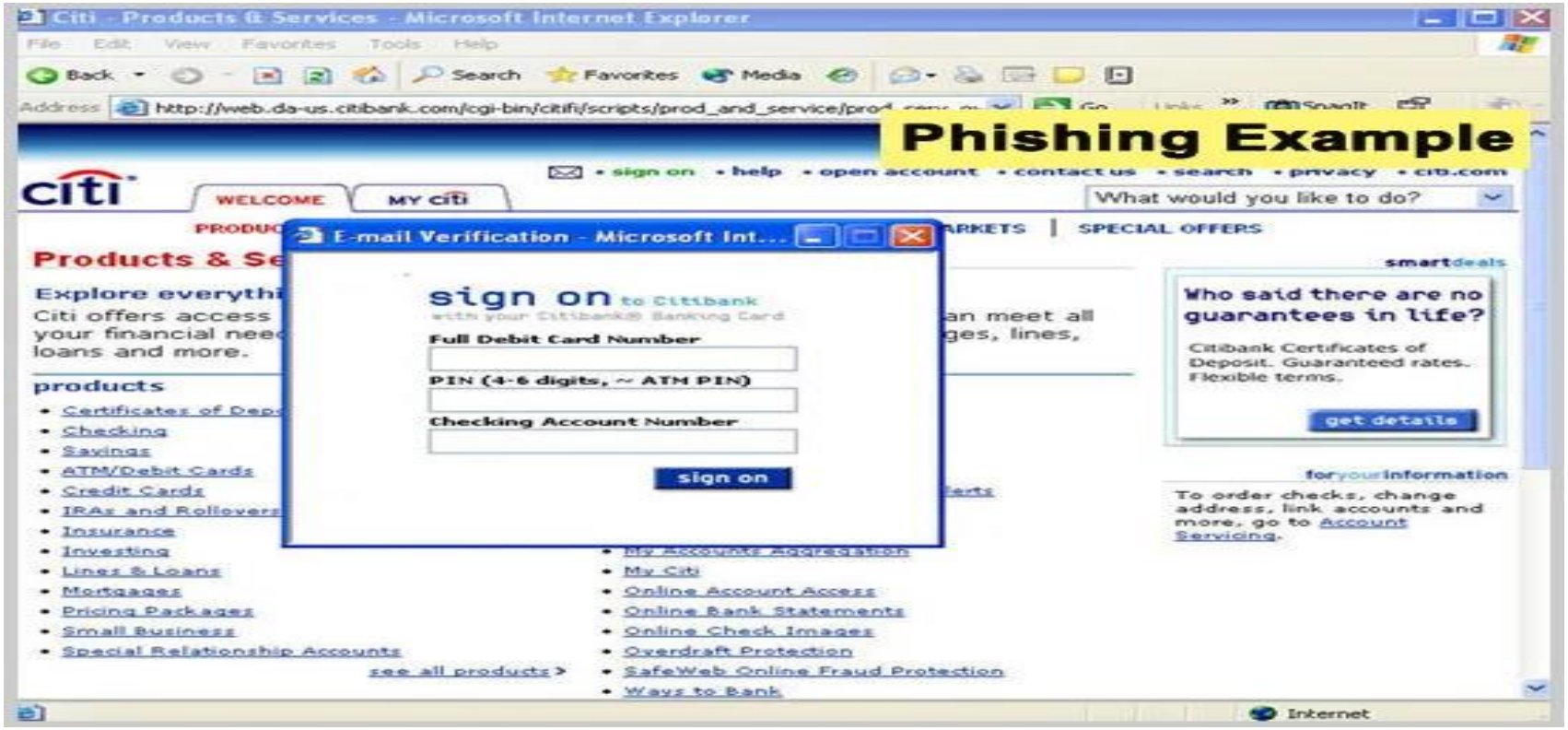
Pop-up messages are the easiest way to run a successful phishing.

Through pop-up messages, attackers get a window to steal the login credentials by redirecting them to a fake website.

This technique of phishing is also known as “In-session phishing.”

Look at the pop-up window given below.

In this example, doesn’t the foreground pop-up seem legitimate enough to mislead customers?



Malware

•**Malware:** A software that is specifically designed to disrupt, damage, or gain unauthorized access to a computer system.

•Stealing information from computer without your knowledge

•In Spanish, **"mal"** is a prefix that means **"bad,"** making the term "badware,“

Malware

Virus v/s Worm

Virus	Worm
• Attaches itself to OS or the programs	• Do not Attaches itself to OS
• Need user action to abet their propagation.	• Self propagates across a network exploiting security in widely used services.
• Damages caused is mostly local to the machine	• It harms the network and consumes n/w bandwidth.
• Spread quite slowly	• Spread much more rapidly Ex. SQL Slammer worm 75,000 victims within ten minutes.

Malware

- A **Trojan** is a type of malicious code or software that looks legitimate but can take control of your computer like modification of file and data theft.

- **Spyware** is a malicious software designed to monitor user activity to recover valuable information such as passwords.

- **Adware:** software that automatically displays or downloads advertising material such as banners or pop-ups when a user is online.

Malware

•Common examples of malware include **viruses, worms,** and **spyware.**

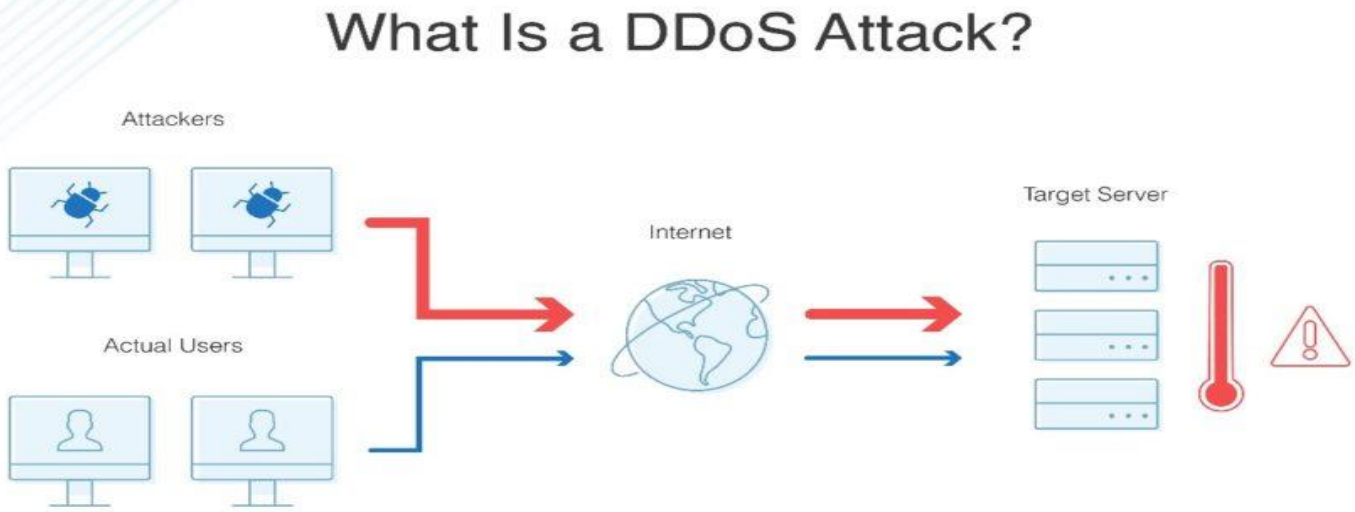
•Malicious little programs can create files, move files, erase files, consume your computer's memory, and cause your computer not to function correctly. Some viruses can duplicate themselves, attach themselves to programs, and travel across networks. In fact opening an infected e-mail attachment is the most common way to get a virus.

•There are many antivirus programs available that scan incoming files for viruses before they can cause damage to your computer. Some of these programs include Norton AntiVirus, McAfee VirusScan, and Virex.

Denial-of-Service (DoS)

•A **DoS** is an attack meant to shut down a machine or network, making it inaccessible to its intended users. These exhaust computer power, memory capacity or communication bandwidth of their targets so that they are rendered unavailable.

•**DoS** attacks accomplish this by flooding the target with traffic, or sending it information that triggers a crash.



Denial-of-Service (DoS)

•Victims of DoS attacks often target web servers of high-profile organizations such as banking, commerce, and media companies, or government and trade organizations.

•Though DoS attacks do not typically result in the theft or loss of significant information or other assets, they can cost the victim a great deal of time and money to handle.

•**Volumetric attacks**. This is the most common type of DoS attack. A bot overwhelms the network's bandwidth by **sending huge numbers of false requests to every open port**.

•Two main kinds of volumetric attacks are called UDP floods and ICMP floods.

Denial-of-Service (DoS)

There are several measures that you can use to protect your business from a DoS attack.

•**Have a plan**. To start, set up a DoS response plan. Defining a clear response from your organization in the event of a DoS attack.

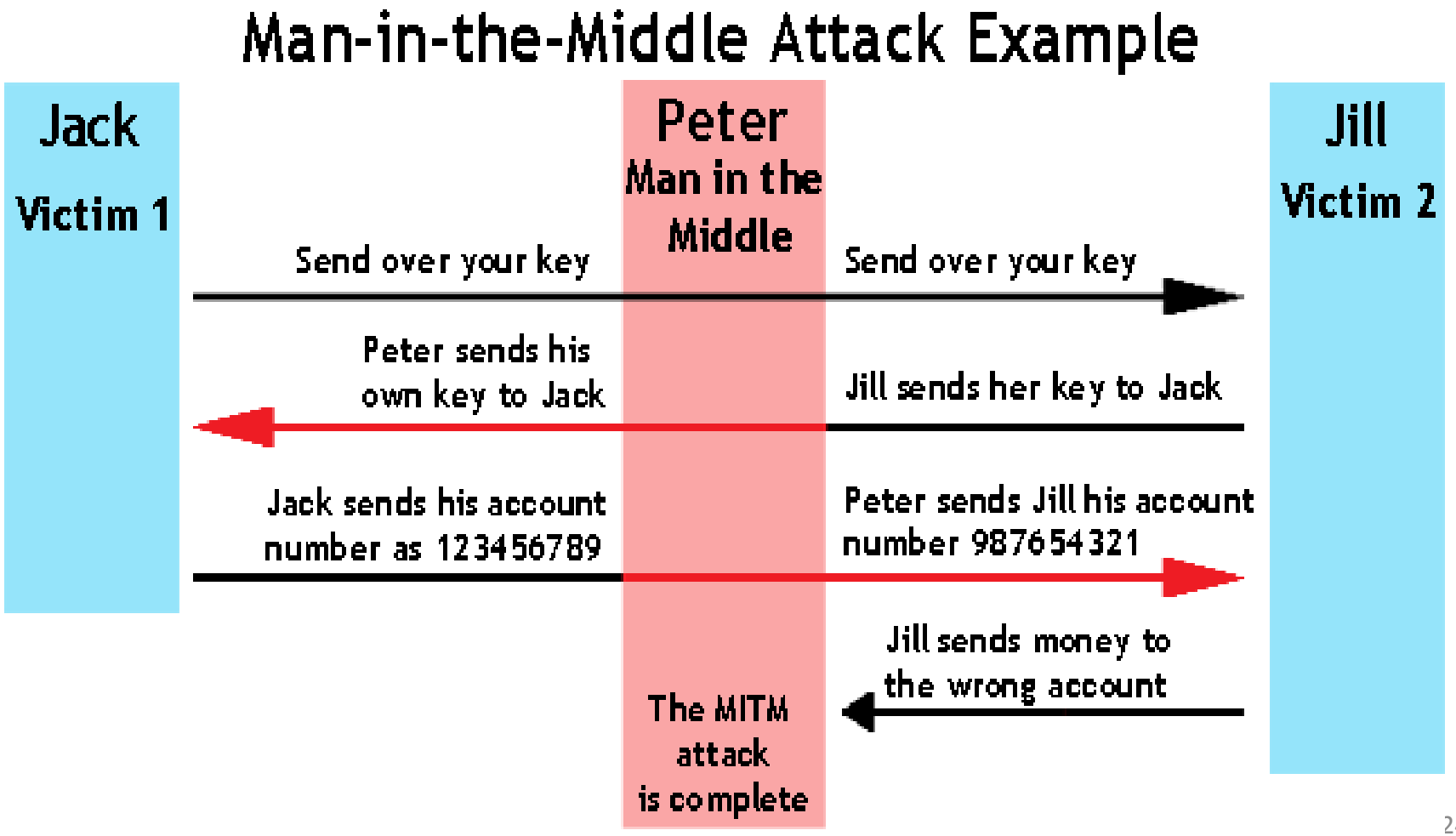
•**Keep everything up to date**. All these systems should be kept up to date, to make sure that any bugs or issues are fixed.

•Install and maintain antivirus software.

•Install a firewall and configure it to restrict traffic coming into and leaving your computer

Man-in-the-Middle Attack (MIMT)

(**MITM**) is an attack where the attacker secretly relays and possibly alters the communications between two parties who believe that they are directly communicating with each other



Man-in-the-Middle Attack (MIMT)

Types of Man-in-the Middle Attacks

•**Wi-Fi Eavesdropping:** Public wi-fi is usually provided "as-is," with no guarantees over the quality of service.

•Another Wi-Fi Eavesdropping attack happens when a hacker creates its own wi-fi hotspot, called an "Evil Twin." They make the connection look just like the authentic one, down to the network ID and passwords. Users may automatically connect to the "evil twin," allowing the hacker to snoop on their activity.

Man-in-the-Middle Attack (MIMT)

Types of Man-in-the Middle Attacks

HTTPS Spoofing

•It's not currently possible to duplicate an HTTPS website.

•However, security researchers have demonstrated a theoretical method for bypassing HTTPS. The hacker creates a web address that looks like an authentic address.

•Instead of regular characters, it uses letters from foreign alphabets. This appears as spam emails you may have seen with strange characters. For instance, Rolex might be spelled Rólex.

•**Email Spoofing, IP Spoofing Attacks, DNS Spoofing and ARP Spoofing**

Man-in-the-Middle Attack (MIMT)

Man in the Middle Attack Prevention

•**Use a Virtual Private Network (VPN)** to encrypt your web traffic. An encrypted VPN severely limits a hacker's ability to read or modify web traffic.

•**Network Security:** Network administrators should be using good network hygiene to mitigate a man-in-the middle attack. Analyze traffic patterns to identify unusual behavior.

•Your network should **have strong firewalls and protocols to prevent unauthorized access**.

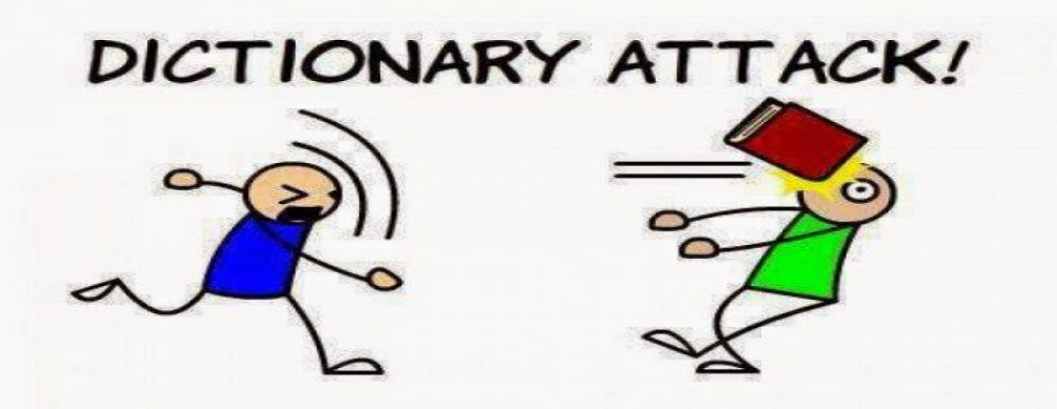
•**Install active virus and malware protection** that includes a scanner that runs on your system at boot.

Dictionary Attacks

- A method used to break security systems, specifically password based security systems, in which the attacker systematically tests all possible passwords beginning with words that have a higher possibility of being used, such as names and places.

- The word "dictionary" refers to the attacker exhausting all of the words in a dictionary in an attempt to discover the password.

- Dictionary attacks are typically done with software (cracx, mortemale) instead of an individual manually trying each password.



Dictionary attacks

- It is estimated that around 80% of people re-use their passwords across online platforms including social media, personal banking, and even work-related systems. While this may seem like a good way to help remember your passwords for important accounts, it is actually leaving you vulnerable to a data breach.

- Facebook CEO, Mark Zuckerberg, who had his social media accounts compromised – including Twitter, where hackers tweeted from his account. His password for his LinkedIn account, *dadada*, was also used for his Twitter.

- Dropbox suffered a breach in 2012, an employee using the same password for LinkedIn that they used for their corporate Dropbox account. Instead of some careless tweets from a hacker, this breach resulted in the theft of 60 million user credentials.

How to Prevent Dictionary Attacks

- The best strategy for creating a long password, that is also memorable, is to make it a passphrase. A passphrase is a sentence or phrase, with or without spaces, typically more than 20 characters longer. The words making up a passphrase are less susceptible to social engineering.

- Your passphrase might be based upon a favourite childhood memory, favourite food, or place you've visited, experiences you've had, or some combination of these things.

SQL Injection Attack

• SQL is standard query language for accessing and manipulating databases.

What does SQL do?

- Executes queries
- Insert update and delete record
- Create new database
- Create new tables
- Create stored procedures
- Create Views
- Set permission on tables, procedures, and views

SQL Injection Attack

- SQL injection is a code injection technique, used to attack data-driven applications, in which malicious SQL statements are inserted into an entry field for execution.

- This is a method to attack web applications that have a data repository.

- The attacker would send a specially crafted SQL statement that is designed to cause some malicious action.

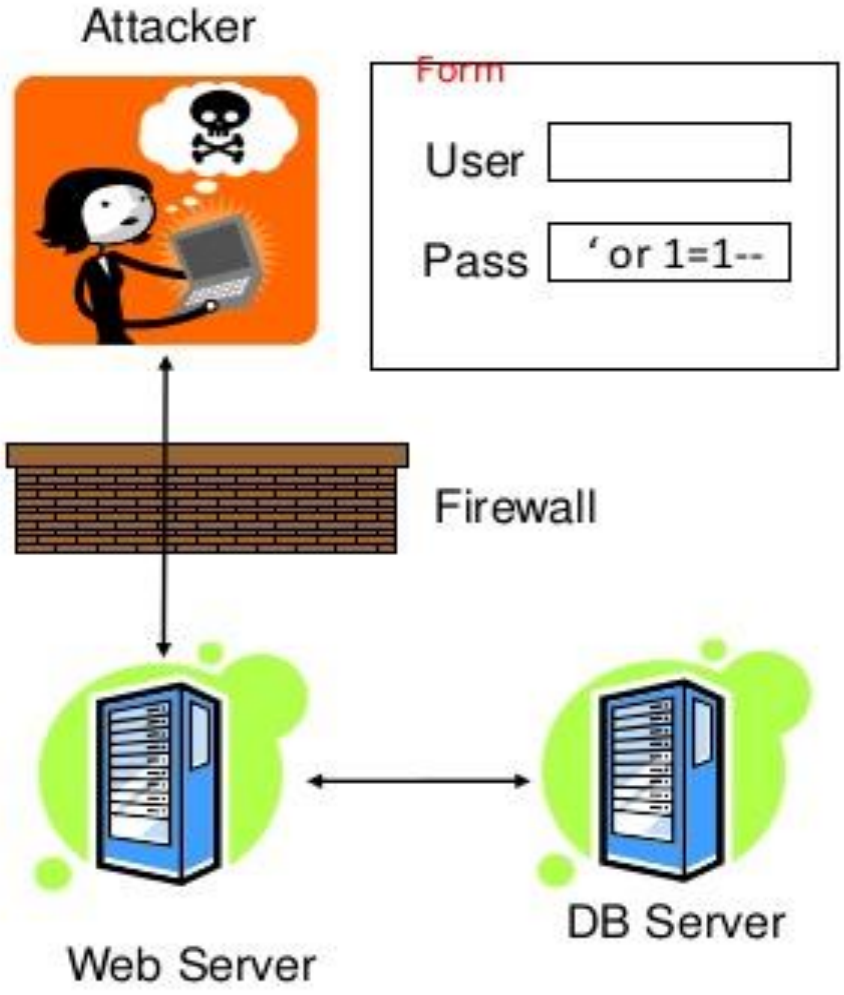
Attack Intent

- Determining database schema
- Extracting data
- Adding or modifying data
- Bypassing authentication

- In August 17, 2009, the United States Justice Department charged an American citizen Albert Gonzalez and two Russians with the theft of 130 million credit card numbers using an SQL injection attack.

How SQL Injection works?

1. App sends form to user.
2. Attacker submits form with SQL exploit data.
3. Application builds string with exploit data.
4. Application sends SQL query to DB.
5. DB executes query, including exploit, sends data back to application.
6. Application returns data to user.



SQL Injection Attack #1

Unauthorized Access Attempt:

`password = ' or 1=1 --`

SQL statement becomes:

select count(*) from *users* **where** *username = 'user'*
and *password = '' or 1=1 --*

Checks if password is empty OR 1=1, which is always true, permitting access.

Defence Against SQL Injection

1. Comprehensive data sanitization

- Web sites must filter *all* user input
- For example, e-mail addresses should be filtered to allow only the characters allowed in an e-mail address.
- Its SQL injection defenses can catch most attempts to sneak SQL through web channels.

Defence Against SQL Injection

2. Use a web application firewall

- A popular example is the free, open source module ModSecurity.
- ModSecurity provides a sophisticated and ever-evolving set of rules to filter potentially dangerous web requests.

Defence Against SQL Injection

3. Limit database privileges by context

- Create multiple database user accounts with the minimum levels of privilege for their usage environment.
- For example, the code behind a login page should query the database using an account limited only to the relevent credentials table.
- This way, a breach through this channel cannot be leveraged to compromise the entire database.

Crytptojacking

• **Cryptojacking** is the unauthorized use of someone else's computer to mine cryptocurrency.

What are cryptocurrencies?

• Cryptocurrencies are forms of digital money that exist only in the online world, with no actual physical form.

• One of the earliest, most successful forms of cryptocurrency is Bitcoin, came out in 2009.

•By December 2017, the value of a single bitcoin had reached an all-time high of nearly $20,000 USD.

Crytptojacking

- Hackers do **Cryptojacking** by either getting the victim to click on a malicious link in an email that loads cryptomining code on the computer, or by infecting a website or online ad with JavaScript code that auto-executes once loaded in the victim's browser.

- In January 2018, researchers discovered the Smominru cryptomining botnet, which infected more than a half-million machines, mostly in Russia, India, and Taiwan.

- Cryptojacking doesn't even require significant technical skills.

- The simple reason why cryptojacking is becoming more popular with hackers is more money for less risk.

How Crytptojacking Works

Hackers have number ways to get a victim's computer to secretly mine cryptocurrencies.

One is to trick victims into loading cryptomining code onto their computers. This is done through phishing-like tactics: Victims receive a legitimate-looking email that encourages them to click on a link. The link runs code that places the cryptomining script on the computer. The script then runs in the background as the victim works.

How to Prevent Crytptojacking

Use endpoint protection that is capable of detecting known crypto miners. Many of the endpoint protection/antivirus software vendors have added crypto miner detection to their products. "Antivirus is one of the good things to have on endpoints to try to protect against cryptomining.

Keep your web filtering tools up to date. If you identify a web page that is delivering cryptojacking scripts, make sure your users are blocked from accessing it again.

Maintain browser extensions Some attackers are using malicious browser extensions to execute cryptomining scripts.

Refer Book for the below topics: imp

- **Vulnerabilities**
  - Human Vulnerabilities
  - Protocol Vulnerabilities
  - Software Vulnerabilities
  - Configuration Vulnerabilities
- **Defence Strategies and Techniques**
  - Access control : Authentication an Authorization
  - Data Protection
  - Prevention and Detection
  - Response, Recovery and Foresenics
- **Guiding Principles**

Refer Notes for the below topics: imp

- **Mathematical background for cryptography**
  - **Modular arithmetic**
  - **The greatest common divisor**
    - **Euclid Algorithm**
  - **Useful Algebraic structure**
    - **Groups : Definition and problems**
    - **Rings : explanation**
    - **Fields**
      - **Polynomial fields: Explanation Problems**

Basics of Cryptography

- **Preliminaries**
  - **Secret versus Public Key Cryptography**
  - **Types of Attack**
- **Elementary Substitution Ciphers**
  - **Monoalphabetic Ciphers**
  - **Polyalphabetic Ciphers**
- **Elementary Transpose Ciphers**
- **Other Cipher Properties**

Basics of Cryptography

Cryptography :

- Is a method of protecting information and communication through use of codes so that only those for whom the information is intended can read and process it.

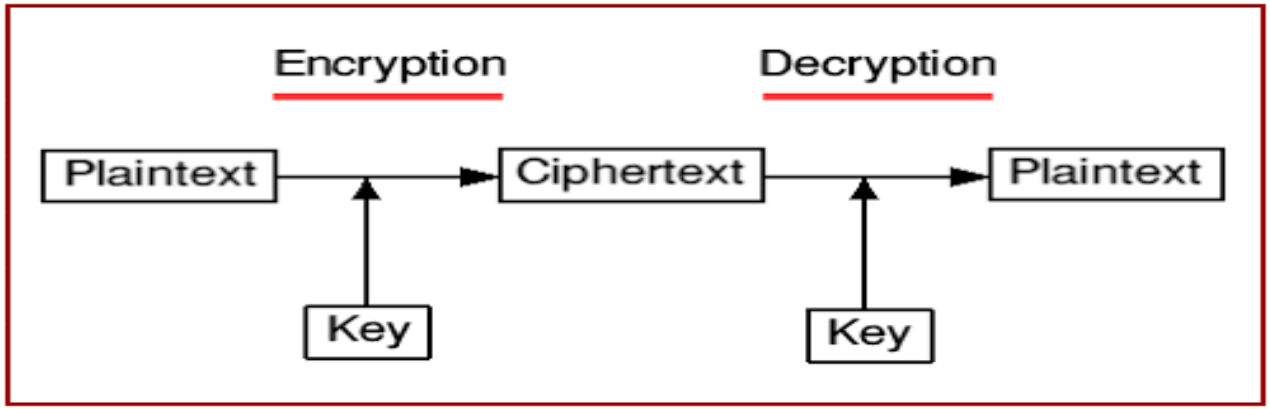
- A cryptographic transformation of data is a procedure by which **plain text** data is disguised or encrypted, resulting in an altered text called **cipher text** that does not reveal the original input.
- Modern cryptography uses mathematical equations(algorithms) to encrypt and decrypt data.

- Today cryptography is used to provide secrecy and integrity of our data and both authentication and anonymity to our communication.

Preliminaries

- The original message to be transformed is called **plain text** and disguised version is called **cipher text.**

- **Encryption** is the process of converting normal message (plaintext) into meaningless message (Ciphertext). Whereas **Decryption** is the process of converting meaningless message (Ciphertext) into its original form (Plaintext).



Preliminaries

- Encryption uses encryption algorithm denoted by *E* and an encryption key *e.*

- Decryption uses decryption algorithm denoted by *D* and an decryption key *d.*

$$c = E_e(p)$$

$$p = D_d(c)$$

p denotes a block of plain text. It is encrypted by sender to produce cipher text denoted by *c*.

Secret versus Public Key Cryptography

There are two types of cryptography

1. Secret key cryptography
2. Public key cryptography

Secret-key cryptography refers to cryptographic system that uses the **same key** to encrypt and decrypt data.

So $e = d$ in the above equation. Hence this from also referred as **symmetric key cryptography**.

Public key cryptography (PKC) is an **encryption** technique that uses a paired **public** and **private key** (or **asymmetric key**) algorithm for secure data communication. A message sender uses a recipient's **public key** to encrypt a message. To decrypt the sender's message, only the recipient's **private key** may be used.

Hence this form also referred as **asymmetric key cryptography**.

Secret versus Public Key Cryptography

–Alice intends to send confidential message Bob

–If Alice and Bob share secret key *k,* then she encrypts the message using the common secret key.

–The encrypted message received by Bob is decrypted using same secret key

$$\textit{Operation performed by Alice} \quad c = E_k(p)$$

$$\textit{Operation performed by Bob} \quad p = D_k(c)$$

–Alice may wish to use public key cryptography assuming that Bob has public-private key pair

–She encrypt her message using her public key

–Bob decrypt the message using corresponding private key

$$\textit{Operation performed by Alice} \quad c = E_{B,pu}(p)$$

$$\textit{Operation performed by Bob} \quad p = D_{B,pr}(c)$$

Secret versus Public Key Cryptography

There are several cryptographic algorithms available today.

- Data encryption standard (DES)
- Advanced encryption standard (AES)
- RSA
- Elliptic Curve Cryptography
- Blow fish
- RC4

Types of Attack

A cryptographic algorithm is secure, if a cryptanalyst is unable to

- Obtain the corresponding plain text from given cipher text
- Deduce the secret key or the private key

Types of attack

- Known cipher text attack
- Known plain text
- Chosen plain text

Known cipher text attack

- Cryptanalyst could accumulate abundant amount of cipher text
- He could then look for patterns in the cipher text in an attempt to reconstruct some plain text and / or deduce the key
- Also called **cipher text-only attack (COA)**

Types of Attack

A cryptographic Algorithm is secure if a cryptanalyst is unable to

- Obtain the corresponding plain text from given cipher text
- Deduce the secret key or the private key

Types of attack

- Known cipher text attack
- Known plain text attack
- Chosen plain text attack

Known plain text attack

- Cryptanalyst have all or part of some plaintext blocks are predictable or guessed.
- Cryptanalyst then build a repertoire of corresponding plaintext, cipher text pairs with the intention of deducing the key.

Types of Attack

A cryptographic Algorithm is secure if a cryptanalyst is unable to

- Obtain the corresponding plain text from given cipher text
- Deduce the secret key or the private key

Types of attack

- Known cipher text attack
- Known plain text attack
- Chosen plain text attack

Chosen plain text attack

- Cryptanalyst carefully choose pieces of plain text and then induce the sender to encrypt such text.
- Cryptographic scheme makes use of pairs of attacker chosen plain text and then corresponding cipher text is called **chosen plain text.**

Types of Attack

Brute force algorithm by trying all possible key values

Let

$(p_1, c_1), (p_2, c_2), \ldots . (p_m, c_m)$, Be plaintext-cipher text pairs

```
For (each potential key, k in the key space)
{
        Proceed = true;
        i = 1;

        While (proceed  = = true  &&  i<=m)
        {
                If (c_i ≠ E_k (P_i))
                {
                        proceed = false ;
                }
                i++;
        }
        If ( i = m + 1)
                Print("key values is k");
}
```

Elementary Substitution Ciphers : Monoalphabetic Ciphers:

Simple substitution: substitute each character by another character or symbol

Monoalphabetic Ciphers: Each letter is always substituted for another unique letter

Let Σ = {A, B, ... Z},

- A Monoalphabetic Ciphers defines the permutations of the elements in Σ.
- There are 26! Permutations. There are 26! possible monoalphbetic substitution ciphers.

- Replace each alphabet ***a*** in text by the alphabet ***k*** position away in mod 26 . This type of scheme refer as Caesar cipher

$$c_i = E(p_i) = (p_i + 3) \bmod 26$$

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

Elementary Substitution Ciphers : Monoalphabetic Ciphers

- Approach to attack **Caesar attack** is to compute the frequencies of different alphabets occurring in the cipher text.
- Number of studies have been conducted on the frequency distribution of the alphabets in regular text.
- For example: Most occurring letters in English are E(12.7%), T(9.1%) and A(9.2%).
- Given string of cipher text, substitute the three most frequently occurring letters in the cipher text for the three most frequently occurring letters in "regular" English.
- We could use other rules such as "the letters R and N never occur consecutively" or the letter "The letter Q followed by a U"

Elementary Substitution Ciphers : Polyalphabetic Ciphers

- In poly-alphabetic ciphers, the cipher text corresponding to a particular character in the plain text is not fixed. It may depend on its position in the block.

- In mono-alphabetic cipher, the relationship between a character in the plain text and characters in the cipher text is one-to-one whereas in poly-alphabetic ciphers is one-to-many.

- **The Vigenere Cipher**
- **The Hill Cipher**

Polyalphabetic Ciphers: Vigenere Cipher

- **The Vigenere Cipher :** is a poly-alphabetic cipher that uses a multi digit key k1, k2, k3,…..km.

- The plain text is split into non-overlapping blocks each containing *m* consecutive characters.

- Then first letter of each block is replaced by the letter k1 position to its right, the second letter of each block is replaced by the letter k2 position to its right.

Plain text:	w	i	s	h	i	n	g		y	o	u		s	u	c	c	e	s	s
Key:	4	19	03	22	07	12	05		11	04	19		05	11	04	19	03	22	07
Cipher text:	A	B	V	D	Y	L			J	S	N		X	F	G	V	H	O	Z

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

Polyalphabetic Ciphers: Vigenere Cipher

- **Encrypt the massage MAKE IT HAPPEN using the Vigenere cipher and key word MATH**

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

M	A	T	H
12	0	19	7

Plain	M	A	K	E	I	T	H	A	P	P	E	N
Key	12	0	19	7	12	0	19	7	12	0	19	7
Cipher	Y	A	D	L	U	T	A	H	B	P	X	U

Plain text: MAKE IT HAPPEN

Cipher Text : YADL UT AHBPXU

Polyalphabetic Ciphers: Hill Cipher

- **The Hill Cipher** is another polyalphabetic cipher proposed by Lester Hill.

- Let p1, p2, p3 . . . . pm be the numeric representation of the characters in the plain text and let c1, c2, c3, ...cm represent the corresponding characters in the cipher text.

- To compute the cipher text we map each alphabet to an integer. We use the mapping A -> 0, B -> 1, ... Z -> 25

- The relationship between a block of plain text and its cipher text is expressed as follows

c = p K mod 26, ***c*** and ***p*** are row vectors corresponding to the plain text and cipher text and ***K*** is the *m x m* matrix comprising the key

At the receiver end, the plaintext is recovered $p = c K^{-1} \bmod 26$

Polyalphabetic Ciphers: Hill Cipher

- **Consider a hill cipher with m = 2 (block size = 2) with key K shown below**

$$K = \begin{bmatrix} 3 & 7 \\ 15 & 12 \end{bmatrix}$$

(i) what is the cipher text corresponding to plain text **HI** and also the encryption decryption process

Solution: $K = \begin{bmatrix} 3 & 7 \\ 15 & 12 \end{bmatrix}$

Encryption ***c = p K mod 26***

$$[7 \quad 8] \begin{bmatrix} 3 & 7 \\ 15 & 12 \end{bmatrix} \text{ mod } 26 = [(7*3)+(8*15) \quad (7*7)+(8*12)] \text{ mod } 26$$

= [141 145] mod 26

= [11 15]

= [L P]

Polyalphabetic Ciphers: Hill Cipher

- **Consider a hill cipher with m = 2 (block size = 2) with key K shown below**

$$K = \begin{bmatrix} 3 & 7 \\ 15 & 12 \end{bmatrix}$$

(i) what is the cipher text corresponding to plain text **HI**

Solution: $K = \begin{bmatrix} 3 & 7 \\ 15 & 12 \end{bmatrix}$

Decryption $p = c\,K^{-1} \bmod 26$

$$K^{-1} = \frac{1}{|k|}\, adj\ k$$

$$|k| = \begin{bmatrix} 3 & 7 \\ 15 & 12 \end{bmatrix} = (3*12) - (7*12) = -69$$

$$\text{adj k} = \begin{bmatrix} 12 & -7 \\ -15 & 3 \end{bmatrix}$$

Polyalphabetic Ciphers: Hill Cipher

$K^{-1} = \frac{1}{|k|} adj\ k$

$= \frac{1}{-69}\begin{bmatrix} 12 & -7 \\ -15 & 3 \end{bmatrix} = 3\begin{bmatrix} 12 & 19 \\ 11 & 3 \end{bmatrix} = \begin{bmatrix} 36 & 57 \\ 33 & 9 \end{bmatrix}$ mod 26

$K^{-1} = \begin{bmatrix} 36 & 57 \\ 33 & 9 \end{bmatrix}$ mod 26 $= \begin{bmatrix} 10 & 5 \\ 7 & 9 \end{bmatrix}$

p = c K^{-1} mod 26

$= [11\ \ 15]\begin{bmatrix} 10 & 5 \\ 7 & 9 \end{bmatrix}$ mod 26

$= [(11 * 10) + (15 * 7) \quad (11 * 5) + (15 * 9)]$ mod 26

$= [215 \quad 190]$ mod 26

= [7 8]

= [H I]

Elementary Transposition Ciphers

• The transposition ciphers shuffles, rearranges or permutes the character in a block of plain text.

Plain Text : *Begin Operation at Noon*

$$\begin{bmatrix} b & e & g & i \\ n & o & p & e \\ r & a & t & i \\ o & n & a & t \\ n & o & o & n \end{bmatrix}$$

(width of the rows and permutations of columns defined by keyword)

Rearrange the rows 1 -> 3, 2 - > 5, 3 - > 2, 4 - > 1, 5 -> 4

Rearrange the columns 1 -> 4, 2 - > 3, 3 - > 1, 4 - > 2

Resulting matrix

$$\begin{bmatrix} o & n & a & t \\ r & a & t & i \\ b & e & g & i \\ n & o & o & n \\ n & o & p & e \end{bmatrix} = \begin{bmatrix} a & t & n & o \\ t & i & a & r \\ g & i & e & b \\ o & n & o & n \\ p & e & o & n \end{bmatrix}$$

Cipher text : A T N O T I A R G I E B O N O N P E O N

Other Cipher Properties: Confusion and Diffusion

Confusion

- Confusion means that each binary digit (bit) of the ciphertext should depend on several parts of the key, obscuring the connections between the two.

- The property of confusion hides the relationship between the ciphertext and the key.

- This property makes it difficult to find the key from the ciphertext and if a single bit in a key is changed, most or all the bits in the ciphertext will be affected.

- Confusion increases the ambiguity of ciphertext and it is used by both block and stream cipher.

Other Cipher Properties: Diffusion

Diffusion

- Diffusion means that if we change a single bit of the plaintext, then half of the bits in the ciphertext should change, and similarly, if we change one bit of the ciphertext, then approximately one half of the plaintext bits should change.

- Since a bit can have only two states, when they are all re-evaluated and changed from one seemingly random position to another, half of the bits will have changed state.

- The idea of diffusion is to hide the relationship between the ciphertext and the plain text.

- This will make it hard for an attacker who tries to find out the plain text

Other Cipher Properties: Block and Stream Cipher

- Both **Block Cipher** and **Stream Cipher** are belongs to the symmetric key cipher. These block cipher and stream cipher are the methods used for converting the plain text into cipher text

Block Cipher	**Stream Cipher**
Block Cipher Converts the plain text into cipher text by taking plain text's block at a time.	Stream Cipher Converts the plaint text into cipher text by taking 1 byte of plain text at a time.
Block cipher uses either 64 bits or more than 64 bits	While stream cipher uses 8 bits
Block cipher Uses confusion as well as diffusion.	While stream cipher uses only confusion
In block cipher, reverse encrypted text is hard.	While in stream cipher, reverse encrypted text is easy.
The algorithm modes which are used in block cipher are: ECB (Electronic Code Book) and CBC (Cipher Block Chaining).	The algorithm modes which are used in stream cipher are: CFB (Cipher Feedback) and OFB (Output Feedback).

Secret Key Cryptography

Secret-key cryptography refers to cryptographic system that uses the **same key** to encrypt and decrypt data.

Hence this from also referred as **symmetric key cryptography**.

There are two types of Secret key ciphers –

1. Stream Cipher
2. Block cipher

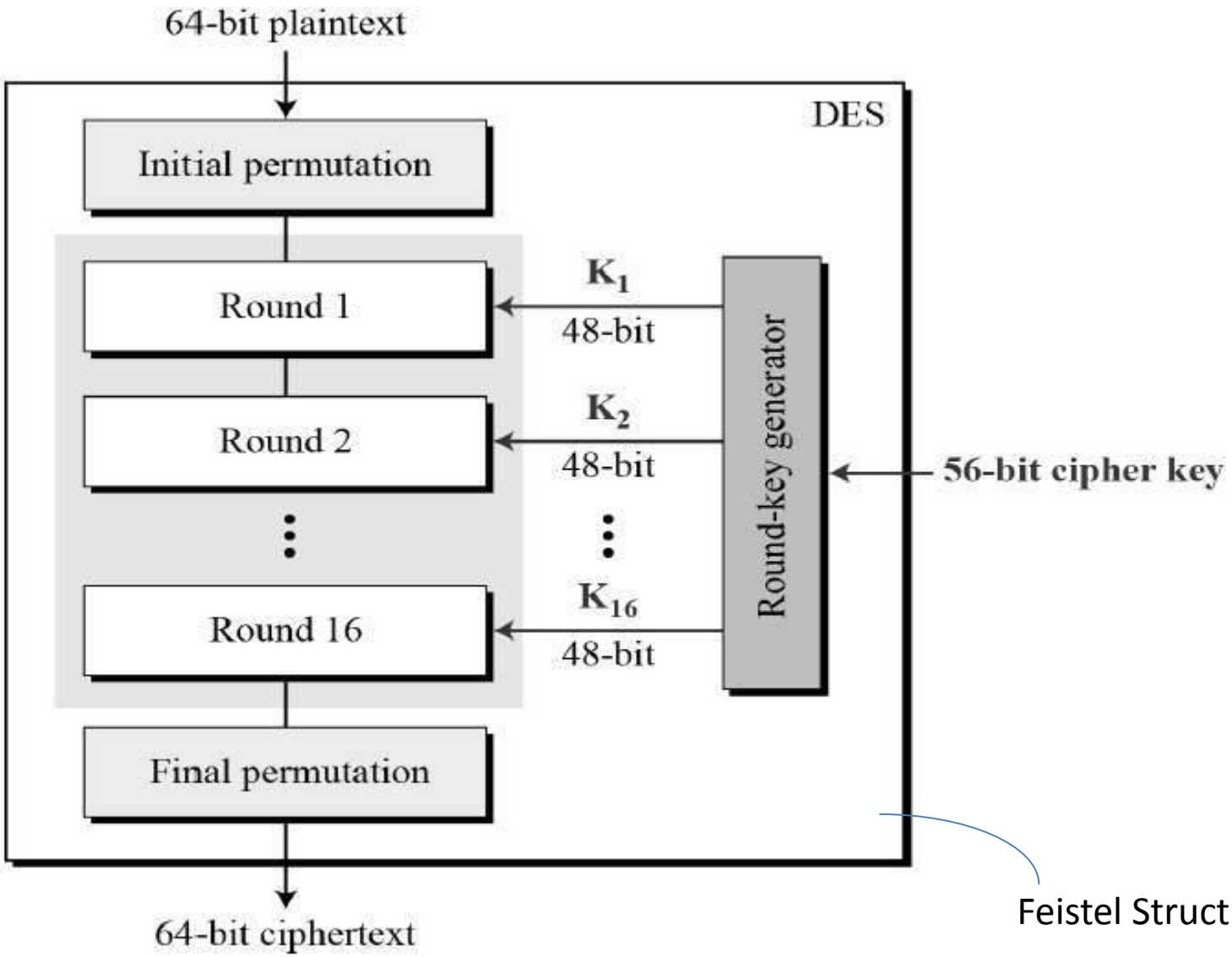
Data encryption standard (DES) is one of the most widely used block ciphers for secret key cryptography

DES (Data Encryption Standard) Algorithm

DES is a block **cipher** algorithm that takes plain text is processed into cipher text by number of rounds.

- Block Size - 64 bits
- No. of rounds – 16 rounds
- Key size - 64 bit
- No. of sub-keys – 16 sub-key
- Sub-key size – 48 bit sub-key
- Cipher Text – 64 bit Cipher Text.

DES (Data Encryption Standard) Algorithm



DES (Data Encryption Standard) Algorithm

Broad-level steps in DES.

- In the first step, the 64 bit plain text block is handed over to an initial Permutation (IP) function.
- The initial permutation performed on plain text.
- Next the initial permutation (IP) produces two halves of the permuted block; says Left Plain Text (LPT) and Right Plain Text (RPT).
- Now each LPT and RPT to go through 16 rounds of encryption process.
- In the end, LPT and RPT are rejoined and a Final Permutation (FP) is performed on the combined block
- The result of this process produces 64 bit cipher text.

DES (Data Encryption Standard) Algorithm

The structure of each DES round is explained below

$L_i = R_{i-1}$

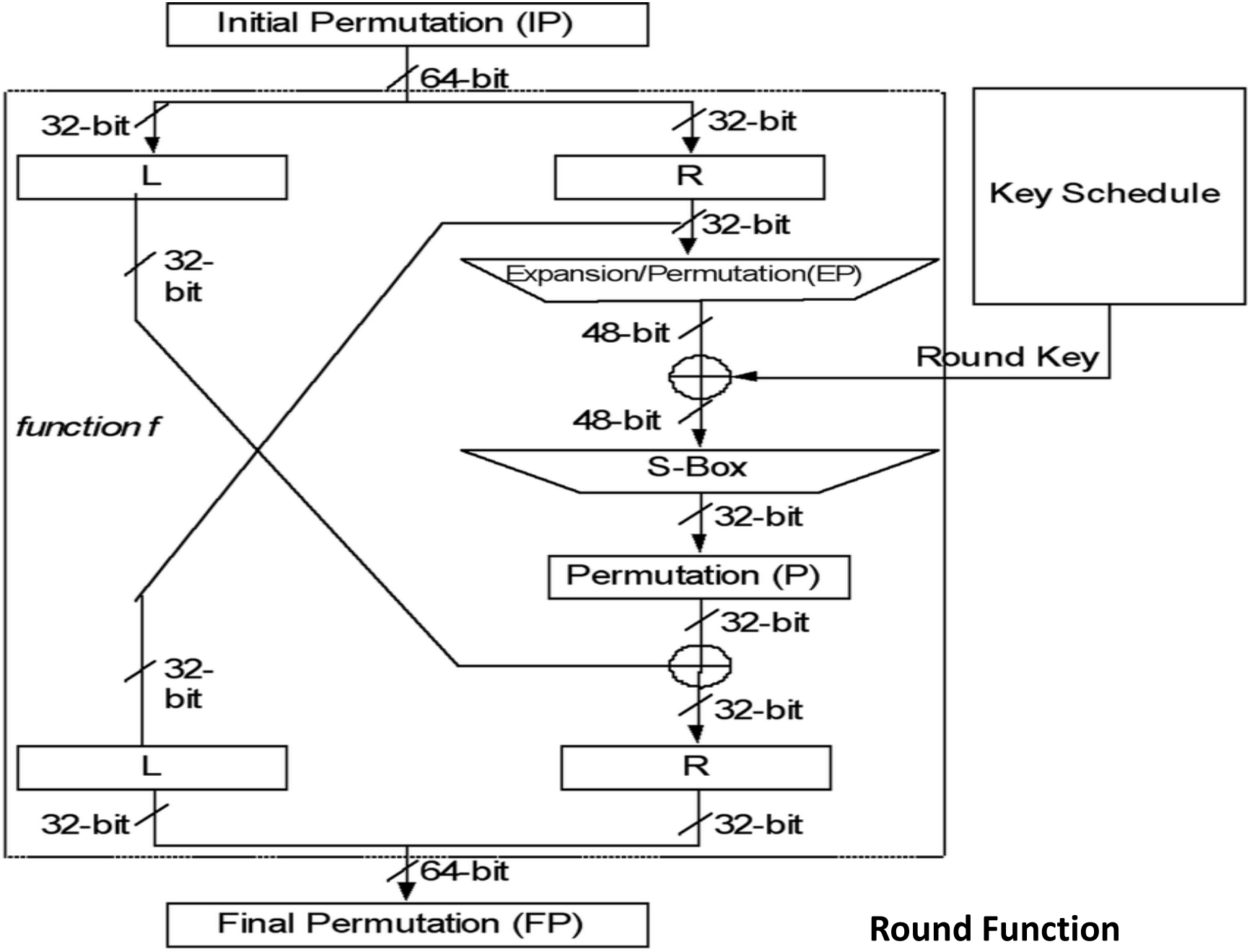
$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$

The function *f* is applied at each round and is referred as the round function. Each round uses round key which is one of inputs to *f*. Each round key is derived from DES key.

Round Function:

The round function involves four operations

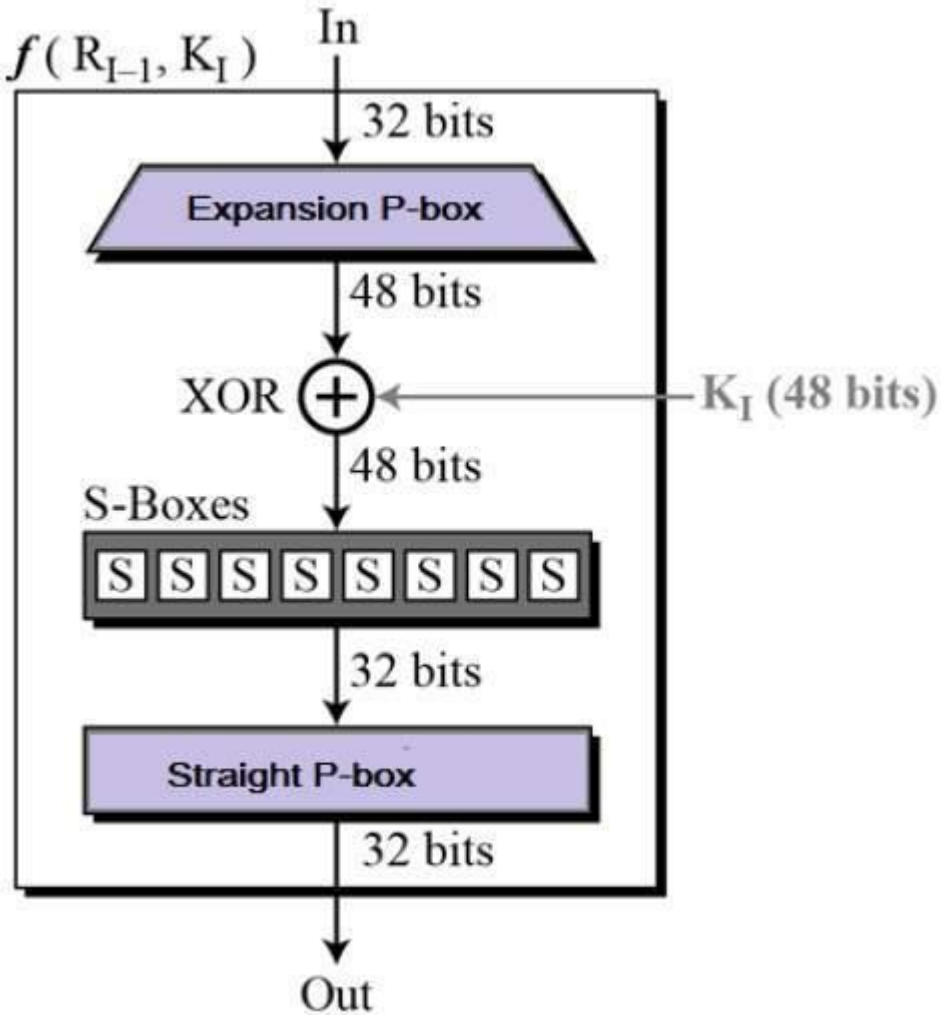
1. Expansion
2. $\oplus$ with round key
3. Substitution
4. Permutation



Round Function

DES (Data Encryption Standard) Algorithm

- The heart of this cipher is the DES function, *f*. The DES function applies a 48-bit key to the rightmost 32 bits to produce a 32-bit output.



Expansion Permutation Box – Since right input is 32-bit and round key is a 48-bit, we first need to expand right input to 48 bits.

XOR – After the expansion permutation, DES does XOR operation on the expanded right section and the round key. The round key is used only in this operation.

Substitution Boxes. – The S-boxes carry out the real mixing (confusion). DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output.

DES (Data Encryption Standard) Algorithm

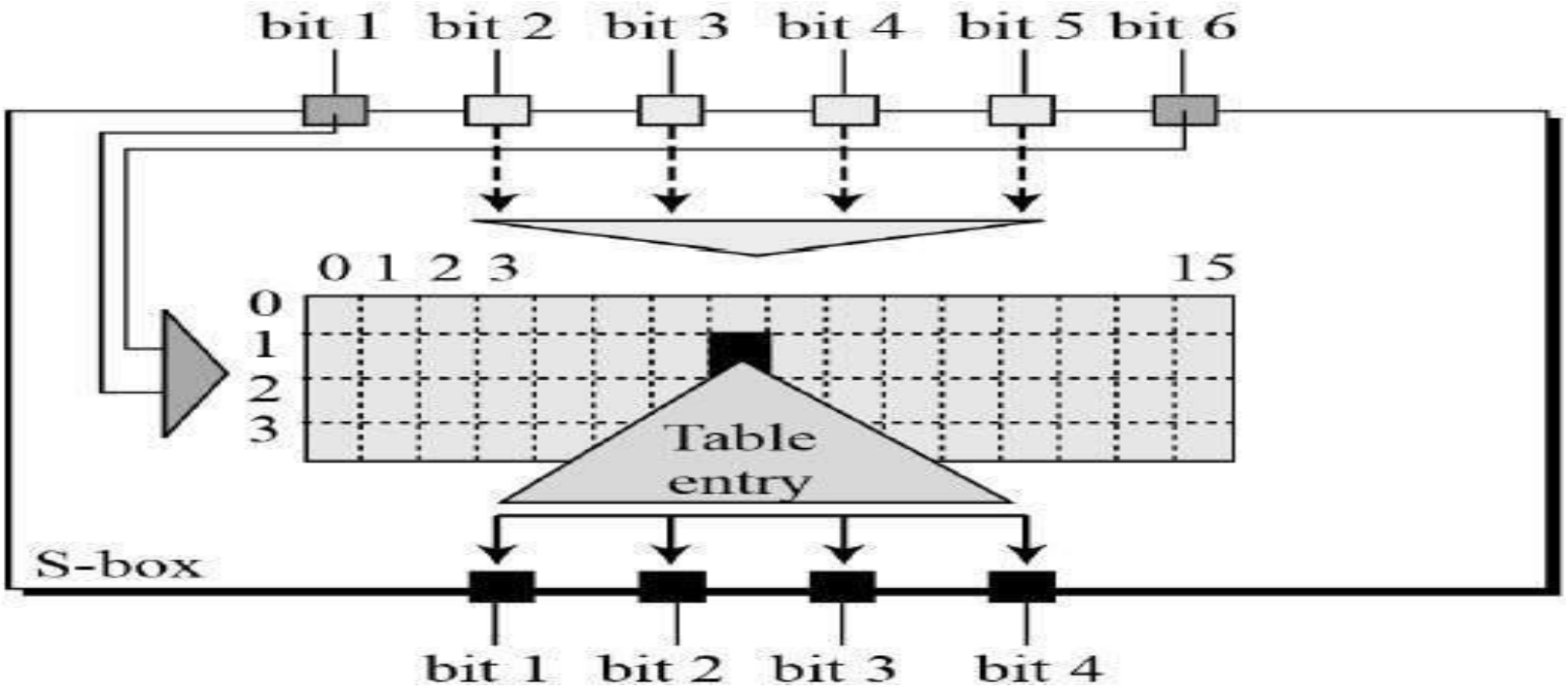
- The input to the round function is $\boldsymbol{R_{i-1}}$ a 32 bit quantity. This first expanded into 48 bits by repeating some bits and interchanging their positions.

- The 48-bit quantity is then $\oplus$ with the round key K_i (Different for each round)

- The result of $\oplus$ operation is divided into eight 6-bit chunks. Each chunk is substituted by 4-bit chunk. A total of 8 different S-boxes provided the eight substitutions.

DES (Data Encryption Standard) Algorithm

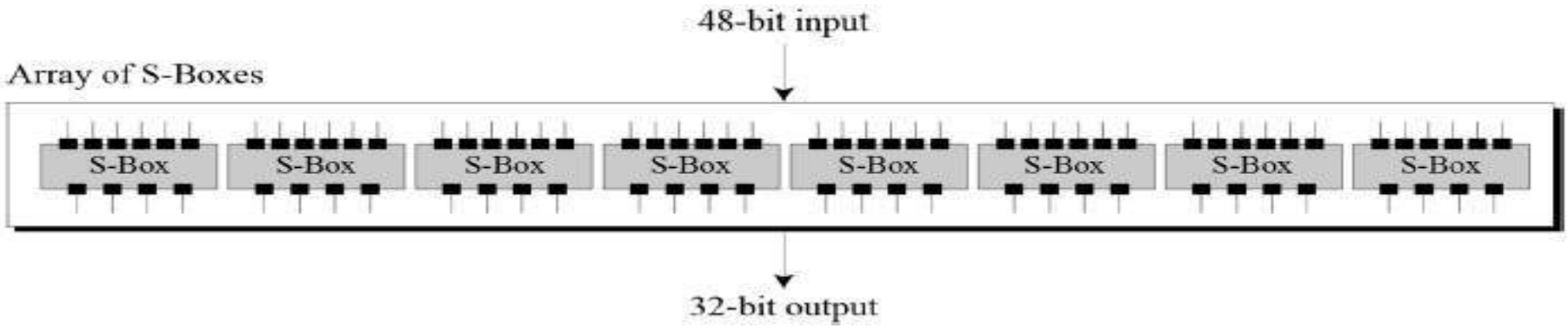
The S-box rule is illustrated below −



The S- box is implemented using 4 x 16 array. Each row of the array is a permutation of the numbers 0 through 15.

Two bit of the i^{th} chunk serve as a row index into the i^{th} table and remaining four bits serves as column index. The o/p of S-box the 4 bit string pointed to by the row and column indices.

DES (Data Encryption Standard) Algorithm



There are a total of eight S-box tables. The output of all eight s-boxes is then combined in to 32 bit section.

Straight Permutation – The 32 bit output of S-boxes is then subjected to the straight permutation

Feistel Structure: Dividing plain text into two equal blocks and performing round function and applying individual keys in each and every round and later swapping of two blocks. If any block cipher algorithm follows all these terms then algorithm is following Feistel structure.

Secret Key Cryptography- Product Cipher

The product cipher combines a sequence of simple transformations such as substitution box, permutation box and modular arithmetic.

- Substitution box (S-box) that takes a binary string of length m and returns binary string of length n.
- An S-box is easily implemented using an array of 2^m rows with row containing an n-bit value.
- A P-box performs a permutation or re-arrangment of the bits in the inputs.
- A P-box and S-box itself is not sufficiently powerful to create secure cipher. By cascading P-boxes and S-boxes alternatively, the strength of a cipher can be increased. Such cipher is called **product cipher.**
- Three operation take place in sequence
  - An operation involving a function of the encryption key
  - A substitution method
  - A permutation

These operations are repeated over many rounds to produce cipher.

Product Cipher

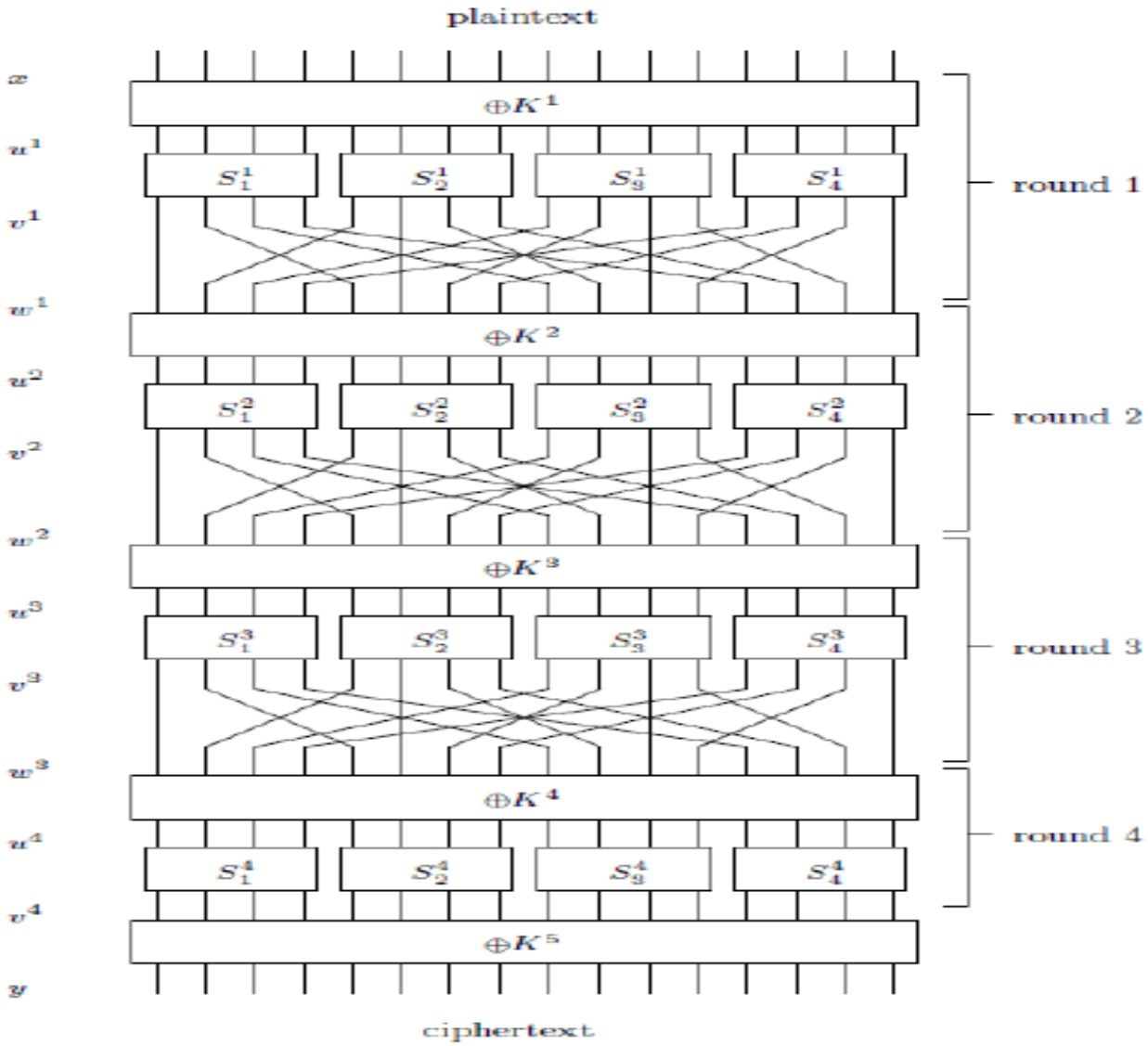


Figure 1: Substitution-permutation-network (SPN)

Module 2

Cryptographic Hash

2.1 INTRODUCTION

- ***Definition:*** A hash function is a deterministic function that maps an input element from a larger (possibly infinite) set to an output element in a much smaller set.
- The input element is mapped to a ***hash value.***
- For example, in a district-level database of residents of that district, an individual's record may be mapped to one of 26 hash buckets.
  - Each hash bucket is labelled by a distinct alphabet corresponding to the first alphabet of a person's name.
- Given a person's name (the input), the output or hash value is simply the first letter of that name (Fig. 7.1).
- Hashes are often used to speed up insertion, deletion, and querying of databases.

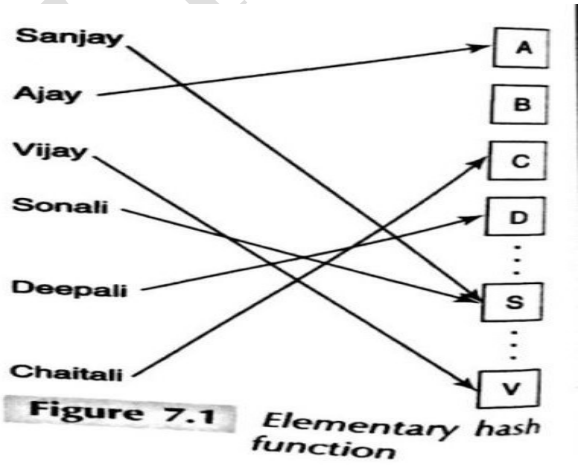


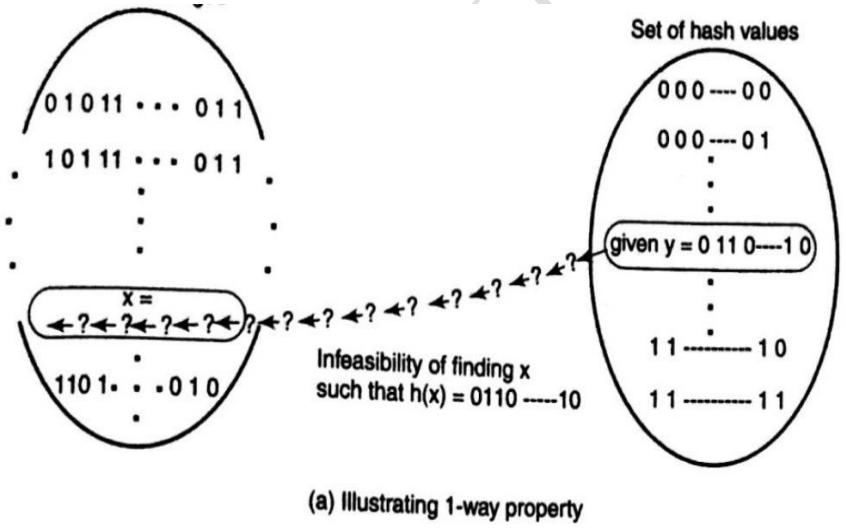
Figure 7.1 Elementary hash function

- In the example above, two names beginning with the same alphabet map to the same hash bucket and result in a collision.

2.2 PROPERTIES

7.2.1 Basics

- A cryptographic hash function, ***h(x),*** maps a binary string of arbitrary length to a fixed length binary string.
- The properties of *h* are as follows:
    1. **One-way property.** Given a hash value, ***y*** (belonging to the range of the hash function), it is computationally infeasible to find an input x such that ***b(x)*** = **y**
    2. ***Weak collision resistance.*** Given an input value x1, it is computationally infeasible to find another input value x2 such that ***h(x1) = h(x2)***
    3. **Strong collision *resistance.*** It is computationally infeasible to find two input values x1 and no x2 such that h(x1)=h(x2)
    4. ***Confusion + diffusion.*** If a single bit in the input string is flipped, then each bit of the hash value is flipped with probability roughly equal to 0.5.



(a) Illustrating 1-way property

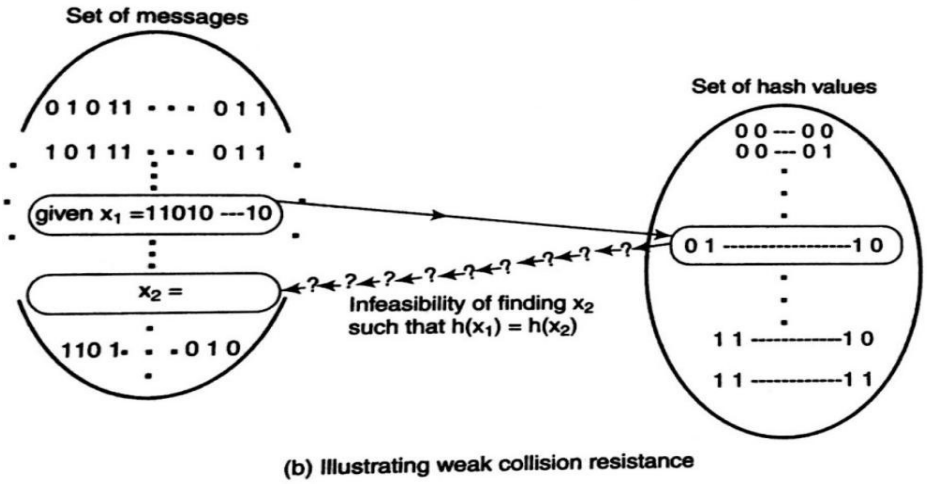


Figure 7.2 Properties of the cryptographic hash

- There is a subtle difference between the two collision resistance properties.
- In the first, the hash designer chooses x1 and challenges anyone to find an x2, which maps to the same hash value as of x1. This is a more specific challenge compared to the one in which the attacker tries to find and x2 such that h(x1)= h(x2).
- In the second challenge, the attacker has the liberty to choose x1.

2.2.2 Attack Complexity

Weak Collision Resistance

- How low long would it take to find an input, x, that hashes to a given value y?
- Assume that the hash value is w bits long. So, the total number of possible hash values is 2^w
- brute force attempt to obtain x would be to loop through the following operations

```
do
{
    generate a random string, x'
    compute h(x')
}
while (h(x') != y)
return (x')
```

- assuming that any given string is equally likely to map to any one of the 2^W hash values, it follows that the above loop would have to run, on the average, 2^{w-1} times before finding an x' such that h(x') = y.

- A similar loop could be used to find a string, x2, that has the same hash value as a given string x1.

Strong Collision Resistance

- A Brute-force attack on strong collision-resistance of a hash function involves looping through the program in Fig. 7.4.
- Unlike the program that attacks weak collision resistance, this program terminates when the hash of a newly chosen random string collides with any of the previously computed hash values.

```
// S is the set of (input string, hash value) pairs
// encountered so far

notFound = true
while ( notFound )
{
    generate a random string, x'
    search for a pair ( x, y) in S where x = x'
     if ( no such pair exists in S )
     {
         compute y' = h(x')
         search for a pair (x, y) in S where y = y'
         if ( no such pair exists in S )
            insert (x', y') into S
          else
              notFound = false
     }
}
return  ( x and x' )  // these are two strings that have
                      // the same hash value
```

Figure 7.4:program to attack strong collision resistance.

THE BIRTHDAY ANALOGY

- Attacking strong collision resistance is analogous to answering the following:
- "What is the minimum number of persons required so that the probability of two or more in the, group having the same birthday is greater than 1/2 ?"
- It is known that in a class of only 23 random individuals, there is a greater than 50% chance that: the birthdays of at least two persons coincide (a "Birthday Collision").
- This statement is referred, to as the Birthday Paradox.

THE BIRTHDAY ATTACK

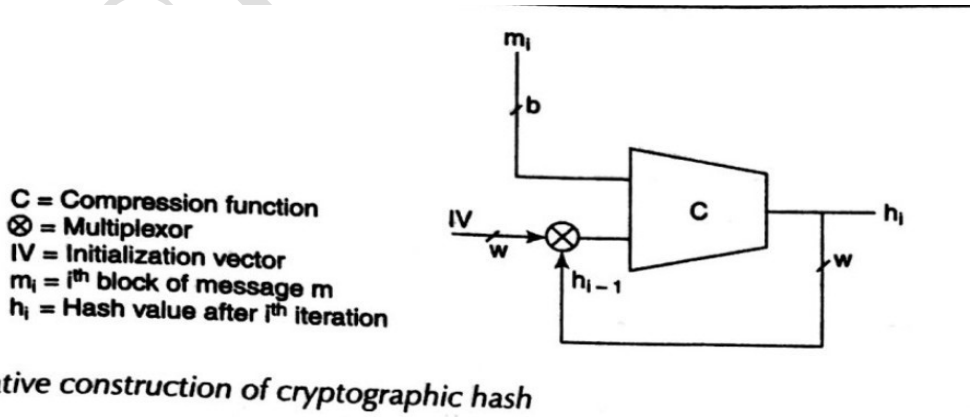
- The following idea, first proposed by Yuval illustrates the danger in choosing hash lengths less than 128 bits.
- A malicious individual, Malloc, wishes to forge the signature of his victim, Alka, on a fake document, F.
- F could, for example, assert that Alka owes Malloc several million rupees.
- Malloc does the following:
  1. He creates millions of documents, Fl, F2,……Fm, etc. that are, for all practical purposes, "clones" of F.
  2. This is accomplished by leaving an extra space between two words, etc.
  3. If there are 300 words in F, there are 2300 ways in which extra spaces may be left between words.
  4. He computes the hashes, h(F1), h(F2), . . . h(Fm) of each of these documents.
  5. He creates an innocuous document, D — one that most people would not hesitate to sign. (For example, it could espouse an environmental cause relating to conservation of forests.)
  6. He creates millions of "clones" of D in the same way he cloned F above.
  7. Let D1, D2, ... be the cloned documents of D.
  8. He computes the hashes, h(D1), h(D2), . . . h(Dm) of each of the cloned documents.
  9. Malloc asks Alka to sign the document D, and Alka obliges.
  10. Later Malloc accuses Alka of signing the fraudulent document
  11. the digital signature is obtained by encrypting the hash value of the document using the private key of the signer.
  12. Thus, Alka's signature on Dj, is the same as that on Fi,.
  13. Hence, at a later point in time, Malloc can use Alka's signature on Dj), to claim that she signed the fraudulent document, F.

2.3 CONSTRUCTION

2.3.1 Generic Cryptographic Hash

- The input to a cryptographic hash function is often a message or document.
- To accommodate inputs of arbitrary length, most hash functions (including the commonly used MD-5 and SHA-1) use iterative construction as shown in Fig. 7.5.
- **C is a compression box.**
- It accepts two binary strings of lengths **b and w** and produces an output string of **length w.**
- Here, **b is the block size and w is the width of the digest**.
- During the first iteration, it accepts a pre-defined initialization vector (IV), while the top input is the first block of the message.
- In subsequent iterations, the ***"partial hash output" is fed back*** as the second input to the C-box.
- The top input is derived from successive blocks of the message.
- This is repeated until all the blocks of the message have been processed.
- The above operation is summarized below:
- **h, = C (IV, m_1)** for first block of message
- **hi = C (h_{i-1}.m_i)** for all subsequent blocks of the message

\



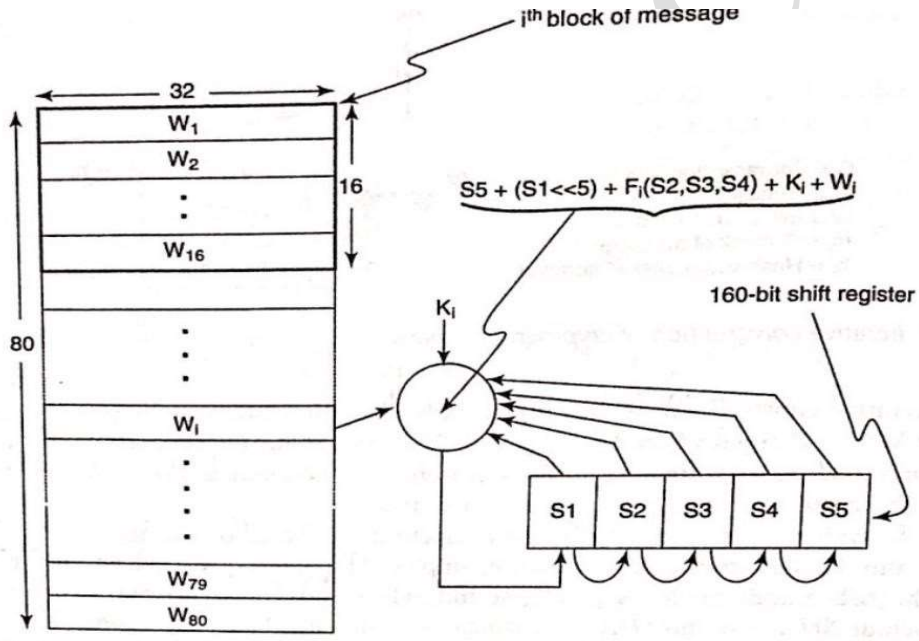
7.5 *Iterative construction of cryptographic hash*

Figure 7.5 Iterative construction of cryptographic hash

- The above iterative construction of the cryptographic hash function is a simplified version of that proposed by **Merkle and Damgard.**
- It has the property that if the compression function is collision-resultant, then the resulting hash function is also collision-resultant.
- MD-5 and SHA-1 are the best known examples. MD-5 is a 128-bit hash, while SHA-1 is a 160-bit hash.

2.3.2 Case Study: SHA-1

- SHA-1 uses the iterative hash construction of Fig. 7.5.



gure 7.6 *Computation of SHA-1*

```
initialize the shift register, S1 S2 S3 S4 S5
for each block of the (message + pad + length field) {
        create the 80-word array [using Eq. (7.2)]
        for i = 1 to 80 {
                temp ← S5 + (S1 << 5) + Fi(S2, S3, S4) + Ki + Wi
                S5 ← S4
                S4 ← S3
                S3 ← S2 >> 2
                S2 ← S1
                S1 ← temp
        }
}
```

F_i is defined b

$$F_i(S2, S3, S4) = (S2 \wedge S3) \vee (\sim S2 \wedge S4), \quad 1 \le i \le 20$$
$$F_i(S2, S3, S4) = S2 \oplus S3 \oplus S4, \quad 21 \le i \le 40$$
$$F_i(S2, S3, S4) = (S2 \wedge S3) \vee (S2 \wedge S4) \vee (S3 \wedge S4), \quad 41 \le i \le 60$$
$$F_i(S2, S3, S4) = S2 \oplus S3 \oplus S4 \quad 61 \le i \le 80$$

- The message is split into blocks of ***size 512 bits***.
- The length of the message, expressed in binary as a 64 bit number, is appended to the message.
- Between the end of the message and the length field, a pad is inserted so that the length of the **(message + pad + 64)** is a ***multiple of 512,*** the block size.
- The pad has the form: 1 followed by the required number of 0's.

Array Initialization

- Each block is split into 16 words, each 32 bits wide.
- These **16 words** populate the first 16 positions, W1, W2 ……W16, of an array of **80 words.**

- The remaining **64 words** are obtained from :

$$W_i = W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16} \quad 16 < i \leq 80$$

- This array of words is shown in Fig. 7.6.

Hash Computation in SHA 1

- A 160-bit shift register is used to compute the intermediate hash values (Fig. 7.6).
- It is initialized to a fixed pre-determined value at the start of the hash computation.
- We use the notation S1, S2, S3, S4, and S5 to denote the five 32 -bit words making up the shift register.
- The bits of the shift register are then mangled together with each of the words of the array in turn.
- The mangling is achieved using a combination of the following Boolean operations: **+, v, ~, ^, XOR ROTATE.**

2.4 APPLICATIONS AND PERFORMANCE

2.4.1 Hash-based MAC

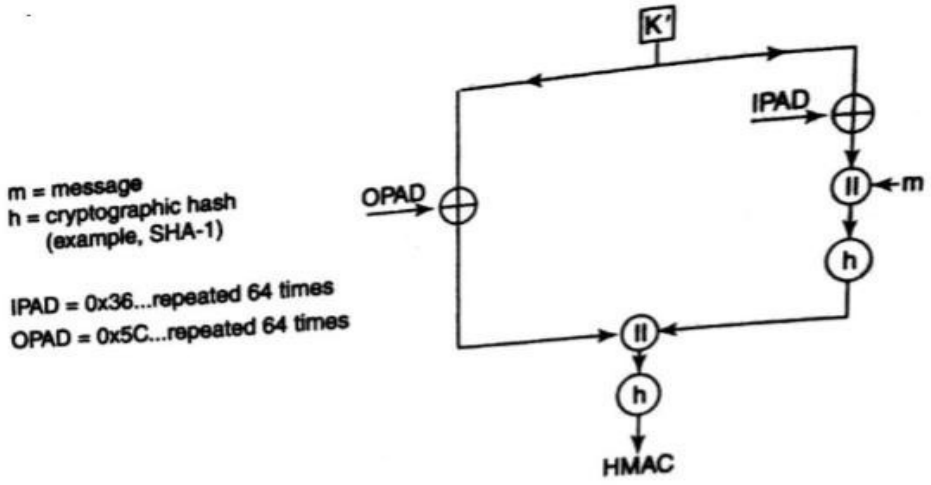
MAC

- MAC is used as a message integrity check as well as to provide message authentication.
- It makes use of a common shared secret, k, between two communicating parties.
- The hash-based MAC that we now introduce is an alternative to the CBC-MAC.
- The cryptographic hash applied on a message creates a digest or digital fingerprint of that message.
- Suppose that a sender and receiver share a secret, k.
- If the message and secret are concatenated and a hash taken on this string, then the hash value becomes a fingerprint of the combination of the message, m and the secret, k.
- ***MAC = h (m|| k)***
- The MAC is much more than just a ***checksum*** on a message.
- It is computed by the sender, appended to the message, and sent across to the receiver.

- On receipt of the **message + MAC,** the receiver performs the computation using the common secret and the received message.
- It checks to see whether the MAC computed by it matches the received MAC.
- A change of even a single bit in the message or MAC will result in a mismatch between the computed MAC and the received MAC.
- In the event of a match, the receiver concludes the following:
- ***(a) The sender of the message is the same entity it shares the secret with — thus the MAC provides source authentication.***
- ***(b) The message has not been corrupted or tampered with in transit — thus the MAC provides verification of message integrity***.
- ***Drawbacks:***
- An attacker might obtain one or more message—MAC pairs in an attempt to determine the MAC secret.
- First, if the hash function is one-way, then it is not feasible for an attacker to deduce the input to the hash function that generated the MAC and thus recover the secret.
- If the hash function is collision-resistant, then it is virtually impossible for an attacker to suitably modify a message so that the modified message and the original both map to the same MAC value.

HMAC

- There are other ways of computing the hash MAC other than this method using HMAC .
- Another possibility is to use key itself as the Initialization Vector (IV) instead of concatenating it with the message.
- Bellare, Canetti, and Krawczyk proposed the HMAC and showed that their scheme is re against a number of subtle attacks on the simple hash-based MAC.
- Figure 7.7 shows how an HMAC is computed given a key and a message.



7.7 Computation of an HMAC

- The ***key is padded with O's*** (if necessary) to form a ***64-byte string*** denoted ***K'*** and ***XORed with a constant*** (denoted IPAD).
- It is then concatenated with the message and a hash is performed on the result.
- ***K' is also XORed*** with **another constant (denoted OPAD)** after which it is prepended to the output of the first hash.
- Once again hash is then computed to yield the HMAC.
- As shown in Fig. 7.7, HMAC performs an extra hash computation but provides greatly enhanced security.

2.4.2 Digital Signatures

- The same secret that is used to generate a MAC on a message is the one that is used to verify the MAC.
- Thus the MAC secret should be known by both parties - the party that generates the MAC and the party that verifies it.
- A digital signature, on the other hand, uses a secret that only the signer is privy to.
- An example of such a secret is the signer's private key.
- A crude example of an RSA signature by A on message, m, is $\mathbf{E_{A.pr}(m)}$
- where A.pr is A's private key.
- The use of the signer's private key is a fundamental aspect of signature generation.
- Hence, a message sent together with the sender's signature guarantees not just integrity and authentication but also non-repudiation, i.e., the signer of a document

cannot later deny having signed it since she alone has knowledge or access to her private key used for signing.

- The verifier needs to perform only a public key operation on the digital signature (using the signer's public key) and a hash on the message.
- The verifier concludes that the signature is authentic if the results of these two operations tally,

$$E_{A.pu}\left(E_{A.pr}(h(m))\right) \stackrel{?}{=} h(m)$$

Question Bank (module 2-chapter 2)

1. Explain generic hash computation and HMAC .
2. Define hashing Explain the properties of hashing with a neat figure.
3. Explain SHA-1 computation with a neat illustration.
4. Explain weak and strong collision resistance.
5. Explain digital signature.
6. Explain birthday analogy and birt

Chethana.C, Dept. of CSE, BMSIT&M

25/3/2020

Discrete logarithm

* Let us consider the multiplicate group $(Z_p^*, \cdot)$
  ↳ p is prime

* Let g is a generator of the group
  i.e $g^1 \bmod p, g^2 \bmod p, \ldots\ldots g^{p-1} \bmod p$
  is a rearrangement of the integers in Z_p^*

* Let $p=29$ & $g=2$, $x \in \{0, 1, \ldots p-2\}$.

$y = g^x \bmod p$ ⟹ modular exponentiation with base g & modulus p.

$x = \log_g y \pmod p$ ⟹ Inverse operation expressed as shown is referred to as Discrete logarithm.

given ⟹ $p, y, g \in Z_p^*$

x	$y = g^x \bmod p$
1	$2^1 \bmod 29 = 2$
2	$2^2 \bmod 29 = 4$
3	$2^3 \bmod 29 = 8$
4	$2^4 \bmod 29 = 16$
5	$2^5 \bmod 29 = 3$
6	$2^6 \bmod 29 = 6$
7	$2^7 \bmod 29 = 12$
8	$2^8 \bmod 29 = 24$
9	$2^9 \bmod 29 = 19$
10	$2^{10} \bmod 29 = 9$
11	$2^{11} \bmod 29 = 18$
12	$2^{12} \bmod 29 = 7$
13	$2^{13} \bmod 29 = 14$
14	$2^{14} \bmod 29 = 28$
15	$2^{15} \bmod 29 = 27$
16	$2^{16} \bmod 29 = 25$
17	$2^{17} \bmod 29 = 21$
18	$2^{18} \bmod 29 = 13$
19	$2^{19} \bmod 29 = 26$
20	$2^{20} \bmod 29 = 23$
21	$2^{21} \bmod 29 = 17$
22	$2^{22} \bmod 29 = 5$
23	$2^{23} \bmod 29 = 10$
24	$2^{24} \bmod 29 = 20$
25	$2^{25} \bmod 29 = 11$
26	$2^{26} \bmod 29 = 22$
27	$2^{27} \bmod 29 = 15$

⟹ using $g=2$, $p=29$ we have, $x = 1 \ldots 28$ generated elements of the group $(Z_p^*, \cdot) = (Z_{29}^*, \cdot)$

now next we are rearranging of value of y in ascending order & list value of x with respect to that

Brute-force algorithm: to compute the discrete logarithm is to

a) prepare a table which pairs each x, $0 \le x \le p-2$, with the corresponding $g^x \bmod p$.

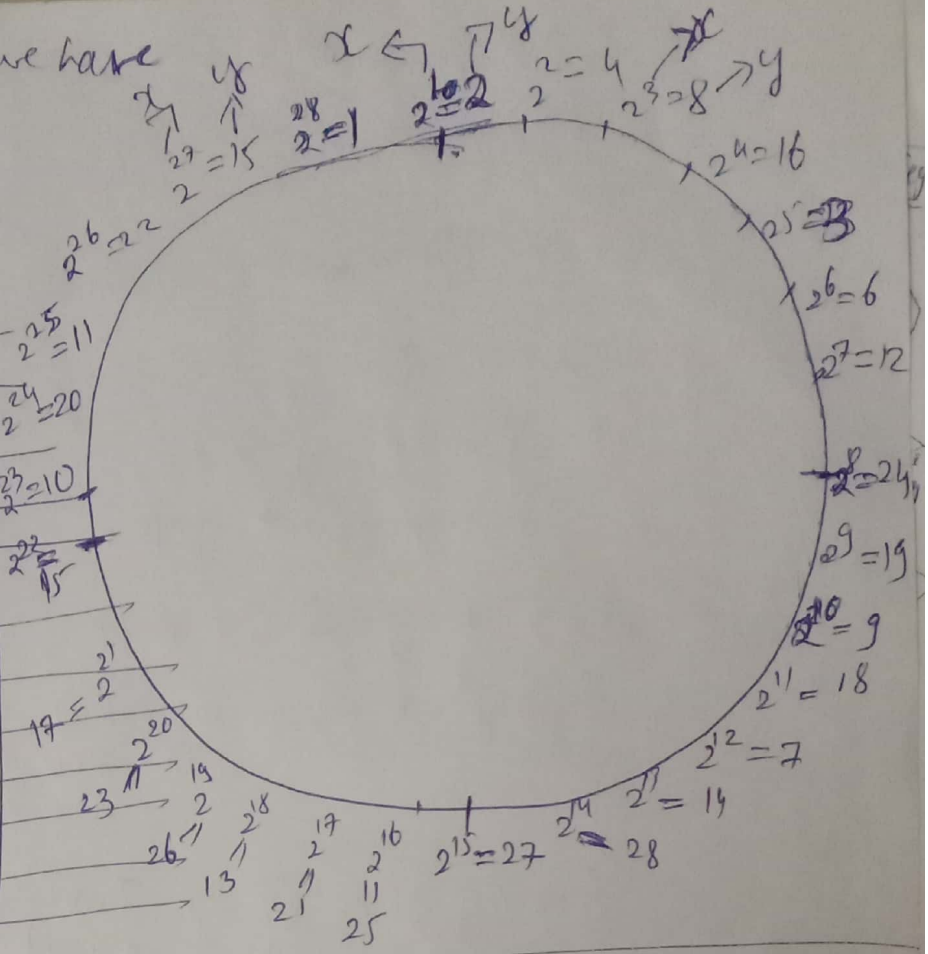
b) we then sort the table on the $g^x \bmod p$ column.

c) Given a value we use it as an index into $g^x \bmod p$ column.

d) The derived discrete logarithm is simply the value in the other column of the same row.

Using Brute force we have

y	$x = \log_2 y \pmod{29}$
1	28
2	1
3	5
4	2
5	22
6	6
7	12
8	3
9	10
10	23
11	25
12	7
13	18
14	13
15	27
16	4
17	21
18	11
19	9
20	24
21	17
22	26
23	20
24	8
25	16
26	19
27	15
28	14

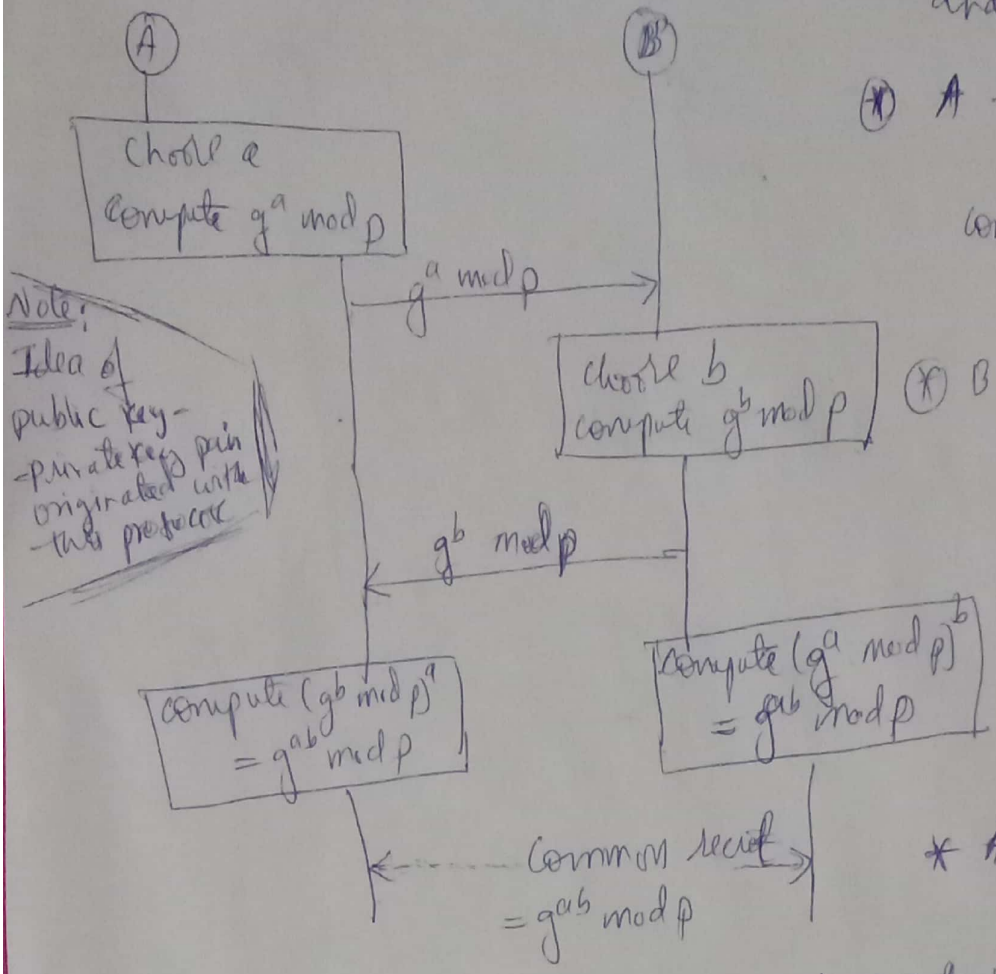


Note: Using Square and multiply strategy the time to compute $g^x \bmod p$ is reduced from a polynomial in p to a polynomial in $\log p$

* The computation cost of the table approach is proportional to p.
* For large p (say 2000 bit p) constructing and storing such a large table is infeasible.

* Other algorithms to speed up the discrete logarithm problem are pollard-rho algorithm & index calculus methods.
Time complexity of these methods is $O\sqrt{p}$ → still prohibitive

Diffie - Hellman Key exchange



Note:
Idea of public key - private key pair originated with this protocol

* A & B knows the base g and modulus p

(A) A → chooses integer a, private key
$1 < a < p-1$, computes partial key (public key)
$g^a \bmod p$ → sends to B

(B) B → choose integer b, private key
$1 < b < p-1$, computes partial key → public key
$g^b \bmod p$ → sends to A

B also computes
$(g^a \bmod p)^b \bmod p$
$= g^{ab} \bmod p$

* A computes $(g^b \bmod p)^a \bmod p$
$= g^{ab} \bmod p$

* Now both A & B shares common secret key

Example Let $p = 131$, $g = 2$

Let A chooses $a = 24$, partial key is $g^a \bmod p = 2^{24} \bmod 131 = 46$ → send to B
B chooses $b = 17$, partial key is $g^b \bmod p = 2^{17} \bmod 131 = 72$ → send to A

A computes $(g^b \bmod p)^a = 72^{24} \bmod 131 = 13$ → shared secret

B computes $(g^a \bmod p)^b = 46^{17} \bmod 131 = 13$

Computational Diffie-Hellman problem: with the knowledge of partial keys $g^a \bmod p$ & $g^b \bmod p$ and public parameters p & g eavesdropper deduce the common secret $g^{ab} \bmod p$, derived by A and B.

He tries to obtain a & b from $g^a \bmod p$ and $g^b \bmod p$

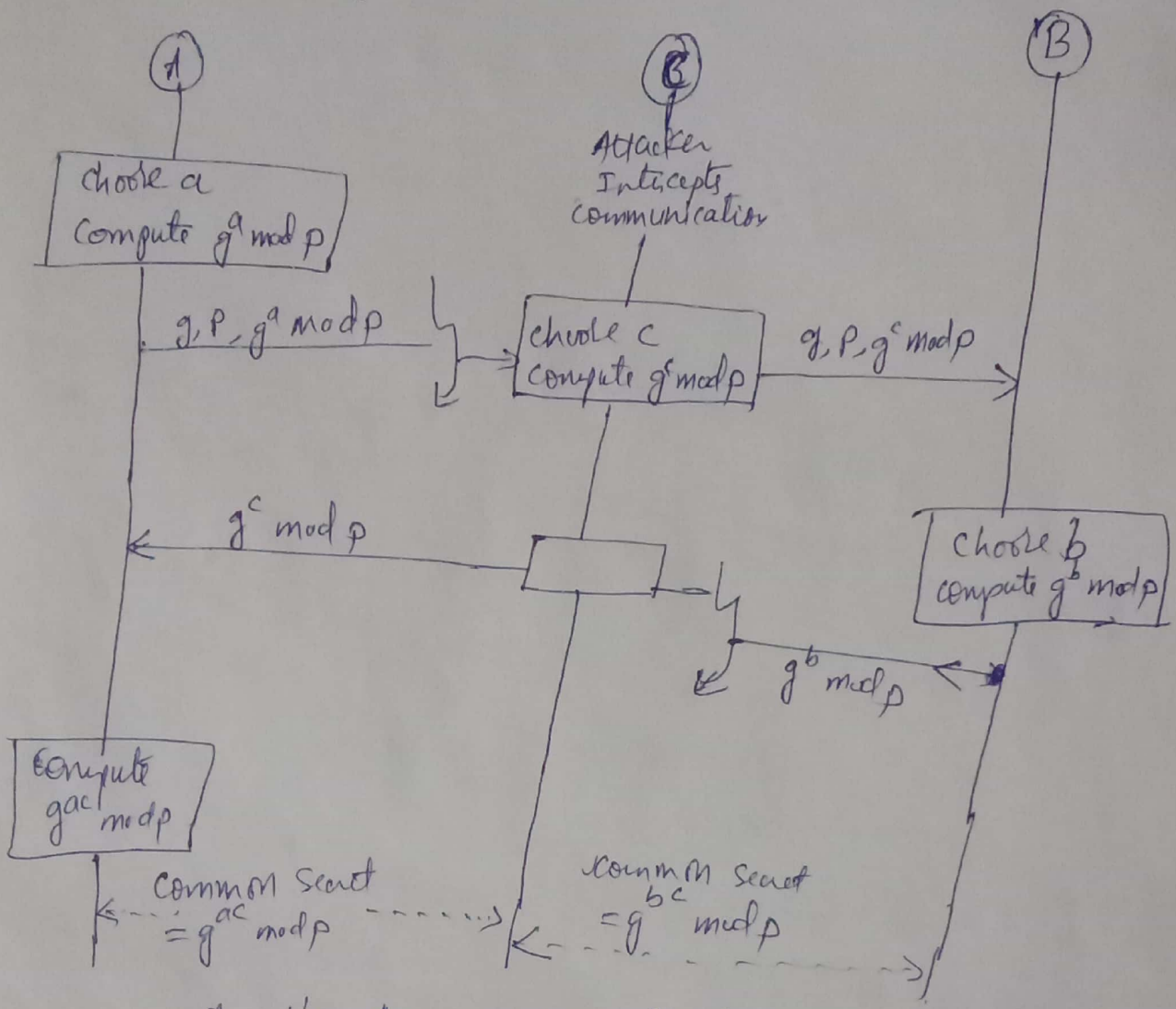
⇒ This is computationally infeasible in the case where p is sufficiently large, because this entails computing the discrete logarithm

Chethana C
Dept of CSE
2SD23CS026

26/3/2020

Figure:

Attacks : Easily attacked : Man in the middle attack



-> C intercepts A's message to B } by substituting with $g^c \bmod p$
-> C intercepts B's message to A }

-> After message transfer, B computes $(g^c \bmod p)^b \bmod p = g^{bc} \bmod p$
, A computes $(g^c \bmod p)^a \bmod p = g^{ac} \bmod p$

-> C, also computes the two secrets
$(g^a \bmod p)^c \bmod p = g^{ac} \bmod p$ & $(g^b \bmod p)^c \bmod p = g^{bc} \bmod p$

-> Every subsequent message encrypted by A & intended for B can be decrypted by C by re encrypted with the secret shared by B & C

-> Vice versa every message from B to A can be decrypted & modified by C

Note: This problem/attack since both A and B neglected to authenticate the source of the partial keys that had received.

Solution: By sending partial keys together with an RSA signature on it.

Chithara.C, Dept of CSE

Choice of Diffie-Hellman parameters

modulus p → prime, g → generator of Z_p^* ?

- → private keys a & b
- → shared secret $g^{ab} \bmod p$
- → To minimize the chance of an attacker guessing any of these secrets, it is desirable that they be chosen (or computed) from a large pool of integers
- → No. of distinct private keys & distinct public keys are both upperbounded by the cardinality of Z_p^*
  - → p-1 for p, prime
  - → is < p-1 for non prime p.
  - → desirable to have large p as possible
  - → choice of size of p is a tradeoff b/w security & performance

g → generator of Z_p^*

→ The space of possible public keys is large (= to p-1)

Small sub group attack

Let p = 29, g = 2
Let A chooses a = 7
public key: $2^7 \bmod 29 = 12$
$2^{14} \bmod 29 = 28$
$2^{21} \bmod 29 = 17$
$2^{28} \bmod 29 = 1$ ...
Sub group of Z_{29}^*
{12, 28, 17, 1}
∴ Common secret key is restricted to an element within the above set, regardless of value of B's private key

$b = 4*i + j$, $j = b \bmod 4$
Secret key by A and B
$g^{ab} \bmod 28 = g^{7(4i+j)} \bmod 29$
$= g^{28i+7j} \bmod 29$
$= (g^{28})^i \bmod 29) * (g^{7j} \bmod 29)$
$\therefore g^{28} \bmod 29 = 1$ by Fermat's theorem
$g^{7j} \bmod 29 \in$ {12, 28, 17, 1} for any value B's private key
* Secret key ∈ "small subgroup" can be easily guessed.
⇒ small subgroup attack

attacker can eavesdrop on the partial secrets exchanged $g^a \bmod p$ & $g^b \bmod p$ & attempt to compute
{$g^a \bmod p, g^{2a} \bmod p, g^{3a} \bmod p$ ...}
{$g^b \bmod p, g^{2b} \bmod p$, ...}
and launch man-in-middle attack

Solution: perform test on private keys a, b chosen by A and B. If test fails, new private key should be chosen.

chethana.C, Dept of CSE 27/03/2020

module-3

Key Management

↳ Related to generation, storage, distribution, and backup of keys

→ In this chapter we are focusing on public key-private key pairs.

How does A knows B's public key

Possibility 1: A may frequently communicate with B in a secure fashion, so he may already have B's public key

→ B would have shared to a securely

↳ if B's private key is compromised, then messages can be decrypted with old public keys.

→ if B's public key-private key pair is changed, then the new public key will swiftly and securely communicated to A. — not always practical.

Possibility 2: centralized directory: to maintain entity's public key.

if A wants to communicate with an e-commerce website B-mart she will obtain B-mart's public key by querying the directory first

questions: who will maintain such a directory

→ scalability problem with directory

→ Infrastructure to support directory could as become bottleneck — leads to spoofing & DOS attacks.

→ non-uniqueness of names: persons with same name → which public key

Possibility 3: A receives a document signed by a trusted source C, containing B's public key
↳ This approach appears to obviate the need for an on-line, centralized directory-site.

Digital certificates : Signed document used to bind a public key to the identity of a person

Identity
- → his/her name
- → national identification no
- → Email & postal address
- → etc

Certification Authority (CA): trusted entity that issues certificates
→ usually government agencies or banks
→ certificates may be issued to individuals, to organizations or even to routers.

Types of certificates
→ Applied through regular e-mail, with the applicant stating his/her public key, name, e-mail address etc
→ CA requires no credentials from the applicant.
↳ assumes that the applicant is in possession of the (uncompromised) private key corresponding to the public key contained in the application received via e-mail.
→ but for the verifier of such a certificate should realize that above certificates are "Trust at your own risk certificate".
⇒ Appropriate for applications with the least demands on security.

(2) To carry more weight

CA can perform identity verification of the applicant:
→ requires credentials such as passport / driver's license
his/her employer, email address etc.

→ This work would be delegated by the CA to a Registration Authority (RA)

→ The role of an RA may be performed by
↳ bank – has existing relationship with the customer

(3) * CA/RA may require the applicant to demonstrate possession of the private key corresponding to the public key presented.

⇒ applicant is asked to decrypt a random string encrypted with the applicant's stated public key.

→ price of such a certificate is higher than the basic certificate

→ Owner can use his/her certificate for more security-demanding applications

3.2.2 X.509 Digital Certificate Format

- X.509 is an ITU standard specifying the format for **public key certificates.**
- The fields of an X.509 certificate together with their meaning are as follows:

1. **Certificate Serial Number and Version:** Each certificate issued by a given CA will have a unique number.

2. **Issuer information:** The distinguished name of an entity includes his/her/its "common name," e-mail address, organization, country, etc.

3. **Certificate signature and associated signing algorithm information:** It is necessary to verify the **authenticity** of the certificate. For this purpose, it is signed by the issuer. So, the certificate should include the **issuer's digital signature** and also the **algorithm** used for signing the certificate.

4. **Validity period:** There are two date fields that specify the ***start date and end date*** between which **the certificate is valid.**

5. **Subject information:** This includes the **distinguished** name of the **certificate's subject or owner.**

 For example, if a customer intends to communicate with an e-commerce web server at **www.B-Mart.com**, then the customer's browser will request **B-Mart's certificate**.

Client-side software will check whether the "**Common Name**" in B-Mart's certificate tallies with **B-Mart's domain name**.

 Other information, such as the subject's country, state, and organization, may be included.

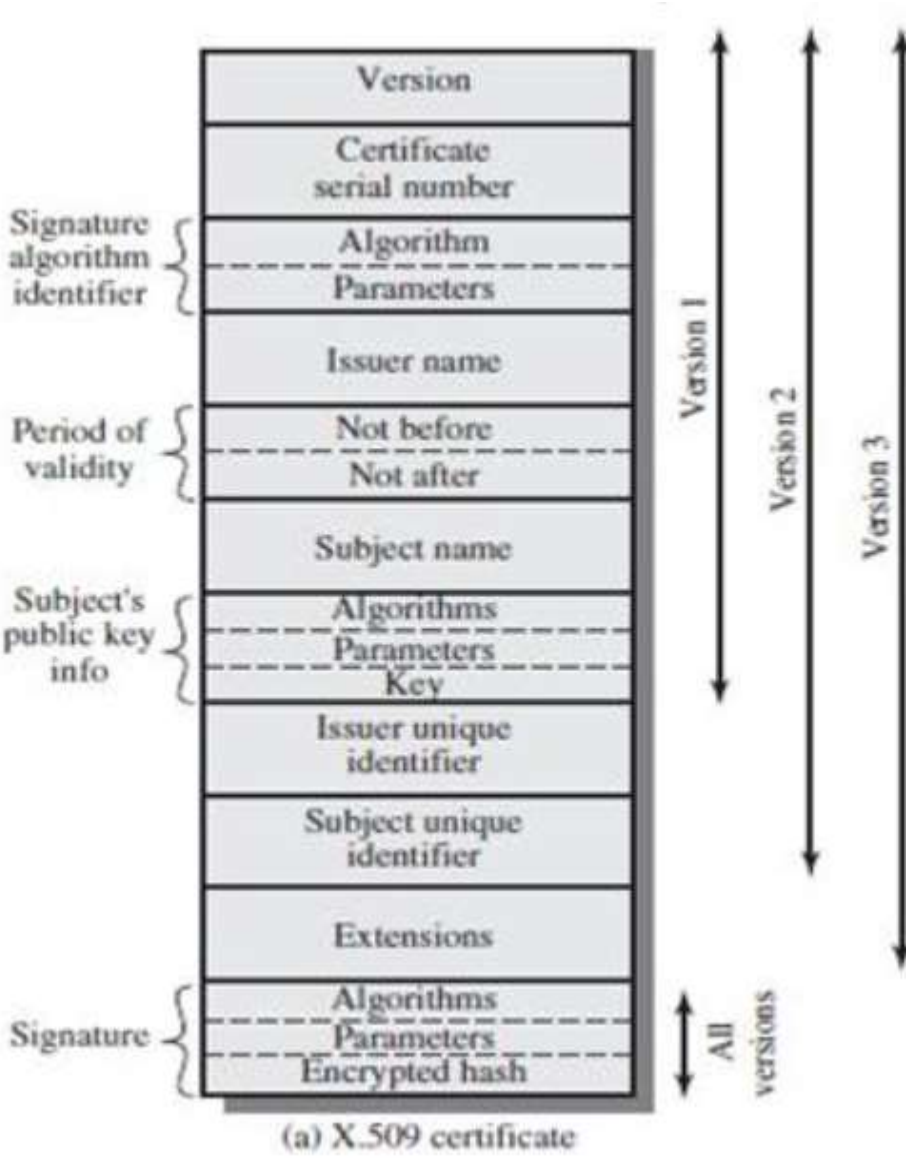
6. **Subject's public key information:** The **public key**, **the public key algorithm** (e.g., RSA or DSA), and the **public key parameters** (modulus in the case of RSA and modulus + generator in case of Diffie-Hellman).

```
Certificate:
    Data:
        Version: 1
        Serial Number: 3174
        Signature Algorithm: md5WithRSAEncryption

        Issuer: C=IN, ST=Maharashtra, L=Mumbai, O=Security Centre,
                OU=Enterprise Security,
                CN=Security Centre CA/emailAddress=agni@SecCent.com.in
        Validity
            Not Before: Jan  1 10:00:00 2010 GMT
            Not After : Dec  31 10:00:00 2010 GMT

        Subject: C=IN, ST=Goa, L=Madgaon, O=Ocean Enterprises,
                 OU=IT Consulting,
                 CN=Ocean emailAddress=zuari@ocean.com.in
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:3b:77:a0:9b:91:28:06:34:9d:54:f2:72:3e:83:
                    fd:a7:26:51:c3:b1:03:cf:bb:27:1e:52:0e:14:2c:
                    16:6a:20:fe:16:12:44:ba:75:eb:e8:1c:9c:5b:66:
                    cb:1e:2d:81:3f:d0:29:c3:ee:f8:17:49:91:b1:3b:
                    a9:0b:95:27:cb:f5:8d:73:10:9c:d8:26:3b:1e:55:
                    3b:12:74:bb:4c:51:b4:ef:92:4a:28:1c:40:da:38:
                    d3:8c:41:88:ba:9d:62:f1:8d:d2:96:b8:e2:b4:22:
                    48:7c:d1:33:90:46:b7:e3:02:bf:1f:77:0a:7b:19:
                    1b:d3:82:0a:1f:f3:80:36:71:
                Exponent: 65537 (0x10001)
    Signature Algorithm: md5WithRSAEncryption
        03:b8:99:fc:2b:71:49:0e:2b:a5:93:50:1d:0f:e2:68:a1:b3:
        9c:88:34:d0:a1:e6:f8:39:27:67:bb:29:0c:7e:47:92:a3:dd:
        f3:7f:1b:02:59:c3:f7:c1:ad:49:c4:47:a9:87:e8:f2:73:44:
        9c:5d:b2:38:74:59:0c:84:5e:1f:9d:4c:26:86:25:ad:28:74:
        13:7d:3b:82:fb:01:ac:8a:3e:45:9a:2b:98:65:ab:3c:72:91:
        27:80:37:b6:cb:a3:44:73:26:8c:b0:8e:f8:91:25:7a:33:53:
        7c:91:33:fc:0f:65:c9:25:11:3b:79:c0:a4:20:c9:42:8a:69:
        32:0b
```

Figure 10.1 *A digital certificate*



(a) X.509 certificate

3.2.3 Digital Certificates in Action

- Assume that A needs to securely ***transmit a session key*** to B. So, she encrypts it with B's public key.
  - A will need to retrieve the public key from B's certificate.
  - A may already have B's certificate or she may send a message to B requesting it.
  - There are a number of checks that A will have to perform on B's certificate prior to using **B's public key.**

- Is this indeed B's certificate?
- This can be determined by checking whether the certificate contains B's name. But the "common name" field alone may be inadequate (since there are probably many John Browns, for example).

- It may be necessary to check other fields in the certificate such as the **subject's web page URL or e-mail address**.
- A should check if the certificate is still valid. Since the **validity period** is contained in the certificate, this is easily done.
- Finally, the certificate must be **signed by a CA or RA.**
- A should verify the signature contained in the certificate.
- A requires the **CA's public key** for signature verification.
- The CA may be globally known or may be known to the community that A and B belong.
- In this case **A** has access to **the CA's public key**.

3.3 PUBLIC KEY INFRASTRUCTURE

3.3.1 FUNCTIONS OF A PKI

- A public key infrastructure includes the **CA's, the physical infrastructure (**encryption technologies, hardwareetc.), and the formulation and enforcement of policies/procedure.
- It includes the following services:
  i. **Certificate creation,issuance,storage and archival**
  ii. **Key generation and key escrow**
  iii. **Certificate/key updation**
  iv. **Certificate revocation**

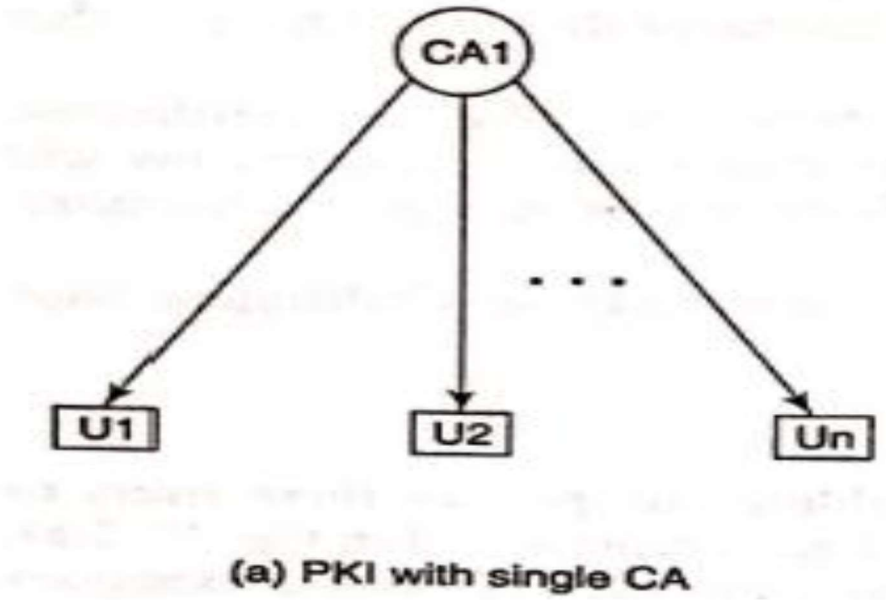
- There are crucial differences in the support required for private keys used for decryption versus those used for signing.
- In the case of encryption/decryption, it is often necessary to have ***a back-up of the decryption key.***
- If not, an employee who looses his decryption key will be unable to decrypt the archives of sensitive data he may have accumulated.
- For this reason, the PKI within an organization, for example, might hold the private keys in **escrow, i.e., they may be securely backed up and made available to the owner or to a trusted authority** (such as a law enforcement agency) under special circumstances.
- On the other hand, there ***is no need to back up a private key used for digital signing.***
- If such a key is lost, the owner could inform the CA or PKI administrator (within an organization).
- He/she could then obtain a new signing key and receive a new certificate carrying the corresponding public key.
- An important function of the PKI is to provide a safe archival facility for all issued certificates.

3.3.2 PKI Architectures

1. PKI with single CA:

 CA1 could issue certificates to multiple users Ul, U2, etc., enabling any pair of these users to communicate securely using certificates exchanged between them.

- This is represented in below Fig.(a).
- Each arc in the figure is a trust relationship.
- For example, the arc from the CA1 to U2 expresses the fact that CA1 vouches for U2's public key in the certificate issued by the CA1 to U2. Such an architecture, however, is not **scalable**.
- There are tens of millions of users who may need certificates. It is not practical for CA1 to issue certificates to all.

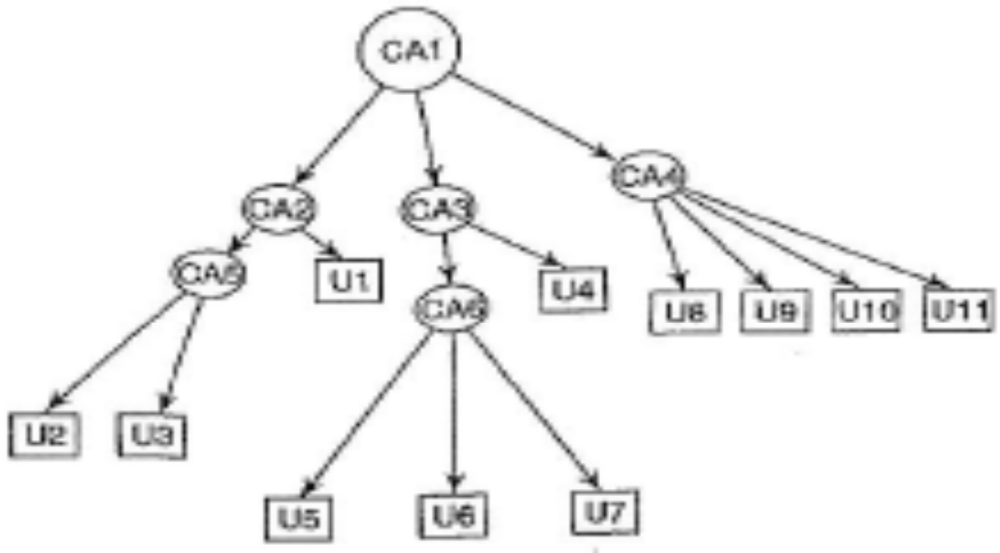


(a) PKI with single CA

2. Hierarchial (tree-based PKI architecture)

- A practical solution to the problem of scalability is to have CA1 certify other CAs who in turn certify other CAs and so on.
- This creates a tree of CAs known as a hierarchical PKI architecture [see below Fig.(b)].
- Here, CA1 issues certificates to CA2, CA3, and CA4.
- CA2 in turn issues certificates to CA5 and end user Ul.
- CA5 issues certificates to users U2 and U3.

- The advantage of this approach is easy **scalability** — each CA is responsible for certifying a limited number of users or other CAs.
- CA1, **the root CA**, is sometimes referred to as the **trust anchor.** every node in the tree will know the root **CA's public key.**
- Suppose U1 in Fig.(b) needs U5's public key.
- U5 would have to provide an entire chain of certificates as follows:
  (1) Certificate signed by CA1 vouching for CA3's public key
  (2) Certificate signed by CA3 vouching for CA6's public key
  (3) Certificate signed by CA6 vouching for U5's public key
- It is assumed that each node has a copy of the root's public key.
- So, upon receiving the above certificate chain, U1 can verify the signature on the first certificate using CA1's (the trust anchor'! public key.



(b) Hierarchical (tree-based) PKI architecture

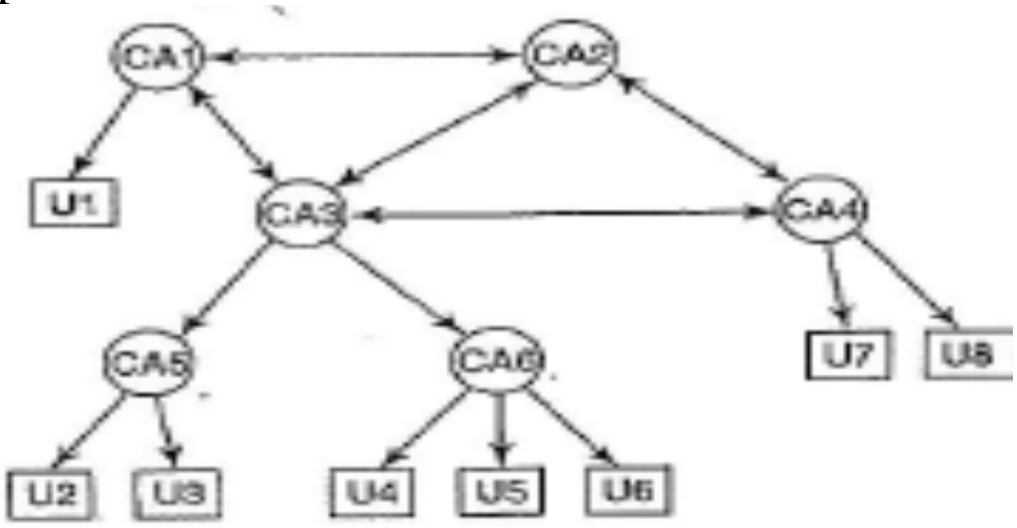
- Public key in the first certificate can be used to verify the signature in the second certificate and so on.

3. Mesh based PKI

- A more dense web of trust is shown in Fig. (c) and is referred to as a **mesh-based PKI.** This could include mutually trusting CAs — CA1 trusting CA2 and. CA2 trusting CA1 shown by a bidirectional arc between CA1 and CA2.
- Unlike in tree based PKI, there may be multiple trust paths between two users.
- Example there could be multiple trust paths between user1 and user 7

  Path 1:CA1,CA3, and CA 4
  Path 2: CA1,CA2, and CA 4.

- Multiple paths provide greater resilience if one or more CAs being compromised.

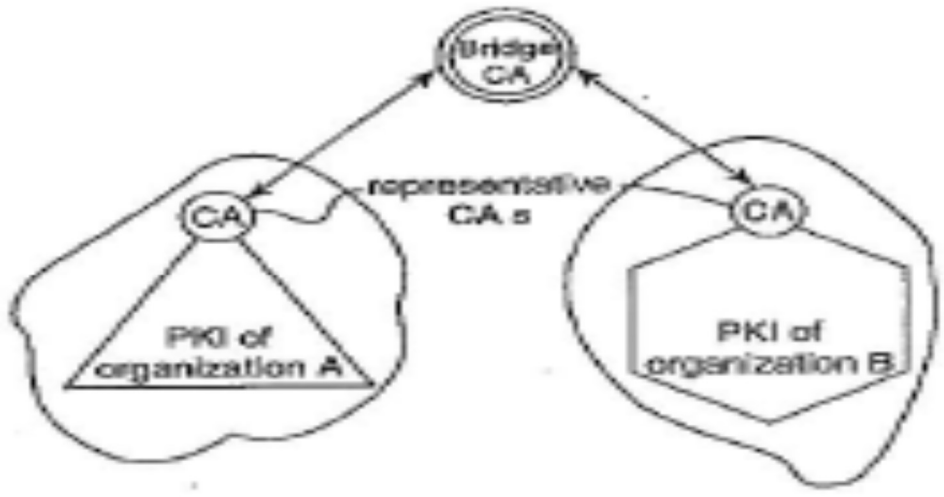


(c) Mesh-based PKI

4. Bridge based PKI

- Another PKI architecture, referred to as ***bridge-based PKI***, is motivated by the need for secure communications between organizations in a business partnership.
- Suppose that the partnering organizations already have their own PKIs.
- **A bridge CA is** introduced that establishes **a trust relationship** with a **representative CA from each organization**.
- This is accomplished by the bridge CA and the organizational representatives issuing certificates to each other.
- The representative CA is one that has a trust path to all (or at least most) of the users in that organization.

Figure 10.2(d) shows a bridge CA that extends the web of trust between two existing organizational PKIs.



(d) Bridge-based PKI

- In the case of organization with a hierarchical PKI the representative CA is the root of the tree.

Note:

- No inter organizational links between two CAS.
- Only trust relationships between the representative CA of each organization and the bridge CA.

3 .3.3 Certificate revocation

Revocation Scenarios

 The validity period of an X.509 certificate is always contained in the certificate.

 However, there are other reasons why a seemingly valid certificate may **actually be invalid.**

Scenario 1: The certificates subject, Prashant, was issued a certificate valid between **Jan 01, 2010, and Dec 31, 2010**.however he quit the organization on **April 1, 2010**.

 Assume that Prashant's certificate is used for key exchange/authentication and that **he has made a copy of it.**

 The **session key** itself is then used to encrypt all messages in both directions for the duration of the ensuing session.

 Generally speaking, it is not legal for Prashant to act on behalf of his company beyond **the date of his resignation**. However, that is precisely what he could do when he attempts to establish official business communication with a customer of his company on say **June 10, 2010.**

 Based on the **expiration date** in Prashant's certificate, the customer would deduce that the certificate was valid.

 Moreover, Prashant would be able to authenticate himself or perform unauthorized decryption since he knows the ***private key***

corresponding to the public key in his certificate. Thus, Prashant might continue to do business on behalf of his company even after **resigning**.

 Based on Scenario 1, we need a mechanism to **revoke a certificate issued by an organization** to an employee when the he leaves or changes roles.

Scenario 2:

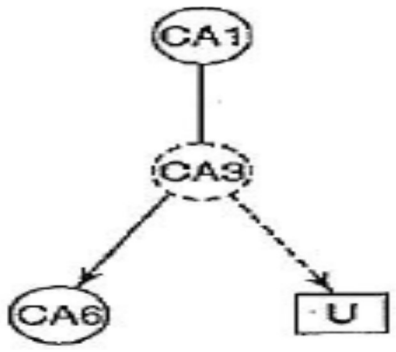


Figure 10.3 *Revocation scenario 2*

 Consider a single chain in a PKI (Fig. 3.3).

 Suppose that the ***private key of CA3*** were compromised.

 An attacker with access to the ***compromised private key*** could then do the following:

- Generate a ***public key, private key pair (X, Y).***
- Create a **certificate** containing the public key X with subject name = **U'.**
- Sign the above certificate using the compromised private key of CA3.

 The attacker has thus created a **fictitious entity U'**, masquerading as a legitimate subject, U (see Fig. 3.3).

 Now the attacker can forge the signature of U on any message by signing with the private key, Y.

The attacker would provide a certificate chain of two certificates — the certificate issued by CA1 vouching for CA3's public key and the above certificate created by him.

 This chain is a valid trust path from the root CA to the subject U.

 Using the public key of CA1 and the certificate chain, the verifier would accept the fraudulent signature generated using Y as an authentic signature of U.

 Scenario 2 is that if a CA's private key is compromised, then any certificate issued by that CA is invalid and it should not be included in any trust path or certificate chain.

Handling Revocation

Solution 1:

 One possible solution to the problem of certificate revocation is to use an ***on-line facility*** that provides information on the ***current status of digital certificates.***

 For this purpose, a ***protocol called On-line Certificate Status Protocol (OCSP)*** is employed.

Solution 2:

 Another proposed solution is to distribute lists of revoked certificates — ***Certificate Revocation Lists (CRLs).*** The frequency of list updation is an important consideration.

 If CRLs are distributed, too frequently, they could consume considerable bandwidth.

 On the other hand, if they were distributed infrequently, information on recently revoked certificates may not reach those who need it in a timely fashion.

Solution 3

 Design a system wherein the signer requires the cooperation of a **Trusted Third Party (TTP)** in generating a signature.

 Both, the signer and the TTP have a part of the private key with neither party knowing the other part.

 To sign a document, the signer would contact the TTP.

 Before requesting to sign , the TTP could check whether the signer's certificate has been revoked and participate only if the signer's certificate has not been revoked.

 Indeed, the TTP may itself maintain **certificate revocation information.**

 The TTP may also act as a **timestamp authority** and certify the time at which the document is signed.

 This may be done, for example, by signing a value obtained by **concatenating a timestamp with the hash of the document.**

FIG 10.4: Summarizes the certificate revocation problem from perspective of signature verification.

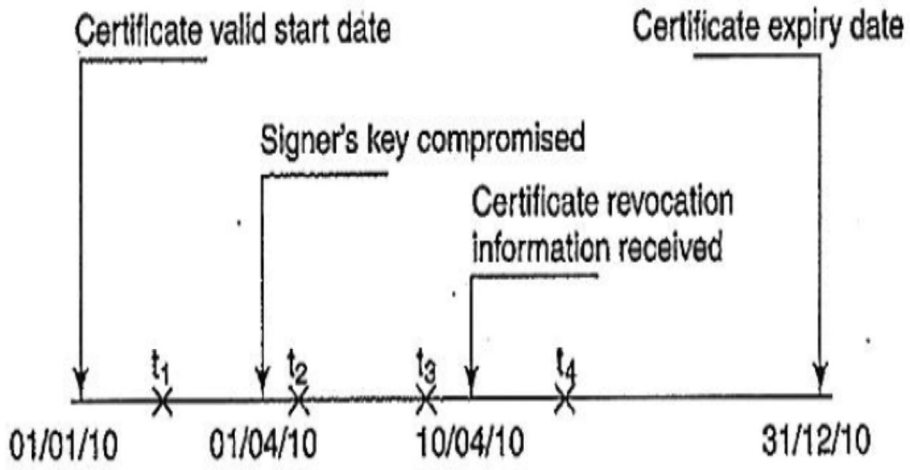


Figure 10.4 *Effect of time lag between key compromise and receipt of certificate revocation information*

Table 10.1 *Effect of certificate revocation lag time on signature acceptance*

Case	Signature generation time	Signature verification time	Does verifier accept signature?	Should verifier accept signature?
1	t_1	t_3	Y	Y
2	t_1	t_4	N	Y
3	t_2	t_3	Y	N
4	t_2	t_4	N	N

Solution: solution 3 that involves TTP at signing time together with a time stamp helps to alleviate the problems identified here.

3.4 IDENTITY-BASED ENCRYPTION

3.4.1 Preliminaries

- The digital certificate is a verifiable way of communicating the public key of a entity .
- Certificates are transmitted along with messages for purposes such ***as authentication, signature verification, and encryption.***
- An alternative to digital certificates emerged in 1984 in the form of **Identity-based Encryption (IBE).**
- Shamir's used a scheme wherein a person's public key could be computed as a function of that **person's unique credential such as his/her e-mail address**. Thus, anyone can reliably compute A's public key only knowing A's e-mail address, for example.
- IBE assumes the use of a **TTP called the Private Key Generator (PKG).**

Here is how a generic IBE scheme works:

- The PKG has a **private key and associated public key parameters.**(K**pr ,public key parameters)**
- To obtain a private key, A informs the PKG that she wishes to receive a private key corresponding to her ID, say alka@iitb.ac.in
- The PKG makes sure that that the credential does indeed belong to A.
- The PKG also makes sure that this ID is universally unique, i.e., there is no other individual with the same credential (in this case alka@iitb.ac.in).

- If so, it generates a **private key for A,** which is a **function of her ID** and the **private key of the PKG**.
- The PKG then securely transmits the private key to A.
- **Disadvantage:**With knowledge of the PKG's public parameters and A's unique ID, anyone can compute A's public key

3.5 Bilinear mapping

 A bilinear mapping ,B(x,y) maps any pair of elements from one given set to an element in a second set.

 The term bilinear follows from the following property mapping:

$$B(k_1 \times u_1 + k_2 \times u_2, v) = k_1 \times B(u_1, v) + k_2 \times B(u_2, v)$$

- Here u1,u2 and v are elements of the first set and k1 and k2 are integer constants.
- An example of dot product of vectors

Let $u = (2, 4, 1)$ and let $v = (5, 3, 2)$.
Then, $(2, 4, 1) \bullet (5, 3, 2)^T = 24$.
We next verify that the dot product is a bilinear operation.
Now let $u_1 = (2, 4, 5), u_2 = (7, 1, 2), k_1 = 3, k_2 = 4$
So,

$$\begin{aligned} k_1 u_1 + k_2 u_2 &= 3(2, 4, 5) + 4(7, 1, 2) \\ &= (6, 12, 15) + (28, 4, 8) \\ &= (34, 16, 23) \end{aligned}$$

So,

$$\begin{aligned} (k_1 u_1 + k_2 u_2) \bullet v &= (34, 16, 23) \bullet (5, 3, 2)^T \\ &= 264 \end{aligned}$$

Next, consider

$$\begin{aligned} k_1 (u_1 \bullet v) + k_2 (u_2 \bullet v) &= 3 ((2, 4, 5) \bullet (5, 3, 2)^T) + 4 ((7, 1, 2) \bullet (5, 3, 2)^T) \\ &= 3 * 32 + 4 * 42 \\ &= 264 \end{aligned}$$

In general,

$$(k_1 u_1 + k_2 u_2) \bullet v = k_1 (u_1 \bullet v) + k_2 (u_2 \bullet v)$$

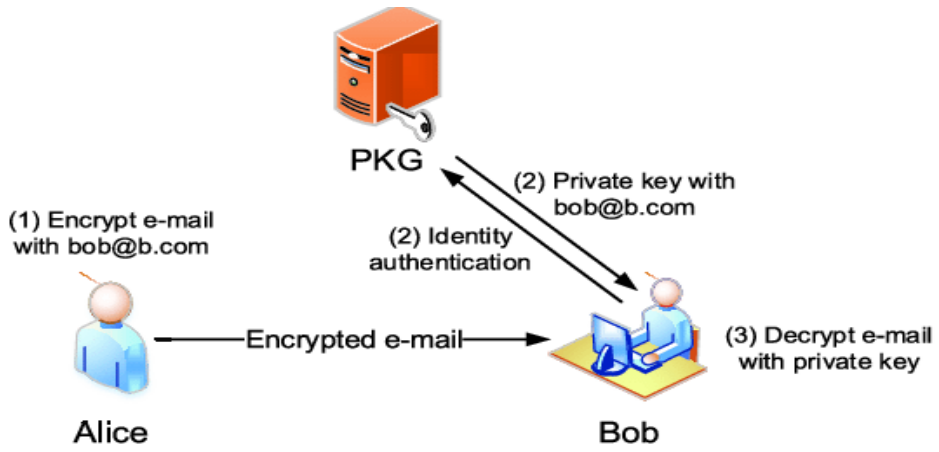
3.4 IDENTITY-BASED ENCRYPTION

3.4.1 Preliminaries

- The digital certificate is a verifiable way of communicating the public key of a entity.
- Certificates are transmitted along with messages for purposes such ***as authentication, signature verification, and encryption.***
- An alternative to digital certificates emerged in 1984 in the form of **Identity-based Encryption (IBE).**
- Shamir's used a scheme wherein a person's public key could be computed as a function of that **person's unique credential such as his/her e-mail address**. Thus, anyone can reliably compute A's public key only knowing A's e-mail address, for example.
- IBE assumes the use of a **TTP called the Private Key Generator (PKG).**

Here is how a generic IBE scheme works:

- The PKG has a **private key and associated public key parameters.**(K**pr ,public key parameters)**
- To obtain a private key, A informs the PKG that she wishes to receive a private key corresponding to her ID, say alka@iitb.ac.in
- The PKG makes sure that that the credential does indeed belong to A.
- The PKG also makes sure that this ID is universally unique, i.e., there is no other individual with the same credential (in this case alka@iitb.ac.in).
- If so, it generates a **private key for A,** which is a **function of her ID** and the **private key of the PKG**.
- The PKG then securely transmits the private key to A.



b. E-mail system based on IBE

https://www.researchgate.net/figure/Identity-based-encryption-scheme-and-pplication_fig4_324071308

- Identity-based encryption is a type of public-key encryption in which a user can generate a public key from a known unique identifier such as an email address), and a trusted third-party server calculates the corresponding private key from the public key.
- In this way, there is no need to distribute public keys ahead of exchanging encrypted data.
- The sender can simply use the unique identifier of the receiver to generate a public key and encrypt the data.
- The receiver can generate the corresponding private key with the help of the trusted third-party server – the private-key generator (PKG).

Disadvantage: With knowledge of the PKG's public parameters and A's unique ID, anyone can compute A's public key

Basics of Bilinear Pairing and use of it to implement IBE

3.5 Bilinear mapping

- A bilinear mapping , B(x,y) maps any pair of elements from one given set to an element in a second set. The term bilinear follows from the following property mapping:

$$B(k_1 \times u_1 + k_2 \times u_2, v) = k_1 \times B(u_1, v) + k_2 \times B(u_2, v)$$

Here u1, u2 and v are elements of the first set and k1 and k2 are integer constants. An example of dot product of vectors as an example of mapping.

Example 10.1

Let $u = (2, 4, 1)$ and let $v = (5, 3, 2)$.
Then, $(2, 4, 1) \bullet (5, 3, 2)^T = 24$.
We next verify that the dot product is a bilinear operation.
Now let $u_1 = (2, 4, 5), u_2 = (7, 1, 2), k_1 = 3, k_2 = 4$

$$\text{So, } k_1 u_1 + k_2 u_2 = 3(2, 4, 5) + 4(7, 1, 2) = (6, 12, 15) + (28, 4, 8) = (34, 16, 23)$$

$$\text{So, } (k_1 u_1 + k_2 u_2) \bullet v = (34, 16, 23) \bullet (5, 3, 2)^T = 264$$

Next, consider

$$k_1 (u_1 \bullet v) + k_2 (u_2 \bullet v) = 3 ((2, 4, 5) \bullet (5, 3, 2)^T) + 4 ((7, 1, 2) \bullet (5, 3, 2)^T) = 3 * 32 + 4 * 42 = 264$$

In general,

$$(k_1 u_1 + k_2 u_2) \bullet v = k_1 (u_1 \bullet v) + k_2 (u_2 \bullet v)$$

s

Bilinear Pairings: First practical scheme that implements IBE

- Use two cyclic groups G and Γ of large prime order p.
- The bilinear pairing $\beta(X, Y)$ maps any pair of elements X and Y form G to an element in Γ.
- G is an additive group has identity 0_G
- Γ is an multiplicative group has identity 1_r
- The property of β is as follows

Bilinearity: For all $X, Y, Z \in G$,

$$\beta(X + Z, Y) = \beta(X, Y) * \beta(Z, Y) \text{ and}$$
$$\beta(X, Y + Z) = \beta(X, Y) * \beta(X, Z)$$

Non-degeneracy: For a given $X \in G$, $\beta(X, Y) = 1_\Gamma$ for all $Y \in G$ if and only if $X = 0_G$.

Ease of computability: There exist efficient algorithms to compute $\beta(X, Y)$ for all $X, Y \in G$.

- Once the group $\mathbb{G}$, Γ and the bilinear pairing β are decided, the PKG proceeds to set up its public parameters.
- Thereafter the clients may request the PKGto generate private keys on their behalf

PKG Parameter Setup

- PKG chooses a generator Ꝑ of the group G.
- Also chooses its private key- random integer, k belongs to $Z_{p.}$
- PKG chooses two kinds has functions.
  - The first hash function i maps a person's ID (ex: email address of arbitrary length.) to an element in G.
  - The second hash function μ maps an element in r to an l-bit binary string. l is the length of message block.
- The two groupf G, r, their order p, the generator **Ꝑ,** PKG public key, k,bilinear mapping β two hash functions i, μ all are publicly known.

Use public and private key Generator

- Let the clients ID be $ID_{A.}$
- The client contacts the PKG and request a private key based on her ID, ID_A
- The PKG verifies the actual owener of the ID_A.
- If so PKG computes the public and Private pair of the requesteras follows

Public Key: $\mathcal{A} = i(ID_A)$ (10.1)

Private Key: $\alpha = k\,\mathcal{A}$ (10.2)

The PKG communicates A and α to A (offline channel)

Encryption: Suppose B wishes to send a message to A

- B requests the public key parameters of the PKG if he does not already have them.
- We assumes B knows ID_A id of A.
- To encrypt an l-bit message ,m to A, Bdoes the following things

.

ss

- B chooses a random number, $r \in Z_p$.
- B computes the following:

$$C_1 = rP \quad (10.3)$$

$$C_2 = m \oplus \mu(\beta(A, rK)) \quad (10.4)$$

- The pair, (C_1, C_2), is the ciphertext. B communicates the ciphertext to A.

Decryption

To recover the plaintext, A uses her private key, α, to compute

$$\begin{aligned}
& C_2 \oplus \mu(\beta(\alpha, C_1)) \\
&= C_2 \oplus \mu(\beta(kA, rP)) && \text{from Eqs 10.2 and 10.3} \\
&= C_2 \oplus \mu(\beta(A, krP)) && \text{from bilinearity property of } \beta \\
&= C_2 \oplus \mu(\beta(A, r(kP))) \\
&= C_2 \oplus \mu(\beta(A, rK)) && \text{from definition of PKG's public key} \\
&= m && \text{from Eq. 10.4}
\end{aligned}$$

Thus, A is able to recover the plaintext using her private key.

Authentication

It is an process in which a principal proves that he/she/it is the entity it claims to be.

 The principal is referred to as the ***prover,*** while the party to whom proof is submitted identity verification is called the ***verifier***.

 Authentication may be based **on what the principal knows** (e.g., a password or a passphrase) or has (an identity card or passport, for example).

 A principal is often a ***human , a computer, an application, or a robot.***

 In the case of a human principal, authentication may use physical characteristics such as ***voice, a fingerprint, a retinal scan, or even a DNA sample*** — this form of authentication is referred to as ***biometric authentication.***

 With **password-based authentication**, an individual is often expected to communicate his/her password to a verifying entity. However, in many cases it may not be advisable for the individual to reveal his/her password.

 Instead, he/she may be required to perform some **"one-way" cryptographic** operation using his/her secret, which cannot be performed without knowledge of it.

 Finally, many authentication systems today use a combination of techniques. This is referred to as ***multi-factor authentication.***

- *Authentication using passport with embedded photograph.*
- *New generation passports and smart cards can be used to store individuals fingerprint*

Authentication

- One way authentication
  - Password based authentication.
  - Certificate based authentication.
- Mutual authentication
  - Shared secrete based authentication.
  - Asymmetric based authentication.
  - Authentication and key management.

3.6 ONE-WAY AUTHENTICATION

 In client—server communications over a campus, network, for example, it is often the case that the client authenticates itself to the server.

 The server may or may not be authenticated to the client. This is referred to as ***one-way authentication.***

Categorized to

1. password based authentication

2. certificate based authentication

3.6.1 Password-based Authentication

- One of the most common mechanisms to implement authentication is the ***password.***
- To login to a server, a user enters his/her ***login name and password.***
- The password is the secret that is known only to the ***user and server.***
- The ***login name identifies a user,*** while the user's knowledge of thecorresponding ***password constitutes proof*** that he/she is the person with the given login name.
- As shown in below Figthe server uses the login name "Alka" to index into a database of (login name, password pairs),
- It Verifies that the ***submitted password matches*** the one stored against "Alka."

Disadvantage:

- First, the **password is sent in the clear**, so an attacker can eavesdrop on the message containing the password and later impersonate the real user.
- Second, the ***passwords are stored in unencrypted form in a file on the server.***
- If an internal attacker obtains access to that file, all passwords stored on that server could get compromised.

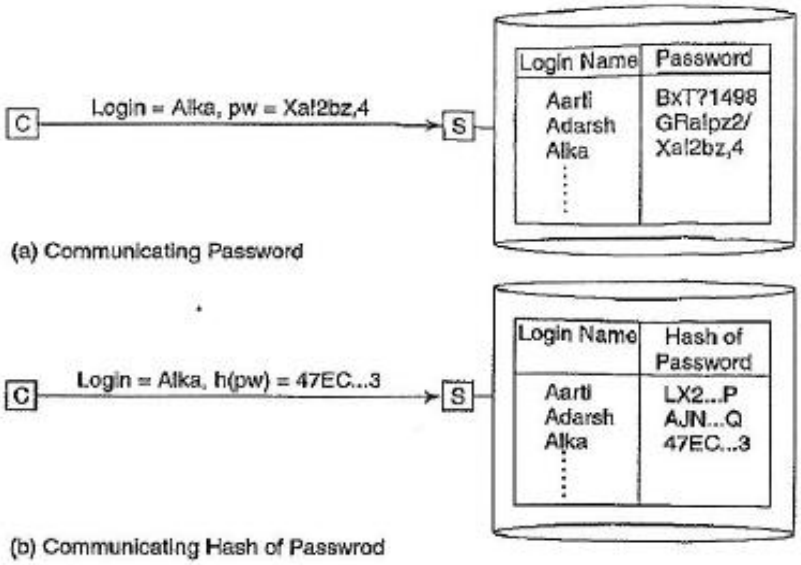


Figure 11.1 *Password-based one-way authentication*

- In Fig(b), the cryptographic hash of the password is stored on the server.
- Also, the login software prompts the user for his/her password and computes its hash which is transmitted.
- **The one-way property of the cryptographic** hash helps prevent an attacker from deducing user passwords from information in the password file or from communications on the transmission line.

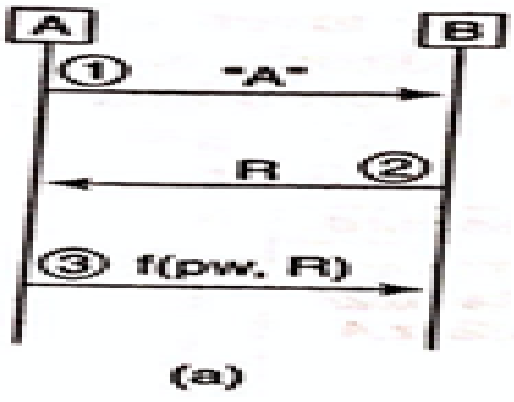
Drawback: However, an attacker could snoop on the communications between Alka and the server and obtain the hash of the password.

- He can, at a later point in time, replay it to the server thus impersonating Alka.
- Such an attack in which one plays back all or a part of one or more previous messages, with the intent of impersonating a legitimate user, is referred to as a **replay attack**.

Solution to Replay attack:

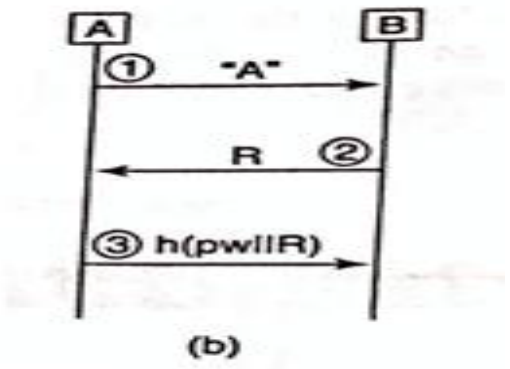
- An effective strategy to thwart a replay attack is for the verifier (in this 'case the server) to offer a fresh challenge to the prover (the client).
- In response, the client **does not communicate its password** but rather proves that it knows the password.
- The server is thus able to verify whether the client is genuine or not.
- The freshness of the challenge requires previous response to answer the current challenge. Such an authentication protocol is commonly referred to as a **Challenge—Response Protocol.**

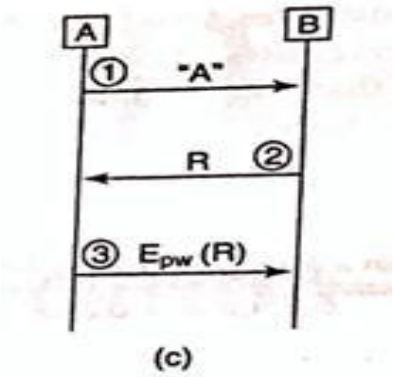
One way authentication using challenge response protocol

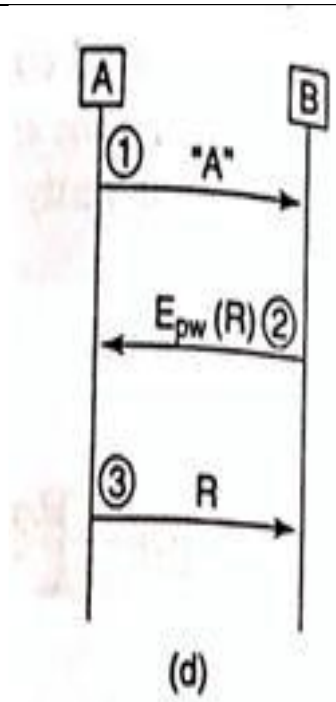
In the first message, A conveys its identity. The second message contains the challenge from the server. The challenge is a random number called a ***nonce (number used only once)*** in security parlance.	 (a) The function, ***f(pw, R),*** has the following properties:

<table>
<tr>
<td>
 The third message is the client's response - function of the challenge and the password.
</td>
<td>
• Given x and y, it should be easy to compute f(x, y).
• f is one-way; so, knowing f(pw, R) and R, it should be infeasible to compute pw .
• Given an R, it should be infeasible to compute f(pw, R) even if one knows

f(pw, R1,), f(pw, R1), f(pw, R3)
The corresponding R1, R2, R3 ..

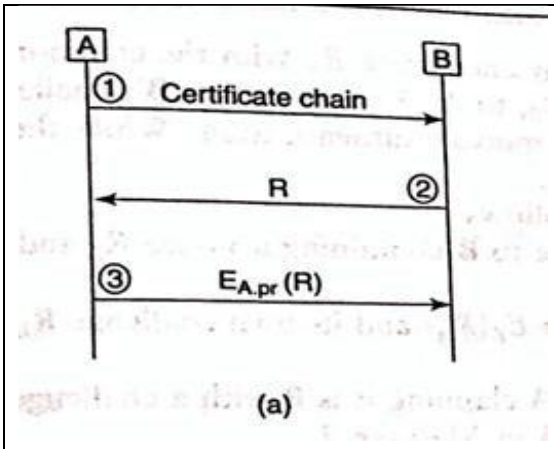
</td>
</tr>
<tr>
<td>


(b)
An obvious choice for f is the cryptographic hash [Fig. (b)], which is applied over the concatenation of the password and the nonce.
</td>
<td>


(c)
Another choice is a secret key encryption function with the where password is used as a key for encryption of random number R [Fig. (c)].
</td>
<td>


(d)
</td>
</tr>
<tr>
<td colspan="3">
In Fig. (d), the challenge sent by the server is an encrypted nonce.
 the client would need to decrypt the challenge to obtain the nonce and return it to the server to prove knowledge of his/her password.
 The underlying assumption in these and other protocols is that nonces are random and non-recurring.
 It is the "freshness" of a nonce that precludes a replay attack.
 The term nonce means "used only once."
 In actual implementations, neither the sender nor receiver keeps track of nonces generated or received.
</td>
</tr>
</table>

3.6.2 Certificate-based Authentication



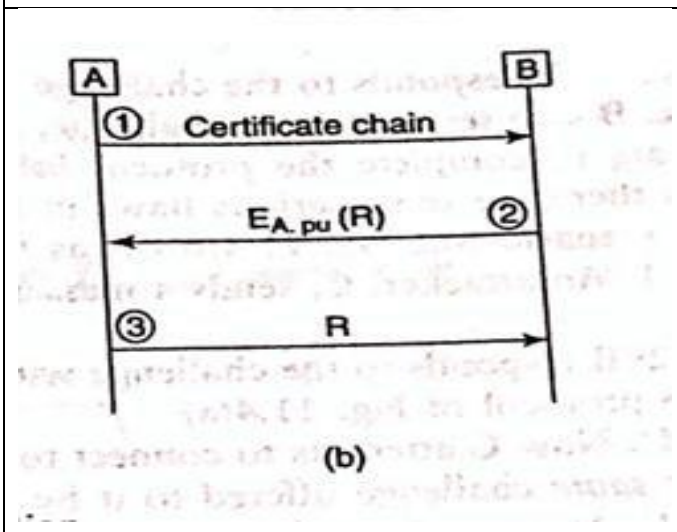
(a)

Fig a:

- **MSG 1:** A client need not share a secret with the server but sends a public key certificate.
  - As shown in Fig.(a), ***A sends her certificate in Message 1.***
  - B performs certain checks such as on the ***validity period and name of principal***.
  - He also verifies the ***signature of the CA*** on the certificate.

Fig a:

- **MSG 2:** He then sends his challenge — a nonce **R**.
- **MSG 3:** A responds by "encrypting" the challenge with her **private key.**
  - When B receives **EA.pr(R),** he "decrypts" it with **A's public key** and compares it with nonce he transmitted in Message 2.
  - If they match, he concludes that A has used the private key corresponding to the public key in her certificate.
  - Assuming that A's private key is safely protected, she must be the entity who created the correct response in Message 3.



(b)

- **MSG 2:** Figure (b) is a slight variation of the protocol in which ***B chooses a nonce, R,*** and ***encrypts it with A's public key*** to create the challenge.
- **MSG 3:** ***A decrypts*** the challenge and sends it to B.

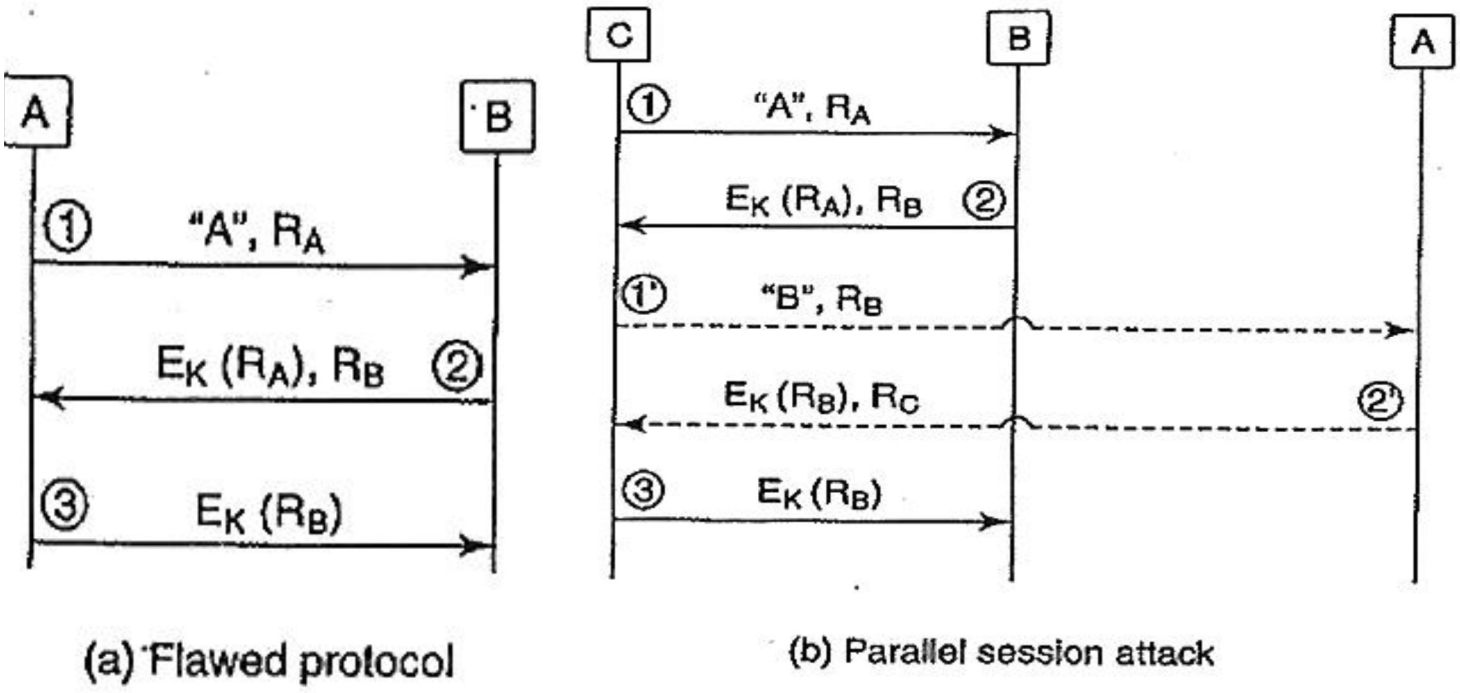
 Authentication of A to B succeeds if what B receives in Message 3, is R, the nonce he just chose.

B. MUTUAL AUTHENTICATION

- It is often necessary for ***both communicating parties to authenticate themselves to each other.***
- For example, in Internet banking, it is imperative that a customer interacts with his/her bank and not some entity posing as the bank.
- Likewise, it is important that a bank to verify the identity of the customer.

1. Shared Secret-based Authentication

- This is a mutual authentication using ***a secret key shared by both parties.***



(a) Flawed protocol (b) Parallel session attack

- **Message 1:** A communicates its **identity A** and its challenge in the form of a nonce **RA.**
- **Message 2:** B responds to the challenge by encrypting, **RA** with **common secret key , K, that A and B share.**
  B also sends its own challenge, RB, to A.
- **Message 3:** A's response to B's challenge in the third message appears to complete the protocol for mutual authentication. , there are some serious flaws in it.

One attack scenario [figure (b)]is as follows:

- **Message 1:An attacker, C, sends a message to B containing a nonce R_A and claiming to be A**
- **Message 2:** B responds to the challenge with **$E_K(R_A)$** and its own challenge **R_B** as required by the above protocol of Fig.(a).
- **Message 1'**: Now C attempts to connect to A claiming it is B. with a **challenge R_B.** Note that this is the same **challenge offered to it by B** in Message 2.
- **Message 2':** A responds to the challenge with **$E_K(R_B)$** and a nonce of its own.
- **Message 3:** C uses A's response **$E_K(R_B)$** to complete the three-message authentication **protocol with B**.

What has the attacker C accomplished?

 C has successfully impersonated A to B.

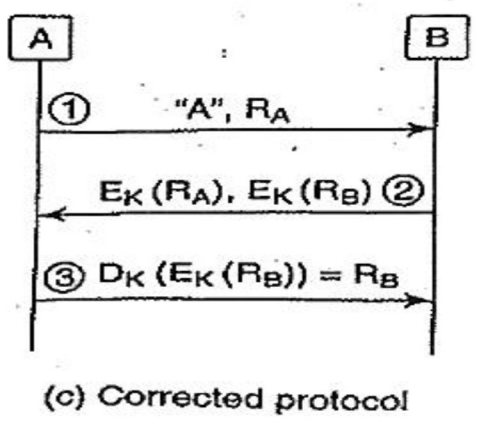
 Message 3 was required to complete the authentication of C (posing as A) to B.

 C initiated the authentication protocol with A, presenting to A the same challenge it had received from B.

 A's response to the challenge in Message 2' was used by C to convince B that it was A that was trying to establish communication with him. This attack is termed a **Reflection Attack** since a part of the message received by an attacker is reflected back to the victim.

 In this case, the reflected message fragment is $E_K(RB)$.

- This attack is also called a **Parallel Session Attack:** In the midst of the protocol run with one entity, opens another protocol run or session with the same entity or another entity.



(c) Corrected protocol

Solution1 : for Reflection Attack

- Figure c: the protocol might require the responder to encrypt his challenge, while the initiator would be required to decrypt her challenge.
- Encrypting both R_A and R_B

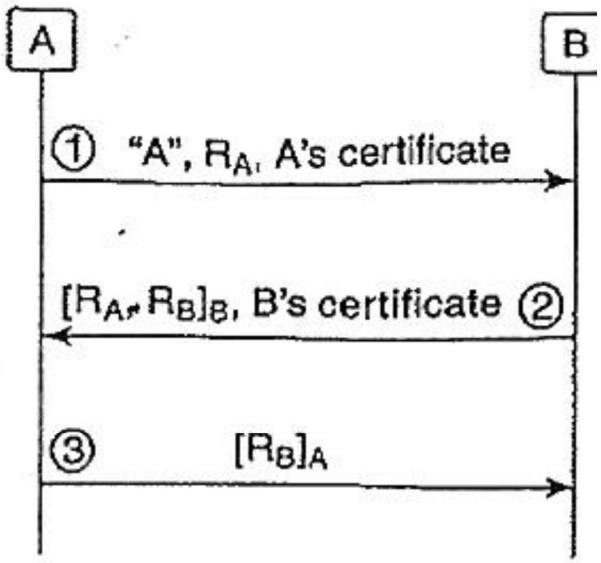
Solution2 : for Reflection Attack: Initiator and the responder to draw the challenges from different disjoint set so in figure (a) above A could use the nonces which are odd numbers, while B could use the nonces which are even numbers. With this modification the RB used in message 2 of figure (b) above cannot be reused in message 1'

2. Asymmetric Key-based Authentication (using public key encryption)

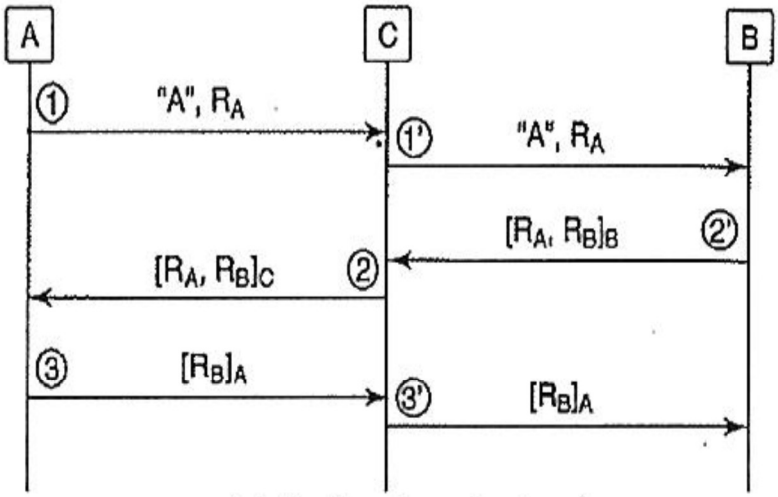
We assume that both ***A and B have public key/private key pairs.***

 The ***notation $[m]_A$ means a message m, sent together with A's signature on m.***

 In the protocol of Fig. (a), each party transmits its own nonce and challenges the other to sign it.



(a) Flawed protocol



(b) Attack on flawed protocol

- Figure (a) shows Mutual authentication using public key cryptography /asymmetric based authentication

- MSG1: Identity of A, challenge sent by A , which is R_A , A's certificate

- MSG2: the string obtained by concatenating RA, RB signed by B, B's certificate.

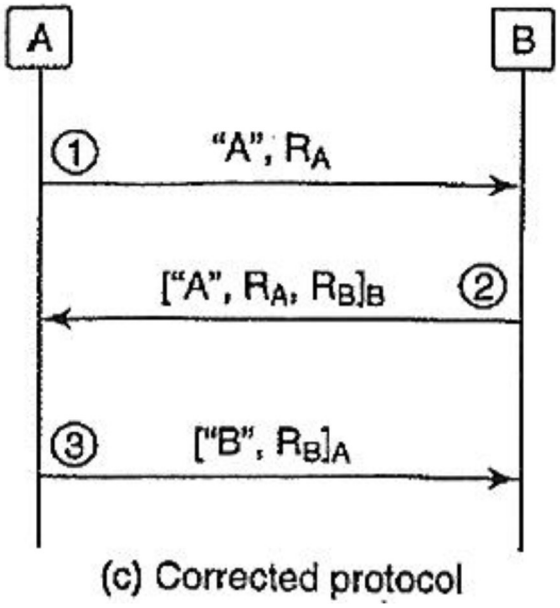
- MSG3: RB is the challenge signed by A

Figure (b) shows attack on flawed protocol:

- **MSG1: A initiates communication with C, sending the challenge RA.**
- **MSG 1': C initiates communication with B using the same nonce RA.**
- **MSG2'**: B responds to "A's challenge" and includes a challenge of his own, **RB**
- **MSG 2**: C responds to A's challenge and uses B's random number, RB, as his challenge to A.
- **MSG3**: A responds to C's challenge (which was actually generated by B). A thus completes the mutual authentication protocol with C.
- **MSG3'**: C forwards A's response to B.

It is clear from Fig.(b) :
That Once A communicates wit C in message 1,

- Message 1' is sent by C includes A's identity and attempts to convince B that A intends to talk to him.
- B responds to what appears to be A's intention to communicate with him.
- Note that, in the current scenario, A may not wish to communicate with B and is not aware that C is attempting to do so on her behalf.
- Yet, after B receives Message 3', he feels A intends to communicate with him since Message 3' contains her signature on a nonce chosen by him.

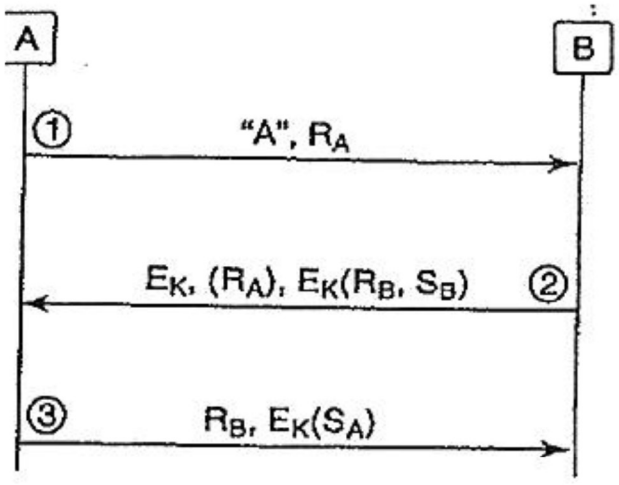


(c) Corrected protocol

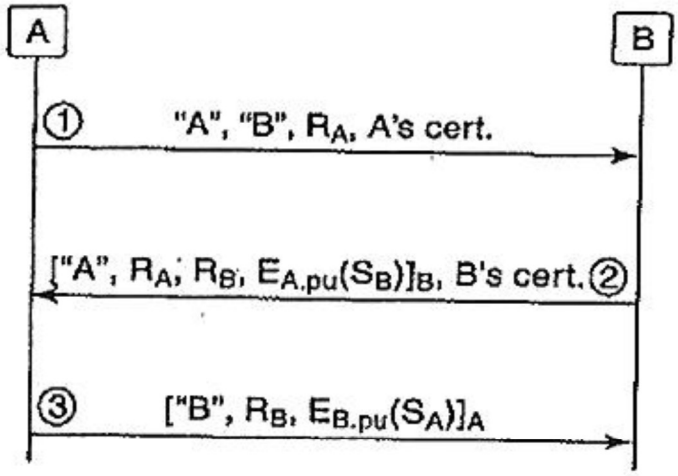
 One solution to the above problem is for the entities to include the **identity of the recipient in all messages signed.**
This is shown in Fig.(c).
 MSG 2: The string obtained by concatenating nonce **RA** and **RB** is **signed by B is sent** . (Means encrypted using B's private key)
 MSG 3: RB is the challenge provided by B and signed by A in response .(means encrypted using A's private key)
Note in fig (b) if this ws [C,R_B]$_A$ in msg 3 then if it was send to B by C , B will understand that it is for C, not for me

3. Authentication and Key Agreement

 In previous sections, authentication was performed using operations involving a long-term, shared secret or a private key.
 Since private key operations are very expensive, the communication can be **integrity-protected** and/or encrypted using **short term keys or session keys.**



(a) Using secret key cryptography



(a) Using public key cryptography

Session key = $S_A \oplus S_B$

Combined mutual authentication and key exchange

- **S_A and S_B** are the contributions to the secret key by A and B, respectively.

 They are freshly chosen random numbers that are encrypted and sent so that they cannot be eavesdropped upon

 The key finally chosen could be a simple function of S_A and S_B, S=S_A XOR S_B.

In figure (a) they are encrypted using shared secrete key in message 2 and 3	In figure (b) they are encrypted using recepient's public key in message 2 and 3

4 .Use of timestamps

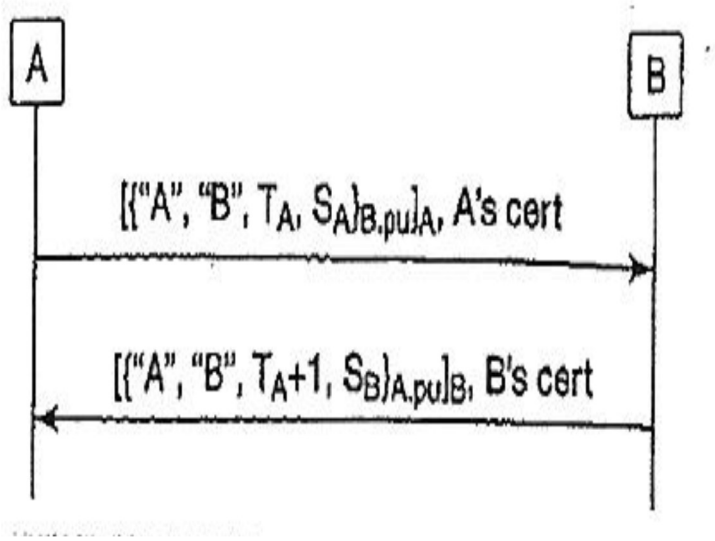


Fig: Mutual authentication using timestamp

 The use of nonces was introduced to prevent replay attacks.

 Basically, each party generates a nonce which is used as a fresh challenge to the other party.

 The recipient is often expected to sign or encrypt the challenge using a secret known to only the recipient (and the sender).

 The key idea here is the freshness of the nonce — if nonces were re-used, the response to the challenge could be **replayed** from a previous session.

 An alternative to nonces are timestamps.

 Ideally**, by securely "stamping" a message with the current time,** you convince the receiving party of its freshness.

Figure shows the use of timestamps in conjunction with public key cryptography for authentication.

- In Message 1, A inserts a timestamp, TA, in her message and signs it.
- B, on receiving the message, checks whether the timestamp is sufficiently recent and then verifies the with timestamps signature.
- He increments the received timestamp, inserts it into his response message to A, and signs the message.
- The notation $\{m\}_{X.pu}$, denotes a message, encrypted using the public key of X
- If the clocks maintained by A and B are **synchronized**, the timestamp in Message 1 signed by A convinces B that the message was freshly created by A.
- The timestamp implicitly serves as A's challenge to B.
- By signing the incremented timestamp, B hopes to satisfy A that he is indeed responding to her message.

C. DICTIONARY ATTACKS

1. Attack Types

- Dictionary attacks are typically launched in the context of passwords.
- Some passwords have too few characters.
- Others may be common celebrity names, place names, etc.
- Some individuals use permutations of characters in the names of their near relatives or friends so that they are easily memorisable.
- Based on such clues, an attacker can build a dictionary of strings which are potential passwords of his/her victim.

Password	*Reason for Weakness*
123 or abcd	Common default passwords
sY,u!	Anything less than 8 characters is too short
nahkhkurhahs	Celebrity name – Shahrukh Khan spelt backwards
23-05-86	Birthdays/anniversaries are convenient but would almost always be part of the attacker's password dictionary
atimuhdam	Permutation of letters in mother's or spouse's name, Madhumita, is a poor choice especially if the attacker has personal information about victim
Kolkata	Place names are oflen part of password dictionaries

- There are two types of dictionary attacks —

1. **on-line**
2. **off-line.**

A. **on-line attack:**

✓ In on-line attacks, an intruder attempts to **login** to the victim's **account** by using the victim's **login name and a guessed password**.

✓ There is usually a system-**imposed**, **limit** on the **number** of **failed** login attempts. So, unless the attacker is particularly insightful or lucky an on-line attack has a limited chance of success.

B. **off-line attack:**

✓ Unlike an on-line attack, an off-line dictionary attack leaves few fingerprints.

✓ One possibility is for the attacker to get a hold of the **password file**.

✓ Passwords are typically transformed in some way (by, for example, performing a cryptographic hash on them) before **being stored** on the authentication server.

✓ The cryptographic hash is a one-way function, so it is not easy for the attacker to deduce the password given its cryptographic hash.Not feasible to find f(pw,R) knowing R and f(pw,R)

✓ However armed with password file or with f(pw,R) **t**he attacker could use his/her dictionary of passwords to implement the following attack.

```
// Let D be an array containing the dictionary
// Let F denote f(pw, R) where pw is the client's password
// Let n be the number of permissible guesses (size of D)

found = false
i = 0
while ( ~ found && i < n)
{
        x = f(D[i], R)
        if (x = = F)
        {
                print ("CORRECT PASSWORD is D[i]")
                found = true
        }
}
```

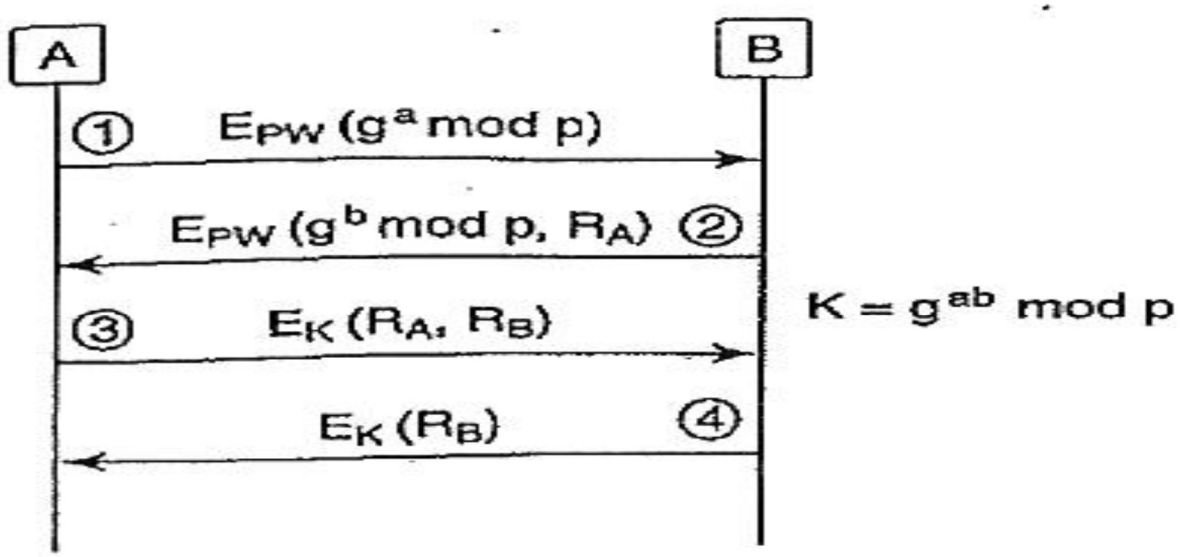
2. Defeating Dictionary Attacks

- One approach to frustrating a dictionary attack is to increase the cost of performing such an attack. The cost is **the time to successfully complete the attack.**
- The most time-consuming operation in each iteration of the dictionary attack program is **f(D[i], R).** Hence, to decrease the attacker's chance of success, the function f(D[i], R) could be made more computationally **expensive**.
- Suppose, for example, instead of the function f being a simple cryptographic hash, it was the cryptographic hash, h, applied successively a hundred times, that is,

  h (... h (h (D[i], R)) .....)
- If the above function were used in the loop of the program, we would expect the program to run about 100 times slower.

Encrypted Key Exchange (EKE) protocol.

- A protocol that virtually eliminates off-line dictionary attacks is the Encrypted Key Exchange (EKE) protocol.
- This is a password-based protocol that combines Diffie—Hellman key exchange with mutual authentication based on a shared secret.
- the Diffie—Hellman protocol is vulnerable to a man-in-the-middle attack which is due to the unauthenticated exchange of "partial secrets", **g^a mod p and g^b mod p.**
- To mitigate this attack, EKE uses a novel idea — each side transmits its partial secret after encrypting it.
- The encryption key, **PW,is the hash of the password.**
- Below Figure shows the four messages that are exchanged in EKE.



s **Figure 11.8** *EKE protocol*

- After MSG 2, both sides should be able to compute the **new session key** **$k = g^{ab} \bmod p$** denoted by K in the figure.

- Mutual authentication is accomplished using the familiar challenge—response protocol in which each side selects a random nonce and challenges the other side to encrypt it with the newly computed session key.

- It is assumed that the victim's password is "weak," that is, it can be guessed using moderate effort. That being the case, basic **password-based mutual authentication** protocols could yield to an off-line dictionary attack.

- Assume that an attacker has access **to $E_{pw}(g^a \bmod p)$ and $E_{pw}(g^b \bmod p)$.**
- The attacker would attempt to guess the victim's **password** and hence PW.

- If the attacker guessed correctly, he/she would be able to obtain the true values of $g^a \bmod p$ and $g^b \bmod p$. But even so, he/she would **not** be **able** to obtain the session key, **$g^{ab} \bmod p$.**

- This is so, since the computational Diffie—Hellman problem is infeasible in large groups that are carefully chosen,

- Thus, EKE is not susceptible to an off-line dictionary attack.

- Another property of EKE is that it provides perfect forward secrecy

- A protocol is said to have perfect **forward secrecy** if it is not possible for an attacker to decrypt a session between A and B even if he/she records the entire encrypted session and then at a later point in time (say a week later) obtains or steals all relevant long term secrets of A and B.
    - Knowing the long term secrets which is the password shared between two parties and two partial secretsccg^a mod p and g^b mod p which are encrypted by password.
    - Even these password is stolen , attacker can find g^a mod p and g^b mod p,but because of infeasibility of the computational Diffie—Hellman problem ,attacke will not be able to deduce the session key **$g^{ab} \bmod p$.**

Authentication-II

- **Key Distribution Centre (KDC) – a trusted third party that shares long-term keys-with clients and servers alike.**
- Two protocols that make use of a KDC—the ***Needham-Schroeder protocol and Kerberos.***
- We then look at the biometric authentication as a complement to and, in some cases, as a substitute for cryptographic authentication.

CENTRALISED AUTHENTICATION

There are a number of advantages of secret key cryptography over public key cryptography in authentication protocols.

- First, digital certificates and a public key infrastructure (PKI) are needed in support of public key cryptography.
- There is a substantial cost to set up and maintain a PKI.
- Also, public key/private key operations are relatively slow compared to secret key operations.
- In secret key cryptography, If the entity communicates with a large number of other entities over time, it must share a secret with each of those parties.
- Managing and securely **storing a large number of keys** is a nontrivial task.
- One approach to alleviating the risk is to employ a ***trusted third*** **party** which, in this case, functions as a ***key distribution centre*** **(KDC).**
- Each user registers with a KDC and chooses a password.
- A ***long-term secret,*** which is a function of the password, is to be exclusively shared by that user and the KDC.
- The main function of the KDC is to securely communicate a fresh, **common session key** to the two parties who wish to communicate with each other.

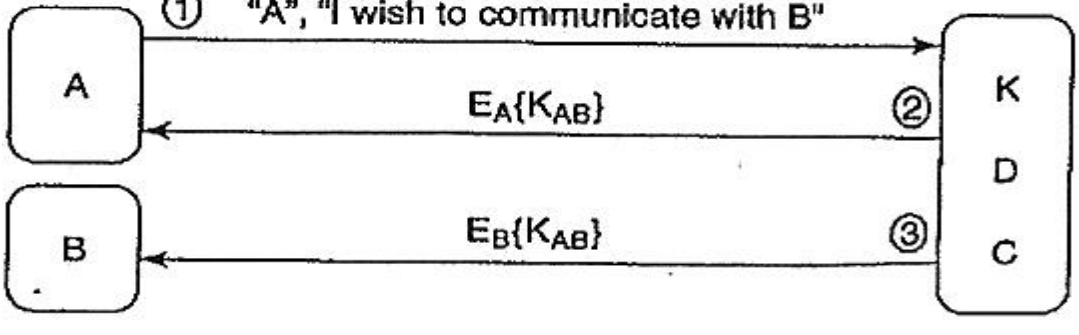


Fig 1: *Message confidentiality using a KDC*

- ***Message 1:*** **A** informs the KDC that it intends to communicate with B
- The KDC generates a random secret, $\mathbf{K_{AB}}$, and dispatches this to **A** and **B** through two encrypted messages.
- ***Message 2*** is encrypted using the long-term secret, $\mathbf{K_A}$, that A shares with KDC.
- ***Message 3*** is encrypted with $\mathbf{K_B}$, the secret shared between B and the KDC.
- Both A and B **decrypt** their messages and obtain the ***short-term session key.***
- A and B then all subsequent messages during the session using $\boldsymbol{K_{AB}}$.
- The above Figure was meant to convey the general idea in using a KDC but the protocol is susceptible to numerous types of replay and man-in-the-middle attacks.

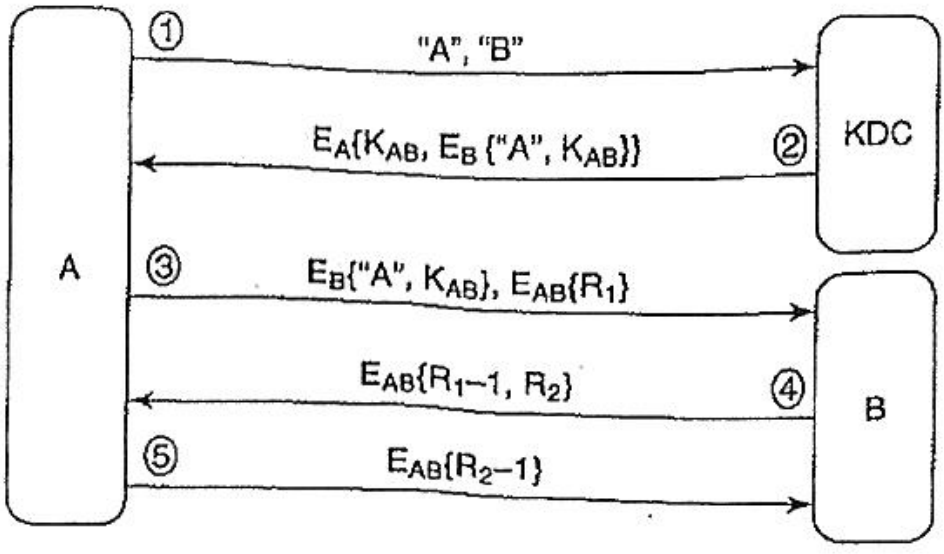
THE NEEDHAM-SCHROEDER PROTOCOL

- There are four versions
  1. **Preliminary version 1**
  2. **Preliminary version 2**
  3. **Preliminary version 3**
  4. **Final version**

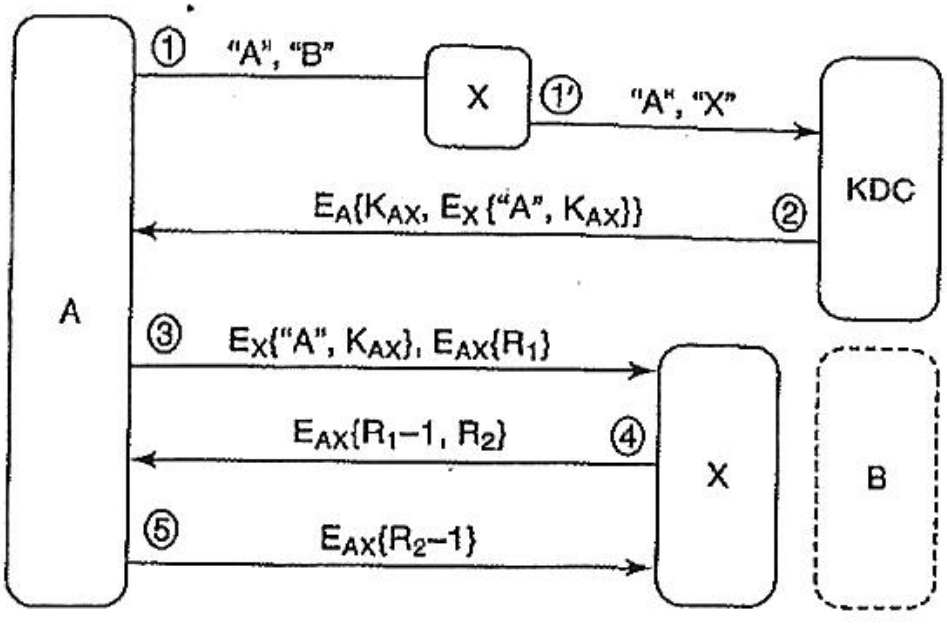
1. Preliminary version 1

- Below Figure Fig2 (a) enhances the protocol of Fig1.to provide **mutual authentication** by including, **challenge—response phase in message 3,4,5**.
- Here, both sides proceed to challenge the other to prove knowledge of the session key, $\boldsymbol{K_{AB}}$.
- The challenge is a **nonce.**
- The response involves ***decrementing the nonce*** and ***encrypting the nonce*** with the **session key,** $\boldsymbol{K_{AB}}$.

- ***MSG 1:*** Identity of A and B sent from A to KDC
- ***MSG 2:*** *In* ***response KDC*** encloses the ticket to B.ie $\mathbf{E_B\{A,K_{AB}\}}$
- ***MSG 3:*** A then forwards the **ticket** along with the challenge to B (R1)
- ***MSG 4: R1 is decremented and B challenges A with R2.***
- ***MSG 5:R2 is decremented by A and forwarded to B.***



(a) : Preliminary version 1



(b) : Man-in-the middle attack on preliminary version 1

Needham–Schroeder protocol: Preliminary version 1

Man in the middle attack on Preliminary Version 1

- The protocol in Fig. 2(a) is susceptible to an impersonation attack shown in Fig. 2(b).
- The attacker, X, is an insider who shares a long-term key with the KDC.
- The attacker, X, intercepts Message 1, substitutes "B" for "X" and sends the modified message to the KDC.

- In response, the KDC creates a ticket encrypted with X's long-term key and sends it to A in Message 2.
- Now X Intercepts Message 3. He decrypts the ticket using the long-term secret he shares with the KDC. He thus obtains the session key, K_{AX}.
- Message 3 also contains A's challenge R1.
- X uses the session key, K_{AX} to decrypt the part of the message containing A's challenge. He successfully responds to A's challenge in Message 4.
- Thus, X successfully impersonates B to A.

2. Preliminary version 2

- A simple fix to the protocol is to include B's identity in the encrypted message from the KDC to A (Message 2). The modified message is

$$E_A\{K_{AB}, "B", E_B\{"A", K_{AB}\}\}$$

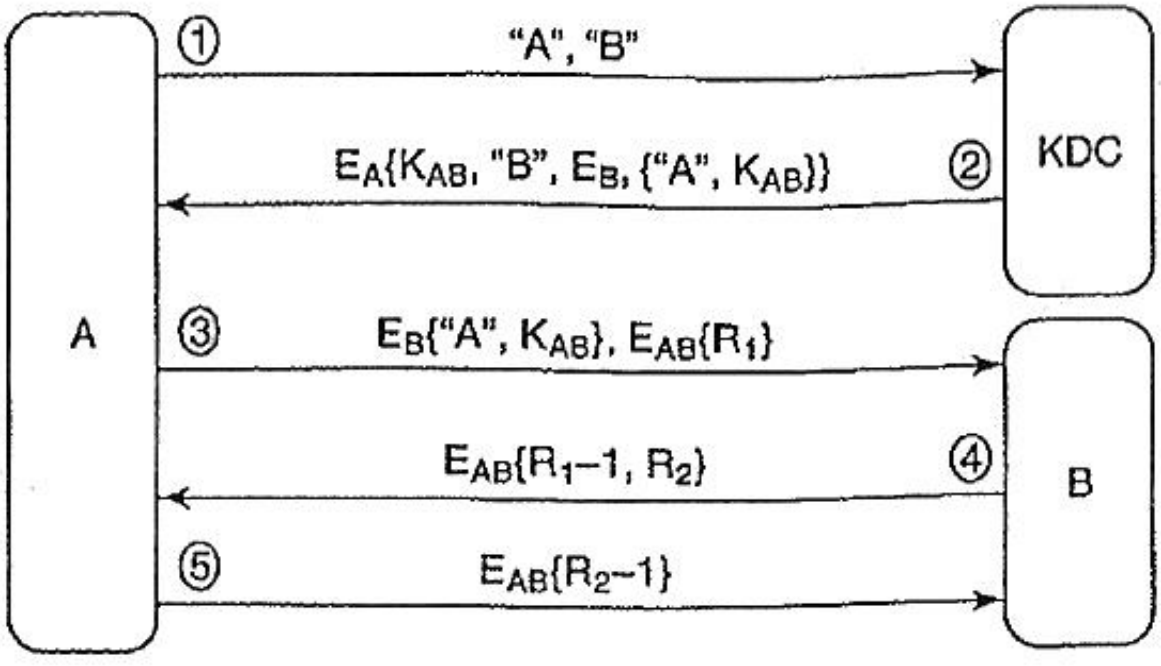
Fig3 a

- ***MSG 1:***Identity of A and B
- ***MSG 2:*** *In* ***response KDC*** encloses the ticket to B.ie $E_B\{A,K_{AB}\}$**, b's identity**
- Now, after A receives and decrypts Message 2, she checks whether B's identity is contained inside the message. The presence of B's identity confirms to A that the KDC knows that A wishes to communicate with B.
- ***MSG 3:*** A then forwards the **ticket** along with the challenge to B (R1)
- ***MSG 4: R1 is decremented and B challenges A with R2.***
- ***MSG 5:R2 is decremented by A and forwarded to B.***

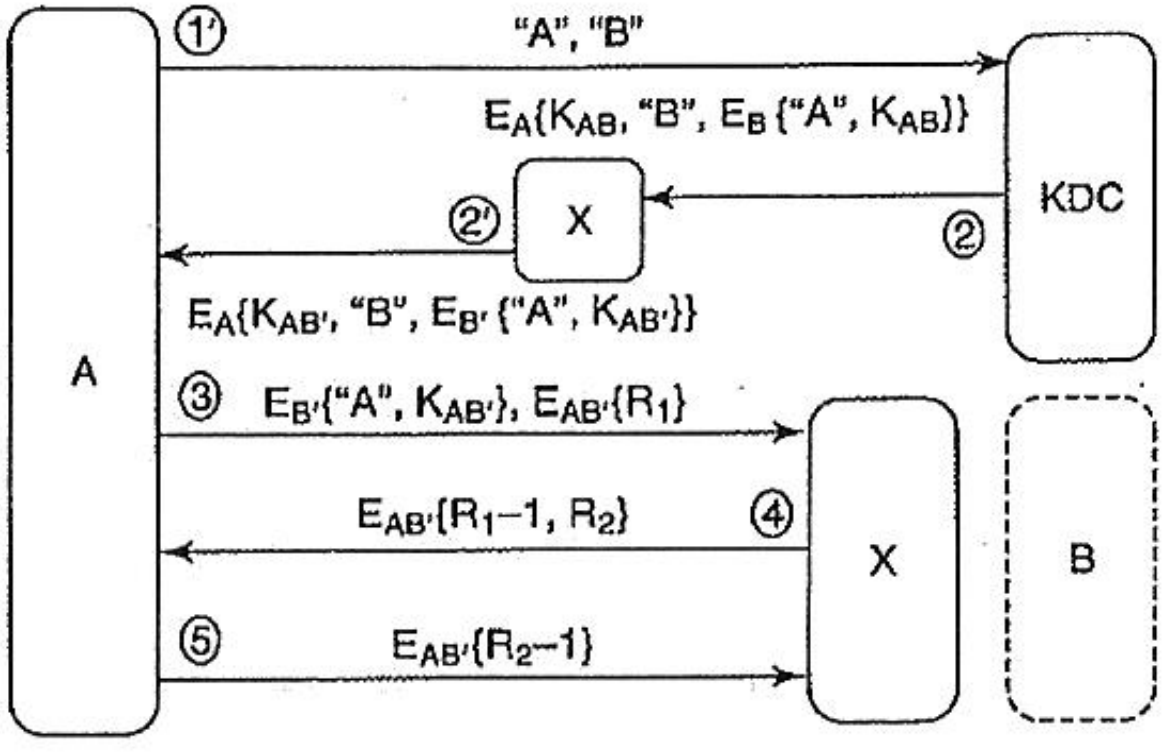
Man in the middle attack and replay attack on preliminary version 2

- The attacker, X, does the following:
- X eavesdrops and records many of A's sessions with the KDC and with B over a period of time and steals **B's password or long-term key**.
- B recognizes that his password has been stolen and immediately reports the incident to the KDC.
- He obtains a **new long-term key,** K_B**,** which he uses subsequently.
- Even then , the following scenario shows X successfully impersonates B to A.

FIG3



(a) Preliminary version 2



(b) Man-in-the middle and replay attack on preliminary version 2

Needham–Schroeder protocol: Preliminary version 2

1. A wishes to communicate with B and sends Message 1 in Fig.3 (b).
2. X intercepts the KDC's response (Message 2) and instead plays a previous recording of Message 2.
3. This message contains a ticket encrypted with **B's old key, $K_{B'}$.**
4. X then intercepts Message 3 from A, which contains the old ticket and a fresh challenge to B. Because X has access to B's old key, he can decrypt this ticket and recover the session key, $K_{AB'}$.
5. Because X knows **$K_{AB'}$,** he can respond to A's challenge in Message 4. X's response is exactly what A expected to receive from B. Hence, A is convinced that she is talking to B.

3. Preliminary version 3

- We can fix this vulnerability in version 2 by ensuring the *freshness* of Message 2.
- This is accomplished by **A sending a (fresh) nonce in Message 1** [Fig. 4(a)] and receiving **confirmation of its receipt by the KDC in message 2.**

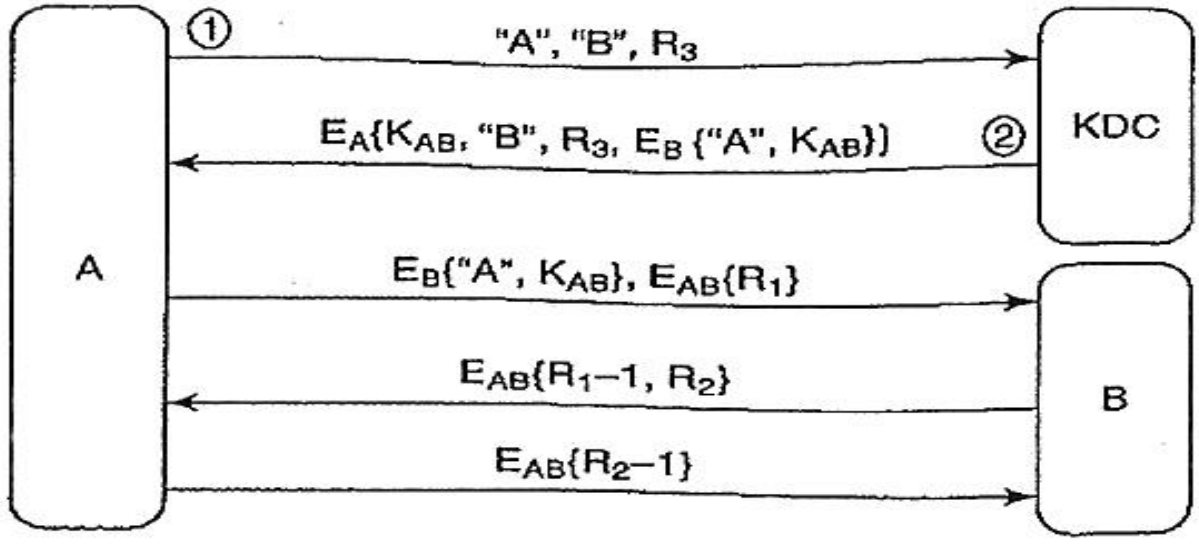
Replay attack on preliminary version 3

- The version 3 is still not secure despite the modifications made.
- X could still attack the protocol by recording previous messages and selectively replaying them when the right opportunity presents itself.
- Such as he attempts to steal A's password or long-term key.
- Assume again that A suspects the compromise of her password and promptly reports this to the KDC without delay.
- X then manages to steal A's long-term key that she shares with the KDC and perform a**n impersonation attack.**
- **A'** is the old password generated key
- **A** is new password generated key.
- Using the compromised (old) key, X can decrypt this message and recover
  - The old session key, $\mathbf{K_{A'B}}$
  - The old ticket $\mathbf{E_B\{A,K_{A'B}\}}$
- To impersonate A, X does the following [see Fig.4(b)]:
  1. X sends,in Message 1 to B, the old ticket and a challenge, R1, encrypted with the old session key.

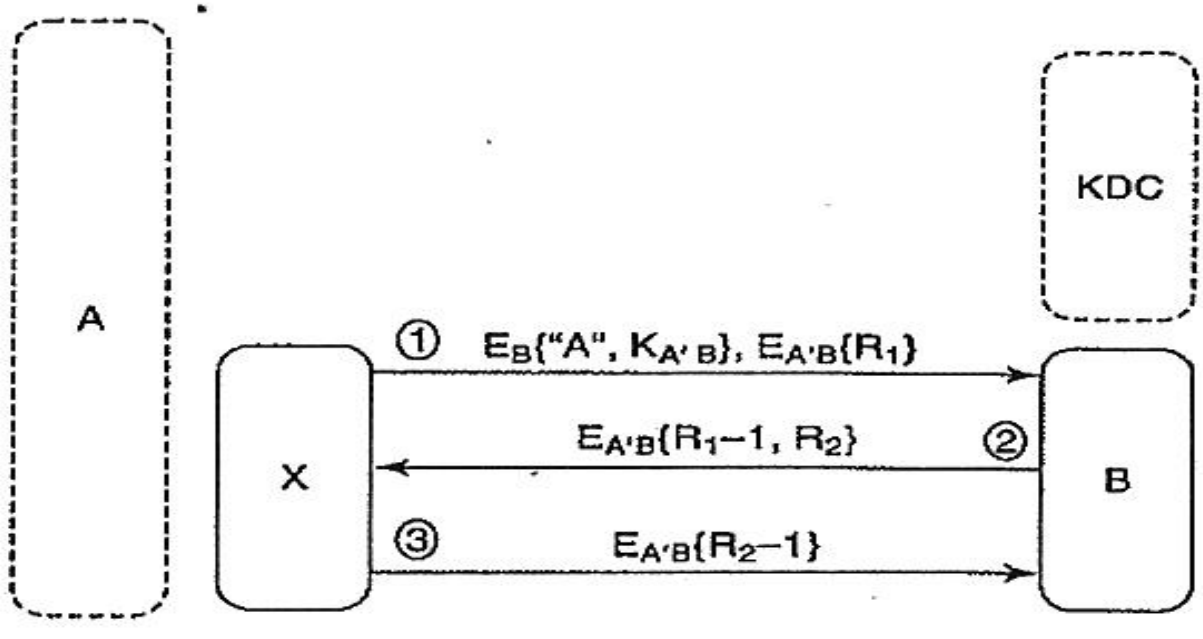
2. B responds to X's challenge and also communicates his own challenge, R2.
3. Because X has the session key, he responds to the challenge by encrypting R2 with the old session key.

- B receives the response and is convinced he is talking to A (impersonated by X).

Fig 4



(a) : Preliminary version 3

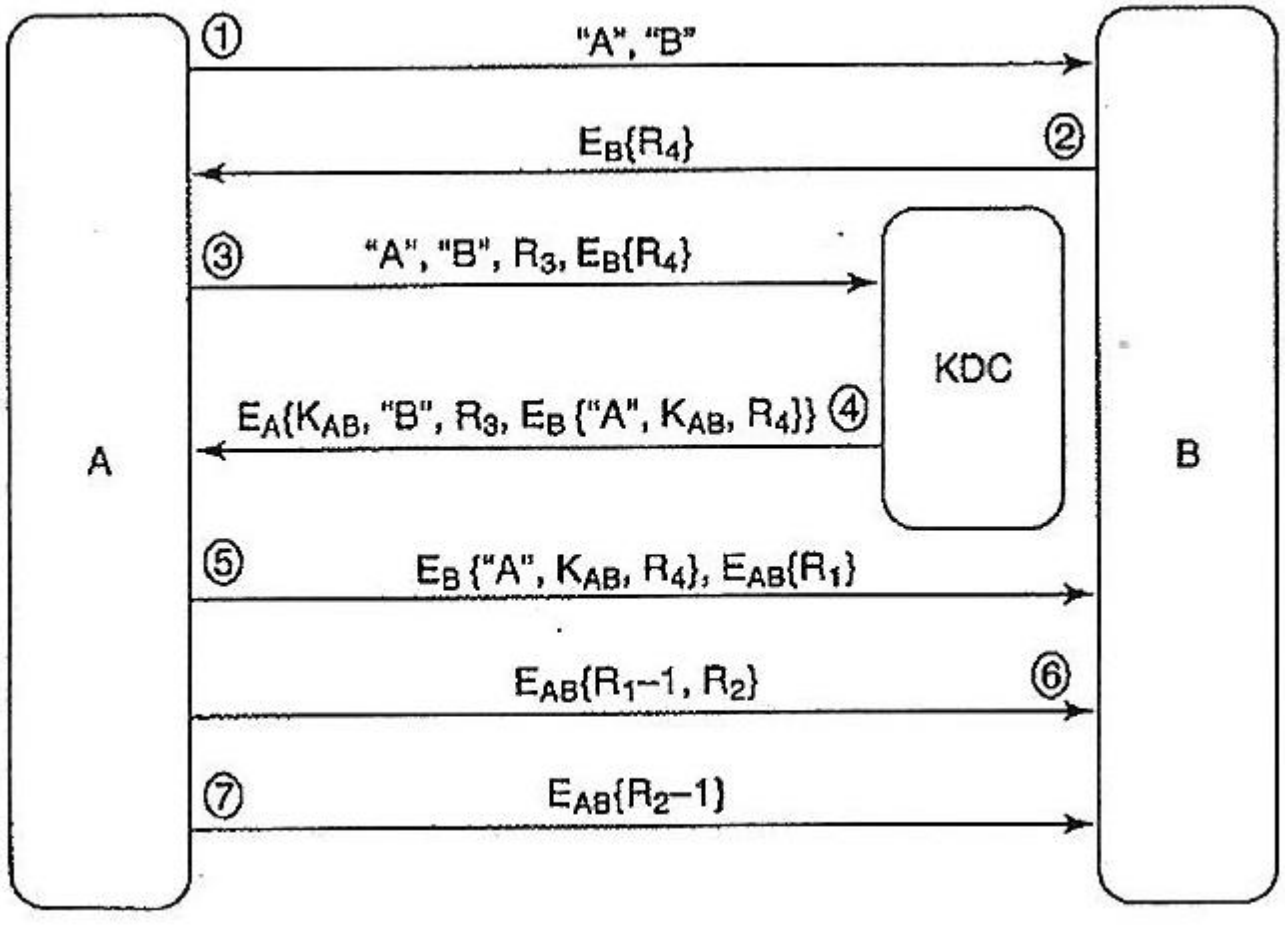


(b) : Replay attack on preliminary version 3

Needham–Schroeder protocol: Preliminary version 3

Needham Schroeder Protocol: Final Version

FIG 5:



Needham–Schroeder protocol: Final version

 The problem in previous versions could be fixed if B were allowed to choose a nonce (R4) and he same nonce were enclosed by the KDC in the **ticket it generates.**

 MSG 1:Identity of A and B sent from A to B

 MSG2: random number R4 generated by B

 MSG3: A forwards his challenge as R3 along with R4.

 MSG 4:KDC generates ticket and includes R4

 When B receives this ticket in msg5, B can verify the random number/nonce generated in msg2 is same or not.

Kerberos

➢ A user could use the same password for all servers but distributing and maintaining a password file across multiple servers poses a securit risk.

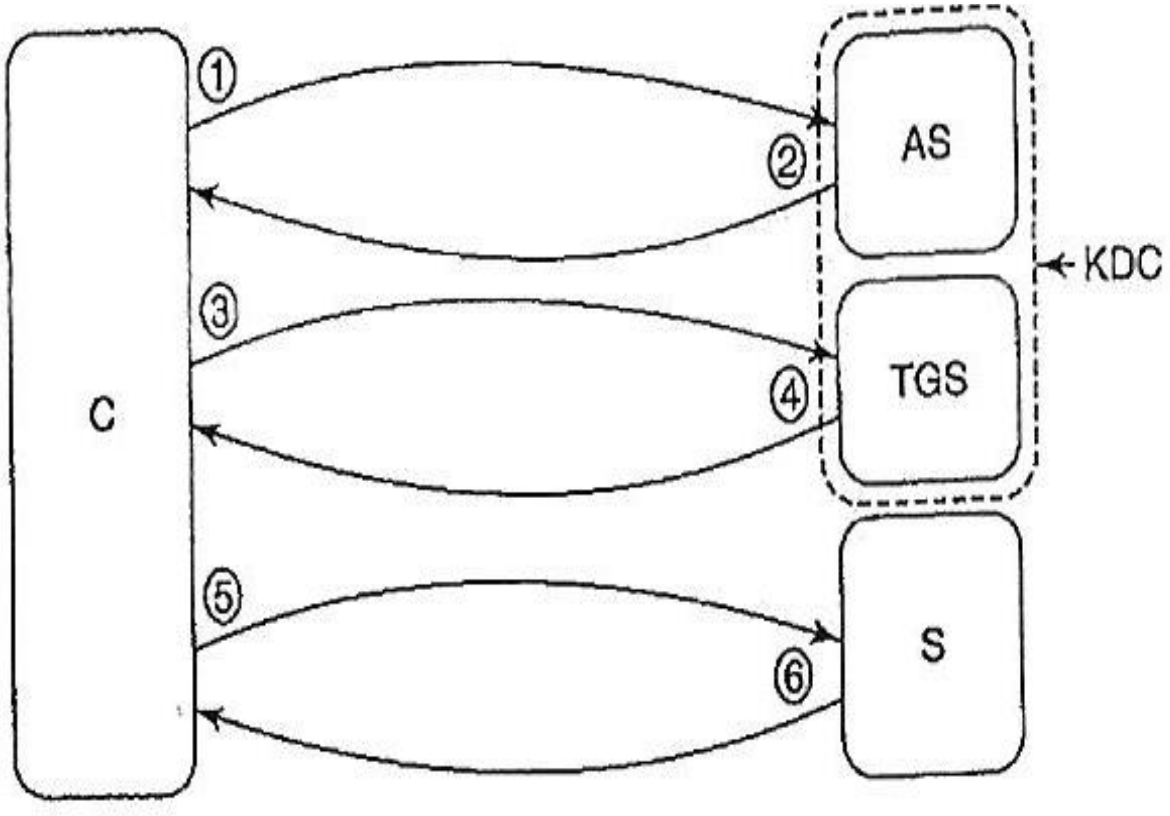
➢ A password-based system should ensure the following:

1. The password should not be transmitted in the clear.
2. It should **not** be possible to launch **dictionary attacks**.
3. The password itself should not be stored on the authentication server, rather it should be cryptographically transformed before being stored.
4. It should not be possible to launch dictionary attacks by obtaining a file containing cryptographically transformed versions of the password.
5. A user enters her password only ONCE during login. Thereafter, she should not have to re-enter her password to access other servers for the duration of the session. This feature is called **single sign-on.**
6. The password should reside on a machine for only a few **milliseconds** after being entered by the user.

The Kerberos protocol elegantly addresses many of these issues.

- Developed at MIT, Kerberos has been through many revisions.
- The latest is Kerberos Version 5.
- The KDC used in the Needham—Schroeder protocol is logically split into two entities here — the Authentication Sewer (AS) and the Ticket Granting Server (TGS).
- The sequence of messages exchanged between the client (C), the Kerberos servers (AS and TGS) and the requested server(S) is shown in Fig.6

FIG 6



① C request Ticket-Granting Ticket

② C receives Ticket-Granting Ticket

③ C request Service-Granting Ticket

④ C receives Service-Granting Ticket and session key

⑤ C authenticates itself to S

⑥ S authenticates itself to C

Kerberos message sequence

- **There are three steps — each involving two messages**

Step 1: Receipt of Ticket-Granting Ticket

Message 1 $C \rightarrow AS$: "C", "TGS", Times, R_1

Message 2 $AS \rightarrow C$: "C", $Ticket_{TGS}$, E_C {"TGS", $K_{C,TGS}$, *Times*, R_1}

where

$Ticket_{TGS} = E_{TGS}$ {"C", "TGS", $K_{C,TGS}$, *Times*}

Step 1: Receipt of Ticket-Granting Ticket

Message 1: $C \rightarrow AS: \quad \text{“C”}, \quad \text{“TGS”}, \text{Times}, R_1$

- In Message 1, the client informs the AS that it wishes to communicate with the TGS.
- *"Times"* field specifies the start time and expected duration of the login session.
- "C," is the ID of the user/client who has logged in.
- R1 is a nonce generated by C

Message2:

$$AS \rightarrow C: \quad \text{“C”}, \quad \text{Ticket}_{TGS}, E_C \{\text{“TGS”}, K_{C,TGS}, \text{Times}, R_1\}$$

where

$$\text{Ticket}_{TGS} = E_{TGS} \{\text{“C”}, \text{“TGS”}, K_{C,TGS}, \text{Times}\}$$

- The response from the AS (Message 2) contains a session key, $\mathbf{K_{c,TGS}}$, to be used for communication between C and the TGS.
- This key is encrypted with the long-term key, K_C known to C and the AS.
- This key is a function of the user's password.
- AS encrypts the nonce, that it received in Message 1.
- The nonce is used to prevent replay attacks.
- The AS also includes a TGT **(Ticket TGS)** in connection with C's request. Contains fresh session key $\mathbf{K_{c,TGS}}$ and is encrypted using long term key shared between AS and TGS

Step 2: Receipt of Service-Granting Ticket

Message 3 $C \rightarrow TGS: \quad \text{“S,”} \ \text{Times}, \text{Authenticator}_C, \text{Ticket}_{TGS}, R_2$

where

$$\text{Authenticator}_C = E_{C,TGS} \{\text{“C”}, TS_1\}$$

Message 4 $TGS \rightarrow C: \quad \text{“C”}, \text{Ticket}_S, E_{C,TGS} \{\text{“S”}, K_{C,S}, \text{Times}, R_2\}$

where

$$\text{Ticket}_S = E_S \{\text{“C”}, K_{C,S}, \text{Times}\}$$

Step 2: Receipt of Service Granting Ticket

Message 3:

$$C \rightarrow TGS: \quad \text{“S,” } Times, \ Authenticator_C, \ Ticket_{TGS}, \ R_2$$
where
$$Authenticator_C = E_{C,TGS} \{\text{“C”}, TS_1\}$$

- In Message 3, C forwards the TGT (Ticket TGS), Authenticator C to the **TGS**
- **Using** this Ticket TGS ,**TGS server** extracts the session key, $\mathbf{K_{C,TGS}}$, known only to **C** and the TGS
- As shown above, the **Authenticator C** encrypts the current time (timestamp) and ID using $\boldsymbol{K_{C,TGS}}$

Message 4:

$$TGS \rightarrow C: \quad \text{“C”}, Ticket_S, E_{C,TGS} \{\text{“S”}, K_{C,S}, Times, R_2\}$$
where
$$Ticket_S = E_S \{\text{“C”}, K_{C,S}, Times\}$$

- The *TGS* generates a fresh session key, $K_{c,s}$, to be shared between **C** and S.
- This key is encrypted using the session key $\mathbf{K_{C,TGS}}$, so only **C** can decrypt it.
- The fresh nonce, **R2, from C is also encrypted by** the TGS using $\mathbf{K_{C,TGS}}$
- This convinces **C** that the received message is from the **TGS**
- Finally, the fresh session key $K_{c,s}$ is enclosed in a *service-granting ticket* to be forwarded by C to **S.**

Step 3: Client-Server Authentication

Message 5 $C \rightarrow S$: $Ticket_S, Authenticator_C$
where
$$Authenticator_C = E_{C,S} \{\text{“C”}, TS_2\}$$

Message 6 $S \rightarrow C$: $E_{C,S} \{TS_2 + 1\}$

Step 3: Client-Server Authentication

Message 5:

$$C \rightarrow S: \quad Ticket_S, Authenticator_C$$

where

$$Authenticator_C = E_{C,S}\ \{\text{“C”}, TS_2\}$$

- C forwards to **S** the ticket containing the session key, $K_{c,s}$.
- **C** also creates and sends to **S** an authenticator by encrypting a timestamp with the session key $K_{c,s}$

Message 6:

$$S \rightarrow C: \quad E_{C,S}\ \{TS_2 + 1\}$$

- **S** retrieves Kc,s from the service-granting ticket.
- **S** verifies the authenticator from **C.**
- **S** then increments the timestamp and encrypts it with the fresh session key.
- The encrypted timestamp serves to authenticate S to C.
- Use of timestamps prevents replay attack

BIOMETRICS

Preliminaries

- A biometric is a ***biological feature*** or ***characteristic of a person*** that *uniquely identifies* him/her over his/her lifetime.
- Common forms of biometric identification include face recognition, voice recognition, manual signatures, and fingerprints.
- More recently, patterns in the iris of the human eye and DNA have been used.
- Behavioural traits such as keystroke dynamics and a person's walk have also been suggested for biometric identification.
- Biometric forms were first proposed as an alternative or a complement to passwords.
- Passwords are based on what a user knows.

- Commonly used ID cards, including personal smart cards, are based on what a person has.
- A biometric, on the other hand, links the identity of a person to his/her physiological or behavioural characteristics.
- The two main processes involved in a biometric system are enrolment and recognition.

1. *Enrolment:*

- ✓ In this phase, a subject's biometric sample is acquired.
- ✓ The essential features of the sample are extracted to create a ***reference template.***
- ✓ Sometimes multiple samples are taken and multiple templates are stored to increase the accuracy of a match in the subsequent recognition phase.

2. Recognition:

- ✓ A fresh biometric sample of a person is taken and compared with the reference templates to determine the extent of a match.

- Biometrics are used in two different scenarios:

1. Authentication

- ✓ Biometric system stores (login name and biometric sample)
- ✓ authentication involves a one-to-one match

2. Identification

- ✓ As in authentication, a biometric sample of the subject is taken but the subject's identity is not presumed to be known beforehand.
- ✓ It is assumed that a database of biometric samples of several users already exists.
- ✓ The subject's biometric sample is compared with the samples in the database to determine if a match exists with any one of them.
- ✓ identification involves a one-to-many match
- ✓ A typical application of authentication is in access control, while identification finds widespread uses in forensics/criminology.

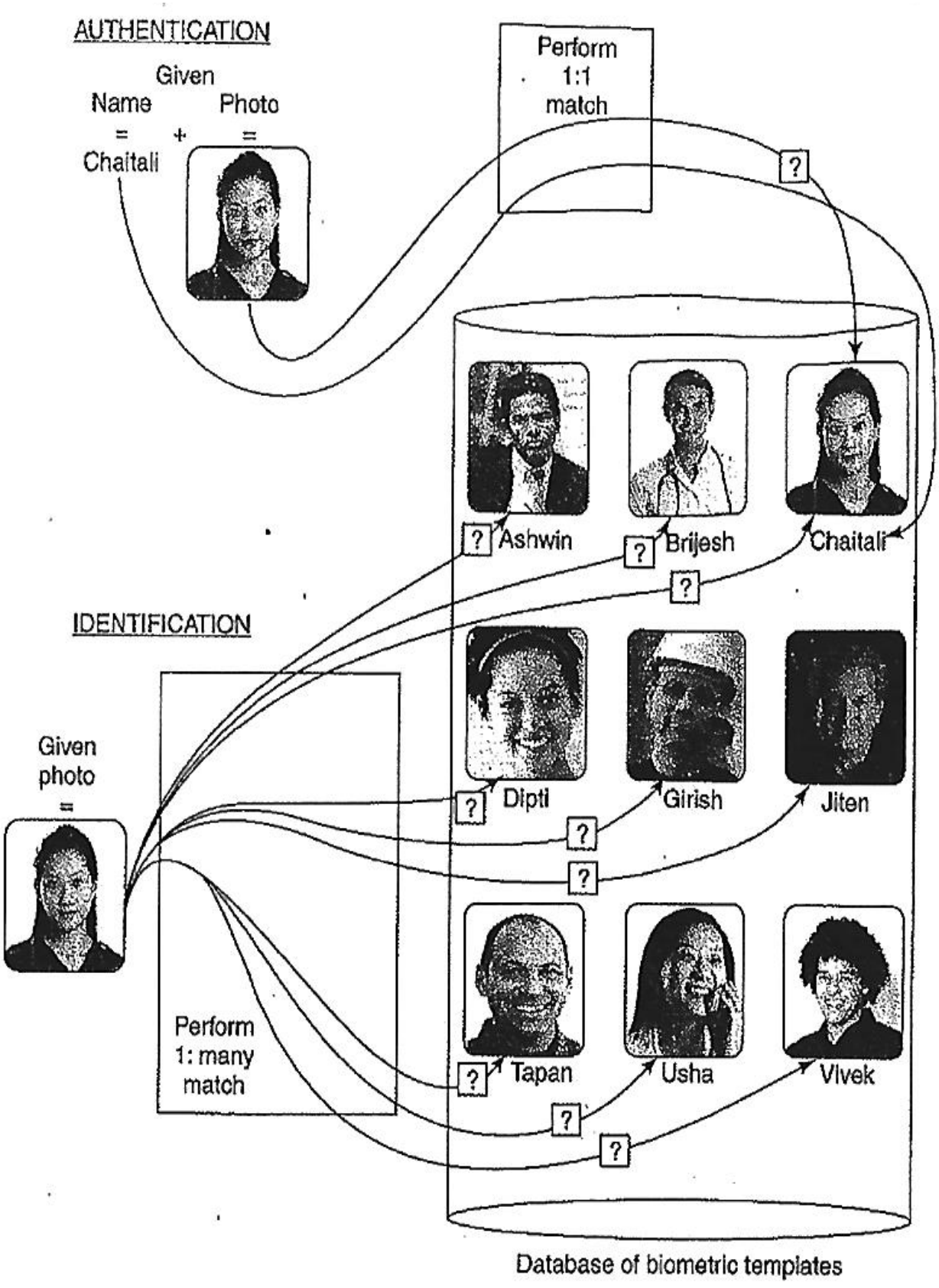


Figure 12.7 *Authentication versus identification*

The characteristics of a good biometric include the following:

✓ **Universality:** All humans should be able to contribute a sample of the biometric. For example, the speech-impaired may not be able to contribute towards a voice recognition system.

✓ **Uniqueness.** Biological samples taken from two different humans should be sufficiently different that they can be distinguished by machine intelligence.

✓ One litmus test of uniqueness is whether the biometric samples of two identical twins serves to unambiguously identify them.

✓ **Permanence**. The biometric should not change over time. The samples acquired during enrolment may be several years old (even tens of years old). Still, it should be possible to detect a match between the newly acquired sample and that stored in a database of samples of thousands of individuals.

✓ Permanence is not a given. For example, a person's voice may temporarily change due to a cold, the manual signature of a senior citizen may change and fingerprints of people in certain professions may wear out over time.

Case studies

1. Fingerprints

2. Iris scan

1. Fingerprints:

✓ A fingerprint is an impression left by the ridges and valleys of a human finger.

✓ Each individual fingerprints exhibit distinctive patterns.

✓ During the enrolment and recognition phase ,an image of the fingertip is taken by placing it on the plane surface of a scanner.

✓ During the recognition phase the input template must match with the patterns stored in database.

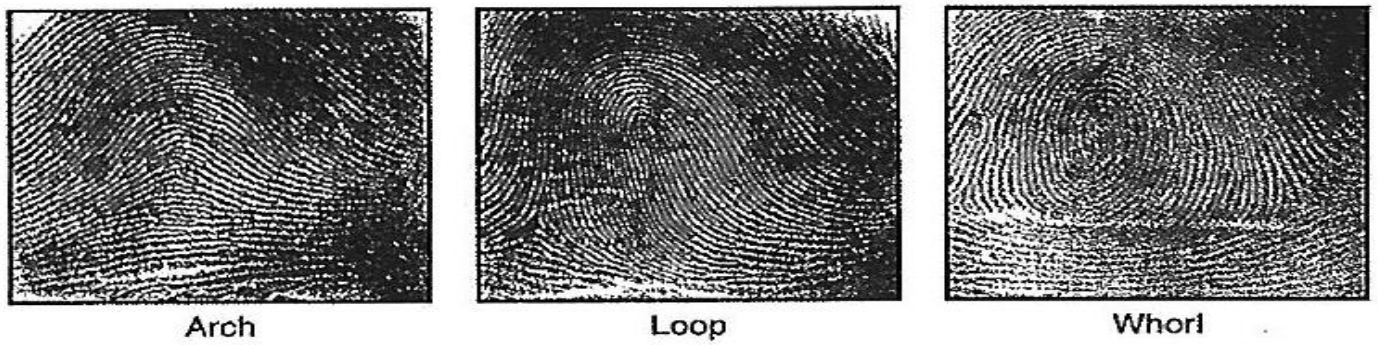
✓ The simplest approach involves identification of distinctive patterns formed by ridges.these are called as singularities.

✓ They are:arch,loop and whorls.

✓ Arch : the ridge starts from one side of the finger and forms an arc and ends on other side.

✓ Loop: the ridge starts and ends at the same side of the finger.

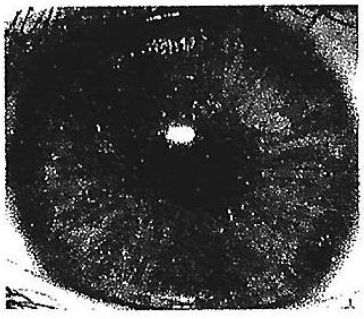
✓ Whorls:appear as closed cycles or spirals in a fingerprints.



9 *Fingerprint singularity patterns*

2. Iris scan

- ✓ The **iris is a thin opaque diaphragm of smooth muscle** situated in front of the lens in the human eye.
- ✓ Its annular shape surrounds the pupil.
- ✓ The intricate patterns on the iris appear to be unique.
- ✓ Two identical twins have iris pattern s that are different as those of two unrelated individuals.
- ✓ The patterns of an iris are also stable with age.



Iris pattern

Binomial Distribution use to model the distribution of distances between two distint iris

$$f(m) = \binom{n}{m} p^m (1-p)^{n-m}$$

$$= \frac{n!}{m!(n-m)!} p^m (1-p)^{n-m}$$

S **S**

INTRODUCTION

- Developed by Netscape in 1994, the **Secure Sockets Layer (SSL)** protocol has emerged as the principal means of ***securing communications between an Internet client (such as a browser) and a server.***
- It was standardized by IETF in 1999 and called **Transport Layer Security (TLS).**

SSL (Secure Sockets Layer)

- SSL is sandwiched between ***TCP (it only runs over TCP) and an application layer protocol.***
- It is application protocol independent.
- Protocols such as HTTP, FTP, SMTP, IMAP, and POP can all be run over SSL.
- Application protocols **secured by SSL are usually suffixed by an "S"** and run on different port numbers.
- For example, ***HTTP runs on port 80 but HTTPS runs on port 443***.
- FTP runs on ***port 21 but FTPS runs on port 990.***

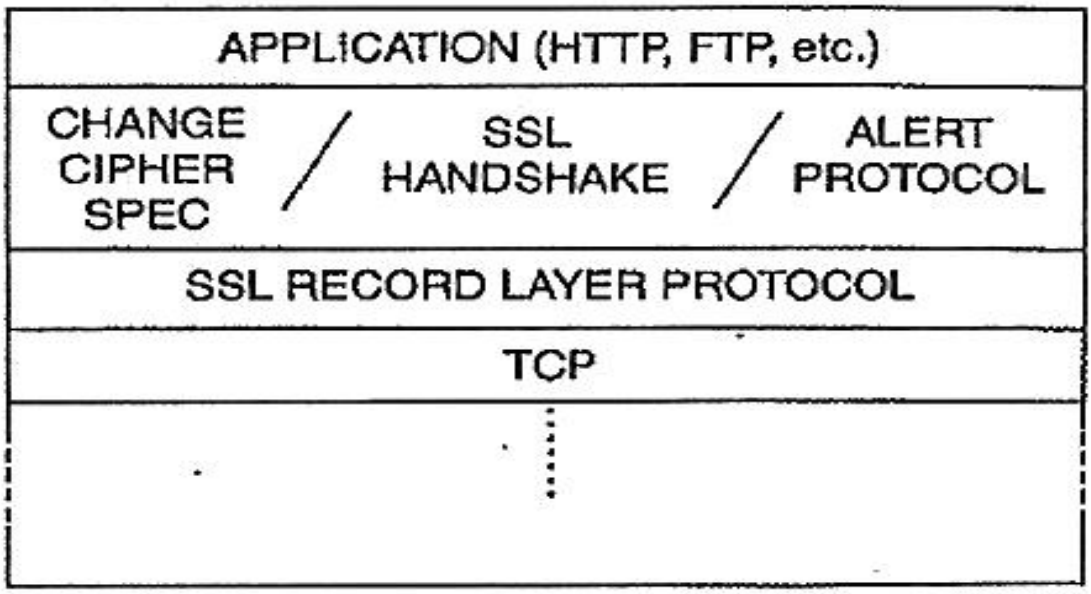


Figure 14.1 *SSL on the protocol stack*

- **SSL is comprised of two main protocols (see Fig. 14.1)**

1. **The Handshake Protocol :**
   - The SSL handshake protocol is used to negotiate the set of algorithms to be used for securing the communication link.

- **Server authentication** in SSL is mandatory and performed as part of the **handshake**.
- The hand-shake protocol is also responsible for deriving keys, for encryption and MAC computation

2. The Record Layer Protocol

- The actual job of providing ***message authentication + integrity checking and encryption*** is performed by the **SSL record layer protocol.**
- It sits just below the handshake protocol and **protects** each message exchanged by the two communicating parties.
- The record layer protocol also **detects *replayed, re-ordered, and duplicate packets.***

SSL HANDSHAKE PROTOCOL

Steps in the Handshake

- The client initiates a handshake with the server to either
  (a) Start a new session or
  (b) Resume an existing session or
  (c) Establish a new connection within an existing session.
- The main steps in the SSL handshake for establishing a new session are as follows:
  (1) Agreement on a ***common cipher suite*** to be used in the new session.
  (2) Receipt and validation of the ***server certificate*** by the client.
  (3) Communication of a ***"pre-master secret"*** and computation of **derived secrets**.
  (4) ***Integrity verification*** of handshake messages and ***server authentication***
- These steps are realized by the sequence of messages shown in below figure
- **The steps are:**
  - ***Step 1:*** **Two messages are communicated in this step —*Client Hello* and *Server Hello.***
    The following decisions are taken here:
    - Should a new session be established or should an existing one be re-used?

- For a **new session** the session ID field in the Client Hello message is **0**; else the field is set to the ID of the session to be re-used,
- The session ID field in the Server Hello message is the ID of the new session to be established or the ID of an existing session.

➢ The algorithm to be used in computing the MAC for message integrity include **MD5** and **SHA-1**.

➢ The key exchange method used for communicating the pre-master secret.

➢ In addition to agreeing on a cipher suite, both sides choose and exchange two 32-byte *nonces*, $\mathbf{R_A}$ and $\mathbf{R_B}$**,** in this step.

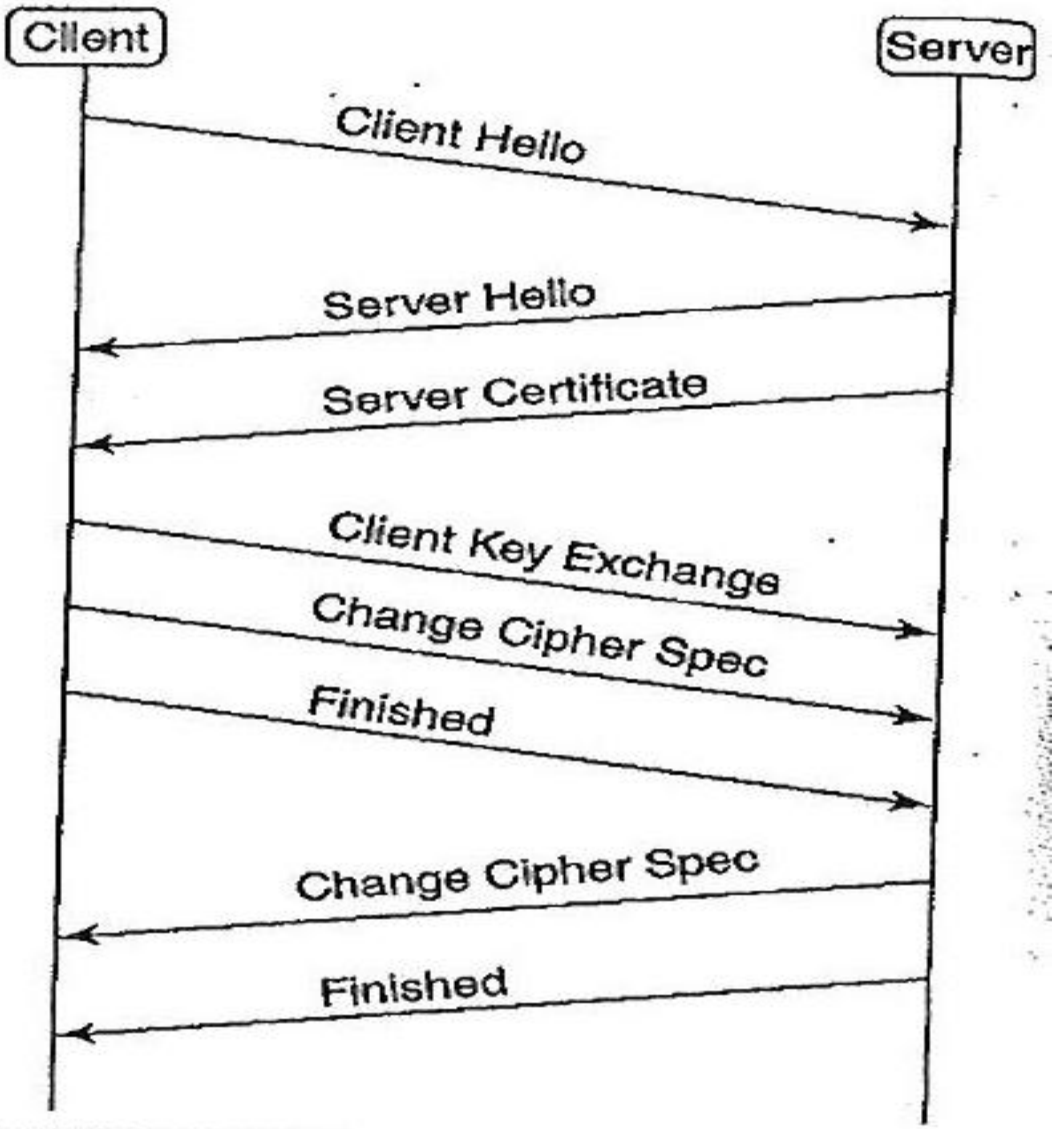


Figure 14.2 *SSL handshake*

➢ ***Step* 2. The server communicates its *certificate* to the client (see Fig. 14.2).**

- On receipt of the certificate, the client checks the owner's name/URL and validity period.
- It also verifies the signature of the CA on the certificate.
- Successful verification of these fields does not guarantee the authenticity of the sender
- Authentication of the server only occurs at the end of Step 4,

- ***Step 3.***
- The client chooses a *pre-master secret* — a 48-byte random number.
- The **pre-master secret** is encrypted with the server's public key and sent to the server in the Client key exchange messages.
- Thereafter, both client and server compute the **master secret**. This is an HMC style function, f , of the pre master secret, the two nounces exchanged in step 1 and some pre defined constants.
- The computation uses a standard cryptographic hash function such as the SHA-1 or the MDS.

Master_Secret = f(Pre-Master_Secret RA, RB, constants)

- Finally six secrets are derived using HMAC-style functions of the master secret, the two nonces, anddifferent pre-defined constants

Derived_Secret_i =

f(Master_Secret, RA, RB, constants), 1<i<6

The six derived secrets are:

- Initialization vector for encrypting messages from client to server
- Initialization vector for encrypting messages from server to client
- Secret key for encrypting messages from client to server
- Secret key for encrypting messages from server to client
- Secret for computing keyed hash on messages from client to server(Client MAC Secret)
- Secret for computing keyed hash on messages from server to client (Server MAC Secret)

Step 4: **This step involves the exchange of two messages in each direction.**

- The first of these is the **"Change_Cipher_Spec"** message (Fig. 14.2).

- The party that sends this message signals that from now on the cipher suite and the keys computed will be used.
- The second message in this step is the "Finished" message.
- This message includes a keyed hash on the concatenation of *all* the **handshake messages** sent in the preceding steps + a pre-defined constant.
- The keyed hash serves as an ***integrity check*** on the previous handshake messages.
- After the server receives the "Change_Cipher_Spec" and "Finished" messages from the client, it verifies the computation of the keyed hash.

- It then computes its own keyed hash that covers the previous handshake messages + a pre-defined constant, which is distinct from the one used by the client.
- The client receives the keyed hash and verifies it. Only at this point is the server authenticated to the client.
- On the other hand, client authentication as part of the SSL handshake is optional.

Key Design Ideas

Key Exchange Methods

- In Step 2, the server dispatches its certificate so the client can use the public key contained in the certificate to encrypt the pre-master secret.
- In some cases, however, the server's certificate may be a "signature-only certificate."
- This means that the public key in the certificate and the corresponding private key may only be used exclusively for signature generation/verification, not for encryption.
- In that case, SSL permits the server to create a **temporary public key/private key pair**. The public key (including modulus) are signed by the server using the private key corresponding to the public key in the **signature-only certificate.**
- The signed public key and certificate are communicated by the server to the client.
- The client verifies the signature on the public key and then uses it to encrypt the pre-master secret.
- SSL offers a rich set of options for key exchange.
- Such as RSA-based key exchange methods, Diffie—Hellman key exchange may be used.

Server Authentication

- The MAC computed by both parties and sent in step 4 is used as an ***integrity check*** on the previous handshake messages.
- All the handshake messages are sent in the clear (except for encryption of the pre-master secret).
- It is possible for an attacker to alter one or more of the handshake messages.
- For example, he may replace **128-bit DES by a 56-bit DES.**
- This may induce both parties to use a weaker cipher, which can be compromised by the attacker.
- The MAC detects any modification in the handshake messages.
- The hash computed by the server and verified by the client uses the *server* MAC s*ecret*,
- It is a function of the master secret which in turn is a function of the pre-master secret.
- Recall that the pre-master secret is chosen by the client and encrypted with the server's public key so that the server alone can read it. So, nobody but the server and client could compute the six secrets.
- Only after the client receives and verifies the keyed hash from the server, is it convinced that it is talking to the authentic server.

Sessions and Connections

- It is good security practice to **change keys** during a long-lasting session.
- SSL has provision for changing keys by creating **new connections** within an existing session.
- In creating a new connection, the pre-master secret which is part of the existing session state is *not* chosen a new.
- Instead, a new master secret is computed as a function of the *existing pre_ master secret* and two *fresh notices* contributed by the client and server.
- The *session state* includes the *pre-master secret*, the negotiated *cipher suite* and, of course, the *session* ID.
- The *state of a connection* includes the two *nonces*, the *master secret*, the six *derived secrets*, and two message *sequence* (one for each direction of message transfer).

SSL RECORD LAYER PROTOCOL

- The SSL record layer protocol is used to securely transmit data using the negotiated cipher suite and the keys derived during the SSL handshake.
- Its main tasks are computation of a **per-message MAC and encryption**.
- If the data to be transmitted is very large, it performs fragmentation.
- Each fragment is **16 kb or less**.
- When a connection is established, both sides initialize a sequence counter to **zero**.
- The counter is **incremented** for each packet sent.
- The sequence number itself is not sent. However, it is used in the computation of the **MAC** (at the sender) and in its verification (at the receiver).
- The MAC is computed on the concatenation of the 64-bit sequence number and the compressed fragment (if compression is used).
- The next step after computing the MAC is **encryption**.
- If the combined size of the data fragment and MAC is not a multiple of block size, a pad is appended.
- The data fragment, MAC, and pad (if any) are then encrypted, prepended with a header, and passed on to the TCP layer for further processing.
- The SSL record layer protocol header: there is a 1-byte Content Type field, which identifies the higher layer protocol used to process the fragment.
- Two bytes are used to specify the Version number.
- Finally, the Length field indicates the fragment size in bytes.

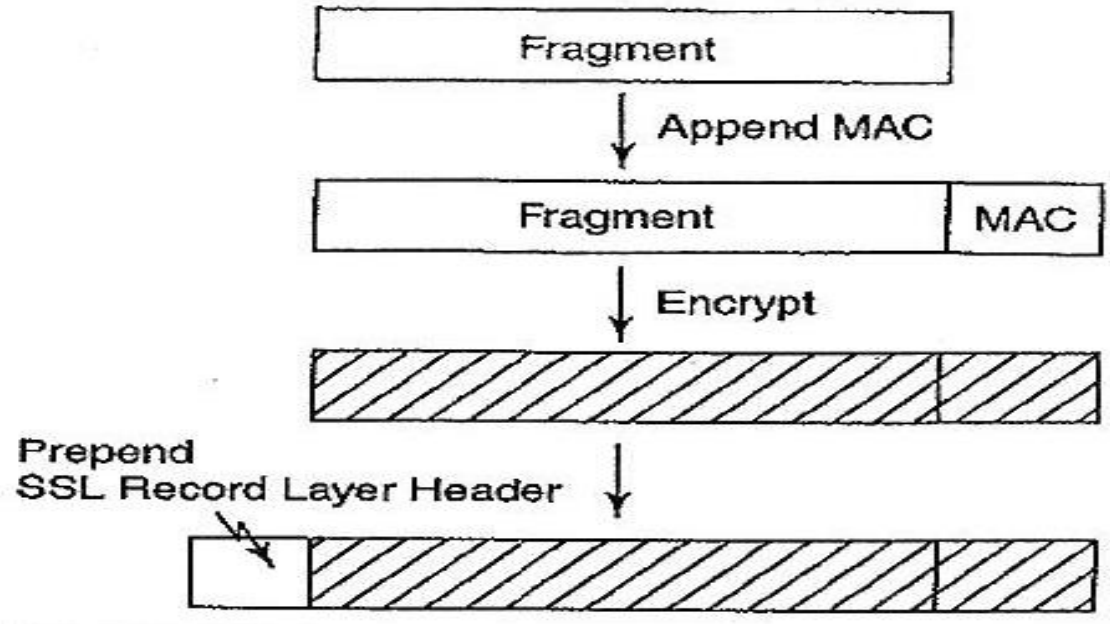


Figure 14.3 *Function of the SSL record layer protocol*

OpenSSL

- **OpenSSL** is open source software that implements the SSL/TLS protocol.
- It is comprised of a number of libraries that implement various cryptographic algorithms.
- It provides extensive support for communicating and validating digital certificates.
- OpenSSL is based on the **SSLeay** library developed by Eric A. Young and Tim J. Hudson.
- OpenSSL enhances the productivity of application developers by providing a rich set of APIs that handle diverse aspects of SSL-enabled communication from connection set-up and tear-down to certificate storage, management, and verification.
- The developers can rely on the OpenSSL APIs to implement the required security.

- Wireless networks present formidable challenges in the area of security.
- The open nature of such networks makes it relatively easy to sniff packets or even modify and inject malicious packets into the network.
- The ease with which such attacks are launched necessitates careful design and deployment of security protocols for wireless networks.

1. BACKGROUND

Wired network

➢ In many organizations, the wired network is an Ethernet LAN with an existing security infrastructure that includes an authentication server (AS).

➢ ***AAA (Authentication/Authorization/Accounting)*** functionality is often provided by a RADIUS (Remote Authentication Dial in User Service) server.

WLANs (wireless LANs)

- There are ***two principal types of WLANs*** —

1. ***Ad hoc networks: where*** stations (possibly mobile) communicate directly with each other.

2. ***Infrastructure WLANs:*** which use an access point (AP) as shown in below figure.

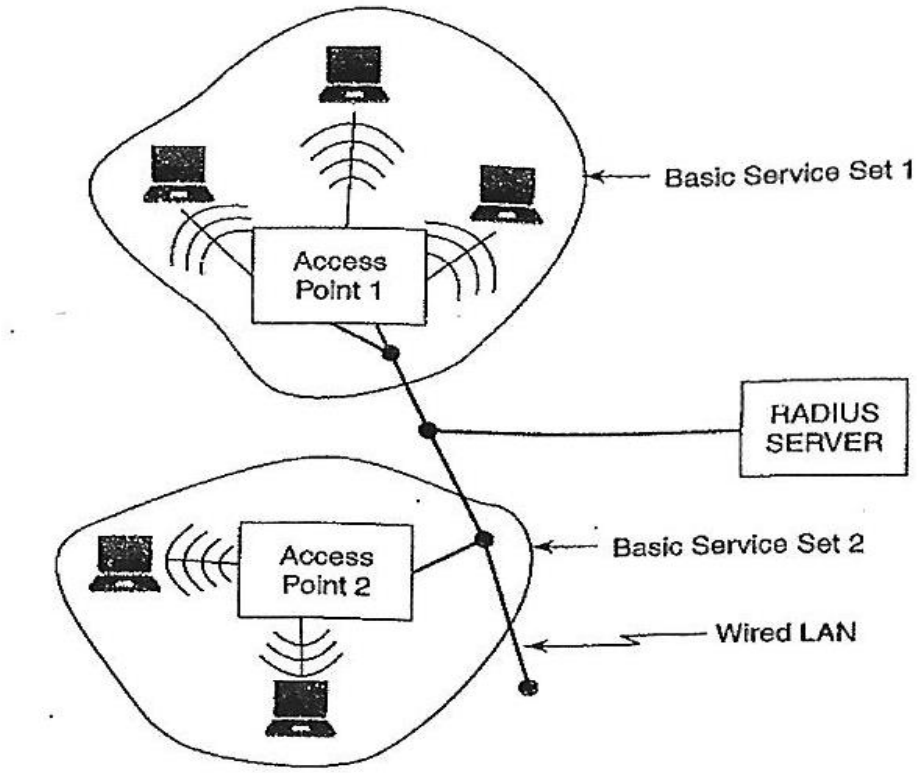


Figure 15.1 *Infrastructure wireless LAN*

Infrastructure WLANs :

➢ A station first, sends a frame to an AP and the AP then delivers it to its final destination.

➢ The destination may be another wireless station or it may be a station on the wired network that the AP is connected to.

➢ The **AP** thus serves as a **bridge** between the WLAN and the existing wired network.

➢ The challenge then is to develop protocols that seamlessly integrate the WLAN with the security infrastructure of the wired network.

➢ A network of wireless stations associated with an AP is referred to as a ***basic service set.*** Such a network may be adequate for a home or small enterprise.

➢ The union of the basic service sets comprises an ***extended service set (ESS).***

➢ Each station and AP in the ESS is uniquely identified by a MAC address — ***a 48-bit quantity.***

➢ Each AP is also identified by an ***SSID (service set ID),*** which is a character string of length at most ***32 characters.***

➢ A wireless station, on power-up, needs to first discover an AP within its range.

- This can be done by monitoring the wireless medium for a special kind of frame called a ***Beacon,*** which is periodically **broadcast by the AP.**
- The Beacon usually contains the ***SSID*** of the broadcasting AP.
- Alternatively, a station may send a ***Probe Request frame,*** which probes for APs within its range.
- An AP, on hearing such a request, responds with a Probe Response frame.
- Like the Beacon, the ***Probe Response frame*** contains the SSID of the AP and also information about its capabilities, supported data rates, etc.

➢ A station that wishes to associate with an AP sends it an ***Associate Request frame.***

➢ The AP replies with an ***Associate Response frame*** if it accepts the request for associating with it.

1. The earliest protocol that incorporated security in WiFi was **WEP (wired equivalent privacy).**

✓ Designed to provide authentication/access control, data integrity, and confidentiality, it failed on all three counts.

2. WiFi Protected Access (WPA)

✓ WPA was intended to fix the shortcomings of WEP without requiring new wireless network cards.

✓ But WPA is not perfect — it too is susceptible to attacks on its cryptographic algorithms.

3. WPA2

✓ All the deficiencies in WEP have been addressed in the IEEE 802.11i (implemented in WPA2).

2. AUTHENTICATION

2.1Pre-WEP Authentication

1. Early versions of 802.11 use naïve approaches: knowledge of SSID sufficed for a station to be authenticated to the AP

➢ Drawbacks:

- An attacker could easily sniff the value of SSID from frames such as the beacon or probe response and then use it for authentication.

2. Another approach was to restrict admission to the WLAN by MAC address.

✓ The AP would maintain a list of MAC addresses (access control list) of stations permitted to join the WLAN.

✓ valid MAC addresses could be obtained by sniffing the wireless medium.

✓ The attacker could then modify his network card to spoof a valid MAC address. So, neither of these approaches was truly secure.

2.2 Authentication in WEP

➢ In WEP, the station authenticates itself to the AP using a challenge—response protocol.

➢ Basically, the AP generates a challenge (nonce) and sends it to the station.

➢ The station encrypts the challenge and sends it to the AP.

➢ The stream cipher, RC4, is used for encryption.

➢ **Response From Station**: the station computes a key stream, which is a function of a 40-bit shared secret, S, and a 24-bit Initialization Vector (IV).

➢ The challenge is then XORed with the keystream to create the response.

RESPONSE = CHALLENGE (XOR) KEYSTREAM(S, IV)

➢ The response together with the IV is sent by the station to the AP.

➢ The shared secret, S, is common to all stations authorized to use the WLAN.

Drawbacks:

➢ All an attacker needs to do is to monitor a challenge—response pair.

➢ From this, he can compute the keystream.

➢ To authenticate himself to the AP, he needs to XOR the challenge from the AP with the computed keystream.

➢ It may also be possible for an attacker to obtain S itself.

➢ By eavesdropping on several challenge—response pairs between the AP and various stations, an attacker could launch a **dictionary attack** and eventually obtain S.

2.3 Authentication and key agreement in 802.11

Authentication

➢ 802.11i uses IEEE 802.1x — a protocol that supports authentication at the link layer.

➢ Three entities are involved:

1. **Supplicant (the wireless station).**
2. **Authenticator (the AP in our case).**
3. **Authentication server.**

➢ Different authentication mechanisms and message types are defined by the ***Extensible authentication Protocol (EAP)*** standardized by Internet Engineering Task Force (IETF).

➢ EAP is not really an authentication protocol but rather a ***framework*** upon which various authentication protocols can be supported.

➢ EAP exchanges are mostly comprised of **requests and responses.**

➢ For example one party requests the ID of another party.

➢ The latter responds with its user_name or e-mail address.

➢ EAP also defines messages that may contain **challenges** and **responses** used in **authentication protocols**.

➢ The AP broadcasts its security capabilities in the Beacon or Probe Response frames.

➢ The station uses the Associate Request frame to communicate its security capabilities.

➢ 802.11i authentication takes place after the station associates with an AP.

IEEE 802.11i

➢ The generic authentication messages in IEEE 802.11i are shown in Fig. 15.2.

➢ The protocol used between the ***station and the AP is EAP*** but that used between the AP and the authentication server depends upon the **specifics**.

➢ For example, the authentication server is often a RADIUS server which uses its **own message types and formats.** (RADIUS stands for Remote Authentication Dial in User Service. It is a client—server protocol used for authentication, authorization, and accounting.)

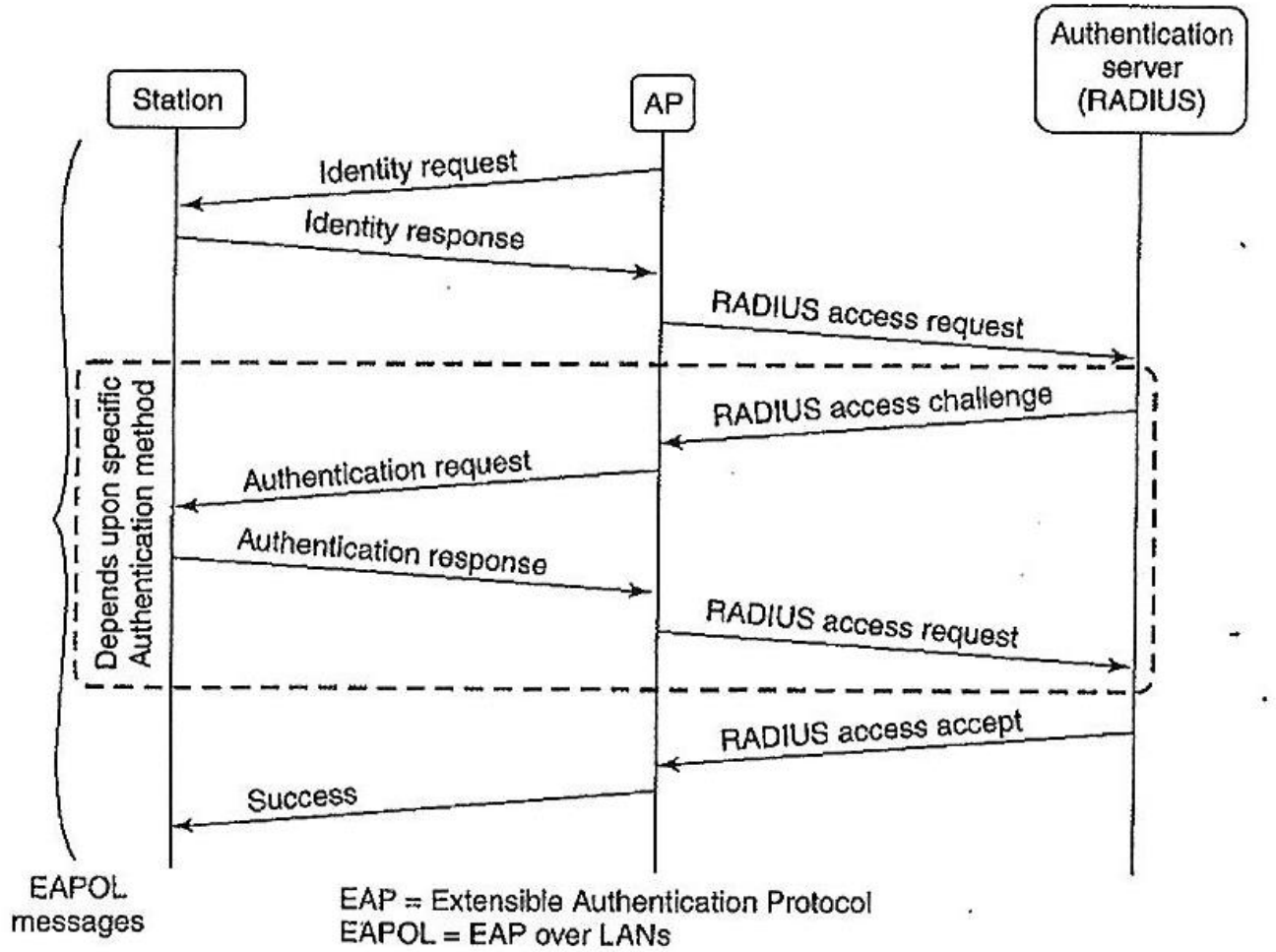


Figure 15.2 *Authentication and master session key exchange in 802.11i*

EAP = Extensible Authentication Protocol messages
EAPOL = EAP over LANs

➢ The main authentication methods supported by EAP include the following:

1. EAP-MDS

2. EAP-TLS

3. EAP-TTLS

4. **EAP-PEAP**

1. EAP-MDS

✓ This is most basic of the EAP authentication methods.

✓ Here, the authentication server challenges the **station** to transmit the MD5 hash of the user's password.

✓ The station prompts the user to type his/her password.

✓ It then computes the hash of the password and sends this across.

✓ This method is insecure since an attacker could eavesdrop on such a message exchange and then **replay** the hashed password thus impersonating the owner of the password.

✓ Also, this method does not support authentication of the AP to the station.

2. EAP-TLS

✓ EAP-TLS is based on the SSL/TLS protocol

✓ most secure and provides mutual authentication and agreement on a ***master session key.***

✓ It requires the AP as well as the user (station) to have digital certificates.

✓ It is relatively straightforward to equip each AP with a digital certificate and a corresponding private key but extending the via to each user of the WLAN may not be feasible.

3. EAP-TTLS

✓ (tunnelled TLS) requires certificates only at the AP end.

✓ The AP authenticates itself to the station and both sides construct a secure tunnel between themselves.

✓ Over this secure tunnel, the station authenticates itself to the AP.

✓ The station could transmit **attribute-value** pairs such as

user_name = akshay
password = 4rP#mNaS&7

4 Protected EAP (PEAP)

✓ This was proposed by Microsoft, Cisco, and RSA Security, is very similar to EAP-TTLS.

✓ In PEAP, the secure tunnel is used to start a second EAP exchange where in the station authenticates itself to the authentication server.

✓ The enhanced security offered by EAP-TLS, EAP-TTLS, and PEAP does, however, come at a steep price in performance measured by the message and computational overheads incurred during authentication.

Key Hierarchy

➢ There are two types of keys used in WLANs.

➢ The first are ***pairwise keys*** used to protect traffic between a station and an AP.

➢ The second type of key is the ***group key*** intended to protect broadcast or multicast traffic between an AP and multiple stations.

The hierarchy of 802.11i keys:

➢ The root of the key hierarchy is the ***Pairwise Master Key*** **(PMK).**

✓ This is obtained in one of two ways

✓ The station and the authentication server may agree on a Master Session Key (MSK) as part a of the authentication procedure.

✓ The authentication server communicates this key to AP

✓ The AP and station then derive the PMK from the MSK.

➢ An alternative to computing a fresh PMK for each session is the ***Pre-Shared Key, (PSK),*** which is used as the PMK.

➢ Pairwise Transient Key (PTK).

✓ The 256-bit PMK is used to *derive* a ***384-bit pairwise Transient Key (PTK).***

✓ The PTK is a pseudo random function of the PMK.

✓ PRF(nonce of AP,nonce of station,MAC address of AP.MAC address of station, PMK).

➢ Three **128 bit** chunks are extracted from the 384 bit PTK for the following purposes:

1. A *Temporal Key* (TK) is used for both encryption and integrity protection of data between the AP and the station.
2. A *Key Confirmation Key* (KCK): Integrity protection is supported by a MAC computed as a function of the message and the KCK.
3. A *Key Encryption Key* (KEK) is used to encrypt the message containing the group key.

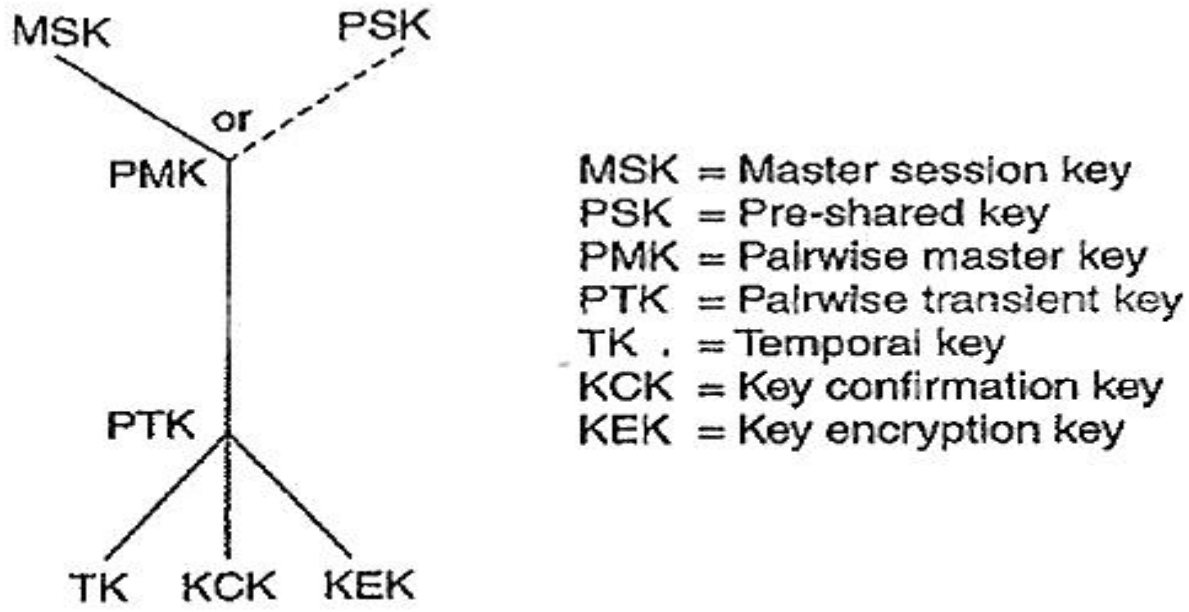


Figure 15.3 *Key hierarchy in 802.11i*

Four-way Handshake

➢ The main goals of the four-way handshake are to

(a) Derive the PTK from the PMK,

(b) Verify the cipher suites communicated in the Beacon and Associate Request Frames and

(c) Communicate the group keys from the AP to the station.

➢ Figure 15.4 shows the messages comprising the four-way handshake.

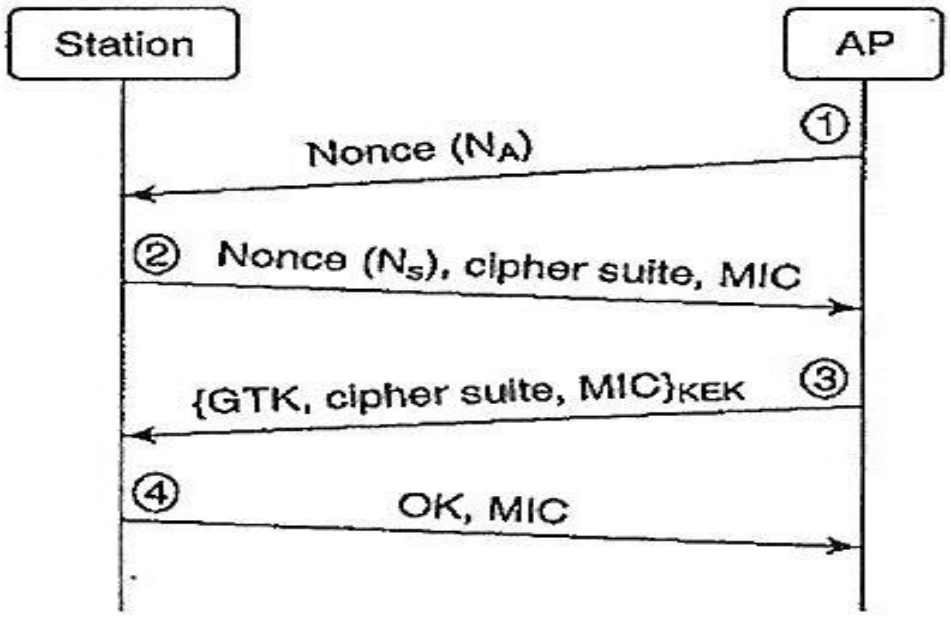


Figure 15.4 *Four–way handshake in 802.11i*

1. Message 1: The AP first sends a nonce, N_A, to the station.
2. Message 2:

   ✓ The station chooses a nonce, N_s station computes the PTK as follows

   ✓ $PTK = \text{prf}(PMK, N_A, N_s, MAC_A, MAC_S)$...

   ✓ The PTK is a pseudo-random function (prf) of the PMK, the MAC addresses of the station and AP and nonces contributed by the station and the AP.

   ✓ The two nonces help prevent replay attacks.

   ✓ Three 128-bit keys — **TK, KCK, and KEK** are extracted from the 384-bit PTK (Fig. 15.3).

   ✓ The station sends nonce, cipher suite and uses KCK to compute MIC (message integrity check).
3. Message 3:

   ✓ On receiving the message 2, AP computes the PTK from the above expression used by the station. Extracts TK,KCK,KEK.

   ✓ AP verifies the **integrity** and **source** of **message 2** using the key KCK.

   ✓ Message 3 contains group transient key (GTK), this is the key used by the AP and all stations to **integrity protect** (and all optionally encrypt) all **multicast** and **broadcast**.

   - Message 3 also contains cipher suite and the message will be encrypted using the KEK and its integrity is protected using KCK.
4. Message 4:

   ✓ This is an acknowledgement from the station that it has received the previous messages without error.

   ✓ It is a signal to the AP that hence forth all messages will be integrity-protected and encrypted with the TK.

1. CONFIDENTIALITY AND INTEGRITY

Data Protection in WEP **(wired equivalent privacy).**

➢ WEP was designed to provide message confidentiality, integrity, and access control but it failed on all three counts.

➢ In this section, we show how plaintext can be recovered and messages can be modified due to flawed design decisions in WEP.

➢ There are many lessons to be learned from WEP — the most important being how not to design protocols for security.

1.1 **WEP Encryption and Integrity Checking**

➢ WEP uses the stream cipher, RC4, for encrypting messages.

➢ It generates a pseudo-random **keystream**, KS, which is a function of a static secret shared between the two communicating parties.

➢ In order to have KS vary from message to message, a random per-message initialization vector, IV, is also used to generate KS.

➢ Early implementations of WEP used a 40-bit secret, S, concatenated with a 24-bit IV to create, in effect, a "64-bit key."

➢ KS is xor[ed] with the plaintext, P, to obtain the ciphertext, C or

$$C = \mathcal{P} \oplus KS(S, IV) \qquad (15.2)$$

The plaintext includes

- Message to be send
- Integrity: which is a 32 bit checksum computed on the message.
- The IV chosen by the sender is included in each frame as shown below

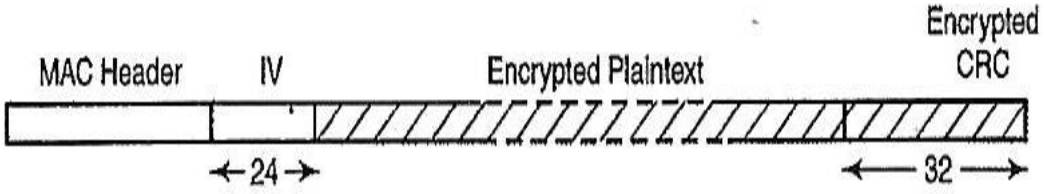


Figure 15.5 *WEP frame*

- The plaintext p is obtained as follows:

The receiver will generates KS from the shared secret Sand the IV retrieved from the received frame. It recovers the plain text from the following equation

$$\mathcal{P} = C \oplus KS\ (S,\ IV) \qquad (15.3)$$

Known plaintext attack

➢ The first problem with WEP is the possibility of keystream re-use.

➢ Since the IV is 24 bits in length, there are only 224 distinct keystreams that could be constructed given a secret S.

➢ Suppose an attacker finds two frames which were encrypted using the same IV.

➢ Let their ciphertexts be C and C'.

➢ Let the corresponding plaintexts be P and P'. using

Eq. (15.2), it follows that:

$$\mathcal{P} \oplus \mathcal{P}' = C \oplus C'$$

So

$$\mathcal{P}' = \mathcal{P} \oplus C \oplus C'$$

➢ Thus knowing c,c', and p, we can obtain p' which is called as known plaintext attack.

Message modification

➢ Consider an attacker who wishes to modify a message sent by a legitimate user.

➢ Let the sender's plaintext (not including the CRC checksum) be M1 F M2 where M1, F, and M2 are each binary strings.

➢ The attacker wishes to substitute the substring, F, with another substring, F',

➢ so that the decrypted message seen by the receiver is M1 F' M2. The attacker does not need to know the values, M1 and M2. However, we assume that he knows F and F'.

➢ Ideally, the message integrity check should detect any modification to an existing message. Can the attacker modify the message (including checksum) in such a way so that the modification is undetected at the receiver end?

➢ For the above plaintext, the ciphertext computed by the sender is:

$$((M_1\ F\ M_2) \parallel CRC(M_1\ F\ M_2)) \oplus KS$$

The attacker intercepts the ciphertext and performs the following operations:

1. He first constructs the string, $0^{|M_1|} \| (F \oplus F') \| 0^{|M_2|}$. Here, $0^{|M_1|}$ is a string of $|M_1|$ zeros where $|x|$ is the length of the substring x.
2. He then computes the CRC on this string, $\text{CRC}(0^{|M_1|} \| (F \oplus F') \| 0^{|M_2|})$.
3. He finally XORs the original ciphertext with $0^{|M_1|} \| (F \oplus F') \| 0^{|M_2|} \| \text{CRC}(0^{|M_1|} \| (F \oplus F') \| 0^{|M_2|})$.

These computations yield

$$\begin{aligned}
&((M_1\ F\ M_2) \| \text{CRC}(M_1\ F\ M_2)) \\
&\quad \oplus KS \\
&\quad \oplus ((0^{|M_1|} \| (F \oplus F') \| 0^{|M_2|} \| \text{CRC}(0^{|M_1|} \| (F \oplus F') \| 0^{|M_2|})) \\
&\quad = ((M_1 \oplus 0^{|M_1|}) \| (F \oplus (F \oplus F')) \| (M_2 \oplus 0^{|M_2|})) \\
&\quad \| (\text{CRC}(M_1\ F\ M_2) \oplus \text{CRC}(0^{|M_1|} \| F \oplus F' \| 0^{|M_2|})) \\
&\quad \oplus KS \\
&\quad = ((M_1\ F'\ M_2) \| \text{CRC}(M_1\ F'\ M_2)) \oplus KS
\end{aligned}$$

The last step follows from the fact that the CRC is a linear operation, i.e.,

$$\text{CRC}(m_1 \oplus m_2) = \text{CRC}(m_1) \oplus \text{CRC}(m_2)$$

The receiver, on decrypting the ciphertext, obtains

$$(M_1\ F'\ M_2) \| \text{CRC}(M_1\ F'\ M_2)$$

➢ The modified message has a ***valid CRC*** and so passes the integrity check at the receiver.

➢ Hence, the receiver accepts the message, ***unaware that it has been modified by an attacker.***

1.2 Data protection in TKIP and CCMP

TKIP

➢ The technical name for WPA is *Temporal Key Integrity Protocol* (TKIP).

➢ By contrast, the encryption key in TKIP is 128 bits.

➢ TKIP generates a random and different encryption key for each frame sent. It employs a process called ***two-phase key mixing***.

➢ The inputs to this process are the 128-bit temporal key, TK, computed as part of the four-way handshake, the sender's MAC address and the four most significant bytes of a 48-bit *frame sequence counter.*

➢ The randomizing capability of the key mixing function and the large size of the key space virtually guarantee that "keystream collisions" never occur.

➢ Thus, known plaintext attacks that could be successfully launched on WEP have no chance of success with TKIP.

➢ The sequence counter is incremented for each frame sent.

➢ It is also carried in the header of each frame.

➢ This helps protect the receiver from ***replay attacks.***

➢ Figure 15.6 shows the two phases used in generating the **RC4 key.**

➢ Two pseudo-random function (PRF1 and PRF2) are employed in the two phases.

➢ The 32 most Significant bits of the sequence counter are input to PRF1.

➢ The least significant 16 bits of the sequence counter are inputs to PRF2 So, the output of PRF2 changes for each frame sent.

➢ The 64-bit message integrity check in TKIP called MIC or is non linear

$$\text{MIC}(m_1 \oplus m_2) \neq \text{MIC}(m_1) \oplus \text{MIC}(m_2)$$

➢ MIC is computed as a function of the data in the frame and also some fields in the MAC header such as the source and destination addresses.

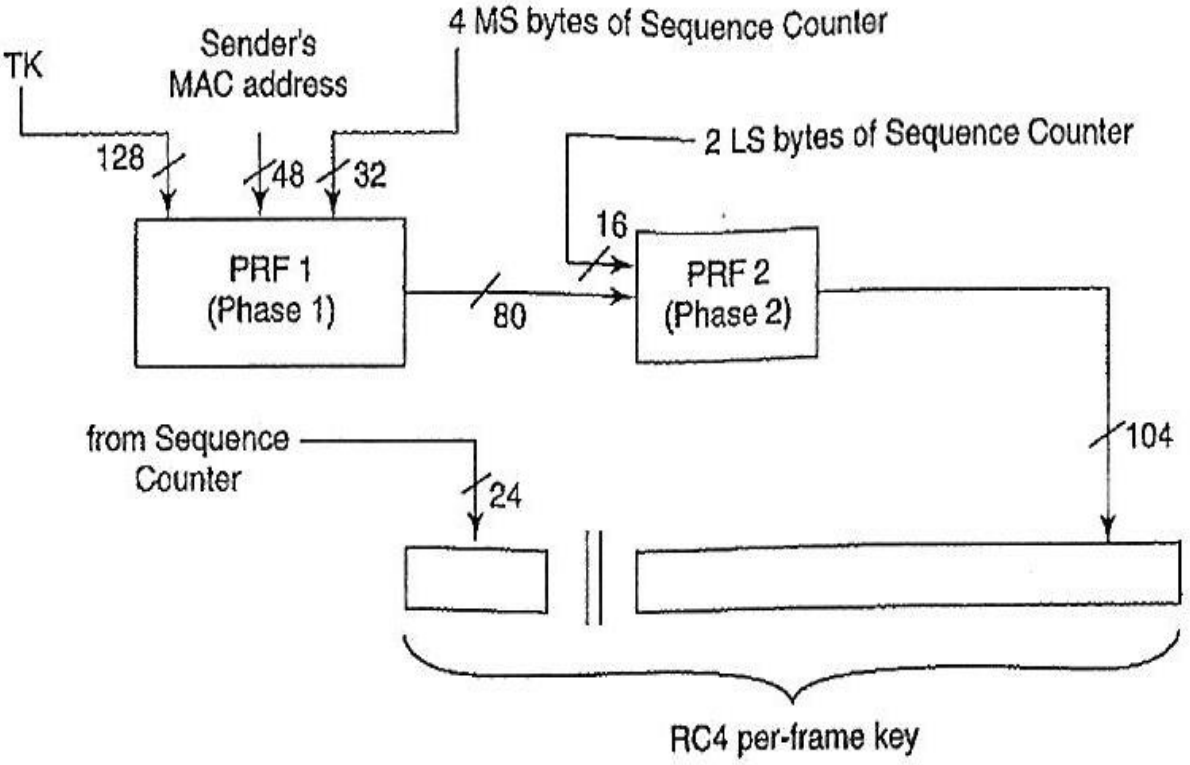


Figure 15.6 *Two-phase key mixing in TKIP*

➢ It also uses as input a key derived from the PTK which was computed during the four-way handshake.

➢ Due to design constraints on WEP cards, MIC's implementation uses simple logical functions, shifts, etc. Hence, it is not as secure as a keyed cryptographic hash.

➢ On the other hand, it is much better compared to the CRC checksum used in WEP.

CCMP

➢ The implementation of 802.11i that uses AES is referred to as WPA-2.
Its technical name is counter mode with CBC MAC protocol (CCMP).

➢ In CCMP terminology, this count is referred to as a packet number (PN).

➢ The count is maintained at both sender and receiver ends.

➢ The PN is also included in a special CCMP header field in a CCMP frame.

➢ The PN is incremented by the sender after each frame is sent.

➢ Upon receipt of a fresh frame in that session, the receiver compares the value of PN in the CCMP header versus the value stored by it.

➢ If the value is less than the stored value, the frame is likely to be a replayed frame and is hence discarded.

The first task in preparing a frame for transmission is to compute a MIC.

➢ The *MIC* is computed using **AES** in *Cipher Block Chaining (CBC) mode* with block **size 128 bits**.

➢ The **key** for performing encryption in each stage of Fig below is **TK**(temporal key).

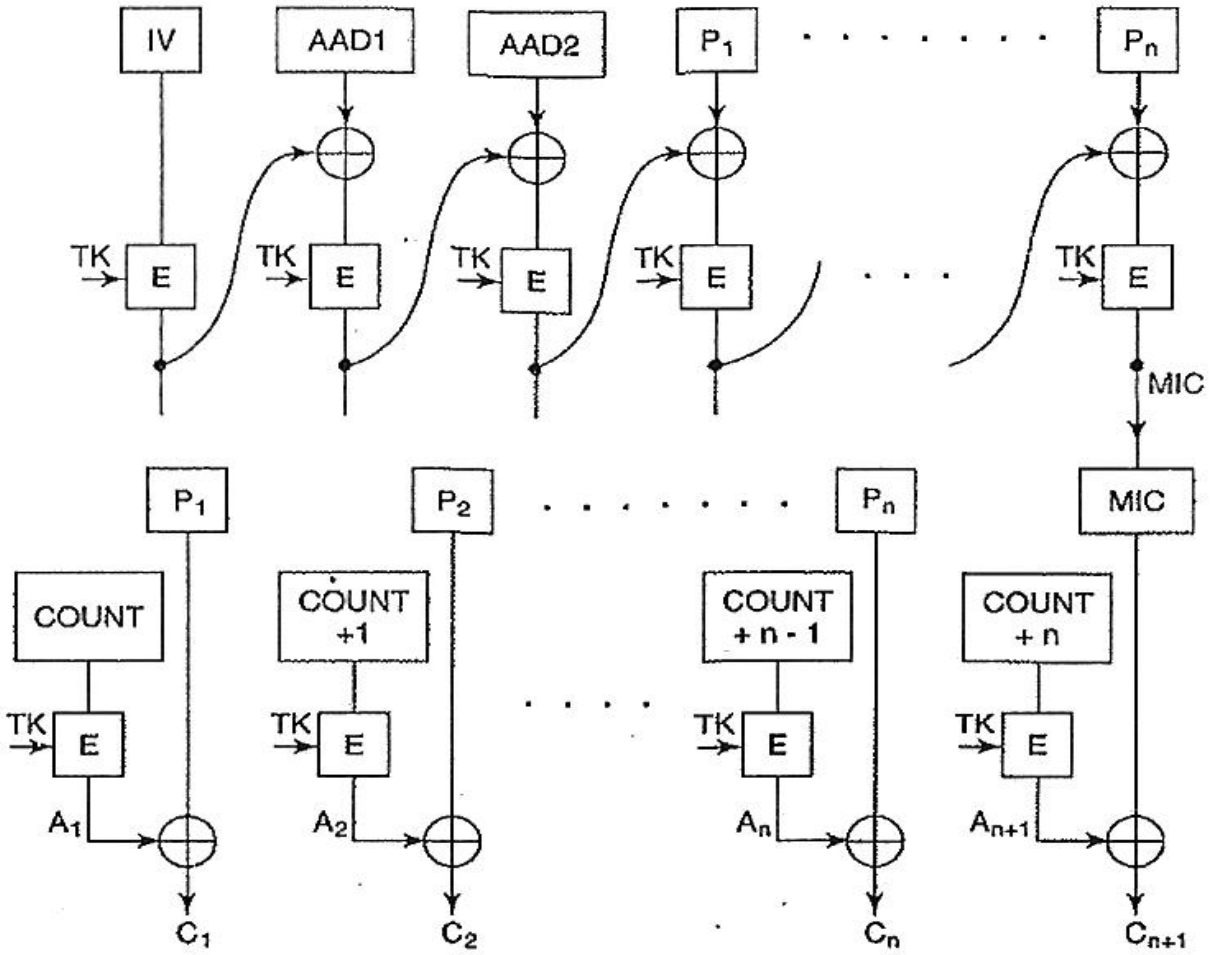
➢ The **IV** for the MIC computation is a "**nonce**," which includes the **48-bit PN**.

➢ The **second** and **third blocks** used in the **MIC computation** are specific fields in the frame header such as the **MAC addresses**, **sequence control**, and **frame type**.

➢ Next, the blocks in the frame data are sequentially processed resulting in an **8-byte MIC**.

The next step is encryption.

➢ The **frame data and the MIC are concatenated** and then **encrypted** using **AES** in **counter mode** (Fig. 15.7).



IV = Initialization Vector (includes 48-bit Packet Number)

AAD1, AAD2 = Additional Authentication Data (includes certain immutable fields of the MAC header)

COUNT is a function of the Packet Number

15.7 *MAC generation and encryption in CCMP*

➢ Let n be the total number of blocks in the ***frame body + MIC.***

➢ The procedure for encrypting the i-th block is:

- Compute Ai= ETK(PN +i*j). Here, PN is the packet number and j is a constant known to both sender and receiver.
- Compute i-th block of ciphertext = A (xor)Pi.

➢ Here, Pi is the i-th block of plaintext.

➢ The frame now includes two new fields — the CCMP header and the MIC.

➢ Upon receipt of the frame, the receiver reverses the operations performed by the sender.

➢ It performs decryption followed by MIC verification.

Firewalls

➢ ***Definition:*** A firewall acts as a ***security*** guard controlling access between an internal protected network and an external untrusted network based on a given security policy.

➢ Besides preventing intruders getting in, a firewall also helps prevent confidential inside data from getting out.

➢ A firewall may be implemented in hardware as a ***stand-alone "firewall appliance" or in software on a PC.***

➢ A single firewall may be adequate for small businesses and homes. However, in several large enterprises, multiple firewalls are deployed to achieve ***defence in depth.***

1. BASICS

1.1Firewall Functionality

The main functions of a firewall are listed as follows:

➢ ***Access Control:***

✓ A firewall filters incoming (from the Internet into the organization) as well as outgoing (from within the organization to the outside) packets.

✓ A firewall is said to be configured with a **rule set** based on which it decides which packets are to be allowed and which are to be dropped.

➢ ***Address/Port Translation.***

✓ NAT was initially devised to alleviate the serious shortage of IP addresses by providing a set of private addresses that could be used by system administrators on their internal networks but that are globally invalid (on the Internet).

✓ It is possible to conceal the addressing schema of these machines from the outside world through the use of NAT.

✓ Through NAT, internal machines, though not visible on the Internet, can establish a connection with external machines on the Internet. NATing is often done by firewalls.

➢ ***Logging.***

✓ A sound security architecture will ensure that each incoming or outgoing packet encounters at least one firewall.

✓ The firewall can log all anomalous packets or flows for later study.

✓ These logs are very useful for studying attempts at intrusion together with various worm and DDoS attacks.

➢ ***Authentication, Caching***, etc. Some types of firewalls perform authentication of external machines attempting to establish a connection with an internal machine.

➢ A special type of firewall called ***web proxy*** authenticates internal users attempting to access an external service. Such a firewall is also used to **cache** frequently requested webpages. This results in **decreased response time** to the client while saving communication bandwidth.

1.2 Policies and Access Control Lists

➢ **High-level policies for access to various types of services are formulated within an organization or campus. Examples of these include the following:**

✓ All received e-mail should be filtered for spam and viruses.

✓ All HTTP requests by external clients for access to authorized pages of the organization's website should be permitted.

✓ DNS queries made by external clients should be allowed provided they pertain to addresses of the organization's publicly accessible services such as the web server or the external e-mail server. However, queries related to the IP addresses of internal machines should not be entertained.

✓ The organization's employees should be allowed to remotely log into authorized internal machines. However, all such communication should be authenticated and encrypted.

✓ Only two **types** of **outgoing** traffic are permitted. First, all e-mail from within the organization to the outside world are permitted. Second, requests emanating from within the organization for external webpages are permitted. However, requests for pages from certain "**inappropriate**" websites should be **denied**.

➢ ***High-level policies are translated into a set of rules that comprise an Access Control List.***

➢ A rule specifies the action to be taken as a function of

(i) the packet's source IP address and port number

(ii) the packet's destination IP address and port number

(iii) the transport protocol in use (TCP or UDP)

(iv) ***the packet's direction — incoming or outgoing***

➢ The Access Control List for the high-level policies is described in Table 21.1.

➢ Policies can, in general, be either ***permissive or restrictive.***

➢ A permissive policy is defined as follows:

✓ ***Permit all packets except those that are explicitly forbidden.***

➢ A restrictive policy, on the other hand, is defined as follows:

- Drop all packets except those that are explicitly permitted.

➢ The ACL in Table 21.1 implements a restrictive policy — the default action is Deny as expressed in rules 5 and 8.

➢ **The rules are scanned top to bottom.**

➢ As soon as a rule is found' that matches the packet's attributes (IP addresses, port numbers, etc.), the action in that rule (usually permit or deny) is taken and no further rules are processed for that packet.

➢ The scanning order is important.

➢ For example, if rules 4 and 5 in Table 21.1 are interchanged, then IPSec traffic will be dropped.

➢ Also, from a performance perspective, it makes sense to put the most frequently acted upon rule earlier on.

➢ By so doing, we can expedite the decision on what to do with a packet.

➢ Finally, it is important to include the default deny rule at the end of the rule set — this prevents ambiguity over what action to take for a packet that has not been matched against the attributes in any of the previous rules.

Table 21.1 *Example access control list*

No.	In-bound (I) or out-bound (O)	Transport protocol	Src. IP addr.	Src. port	Dest. IP addr.	Dest port	Action	Comment
1	I	TCP	Any	Any	MS	25	Permit	Allow incoming e-mail
2	I	TCP	Any	Any	WS	80	Permit	Allow requests for organization's webpages
3	I	UDP	Any	Any	NS	53	Permit	Allow DNS queries
4	I	IPSec	Any	Any	*	*	Permit	Allow incoming VPN traffic
5	I	Any	Any	Any	Any	Any	Deny	Forbid all other incoming traffic
6	O	TCP	Any	Any	Any	25	Permit	Allow outgoing e-mail
7	O	TCP	Any	Any	*	80	Permit	Allow requests for external webpages
8	O	Any	Any	Any	Any	Any	Deny	Forbid all other outgoing traffic

Note: MS, WS, and NS are the IP addresses of the organization's e-Mail Server, Web Server, and DNS server (Name Server), respectively. * depends on configuration

1.3 Firewall Types

➢ Firewalls can be classified into the categories

1. **Packet Filters**
2. **Stateful Inspection**
3. **Application Level Firewalls**

1. Packet Filters

➢ This involves checking for matches in the IP, TCP, or UDP headers.

➢ For example, it may be necessary to check whether a packet carries a certain specific source or destination IP address or port number.

➢ It is often performed by the border router or access router that connects the organization's network to the Internet.

➢ In effect, the border router becomes the first line of defence against malicious incoming packets.

➢ why the packet filtering firewall is inadequate????

Drawbacks:

➢ Consider an external mail server (IP address = ABC) that wishes to deliver mail to an organization.

➢ For this purpose, it should first establish a TCP connection with the organization's mail server, MS.

➢ Consider the arrival of a packet with the following attributes:

Source IP address = ABC
Destination IP address = MS
TCP destination port = 25 (SMTP port)
ACK flag set

➢ Such a packet would be part of a normal flow provided a connection between ABC to MS has been established. But suppose such a connection has not yet been established.

➢ Should the packet still be allowed in? The simple packet filter will allow the packet to enter even if no prior connection between ABC and MS was established.

➢ It should be noted that such packets are often used to perform port scans.

➢ A simple packet filter merely inspects the headers of an incoming packet in isolation. It does view a packet as part of a connection or flow. Hence, it will not be able to filter out such pack `'t arriving from ABC.

2. Stateful Inspection

➢ A firewall uses packet's TCP flags and sequence/acknowledgement numbers to determine whether it is part of an existing, authorized flow.

➢ If it is participating in the establishment of an authorized connection or if it is already part of an existing connection, the packet is permitted, otherwise it is dropped.

➢ In the above example of the packet from ABC, the stateful packet inspection firewall will realize that it has not encountered the first two packets in the three-way handshake and will hence drop this packet.

3. Application Level Firewalls

➢ A packet-filtering firewall, even with the added functionality of stateful packet inspection, is still severely limited.

➢ What is needed is a firewall that can examine the application payload and scans packets for worms, viruses, spam mail, and inappropriate content. Such a device is called a ***deep inspection firewall.***

➢ A special kind of application-level firewall is built using proxy agents. Such a ***"proxy firewall" acts as an intermediary*** between the client and server.

➢ The client establishes a TCP connection to the proxy and the proxy establishes another TCP connection with the server as shown in Fig. 21.1.

➢ To a client, the proxy appears as the server and to the server, the proxy appears as the client. Since there is no direct connection between the client and the server, worms and other malware will not be able to pass between the two, assuming that the proxy can detect and filter out the malware. Hence, the presence of the proxy enhances security.

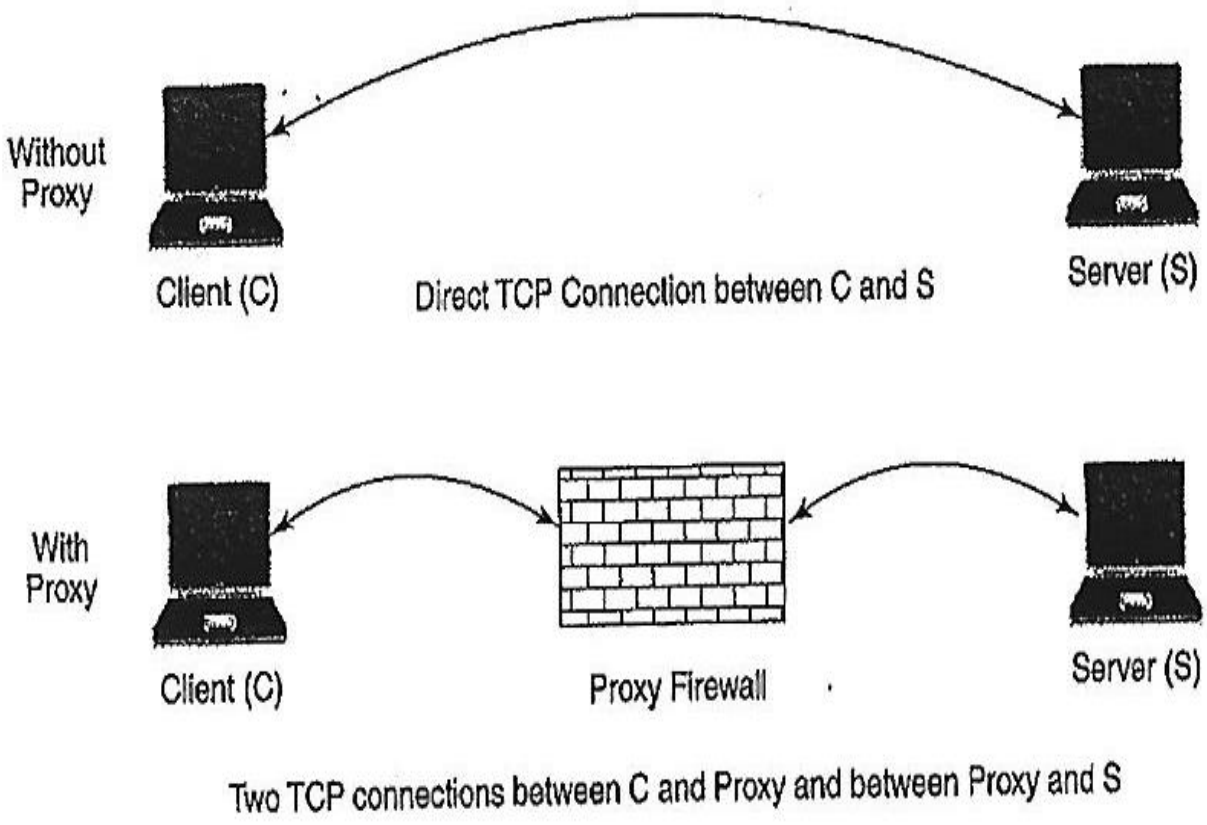


Figure 21.1 *Proxy firewall*

➢ There are proxy agents for many application layer protocols including HTTP, SMTP, and FTP.

➢ In addition to filtering based on application layer data, proxies can perform client authentication and logging.

➢ An HTTP proxy can also cache webpages.

➢ Caching has a major impact on performance.

➢ If the webpage is cached in a web proxy server located in the client's organization, the response time could be greatly reduced compared to that where the page has to be fetched from the external web server.

➢ Also, caching reduces the demand on external communication bandwidth while easing the load on the web server.

➢ Firewalls are a necessary element in the security architecture of an organization that permit access to/from the external world. In the next section, we study firewall deployment.

2. PRACTICAL ISSUES

- The security architecture of a medium size or large organization includes firewalls, proxy servers, VPN terminators, and intrusion detection/prevention (IDS/IPS) devices.

2.1 **Placement of Firewalls**

➢ We first note that firewalls help segregate or isolate the network into ***multiple security zones.***

➢ Each firewall in the organization enforces rules that control the transfer of packets between different security zones.

➢ At the very least, there are three zones —

1.The Internet,

2. The region containing the publicly accessible servers and

3. The internal network.

Figure 21.2 depicts a four-zone layout using three firewalls.

➢ Of the three firewalls, the first is really a router (the Border Router) with some packet-filtering capability.

➢ This is the access router interfaces with the Internet.

➢ It is connected to a stateful firewall, FW-1, which has three interfaces (firewalls that have more than two interfaces are referred to as multi-homed).

➢ The zone connected to the right interface of FW-1 is referred to as ***a screened subnet*** though it is more commonly referred to as a **De-Militarized Zone** (DMZ). It is labelled DMZ-1 in Fig. 21.2. A DMZ, in the true sense, is the area between two firewalls.

➢ In Fig. 21.2, the zone between firewalls FW-1 and FW-2 is a real DMZ labelled ***DMZ-2.***

➢ **Demilitarized zones** are so called because they often host servers that are accessible to the Internet and also to the internal network.

➢ Because they are accessible to the public, they are the most likely machines to be compromised in the entire network.

➢ Once a machine in the DMZ is compromised, other machines in the DMZ could get infected.

➢ DMZ-1 contains the publicly accessible servers.

➢ These include the web server, the external e-mail server, and the DNS server. All incoming mail from the Internet is received by this e-mail server, which checks for virus signatures and spam mail.

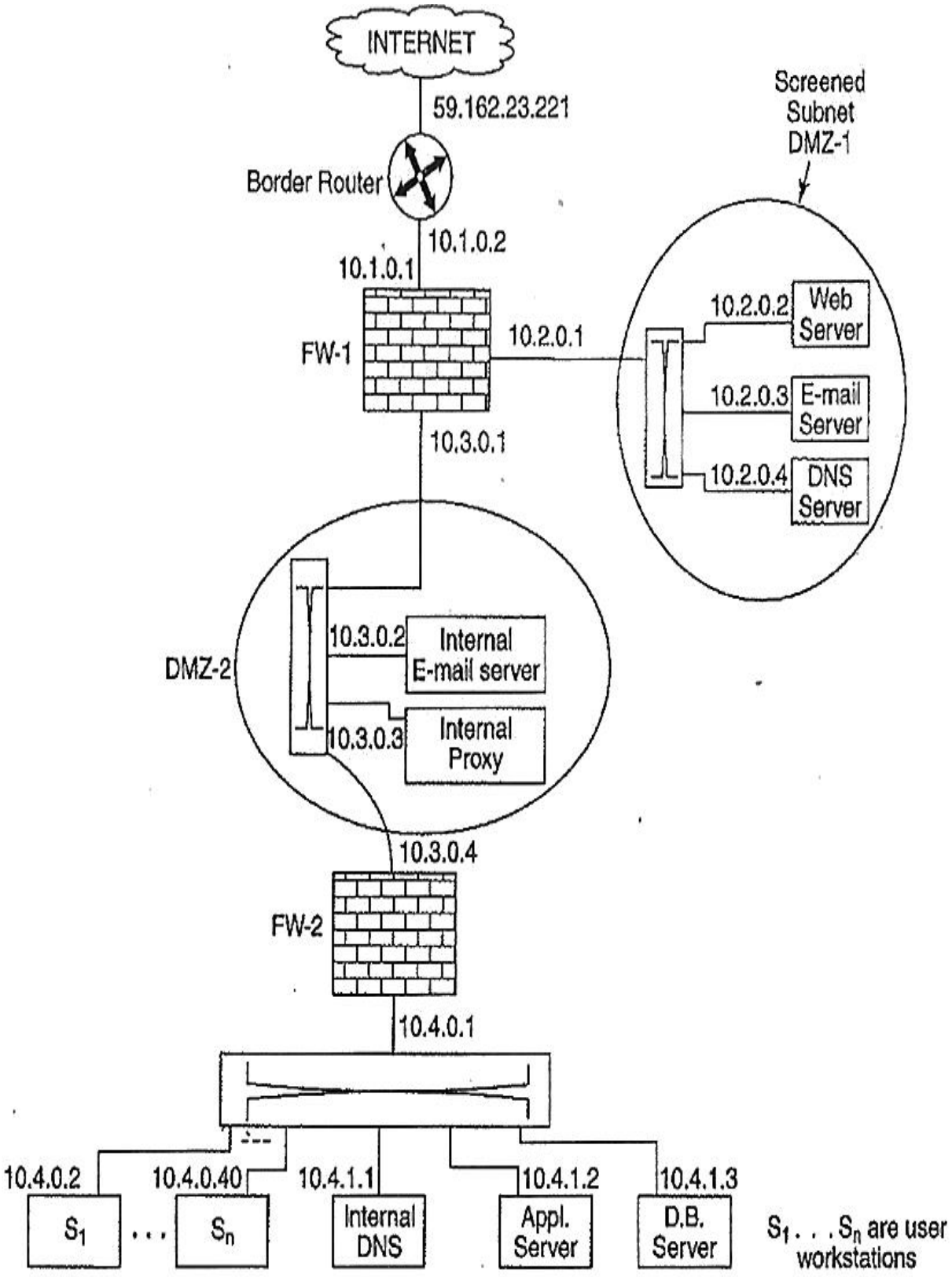


Figure 21.2 *Firewall placement*

➢ The DNS server resolves names of publicly accessible servers. However, care should be taken to ensure that it does not contain address records of any of the internal machines. DMZ-2 contains the internal e-mail server. This is the server that hosts the mailboxes of the company employees. It handles the sending and receiving of all mail between internal parties. It periodically establishes a connection to the external mail server (in DMZ-1) to retrieve all incoming mail.

➢ Outgoing mail (from the internal network to the Internet) can be handled in several ways. The internal mail server can set up an SMTP connection to a remote mail server to transfer mail.

➢ Alternatively, it can connect to the external mail server (in DMZ-1) and use it to relay all outgoing mail.

➢ **DMZ-2 also contains an Internet proxy server.**

➢ All internal users who wish to access external webpages connect to the proxy.

➢ The proxy authenticates the internal user and decides whether a page can be accessed (different restrictions might apply to different classes of users).

➢ The proxy scans incoming webpages for virus signatures and objectionable content. Finally, the proxy also performs caching of webpages.

➢ The internal network contains application servers, database servers, and user workstations.

➢ It also has an internal DNS server. This DNS server is different from the external DNS server in that it provides mappings between the domain names of the internal machines and their IP addresses.

➢ The internal machines all have private addresses. It is neither necessary nor desirable for third parties on the Internet to be aware of the private addresses of the internal machines. Hence, this DNS server is placed in the internal network.

➢ A feature of the security architecture in Fig. 21.2 is that services such as DNS and e-mail are ***split;*** that is, there is an internal DNS server as well as an external one.

➢ Likewise, there is an internal e-mail server and an external one.

➢ Generally, no external connection should be allowed to the internal servers.

➢ Connections in the reverse direction from the internal servers to hosts on the Internet should either be forbidden or severely restricted.

2.2 Firewall Configuration

➢ In order to create a firewall ruleset, we need to identify all the **possible authorized** connections that might be set up between pairs of machines in two different zones **adjacent** to the firewall.

➢ We first present a simplified version of the ruleset for firewall FW-2 (Table 21.2).

➢ Table 21.2 Simplified ruleset for firewall, FW-2

Table 21.2 *Simplified ruleset for firewall, FW–2*

No.	From IP Addr.	From Port	To IP Addr.	To Port	Protocol	Action
1	*	*	Internal	*	*	Drop
2	User	*	Int_Mail_S	25	SMTP	Accept
3	User	*	Proxy	80	HTTP	Accept
4	*	*	DMZ-2	*	*	Drop

*Wildcard

➢ The first rule states that ***no machine*** from any other security zone is permitted to establish a TCP connection to any internal machine.

➢ Rules 2-4 assert that, other than connections from internal stations to the internal mail server (on port 25) and web proxy (on port 80), no other connections are permitted to DMZ-1, DMZ-2, or the Internet.

➢ Table 21.3 shows the ruleset for firewall FW-1.

Table 21.3 *Simplified ruleset for firewall, F1*

No.	From IP Addr.	From Port	To IP Addr.	To Port	Protocol	Action
1	*	*	DMZ-2	*	*	Drop
2	Int_Mail_S	*	Ext_Mail_S	25	SMTP	Accept
3	Internet	*	Ext_Mail_S	25	SMTP	Accept
4	Internet	*	Web_S	80	HTTP	Accept
5	Internet	*	DNS-S	53	UDP	Accept
6	*	*	DMZ-1	*	*	Drop
7	Proxy	*	Internet.	80	HTTP	Accept
8	Ext_mail_S	*	Internet	25	SMTP	Accept
9	*	*	Internet	*	*	Drop

- Rule 1 in Table 21.3 states that no TCP connection is to be established to any machine in DMZ-2 from any machine in DMZ-1 or the Internet.
- Rule 2 states that the external mail server can accept connections from the internal mail server to receive incoming mail or to send outgoing mail.
- Rule 3 allows connection to the external mail server from mail server on the internet to deposit incoming mail.
- Rule 4 and 5 permit connections from the internet to the organizations web server and external DNS server, respectively.
- Rule 6 states that no other connection may be set up to any machines in DMZ-1 for any other purpose.
- Rule 7 and 8: the internet proxy in DMZ-2 and external mail server are permitted to make connections to machines on the internet to access webpages and to send outgoing mail.
- Rule 9: confirms that no other connection from the organizations machine to the internet for any other purpose is allowed.

1. Virus, Worms and Malware

Preliminaries

- **Computer Virus:** its earliest usage was in the context of malware resides in the boot sector of the floppy disk,usually the first sector on the disk and contains the code for bootstrapping
  - The rate of spreading virus is relatively slow depend on rate at which the boot floppies were exchanged.
- **Worms:** Use network to propagate with extraordinary speeds.
  - Can be spread without human intervention or through the human actions.
- **Trojans :** Do not replicate ,typically activated by action on the part of the victim
  - Can spread using email attachments, through file sharing software, from websites or through cell phones downloads

1.1 Virus And Worm Features

1.1.1 Virus Characteristics

 When a virus-infected program is run, the **virus code** is executed first.

 One of the first tasks of virus code is to seek other programs not yet infected and then pass on the infection to one or more of them.

 A truly malicious virus may then perform actions such as deleting certain files.

 An innocuous virus may attempt something benign like printing a "hello world" message.

 Execution of the virus code is usually followed by execution of the **host's original program.**

 All the virus code need not be located at the start of the infected file.

 In some cases, virus code is both **prepended** and **appended** to the host file.

 Virus code could be split into **several segments** and interspersed throughout the infected file using JUMP statements at the end of each virus segment.

 In most of these cases, the size of the infected program is larger than the original host program. This helps **anti-virus** software to detect infected code.

 To evade detection, some viruses modify the file **service interrupt handler that** returns attributes of files. By so doing, the service handler may be programmed to return the uninfected length of the file.

 Another technique is to use **compression** so that the length of an infected file remains the same as the length of its original version. The virus writer includes a

compression routine in the viral code. To infect another file, the virus first compresses that file and then prepends the virus code to the compressed file.

 The infected file must be uncompressed just prior to execution.

 One of the characteristic features of many viruses is the set of system calls they make. **System calls** are used by application programs to request services of the operating system.

 They are made to read/write files, spawn new processes, establish TCP connections, etc. Some viruses make calls to copy their own code to other files, create/modify entries in the Windows registry, or search for e-mail. Such "suspicious" calls are often used to distinguish malicious from benign code.

1.1.2 Worm Characteristics

- Classes and features
- Worms are most commonly classified based on their vector of propogation.
- The main categories include:

1. Internet scanning worms ***2. E mail worms***

3. P2P worms ***4. Web worms*** ***5. Mobile worms***

Over the years worm writers have brought many ingenious techniques to worm design. The below table shows selected malware and innovative aspects of each

Table 19.1 *Selected malware with their innovative features*

Malware name	*Year unleashed*	*Type of malware/ vectors of propagation*	*Claim to fame*
Code Red	July 2001	Internet scanning	First worm that spread rapidly causing billions of dollars in damage
Nimda	September 2001	E-Mail, HTTP, file sharing	One of the first worms to use multiple vectors of propagation
Slammer	January 2003	Internet scanning	First Internet scanning worm to spread through UDP, not TCP. Hence much faster – infected 90% of vulnerable hosts in just 10 min
Sobig	August 2003	E-mail	Worm updated itself at specific points in time. The updated worm code was obtained from specific URLs
Witty	March 2004	Internet scanning	One of the first worms to carry a destructive payload, which wiped out part of its victims' disks
Cabir	June 2004	Bluetooth	One of the first worms to target cellphones
Santy	December 2004	Web	One of the first worms to use a web search engine to locate new targets. Exploited a generic vulnerability in PHP
Commwar	March 2005	Bluetooth, MMS	One of the first mobile worms to use two vectors of propagation
Samy	October 2005	Web	In 24 hours, infected over 1 million users' profiles on MYSpace – a social networking site
Storm	January 2007	Bot. E-mail, infected websites	Multiple infection stages, URLs for secondary infections communicated via encrypted links in P2P network
Conficker	September 2008	Bot. Random scans, USB drives, network file systems	Dynamically generated URLs for code updates. Updates digitally signed by botnet controller

Limited categorization of achievements in worm design and sample features are as follows

- **Enhanced Targeting**

 The most important attribute of a Worm is that it spreads its infection to other computers.
 Many target selection strategies have been proposed and implemented.
 Worms that spread through **e-mail**, for example, have an easy way to figure out their targets.
 All they need to do is look into their **victim's mailbox or e-mail address book** to find a set of targets.
 A mobile worm obtains phone numbers of its potential victims from the phone book in the cellphone hosting the worm.
 Some web worms use **search engines** to harvest URLs of potentially vulnerable targets.
 Internet scanning worms, on the other hand, scan the **IP address space** for vulnerable machines.
 The most straightforward approach is **random scanning** — choosing IP addresses at random. This was adopted by **Code Red Version-I**. However, Code Red Version-II adopted localized scanning.
 Over 80% of the time, it attempted to connect to victims with whom it **shared** the **network address** (most significant 8 or 16 bits of the IP address). This strategy was more successful since hosts in the same network are likely to be closer and be running the same soft-ware.
 Worms like Nimda, unleashed in September 2001, spread aggressively thanks to its five different vectors of propagation. Propagation through HTTP and e-mail were particularly successful in penetrating the perimeter of the enterprise. Once inside, it exploited the Windows file-sharing feature to spread within the enterprise.

- **Enhanced Speed**

 To enhance the infection rate, some worms are designed to spawn **multiple threads.**
 Each thread is responsible for setting up connections to a different subset of hosts, thus increasing the rate at which infection is spread.
 Some worms reduce infection latency by targeting a buffer overflow vulnerability on an application that employs UDP rather than TCP.

 TCP connection establishment involves a three-way handshake and is time-consuming.

 UDP, by contrast, is connectionless.

 This sharply reduces infection latency.

 A steep increase in the number of infected machines at the very outset of a worm epidemic has a multiplicative effect on spreading rate.

 For this purpose, the attacker could create **one or more hit-lists** carrying addresses of several thousand vulnerable machines.

 The first worms to be **let loose could** carry one such **list**.

 As a worm infects each new machine, it splits its list between itself and the machine it has just infected.

 Given that most of the machines on the hit-lists are vulnerable, the **worm** spreads rapidly during the **initial stage** of the epidemic. Thereafter, the infected machines could spread the infection using **random scanning or some other spreading method**.

- **Enhanced Capabilities**

 Most worms (and viruses) have **unique and distinct signatures** — a pattern of bits, usually assembly language code, which appears in all instances of the worm.

 Worm and virus signatures are the key to detecting them. However, there are sophisticated code obfuscation techniques to evade detection.

 One such technique is the use of **encryption** for disguising worm code.

 Different instances of the worm may use different keys for encryption. Thus, they might fail to match any existing worm signatures. Such worms are said to be **polymorphic**.

 A polymorphic worm would have to be decrypted before being executed. This suggests that a decryptor routine "in the clear" would have to be part of the worm code.

 Decryptors themselves may be very simple, involving XOR operations or trivial shift-based substitutions. However, detecting a worm on the assumption that the decryptor routine is invariant would not always succeed.

 Figure 19.1 shows two versions of assembly code that look different but perform the same function.

 The second version is inefficient with spurious instructions.

 The second version also has a spurious branch instruction to confuse worm code detection software that relies on control flow analysis.

 Worms that have multiple such versions with or without relying on encryption are referred to as metamorphic worms.

Assembly Pseudo-code

```
if R5 > 0
    R4 ← R1 + R2
else
    R4 ← 4 × R2 + 3 × R3
```

Assembly Code: Version 1

```
        CMP   R5, #0
        BLE   Second
First:  ADD   R1, R2, R4
        BRA   Finish
Second: ADD   R2, R3, R4
        SLA   R4, #2, R4
        SUB   R4, R3, R4
Finish: . . .
```

Assembly Code: Version 2

```
        XOR   R6, R6, 0
        ADD   R5, R6, R6
        SUB   R6, #0, R6
        BG    First
Second: ADD   R2, R2, R4
        ADD   R4, R4, R4
        ADD   R4, R3, R4
        ADD   R4, R3, R4
        ADD   R4, R3, R4
        BNE   Finish
        CMP   R4, R4
        BE    Finish
First:  ADD   R1, R2, R4
Finish: . . .
```

Figure 19.1 *Polymorphic assembly language versions of same pseudo-code*

- ***Enhanced Destructive Power***

 It is estimated that worms such as Code Red and Nimda caused billions of dollars in damage.

 Analysts estimate costs based on lost productivity, clean-up costs, system downtime which affects business and revenues.

 Fast-spreading worms also caused severe network congestion problems disrupting normal Internet traffic and contributing to system dos time.

 Nevertheless, most worms thus far have been relatively benign.

 Some worms contributed attack packets to a DDoS attack or caused website defacement.

 The Witty worm which appeared in Mar 2004, however, was qualitatively different. It was the first worm to carry a destructive payload. deleted a random section of the victim's hard disk leading to a system crash

1.2 INTERNET SCANNING WORMS

- One characteristic of Internet scanning worms is that they are self-activated.
- The ability to spread without human intervention distinguishes them from most types of e-mail, P2P, and web worms.
- This category of worms is so called because they scan the Internet looking for vulnerable machines.
- The vulnerability could be a buffer overflow problem in a commonly used service provided by a particular version of an OS.
- The worm communicates with and delivers its malicious payload to the victim using standard transport protocols such as TCP or UDP.
- Once installed on the victim, it could erase local files, steal secrets, or deface webpages, but above all it seeks new victims to infect.

1.2.1 Case Studies: Code Red and Slammer

Code Red

- One of the best known examples of Internet scanning worms is the Code Red worm,
- It all started on June 18, 2001, when a buffer overflow vulnerability was discovered in the Microsoft IIS Weh Server.
- A patch for this vulnerability was developed a few days later.
- It is estimated that there were several million IIS servers in active deployment.
- Even assuming that a large percentage of these were patched, that still left plenty of room for the spread of the worm, which was unleashed on July 12, 2001.
- The worm itself was carried in HTTP request messages targeted at IIS servers.
- The first version of the worm used a random number generator to generate new addresses of machines to infect. However, the same seed was used for the random number generator in every instance of the worm.

Slammer

- The SQL Slammer was launched on 25 January, 2003, and targeted a buffer overflow vulnerability on the Microsoft SQL server 2000.
- The worm sent packets on UDP port 1434 — the database software's resolution service.
- It used simple random scanning to propagate.

 Slammer's payload was a mere 384 bytes in length — far smaller than the 4 kb payload of Code Red. Also, UDP, being a connectionless protocol, there is no overhead of connection establishment.

Worm Propagation Models

Simple Epidemic Model

 The Simple Epidemic Model used to study the spread of infectious diseases among humans is an appropriate starting point.

 The model assumes that there are only two types of entities in the population.

 Either an individual is susceptible or he is infected.

 An infected individual can infect a susceptible person.

 Once infected, a person remains infected and does not recover.

Let N be the size of the total population. Let $I(t)$ be the number of infected individuals at time t. The number of susceptibles at time t is then $N - I(t)$. β is the initial infection rate, i.e., each infected person attempts to pass on the infection to β susceptibles in 1 time unit. The following differential equation captures the number of infected persons at time t.

$$dI = \beta I(t)\left(1 - \frac{I(t)}{N}\right)dt \qquad (19.1)$$

or

$$\beta\, dt = \left(\frac{dI(t)}{I(t)\left(1 - \frac{I(t)}{N}\right)}\right) \qquad (19.2)$$

In an infinite population, each infected person infects $\beta\, dt$ susceptibles in time interval dt. However, in a finite population of size N, the probability that the target of an infective is already infected is $\frac{I(t)}{N}$. Such targets do not add to the population of newly infected. The factor $\left(1 - \frac{I(t)}{N}\right)$ in the above equations ensures that only previously uninfected entities are added to the count of the freshly infected in time interval dt.

Integrating both sides of Eq. (19.2) yields

$$I(t) = \frac{I_0 N}{I_0 + (N - I_0)e^{-\beta t}} \qquad (19.3)$$

Kermack—McKendrick Model

 The Kermack—McKendrick (K—M) model more accurately models the spread of human infectious disease by considering three (instead of two) categories of people:

- Those who are susceptible (state S)
- Those who are infectious (state I) and
- Those who are neither, i.e. individuals who are cured or those who have succumbed to the disease (terminal T).

 Initially, all individuals in the population are susceptible.

 It is possible to go from state S to I but not vice versa .

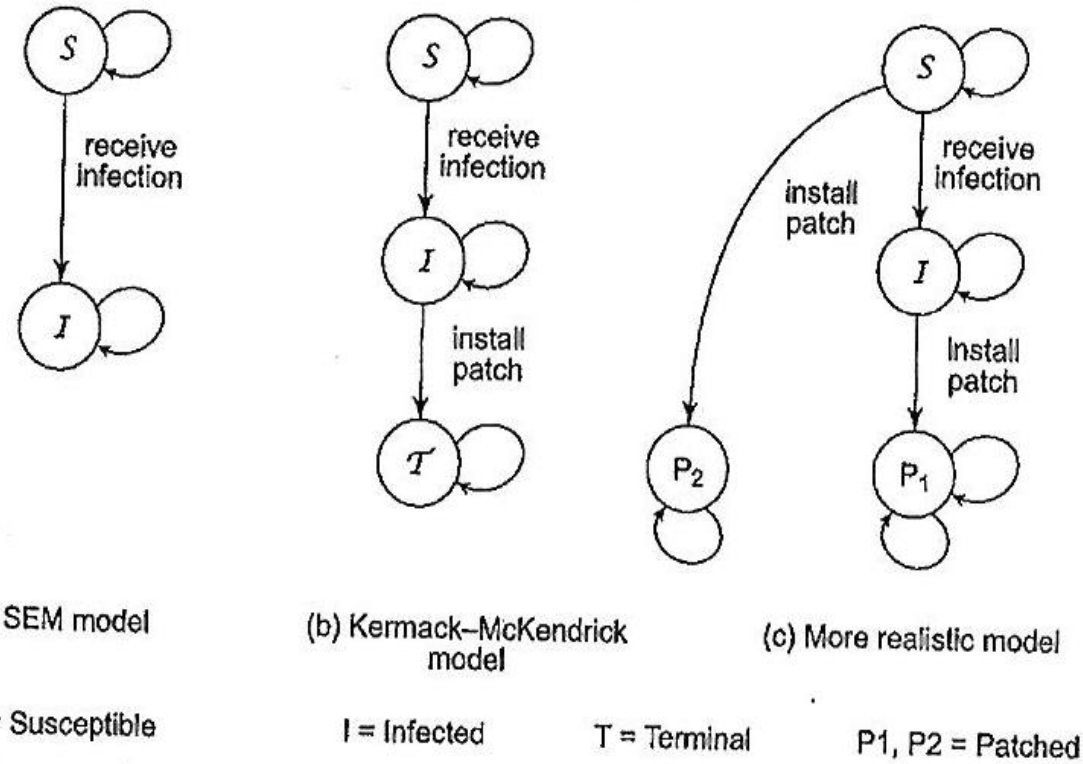


Figure 19.3 *State machine representation of vulnerable machines during a worm epidemic*

 An infectious person may or may not be cured.

 If cured however he is not again vulnerable to disease.

 The transition from S toI corresponds o an infected machine being patched

 Also again such a machine is never vulnerable to a Code Red infection.

As before, let $I(t)$ be the number of infectious machines at time t, let N be the total number of machines, and let β be the infection rate. Let $S(t)$ be the number of susceptibles. So, the number of machines in the terminal state is $N - S(t) - I(t)$. The K–M set of equations is

$$\frac{dS}{dt} = -\beta I(t)\left(\frac{S(t)}{N}\right) \qquad (19.4)$$

and

$$\frac{d(N - S(t) - I(t))}{dt} = \gamma I(t) \qquad (19.5)$$

- Equation 19.4 describes the rate at which the susceptible decrease due to transition to the infectious state.
- Equation 19.5 captures the rate at which machines in the terminal state increase.(neither susceptible nor infectious)

Analogous to β, γ is the rate at which the infectious machines transit to the terminal state.

While the K–M model better explains the spread of Code Red compared to the Simple Epidemic Model, it still falls short. Its implicit assumptions are at variance with the spread of Internet scanning worms. In particular:

- Machines that are *susceptible but not infectious may also be patched*. Thus, there ought to be a transition from state S to state *T*. This is not factored into the model.
- The *infection rate, β, is network-dependent*. As the worm continues to spread, it will consume network resources such as bandwidth leading to traffic congestion. This in turn will slow down the infection rate. However, both the models considered here assume β to be constant.
- The K–M model assumes that the rate of transitioning from the infectious to the terminal state is a constant, γ. During the early stages of the worm epidemic, not much is known about it. So few machines will be patched. However, once the epidemic sets in and there is more public awareness, more administrators will apply patches. This rate will decrease after most worm-aware administrators have applied their patches. Thus, the rate at which machines are patched, and hence γ, *is far from constant.*

A state diagram of a model that factors the patching of susceptible machines is shown in Fig. 19.3(c).

1.4 Topological worms

- Worm can be represented as a graph with the nodes representing the vulnerable machines.
- An edge between Machine A and Machine B exists if A knows /stores the address of B and is capable of directly infecting B by sending it a malicious payload.
- Topological have focused targets.
- Their intermediate targets are their neighbours who in turn spread the infection to their neighbours and so on.
- Their rate of spreading is faster than Internet scanning worms.
- Two types of topological worms are email worms and P2P worms.

1.4.1 Email worms

 Email worm propagates through infected e mail.

 The victim receives email that comes from trusted or familiar source.

 The victim sees an innocent text file attached to the email.

 In reality, the file named such as loveletter.text.vbs contains Visual Basic script.

 By clicking on this attachment, the embedded VB script executes, sending a copy of itself to every person in the victim's contact list.

 Many e-mail worms exploit the fact that documents created by certain word processors embed software macros in them.

 The macros execute when the document is opened

 For example, Melissa was a macro worm (or "macro virus") that propagated by sending copies of itself to the first 50 persons in the victim's address book.

 One of the best-known e-mail worms of more recent vintage is Sobig, which was let loose in 2003. It spread by communicating malicious c-mail or copying itself to an open network share.

 There were several versions of SoBig. One version could update itself by downloading code from certain websites.

 The URLs of these sites were contained in a file that itself was downloadable from geocitics.com this site allows users to host their own free webpages besides providing tools in support of building dynamic webpages).

 Some of the malicious code received, installed a keystroke logger and stole passwords from its victims.

1.4.2 **P2P Worms**

 A P2P network is a massively distributed system of computers where each peer or node plays the role of both client and server.

 They are used principally for sharing files, which may contain songs, images, videos, etc.

 Each peer maintains within itself a shared folder of files that it is willing to share with others.

 Users do not download files from a central server but from their peers located across the globe.

 They are immensely popular as evidenced by the fact that a very large proportion of Internet traffic is comprised of P2P packets.

 To see how P2P worms spread, it is important to understand how a P2P network operates.

 Most P2P networks use an **overlay network**, which is a logical network of peers.

 Two peers are said to be neighbors at any given point of time if there is an active TCP connection between them.

- P2p networks are scalable and resilient

A peer, A, that wishes to obtain a file, say *abc*, creates a "Query Request" message for *abc* which it sends to all its neighbours. Each neighbour that does not have the file, in turn, queries its neighbours and so on. In Fig. 19.4, links labelled Q1 are the first set of links over which the query from A propagates. Links labelled Q2 are the next set of links, etc. Many peers are thus hard at work independently trying to locate a peer who has *abc* and is willing to share it. If a peer has *abc*, it returns a "Query Hit" message. This message traces the path of the "Query Request" message in reverse.

The requesting node may receive multiple positive responses. For example, Node A (the requester in Fig. 19.4) receives two positive responses from nodes B and C. It chooses one of them (say node B) and directly contacts it. Node A sends its IP address to node B with a request that *abc* be downloaded to that IP address. The file is then downloaded using FTP or HTTP.

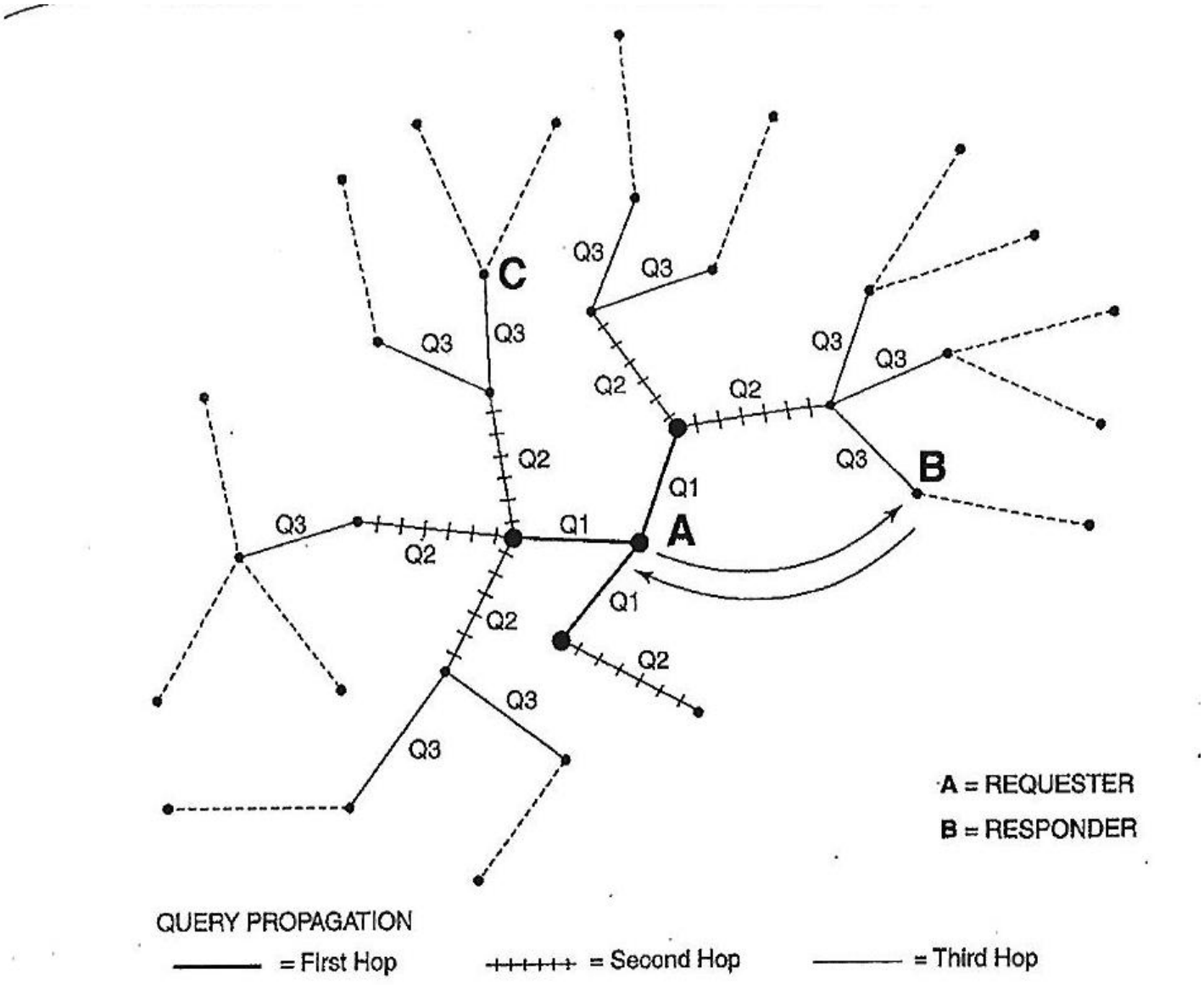


Figure 19.4 *Querying peers for a file in a P2P network*

- **Here are potential ways in which P2P worms may spread:**

1. One of the simplest is for a malicious peer to respond positively to any query.

- If the requester then chooses to download the file from the malicious peer, the latter sends it an infected file whose name is changed to match that of the requested file.
- The infected file contains a worm which passes on its infection to the requester.
- Once infected, the requester mimics the behaviour of the malicious peer thus helping to propagate the infection.
- Alternatively, various "popular" files stored in the shared folder of a peer may be infected.

 When any of them is downloaded, the infection spreads to the shared folder of the requesting peer.

2. Peers in a given P2P network run the same P2P protocol.

 There are usually few different implementations of this protocol leading to little software diversity.

 An exploitable buffer overflow vulnerability in one popular implementation is a familiar starting point.

 This, coupled with the fact that a peer maintains a list of neighbours, implies that a worm has ready targets and does not need to perform random scanning as in the case of Internet scanning worms.

 The first type of worm is said to be passive since it propagates only when requested to download a file.

 The second type of worm is active since it propagates on its own without receiving requests from its peers.

 One aspect of P2P worms is that they may result in no apparent traffic anomaly, so an intrusion detection system monitoring network traffic is unlikely to raise an alert.

1.5 MOBILE MALWARE

1.5.1 Introduction

 New—generation smartphones combine the functionality of a cellphone and a lose-end PC.

 They may be used for storing confidential documents, communicating via e-mail/SMS/MMS, and taking photographs. They support feature-rich applications that run on top of a complete OS.

 The most common OS on smartphones is the Symbian followed by Windows Mobile, Linux, and recently the Mac OS X (on the iPhone).

 They provide a rich set of APIs to access the phone book and other files, send SMS/MMS messages, etc. Unfortunately, these very APIs can also be used by malware to, for example, read a confidential document on the smartphone and ship it to the attacker as an MMS attachment.

1.6.2 Bluetooth

- Bluetooth is both a communication technology and a protocol stack.
- As a communication technology,
- It supports short-range wireless communication —
- A maximum of between 10 and 100 meters between devices.
- Bluetooth uses 2.4 ghz shortwave radio technology.
- Bluetooth is a complex, multi-layered protocol.

Discovery and User Authorization

- Besides the vulnerabilities mentioned above, social engineering is a key factor in the spread of mobile malware,
- Many PC users do not hesitate to click hot links in their e-mail or open attachments even when the sender of the e-mail is unknown.
- This behaviour carries over to the mobile world where many users have no hesitation in accepting a file from an unknown source.

- The combination of Bluetooth implementation vulnerabilities, rich feature set of the smartphone, and unthinking user behaviour has exposed the smartphone to various strains of malware.
- To investigate the spread of malware in smartphones, it is necessary to understand the basics of how smart phones exchange files using Bluetooth.
- To discover other Bluetooth-enabled devices in its neighbourhood, a device initiates an inquiry procedure, which includes broadcasting an inquiry request.
- All devices in the range of the initiator that are in discoverable mode respond sending their bluetooth device address (BD_ADDR).
- This is a 48-bit MAC address — the first 24 bits identify the device manufacturer/model and the last 24 bits specify a particular instance of that model.
- Bluetooth is a connection-oriented protocol.
- A device, A, can set up a connection to any other device B. But to set up such a connection, A should know the BD_ADDR of B.
- One way of obtaining B'sBD_ADDR is through the discovery procedure.
- A large percentage of users keep their phones in discoverable mode, so this is an attractive way of harvesting device addresses especially In crowded areas such as railway stations or malls. However, it should be noted that even with the phone in non-discoverable mode, there are a number of brute force techniques to extract its BD_ADDR.

 Knowing B's BD_ADDR, A could attempt to exchange files with it using the OBEX (object exchange) protocol.

 A session protocol that resembles HTTP, OBEX is used to transfer images, business cards, and other files between Bluetooth devices.

 An attacker, A, could use OBEX Push totransfer a file containing malicious code to user, B.

 User authorization is usually required before a file can be accepted by his smartphone. Each user selects a PIN which varies between 4 and 16 characters long but 4 characters are typically used.

 The smartphone usually prompts a user to enter his/her PIN as a way to confirm whether an external file, for example, should be accepted.

 Some OS versions accept file transfers without user authorization. And some smartphones allow users to disable the "Authorization Required" option for file transfers.

Link Level Security

 The level of security provided by user authorization alone is generally inadequate.

 Another level of security provided by Bluetooth is link-level authentication and encryption.

 For this purpose, both sides compute a common secret called the link key.

 Bluetooth uses a procedure called **pairing** wherein this key is computed by two participating devices.

 Pairing is preceded by discovery/inquiry and paging.

 The latter is a procedure whereby the discovering device, A, establishes a connection with the discovered device, B.

 Computing the Link Key

1. The first step in deriving the common link key between A and B is to compute an initialization key, K_{init}.

 This is a function of the BD_ADDR of B and a nonce, IN_RAND, generated by A as shown in Fig. 19.7(a).

 Before the pairing procedure, the owners of A and B must agree in an off-line manner on a temporary PIN to be used specifically as part of the pairing procedure.

 Both users then type in the temporary PIN. K_{init} is also a function of this temporary PIN agreed to by both parties. Kinit is computed from IN_RAND,

BD_ADDR0, and the PIN using an algorithm, E22, based on a block cipher called SAFER+.

2. To compute the link key,

- A and B each generate a random number (LK_RAND_A and LK_RAND_B, respectively).
- Each party then performs an XOR of its random number with and they each transmit this across.
- Each side recovers the other party's random number by performing an XOR of the received value with Kinit, (see Fig. 19.7(6)).
- Now, each side has LK_RAND_A, LK_RAND_B, and the two device addresses, BD_ADDR_A and BD_ADDR_B.
- They then perform identical operations on these to obtain the link key, KAB as shown in Fig. 19.7(a). The operations involve the use of an algorithm, E21, which, like E22, is based on the cipher, SAFER+.
- Thereafter, each device stores the pair, BD_ADDR, of the other device and the newly computed link key in a database. Each device maintains such a database of BD_ADDR, link key pairs, one pair per device it is paired up with.

Using the Link Key

- The link key is used for both authentication and encryption.
- Suppose A and B were already paired.
- Let K_{AB}, be their common link key.
- Now suppose A wishes to authenticate B.

- For this purpose, it generates a random number, RANDA (a challenge) and sends it to B.
- The response computed by B is $E1(K_{AB},RAND_A,BD\_ADDR_B)$.

Hacking the link key

- It is possible to launch a dictionary attack by sniffing each message involved in pairing and authentication.
- These attacks enable an eavesdropper to obtain the link key KAB.

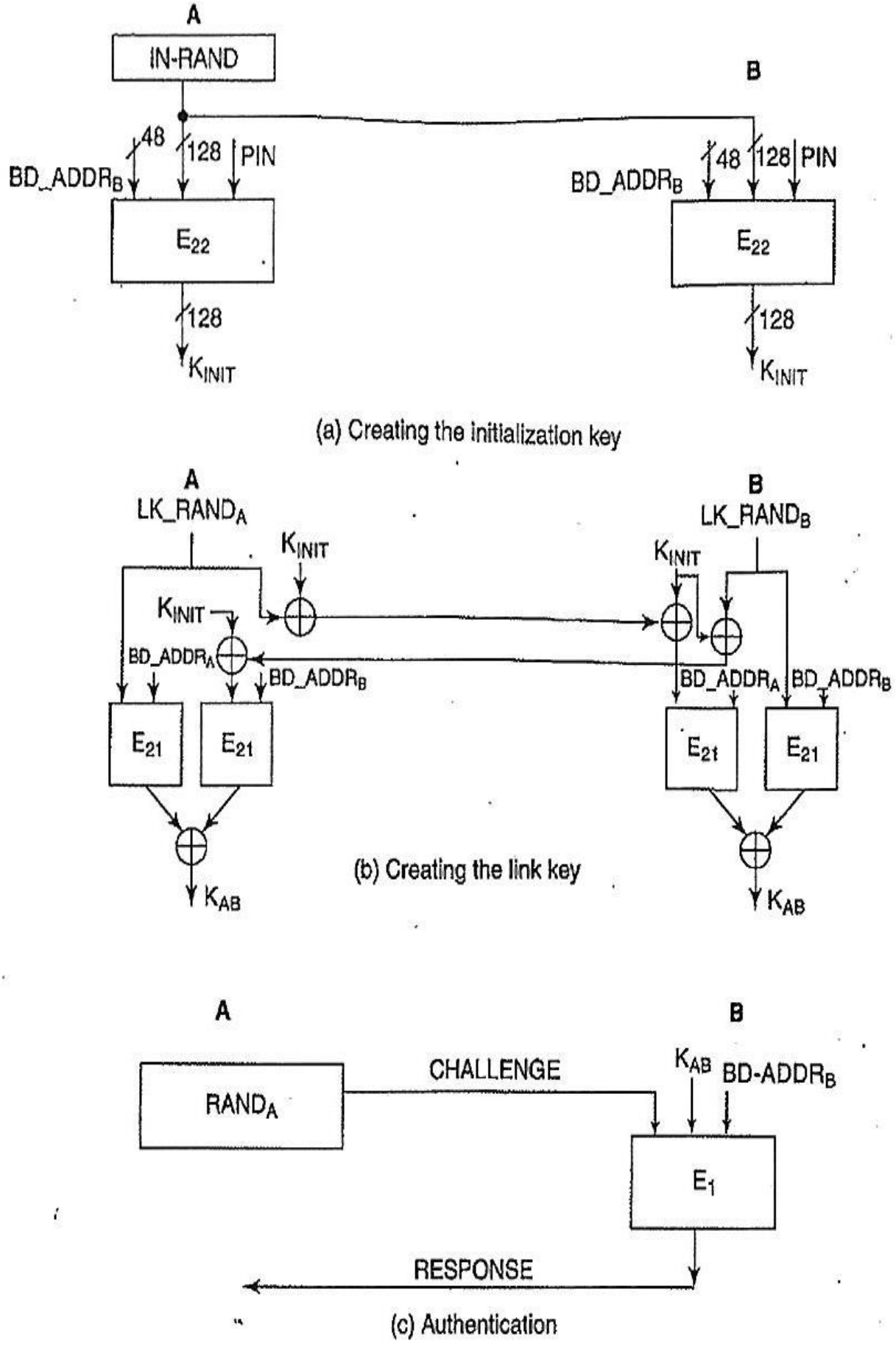


Figure 19.7 *Generation and use of the Link key*

- The latest version of Bluetooth version 2.1 gets rid of this problem by using Elliptic Curve Diffie –Hellman (ECDH) key exchange. The idea is similar to EKE protocol.
- EKE protocol was used to thwart off-line dictionary attack on weak password.
- In the case of smartphone, the PIN which could be just 4 characters long is analogous to the weak password. Once the PIN is guessed the link key can be obtained.

1.5.3 Examples

 Cabir was one of the earliest proof-of concept worms that targeted the Symbian Series 60 OS.
 Unleashed in June 2004, it was authored by the International Virus writing group 29A.
 The worm attempts to discover other Bluetooth-enabled phones set in discoverable mode.
 When it finds such a phone, it sends the worm payload in a SIS file.
 The receiver needs to accept and install the file.
 Its payload was mostly benign typically displaying "Caribe" on the screen. However, the continuous scanning for new victims by an infected phone depletes battery power.
 Commwarrior, which appeared in March 2005, was the first worm to spread through, both Bluetooth and MMS.
 Like Cabir, it targeted Symbian smartphones.

1.6 BOTNETS

1.6.1 Basics

 A botnet is an army of compromised computers or bots connected to the Internet and remotely controlled by a ***"botmaster."***
 The earliest botnets were a collection of zombies that participated in DDoS attacks.
 The emergence of botnets is closely linked to the motive of financial gain that is behind many recent cyber-attacks.
 They are often used to send spam mail on behalf of third parties,
 For example, Bot programs, may contain keyloggers and other forms of spyware that capture sensitive personal information such as passwords and credit card numbers and send these to the botmaster.
 Botnets have also been used as an extortion tool — "Pay up or your website will be bombarded by a DDoS attack".
 How does a computer become a bot?
 Bots are created in ways similar to many of the traditional trojan/worm/virus infections.
 A common vector of propagation is e-mail that contains an infected attachment.
 Another is through downloading a malicious webpage containing scripts that exploit vulnerabilities in certain browsers or application software.
 A bot infection may also be propagated by bots themselves by scanning the Internet for vulnerable machines.

 Finally, open file shares and IRC (Internet Relay Chat) multicast messages have also been widely used to spread infections.

 One important difference between a bot and a computer infected by a traditional worm/virus/ Trojan is that a bot needs to communicate with specific nodes in the bonet to receive fresh commands.

 A bot may be ordered to send spam or to "Launch a DDoS attack on site abc.com beginning 14:00 hours on 01-12-10." Some of the nodes in the botnet play the role of Command and Control (C&C) servers. They receive commands from the botmaster and disseminate these to the rest of the bots.

1.6.2 Case Study: The Storm Botnet

 The Storm botnet was first detected in January 2007.

 Its other names are Peacomm, Nuwar, and Zhelatin.

 Bots in the Storm botnet are infected in stages.

 The most common vectors for propagating the primary infection appear to be **e-mail or infected websites.** E-mail was sent with sensational subject lines like "230 die as Storm batters Europe."

 Likewise, users were lured into downloading free but infected files from websites containing music of various pop artists.

 The primary infection instructed the victim to join the Storm botnet embedded in the Overnet P2P network.

 Once part of the botnet, the bot was programmed to receive the second and subsequent injections of malicious code. One of the injections instructed the bot to propagate e-mail viruses. Another injection received some days later instructed the bot to launch a DDoS attacks on a target specified by the botmaster.

The overnetP2P network

- **The overnetP2P network** is based on the kademlia protocol which employs the distributed hash table based routing protocol: which locates a value corresponding to the given search key.
- Suppose X is a peer who searches for file then it could be replied with IP address and port number hosting file. Let it be Peer Y.
- Then X could contact Y to obtain the file.

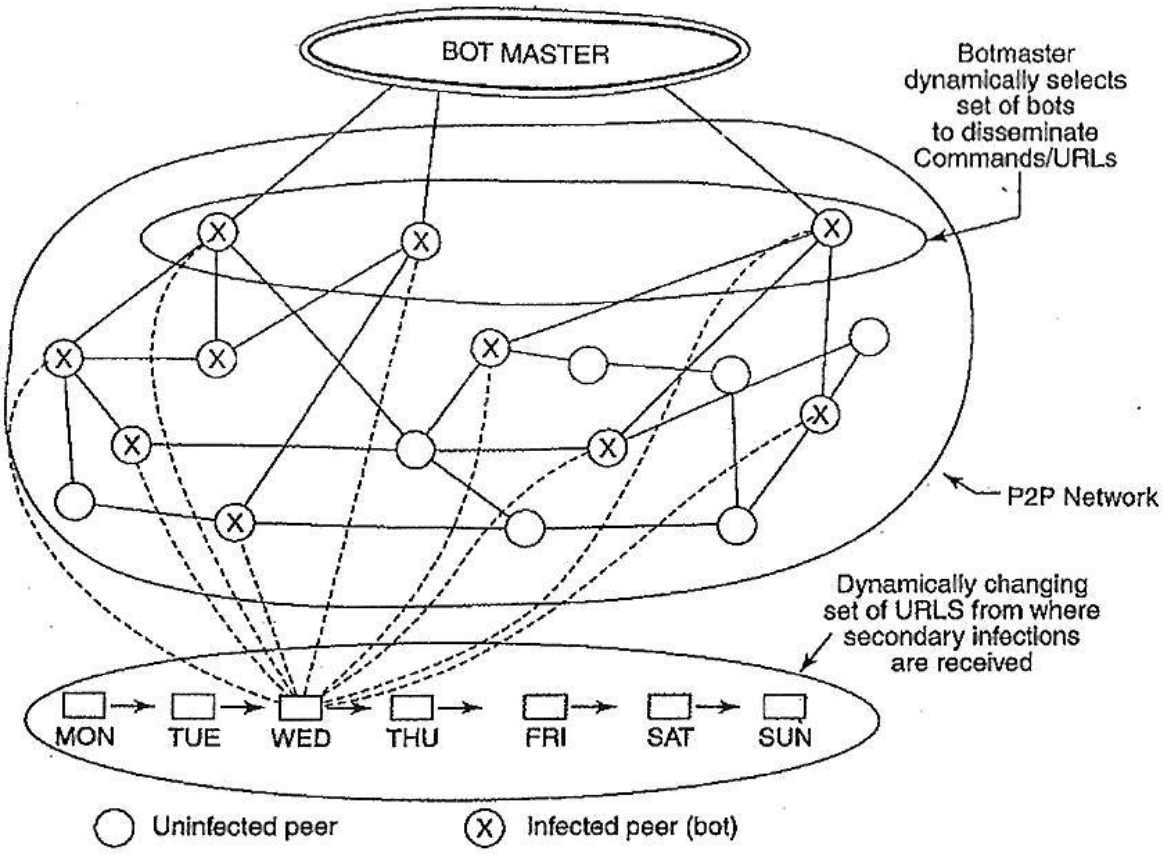


Figure 19.8 *Second-generation botnets using P2P networks*

- The initial infection has had a list of 146 peers used for bootstrapping (procedure which makes infected newly infected machine part of storm botnet.)
- Each entry in the peer list has MD4, 128bit peer hash followed by IP address and the Port number of the peer.
- When searching for the key, bot will search in these list first.
- Bot is programmed to fetch updated code for its subsequent infections.
- The storm botnets designers will update malicious code and also changes URLS from which the code was to be downloaded to confuse security analysts.
- The P2P network is not used to communicate this code but the encrypted URLS from which the code can be downloaded
- The Search value was the encrypted URL, while the search key was computed by each bot as a function of the date and a random integer f(date, rand) 0<rand<31
- In response to the search query bot receives the encrypted URL and also a partial decryption key.
- This partial key in conjunction with hard codded key in the bot is used for decrypting the encrypted URL.
- The bot then proceeds to fetch the infected code form the URL.

- The URL from which the malicious code are downloaded changes daily.

Conficker worm

- Its design is similar to storm worm
- A bot uses domain generation algorithm to dynamically generate domain names from where the malicious code could be downloaded.
- In addition to the domain flux it uses another DNS technique called fast flux in which one domain name was mapped with hundred of IP addresses.
- Conficker attempted to disable the antivirus and other detection software on its victims.

1.4 WEB WORMS AND CASE STUDY

 Web worms differ from malware such as the Internet scanning worms in several ways.

 Many web worms are executed in browsers which run on diverse hardware/OS plat forms.

 Web worms are written in a high-level language making it easy to perform complex operations but difficult to execute low-level operations.

 On the other hand, many other worms are written in assembly language.

 One type of web worm is the XSS worm — so called since it exploits cross-site scripting vulnerabilities in web servers.

 The first step in creating an XSS worm is to inject attack code into a vulnerable web server.

 When a user accesses the infected website through his/her browser, the malicious code (usually Javascript) is downloaded on to the browser.

 As in any XSS vulnerability, malicious code executes on the browser.

 Given that a key function of a worm is to propagate, the challenging question then is "How does an XSS worm propagate?"

 A partial answer to this question may be found through our next case study of the Samy worm.

***XSS Worm Case Study:* Samy worm**

 The XSS worm, Samy was unleashed in October 2005.

 Authored by SamyKamkar, it infected the social networking site, Myspace.

 Social networking sites typically allow users to create and edit their profiles (Fig. 19.5), which are stored on the site and are accessible to other members of the social networking group.

 A user profile may contain information about him including his hobbies, photographs, etc.

 A user profile also contains a list of the user's friends with hyperlinks to their profiles.

 Samy added a bunch of carefully crafted Javascript to his profile.

 When a visitor to Samy's website, say V1, downloaded Samy's profile on to his browser, the Javascript in Samy's profile executed.

 This caused Samy to be added as a friend in V1's profile and also to include the message "but most of all, Samy is my hero.

 " Within 20 hours of the first visit to Samy's profile, Samy had been added as a friend to more than a million user profiles.

 This rate of spread was even faster than that of Code Red.

 How did the worm spread and why did it spread so fast?

 The malicious Javascript uploaded itself on to V_1's profile on the MySpace server, thus infecting it.

 This is done by an HTTP Post-request sent from the browser to the server. However, that would cause the screen to freeze between sending the request and receiving the HTTP response from the server.

 To ensure that the viewer had a normal screen experience, Samy'sJavascript created an XMLHttpRequest object which was used to send the malicious Javascript to the Myspace server. Unlike the regular HTTP request, the message from an XML HttpRequest object is asynchronous and runs in the background.

It is pertinent to ask whether the Myspace server and the browsers that executed the malicious code could have prevented the spread of the Samy worm. For example, why did the Myspace server permit a user to add Javascript to his profile? The fact of the matter is that Myspace was very cautious in *filtering* many suspicious tags. It did filter out tags such as <script> and <body>.

→ *However, Samy included Javascript within the tag* as shown below.

```
<div expr =." * Javascript here * " style="background:url('java
script:eval(document.all.mycode.expr)')">
```

instructed the browser to execute the expression using eval(document.all.mycode.expr). Not all browsers execute Javascript contained in the tag but some did cooperate with Samy and thus aided the spread of the worm.

- **The keyword javascript was filtered by Myspace .So how did it manage to persist in infected user profiles on the Myspace Server?**

→ *Samy split "javascript" on two lines as shown in the tag above.*

Fortunately for Samy, this was not detected by the Myspace server. Second, most browsers were able to fuse "java" and "script" from the two successive lines to create "javascript."

Santy worm

- Written in perl and executed on server.
- The PhpBB application did not carefully check for clients input.
- One of the input received is URLS query string.
- Cleverly the disguised worm code was passed through this parameter.
- The server failed to detect that this input was actually perl code.
- Santy worm attempted to identify other PhpBB application by contacting web search engines such as Google to locate its targets.

INTRUSION PREVENTION AND DETECTION

2.1 **Introduction**

 Definition: An intrusion is the ***act of gaining*** unauthorized access to a system so as to cause loss or harm.

 Examples of intrusions include the following:

  Unauthorized login to a system by illegally acquiring a password (through, for example, a password guessing attack). –

  Worm infections that use the system as a launch pad to spread and infect other machines.

  Injection of spyware that passively monitors the activities of the user and relays this information back to the attacker (over the Internet, for- example).

  Flooding the host with ***spurious connection requests*** that attempt to exhaust the target's resources — processing power, memory, or communication bandwidth.

 Two ways of handling intrusions are ***intrusion prevention and intrusion detection.***

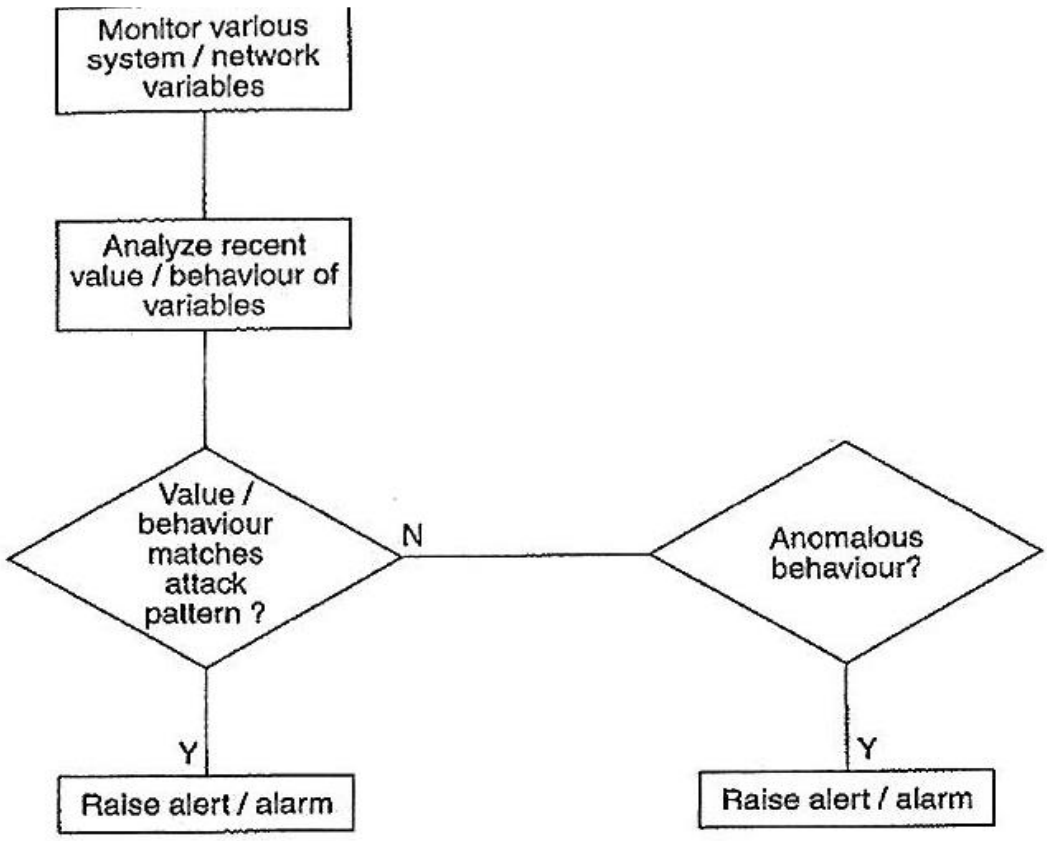


Figure 22.1 *Tasks performed by an IDS*

2.2 Prevention Versus Detection

Prevention

 Intrusion prevention anticipates various kinds of attacks and takes steps to forestall their occurrence.

 On the one hand, programmers should adopt practices that help reduce or eliminate software vulnerabilities.

 The use of safe string manipulation functions in C/C++ and the use of parameterized SQL queries are some of the practices recommended to protect against buffer overflow and SQL injection attacks, respectively.

 Likewise, sanitizing user input from HTML forms is one preventive measure against cross-site scripting attacks. Another set of preventive measures may be taken by the computing system (hardware, compiler, or operating system) to provide a second line of defence.

 Extensive training should be imparted to system administrators on this and related tasks.

 Finally, users should be trained to adhere to sound security practices such as password protection and be educated on the variety of social engineering attacks.

 One final aspect of intrusion prevention is deterrence. Hacking, whether for fun or profit, is a criminal offence.

Detection

 An intrusion detection system (IDS) (Fig. 22.1) performs the following three tasks:

 First, it monitors "events of interest" occurring in the target system or in the network.

 An event of interest may be a system call (a call made to the operating system) to, for example, open a file containing sensitive data.

 Another event of interest may be the attempted establishment of a TCP connection from a specific IP address to a certain port.2.

 An IDS generates a large amount of data which it then analyzes and converts into valuable information to be used by system administrators.

 These are examples of thresholds and parameters set by a human.
 On the other hand, it would be highly desirable if the IDS were capable of learning what is normal behaviour, detecting anomalous events when they occur, and flagging such events.

 There are a number of key questions related to IDS functioning and deployment:

 What are the variables that the IDS should monitor?
 When should an alert be raised? When should an alarm be sounded?
 Where should the IDS be placed?

2.2.1 **Case Study: Unauthorized User Logins**

 To prevent unauthorized logins owing to compromised passwords, the following should be adhered to:
1. A password should be at least **eight-characters** long, hard to guess, and include at least one non-alphanumeric character.
2. A password should be changed at least ***once in two months.***

3. Passwords should be ***stored securely*** (not written on sticker pads) and should not be communicated to friends, relatives, and co-workers.
4. After three consecutive ***unsuccessful attempts*** to a specific account, the system should be designed to disable all further log-in attempts for the next 20 minutes.

- Rules 1 and 2 must be enforced by the system.
- Rule 3 involves the user alone,
- Rule 4 involves the system alone.
- These rules are all measures intended to prevent intrusion.

- As a further preventive measure, a high-security organization may mandate two-factor authentication — ***passwords in conjunction with biometrics.***

- In addition to prevention, an IDS may also be deployed to monitor suspicious log-ins.

2.3 TYPES OF INTRUSION DETECTION SYSTEMS

- A real-world IDS monitors and mines hundreds of variables for interesting patterns.
- Table 22.1 shows a sample of variables together with a condition that may trigger an alert.
- Some of the variables are mere bit patterns in the packet header or the packet payload.
- Other variables are counts of a certain occurrence within a time interval.
- We next classify intrusion detection systems based on their functionality.

1) Anomaly versus signature based IDS

2) Host based versus network based IDS

Table 22.1 *Events of interest to an IDS*

Variable monitored	*Event of interest*	*Possible attack*
No. of accesses to specific file	Tenfold increase over norm	DoS attack or flash event
Login frequency to particular account	Unusually high	Attempted break-in
No. of distinct source IP addresses of arriving packets	Very high	Worm attack
Ratio of ARP request packets to ARP response packets	>> 1	Network scan to identify local active hosts
Ratio of TCP SYN packets to TCP FIN packets	>>1	Possible DoS attack
Percentage of half-open TCP connections	Sudden surge	Possible DoS/DDoS attack
TCP header flags	Invalid combination	Port scan, OS fingerprinting
TCP connection establishment	Unused destination port	Attempt to find which services are open
Payload of incoming packet	Specific byte sequence present	Specific worm attack
O.S. calls	Particular sequence of calls	Specific virus attack

Anomaly versus Signature-Based IDS

AnomalyBased IDS	Signature-Based IDS
□ Anomaly based intrusion detection involves making a determination whether the ***behaviour of the system is a statistically significant departure from normal.***	□ ***Signature-based intrusion detection*** (also called ***misuse detection***) works by identifying specific Patterns of events or behaviour that indicate or accompany an attack.
□ The IDS will have to learn, over time, what constitutes normal activity, usage, and behaviour.	□ Each such pattern is called a ***signature.***
□ The first six conditions in Table 22.1 are examples of what an ***anomaly based IDS would monitor.***	□ A signature-based IDS maintains a database of known ***signatures.***
□ Consider monitoring the number of TCP SYN packets (with the **SYN flag set**) and FIN Packets (with the **FIN flag set)** in each successive 10-second interval.	□ It attempts to obtain a match between the ***currently observed behaviour of the system and an entry in this database***.
□ A disproportionate number of **SYN** packets vis-a-vis FIN packets indicate several half-open TCP connections and possibly the onset of a ***SYN flooding attack.***	□ A real world signature-based IDS will have thousands of attack signatures against which to compare.
	□ ***An example of an attack signature is a specific bit sequence in a worm payload.***
	□ In a signature-based IDS it is the presence of a specific signature that raises an alert.
	□ On the other hand, it is possible that a spread of the worm has caused much network traffic congestion and greatly increased CPU utilization on infected machines.

Host-based versus Network-based IDS

Network-based IDS	Host-based IDS
An IDS that captures information about packets flowing through the network is referred to as ***network-based IDS.***	A host-based IDS is typically implemented in ***software and resides on top of the host's operating system.***
For reasons of performance, it is common to have stand-alone appliances that perform network-based intrusion detection. These typically run only the IDS and are hence not vulnerable to various worm and virus attacks.	Its main job is to monitor the internal behaviour of the host such as the ***sequence of system calls made, the files accessed***, etc.
They may be ***deployed at multiple points*** in a large organization.	For this purpose, it makes use of ***system logs, application logs, and operating system audit trails to*** identify events related to an intrusion.
	Operating system logs, for example, keep track of when ***users log in,*** the number of ***unsuccessful login attempts, the commands executed, network connections made,*** etc.
	Application logs keep track of which files have been opened or which registry keys have been accessed during the run of an application.
	File system integrity checkers, for example, compute a cryptographic hash on the contents of each file. They detect file changes by comparing the computed hash of a file to its stored hash.

Two desirable features of an IDS are ***speed and accuracy.***

Speed is especially important in ***fast-spreading Internet worms,*** for example.

Early worm detection and an early response mechanism such as automated system shutdown can help reduce the number of infected 'machines. The IDS should be able to detect every instance of an intrusion.

An undetected intrusion is referred to as a ***false negative***.

2.4 DDoS ATTACK PREVENTION and DETECTION

2.4.1 DDoS Prevention

1) Preventive Measures At The Host
2) Preventive Measures Inside The Network

Preventive Measures at the Host

 One possible way of handling **SYN** attacks is to drop requests for TCP connections.
 But this could result in collateral damage if the victim is unable to distinguish between SYN packets that are part of the attack and those from its legitimate clients.

1. One way to reduce collateral damage is to categorize IP addresses as "almost certainly genuine", ***"probably spoofed",*** etc.
2. The "***almost certainly genuine"*** addresses are those with whom normal connections were established and terminated in the past.
3. Under rapidly increasing load, packets with ***unfamiliar source addresses*** are discarded with high probability.

 Another strategy under high-load conditions is to allocate a full buffer of about 300 bytes for a given TCP connection request only upon completion of the three-way handshake.

1. While the connection is still half-open, minimal information about it is stored in a **hash table** called the **SYN cache**.
2. This information includes the **TCP sequence numbers** and **source/destination addresses and ports.**
3. An alternative to the SYN cache is the **SYN cookie**, which stores no state information at all for each half-open connection.
4. Instead, the responding machine places a cookie within the ***Sequence Number field of the second handshake message.***
5. The cookie is computed as a hash function of the source address, destination address, source port, destination port, and a secret.
6. The initiator of the connection dispatches the cookie it just received in its ACK message (third message of the three-way handshake).
7. Upon receiving the ACK, the responder re-calculates the cookie and verifies that it matches the value enclosed in the received ACK.
8. ***Only then does it reserve buffer space for the connection.***
9. If the source IP address in the first message of the handshake were spoofed, the cookie in second message would not be received by the initiator but by the machine corresponding to the spoofed IP, address.

10. The initiator would not be able to complete the three-way handshake since it does not know the cookie value. Hence, its connection request would not be granted buffer space.

Preventive Measures inside the Network

 An intuitively appealing approach to frustrating DDoS attacks is to implement measures closer to the source of the attack.

 One such measure is ***egress filtering.***

 Attack: Most DDoS attack packets use ***spoofed source IP addresses***.

 Address spoofing is employed to confuse cyber sleuths making it hard to pinpoint the true source of the attack.

 The perpetrator hopes to continue the attack for as long as desired and perhaps even resume it at a later point without being traced.

 Solution: The ***egress router is the last router*** encountered by any packet generated inside the network before it exits that network and enters the Internet.

 Let A be the set of all externally visible IP addresses within the network (behind the egress router). The egress router examines the source address of each packet leaving it. If the address does not match any address in A, it drops the packet.
By thus detecting and filtering spoofed packets' it helps prevent DDoS attacks.

 The idea of ***egress filtering has been extended to routers*** in the core of the Internet.

 A filter, on the other hand, uses the packet's source address to make a decision on whether or not to discard the packet.

 To implement ***Distributed Route Filtering (DRF)***, a filter maintains, for each of its inter-faces, the set of all ***source addresses*** from which packets arrive en route to some destination.

 The router uses ***BGP routing information*** to obtain the ***latest mapping*** between each of its interfaces and the subset of source addresses using that interface.

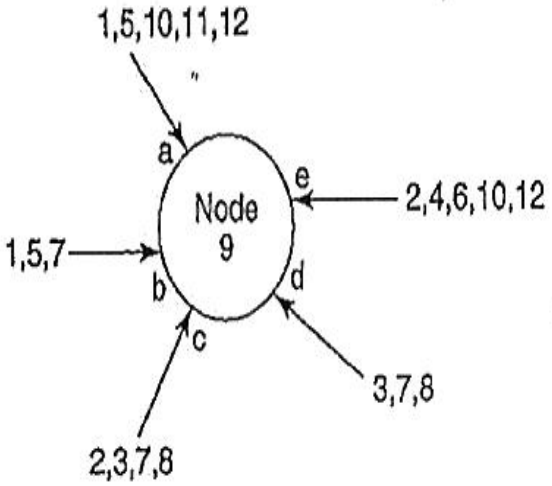
 The filtering decision is straightforward — if a packet with source IP address = S arrives via an interface that it should not have, that packet is assumed to be spoofed and is hence discarded.

 Figure 22.2(a) shows an example of a router implementing DRF.

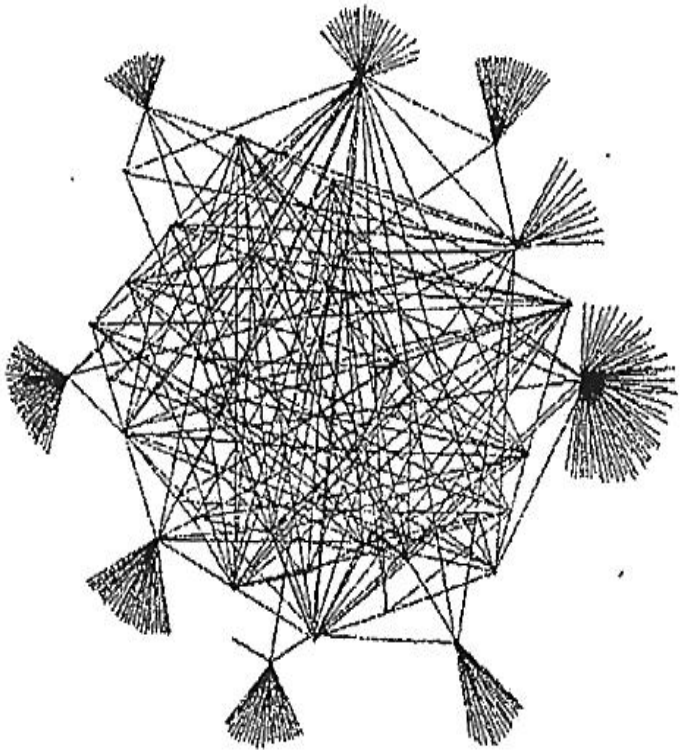
 Each of its interfaces is marked with the source addresses that use that interface en route to some destination.

 Note that packets from the same source may enter the router through different interfaces.

 For example, packets from source address 7 may arrive through interfaces b, c, or d.



(a) Router implementing DRF



(b) Internet Topology (Power Law Graph)

Figure 22.2 *Distributed route filtering*

 In the simplest implementation of the filter, the ***router checks whether a packet has arrived on one of its "acceptable" interfaces based only on the packet's source IP address.***

 For example, a packet bearing source address = 7 arriving on interface **c** would be forwarded. However, another packet with the same source address but arriving on interface **e** would be suspected of having a spoofed source address and would be discarded [see Fig. 22.2a]

Issues in distributed router –based solutions

- Estimating the percentage of the core routers that need to be retro- fitted with a filter for DRF to prevent DDOS attacks.
- The simulations result of [PARKO1] shows that excellent coverage against DDOS attacks is obtained if and only if only about 18% of the core routers are DRF enabled.
- The reason for such a optimistic cost estimate is peculiar power law topology of the Internet. As shown in the figure 22.2b a few nodes in the network are connected to just a few – such a topology is called power law graph

2.4.2 DDoS Detection

 Egress filtering and DRF are preventive mechanisms.

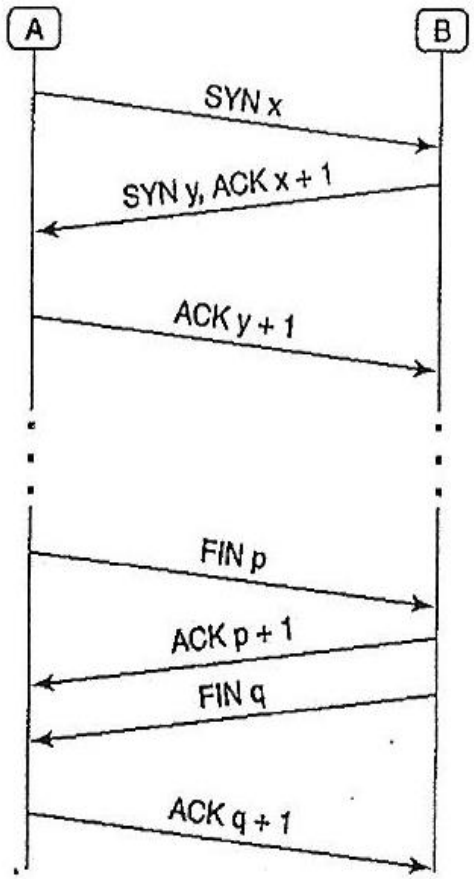
 Another approach is to detect the onset of DoS and then take remedial action.

 In a SYN flood attack, the victim sees a disproportionate number of SYN packets compared to FIN packets.

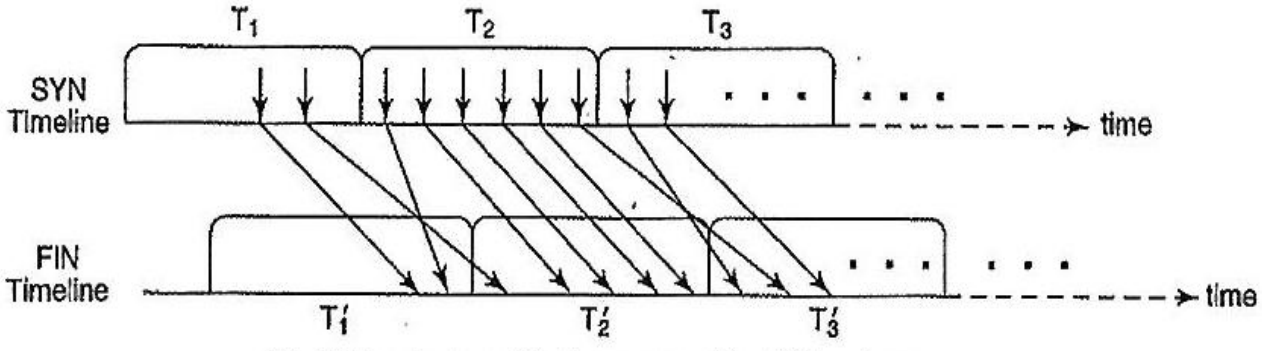
 By a SYN packet, we mean any incoming packet with the SYN flag set.

 A FIN packet is sent by the side that wishes to terminate the TCP connection.

 If the other party agrees to termination, it responds with its own FIN packet. Thus, SYN and FIN packets usually occur in pairs.



(a) TCP connection establishment and termination



(b) SYN packets and their corresponding FIN packets

Figure 22.3 *TCP SYNs and matching FINs*

 Figure 22.3(b) shows two horizontal timelines — the top line shows the times of SYN packet arrivals.

 The bottom line shows the corresponding FIN arrivals.

 Time is slotted into fixed-length observation intervals," T1, during which we record the number of SYN arrivals.

 The corresponding observation intervals for FINs, T1', T2', ... are shifted to the right by the average duration, of a TCP connection.

 To construct an anomaly detection system, we define the following variables as

  Si# of SYN packet arrivals in the i-th observation interval

  Fi = # of FIN packet arrivals in the i-th observation interval

  Di = normalized difference between # of SYN and FIN packets in the i-th observation interval, i.e.,

To construct an anomaly detection system, we define the following variables as in [WANG04].

$S_i \equiv$ # of SYN packet arrivals in the i-th observation interval

$F_i \equiv$ # of FIN packet arrivals in the i-th observation interval

$D_i \equiv$ normalized difference between # of SYN and FIN packets in the i-th observation interval, i.e.,

$$D_i = \frac{S_i - F_i}{F_i}$$

$T \equiv$ threshold for detection

Consider the *time series*,

$$D_1, D_2, D_3, \ldots$$

➢ **The different algorithms that attempt to detect the onset of a SYN Flood Attack by monitoring the above series.**

1. Algorithm 1. Raise an alert if the most recently computed detection variable Di exceeds the threshold, i.e., D, >T1

➢ Figure 22.4(a) shows D versus time with the threshold set at T1 = 90.

➢ Some of the problems with this approach are as follows:

(i) The IDS may raise many *false alarms* since it bases its decision on point values. So, for example, at *time* = 16 in Fig. 22.4(a), the value of D rises to 102 triggering an alarm. However, this alarm is unwarranted since the D values at neighbouring points (around *time* = 16) are well below the threshold, $\mathcal{T}_1$. A modest spike in D at just one point is very unlikely to result in memory exhaustion but it does cause the IDS to raise an alarm.

(ii) The values of D, between *time* 28 and 33 are just below the threshold, $\mathcal{T}_1$ (Fig. 22.4). The *cumulative* effect of the attack packets across the interval will cripple the system but this algorithm will not raise an alarm. Thus, SYN Flood attacks may evade detection by flying below the radar as shown in Fig. 22.4(a).

Our next two solutions, consider the cumulative effect of previous values of D.

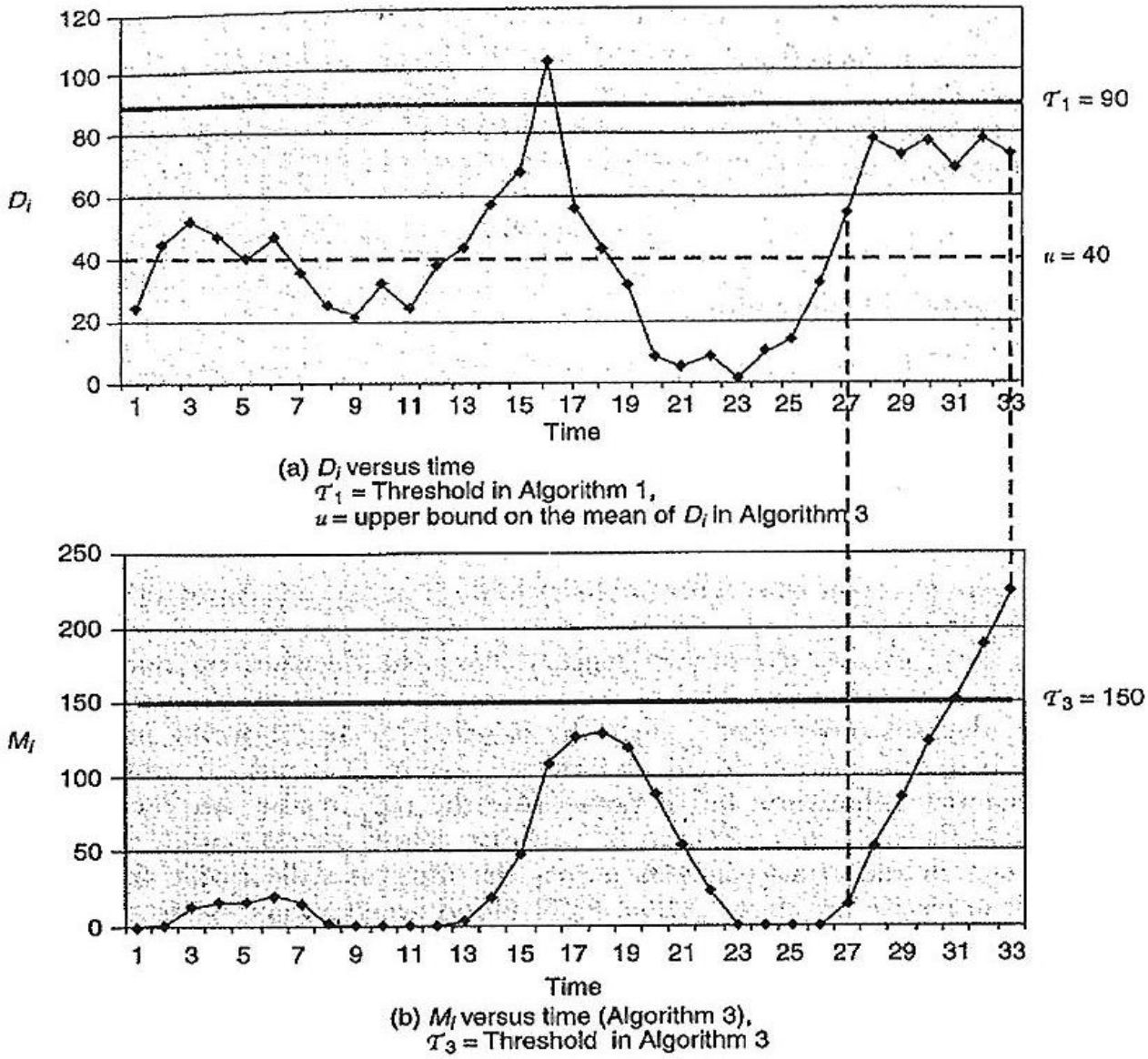


Figure 22.4 *Variables monitored in SYN flood attack detection*

2. Algorithm 2 : Raise an alert if the "smoothed average" of the previous values of D exceeds the threshold.

- This approach uses the well-known technique of ***exponential smoothing.***
- The decision variable at the end of the i-th observation interval is the smoothed average, Si computed using:

$$S_i = \alpha D_i + (1 - \alpha) S_{i-1} \qquad 0 < \alpha < 1 \text{ and } S_0 = 0$$

The above recursive expression for S_i can be expressed iteratively by repeated substitution of S_{i-j} in terms of S_{i-j-1}. This yields

$$S_i = \alpha D_i + \alpha(1 - \alpha) D_{i-1} + \alpha(1 - \alpha)^2 D_{i-2} \ldots$$

For example, for $\alpha = 0.4$, we get

$$S_i = 0.4 D_i + 0.24 D_{i-1} + 0.144 D_{i-2} \ldots$$

The decision variable, S_i, is thus a *weighted sum* of D_i, D_{i-1}, D_{i-2}... with decreasing weights assigned to earlier values of D. Thus, "earlier" values of D count less.

An alarm will be raised if S_i exceeds a threshold T_2. The value of T_2 is set based on empirical data. If it is too low, it will result in many *false positives*. If it is set too high, it will result in *false negatives*. Another design parameter is the "smoothing constant," α. If a value close to 1 is selected, it will give disproportionate importance to the most recent value of D_i. In the limiting case of

3. Algorithm 3. Define a modified cumulative sum of previous values of D. Raise an alert if this value exceeds a threshold.

 During normal operation, the number of FINs will balance out the number of SYNs and hence Di will be close to 0.

Let u be an upper bound on the mean of D_i during normal operations. Let D_i' be a shifted version of D_i, i.e., $D_i' \equiv D_i - u$.

The *decision variable*, M_j, used here is defined as

$$M_j = (M_{j-1} + D_i')^+$$

with $$M_0 = 0$$

Here, the notation x^+ is defined as follows. $x^+ = x$ if $x > 0$, otherwise it is 0.

The IDS sounds an alarm at the end of the j-th interval if $M_j > \tau_3$, where τ_3 is a threshold determined empirically. Figure 22.4(b) shows the value of M_j versus time with a threshold of $\tau_3 = 150$. At *time* = 1, $D_i < u$, so $M_j = 0$. Between *time* 2 and 6, D_i is slightly above u, so M_j *increases* monotonically. Between *time* 7 and 12, D_i' falls below u, so M_j decreases to 0 and remains there until *time* 12.

The interesting interval is between *time* = 27 and 33. Here, D_i is consistently well above u (though it is below the threshold in Algorithm 1). This causes M_j to increase and it *overshoots the threshold* of $\tau_3 = 150$. This causes an alarm to be sounded due to a cumulative build-up of SYN attack packets.

We thus see that with Algorithm 3 (cumulative sum method), the false positive and false negative encountered with Algorithm 1 are both avoided.

2.4.3 IP Traceback

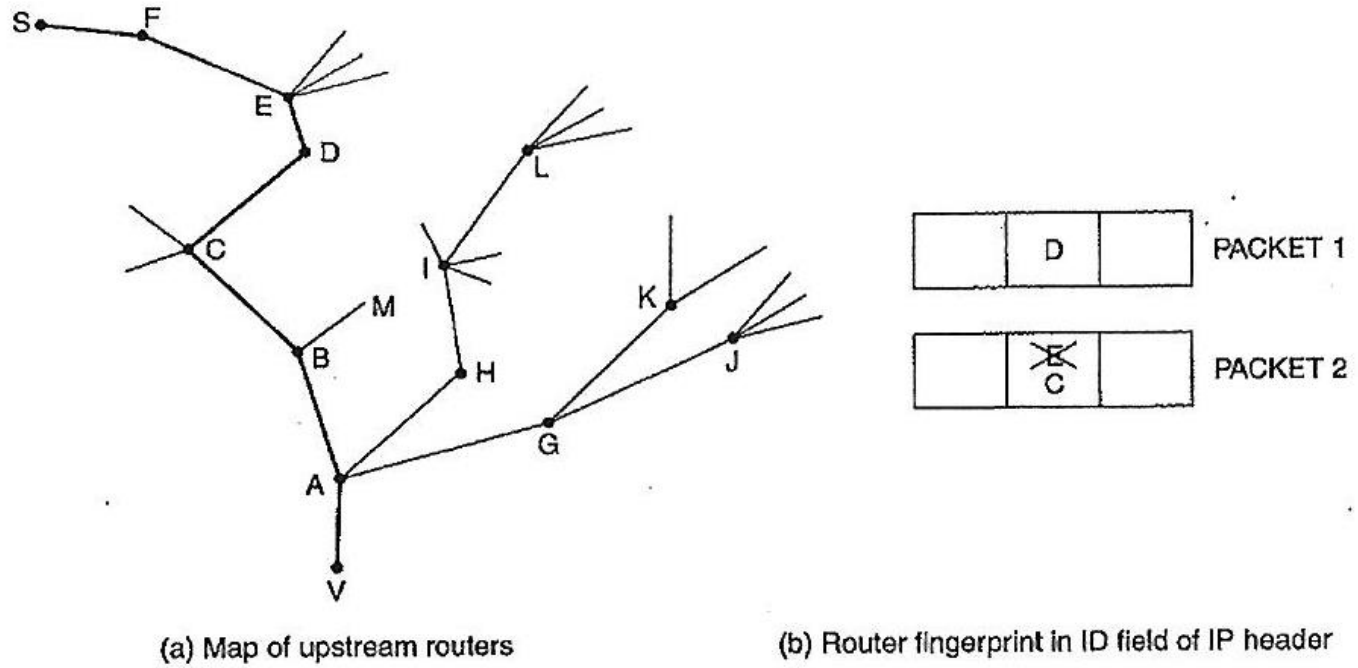
- There are two principal approaches to IP trace back :
- ***packet marking:*** the packet keeps track of the routers it has visited
- ***packet logging :***each router keeps track of the packets passing through it.
- hybrid approaches using a combination of packet marking and packet logging have been proposed.

Probabilistic Packet Marking

- Consider, for a moment that every intermediate router were to ***append its 32-bit IP address*** to each packet it forwards.
- A packet on the Internet traverses about 10 hops on the average, so an extra 40 bytes would be needed to keep track of its path from source to destination.
- This is an unacceptable per-packet overhead.
- Instead, existing but infrequently used fields in the IP header are used to keep track of the routers visited.
- The IP header has a ***16-bit ID field.***
- This field provides support for packet fragmentation and re-assembly.
- Different networks have different restrictions on the size of the datagrams they can carry.
- They may split a datagram into two or more fragments and send each fragment separately.
- The router at the destination end has the responsibility for reassembling the fragments
- To create the original packet.
- All the fragments carry the same number in their ID fields, so they can be identified for re-assembly.
- On the assumption that the ID field is often unused, traceback schemes employing PPM use the ID field to store partial information on intermediate routers.
- But, given that the length of each IP address is 4 bytes, how can a packet store router address information in a 16-bit ID field?
- The answer lies in computing a global fingerprint for each router — this is, say, 16 or fewer bits of the hash of a router's IP address.
- An intermediate router writes its fingerprint value into the ID field of a packet with probability p.
- Note that it could over-write a previously written fingerprint of a router closer to the source of the attack.

 To identify the perpetrator of the attack, the ingress router at the victim end will need to collect a sufficient number of packets that are all part of the same flooding attack.

 We assume that each ingress router has a map of all upstream routers from it.



(a) Map of upstream routers (b) Router fingerprint in ID field of IP header

Figure 22.5 *Probabilistic packet marking*

 V is the victim and S is the source of attack. Since all the packets have been probabilistically marked with fingerprint of the intermediate routers, an ensemble of attack packets will reveal the identities of various intermediate routers, thus helping to reconstruct the attack path.

 Fig. 22.5(b), shows two packets –Packet1 was first marked by D and not overwritten by any downstream router.Packet2 on the other hand was first, marked by Router E and then its ID field was overwritten by Router C.

One problem with this approach is that it is more probable that upstream routers have their marks overwritten. For example, the probability that the mark made by Router F on a packet to V survives is $(1 - p)^5$. The probability that a packet arriving at V is marked by Router F and that the mark survives is $p(1 - p)^5$.

In general, the probability of an incoming packet at V having the mark of a router h hops away is

$$p(1 - p)^{h-1}$$

An important consideration is the number of packets needed at V to reconstruct the attack path. This is closely related to p and the distance between V and the attack source. (Exercise 22.7).

Packet Logging

- ***Each router attempts to keep track of every packet that passes through it.***
- Packet logging makes use of the idea of a packet fingerprint or digest.
- This is computed using a well-designed ***hash function*** — one that distributes the hash values uniformly across all possible hash inputs.
- An interesting feature of packet logging is that it can help track even a single rogue packet.
- First, assume that each router stores each packet received by it in the last 5 minutes.
- Suppose the victim wishes to obtain the exact path followed by a packet received by it.
- The idea is that the victim's ingress router, A, queries each of its adjacent routers whether they have seen the packet.

In Fig. 22.5(a), A would query B, H, and G.

- The router that responds positively, say B, then queries its neighbours, C and M.
- The one that responds positively then contacts its neighbours and so on until the source of the packet is traced.

The storage requirements at each router implementing this scheme could be prohibitive. Consider, for example, a router with six links. Assume that the link speeds are 1 Gb per second and that the router is operating at peak capacity. First, assume that a copy of each packet traversing the router is to be maintained for a period of 5 minutes. So the required amount of storage required in a router is about 1 Terabyte. (We assume that a DoS attack can be detected and traced back to its source in about 5 minutes). At the expense of some computation, we can bring down the storage requirements by storing only the digest or hash of a packet instead of its entire content.

- **The storage requirements can be further reduced by the use of a space-efficient data structure called the Bloom Filter.**
  - Let n be the maximum number of packets to be stored in a router in a given interval, say 7 minutes.
  - Each time an element has to be inserted, one or more hash functions on that element need to be computed.
  - Let k be the number of distinct and independent hash functions used. k is a design parameter.
  - The output of each hash function returns a w-bit quantity.
  - The Bloom Filter is basically a bit array.
  - Let $\mathbf{m = 2^w}$ be the size of this array.

 Packet "Insertion.”: When a packet enters the router, the k hashes are computed on its content.

 To speed up the computation, the hashes are only computed on the invariant parts of the IP header and a small part of the payload, say 10 bytes.

 Suppose the k hash computations yield the values i1, i2, i3,...ik.

 These k hash outputs are used as indices into the bit array.

 To "insert" a packet, the bits in those positions are all set to 1. (If one or more of them were already set, they remain set.)

 Packet Presence Check: To check if a packet, P, is present in the Bloom Filter, compute the k hashes on it as done during packet insertion.

 Suppose the k hash computations yield the values i1, i2, i3, ... ik. Then, check whether each of the elements of the Bloom Filter are set. If even one of these elements = 0, P has not been encountered by this router.

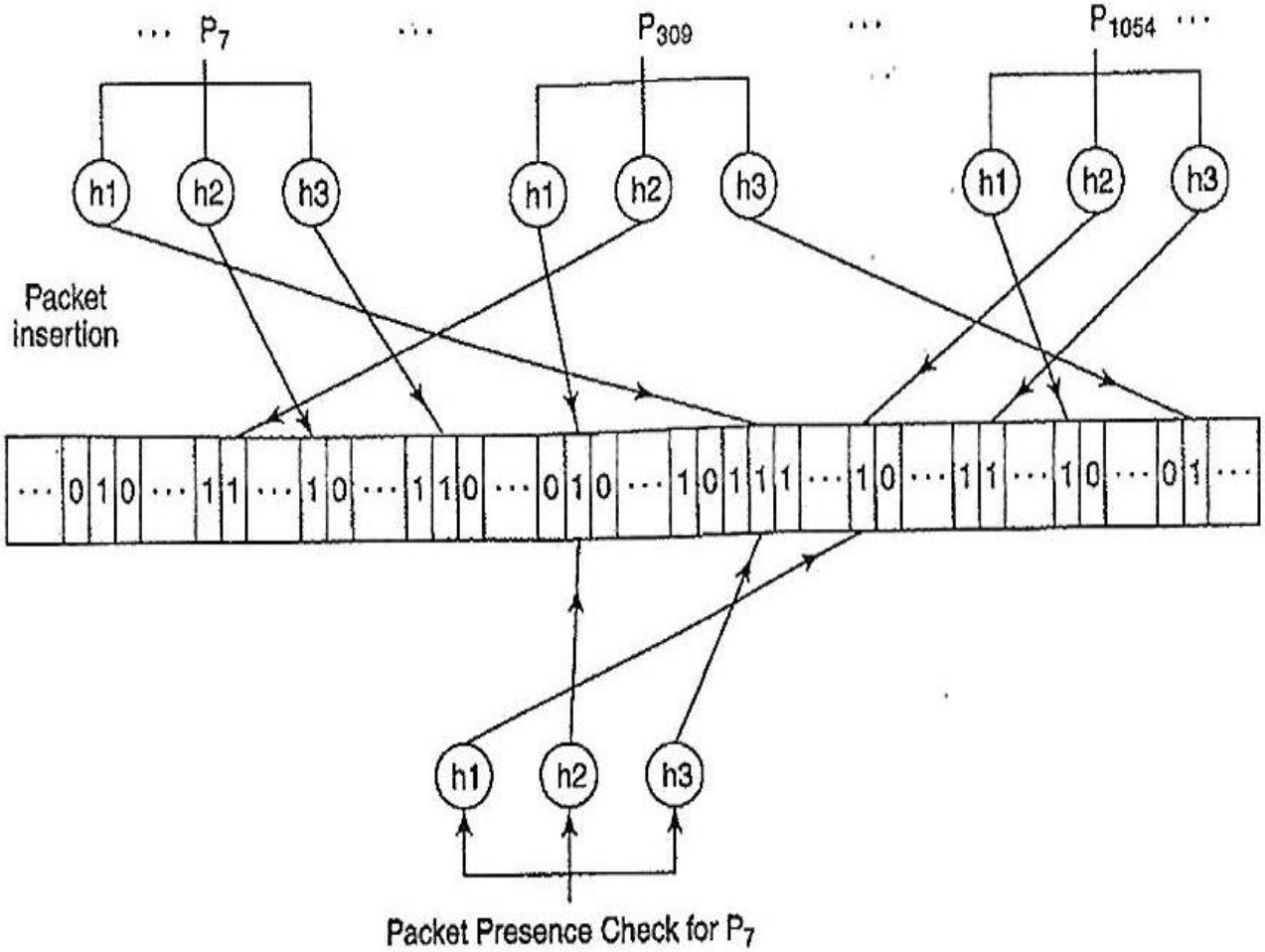


Figure 22.6 *Illustrating false positive in a Bloom Filter*

If, however, all the k elements are set, it is not necessarily true that the router has encountered $\mathcal{P}$ during the current time interval. Suppose that the output of one of the hash functions for at least one packet stored in the Bloom Filter is i_1 and the output of a hash function of at least one packet stored in the Bloom Filter is i_2 and so on. Then, since the values at positions $i_1, i_2, i_3, \ldots i_k$ are all set to 1, we will deduce that $\mathcal{P}$ is stored in the Bloom Filter even though it may not. We will incorrectly conclude that the router has encountered $\mathcal{P}$ in the time interval under observation, resulting in a *false positive*. On the other hand, there is no chance of a false negative with the Bloom Filter.

Figure 22.6 clarifies how a false positive may occur. A router has been presented with Packet P_7 and is being asked whether it has encountered this packet. So, it computes $h1(P_7)$, $h2(P_7)$ and $h3(P_7)$ and it looks to see if the corresponding bits in the Bloom Filter are set. As it turns out, these bits have been set respectively by the application of h2 on Packet P_{1054}, h1 on Packet P_{309} and h1 on Packet P_7. So the system will conclude, rightly or wrongly, that it has encountered P_7 before. We next derive an expression for the probability of a false positive.

Intuitively, for a given n, the chance of a false positive decreases with larger m. On the other hand, a larger value of m incurs a greater storage overhead. We next derive an expression for the probability of a false positive as a function of m and k.

Probability [a packet hashes to i_1] = $\frac{1}{m}$

Probability [a packet does not hash to i_1]

$$= \left(1 - \frac{1}{m}\right)$$

Probability [none of the n packets hash to i_1 with any of the k hash functions]

$$= \left(1 - \frac{1}{m}\right)^{kn}$$

Probability [at least one of the n packets hashes to i_1 with at least one of the k hash functions]

$$= 1 - \left(1 - \frac{1}{m}\right)^{kn}$$

Now the probability of a false positive is given as:

Probability [at least one of the n packets hash to i_1 with at least one of the k hash functions *and* at least one of the n packets hash to i_2 with at least one of the k hash functions

and at least one of the n packets hash to i_k with at least one of the k hash functions]

$$= \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k$$

It turns out that a reasonably acceptable false positive probability of 1% is achievable with $k = 3$ and $m = 12 \times n$. This translates to a storage cost of only 12 bits per packet at the expense of performing three hash computations per packet. This is considerably less than storing each IP datagram (about 500 bytes on average) or even storing just a 32-bit hash of a packet.

WEB SECURITY

1.1 Motivation

1.1.1 Introduction

- Availability of the web and interactive nature of web application have played a role in providing unprecedented convenience to the customer.

- Compared to JSP/Java Servlets or ASP the next generation of component based web technologiessuch as J2EE, .NET,provides scalability and reusability together with support for transaction processing , security etc.

- SSL provides security over communication link

Many of the earlier web applications (such as Internet banking) involved human-to-program interaction. However, applications such as supply chain management differ from traditional web applications in several significant respects:

- Programs communicate with each other over the web with little or no human intervention.
- Services might have a composite nature. Such "composite services" necessitate the involvement of multiple providers, each providing an "atomic service."
- There are potentially a large number of "atomic service" providers offering a given service. So, clients have a choice and can dynamically change providers.
- Clients and providers may be running applications using different web technologies on diverse computing platforms with different operating systems. Inter-operability is thus an important issue.

- **Web Based travel planning** is an application that possesses many of the above feature. A user can visit website to choose his/her travel agent to book airline ticket.
    - Customer could also reserve hotel, car for rentals using the same login session. In this scenario there are atomic service providers' travel agent, the airline, hotel chain and the car rental company. The travel agent would have partnership with airlines, hotels, car rentals which are listed on web page for the user selection option.

- The Computing platform and the software that power their application might be very different from one another.

Web services: The World wide web consortium W^{3}C defines a web service as a software system identified by a URI whose public interfaces and bindings are defined and described by XML. These system may then interact with the web service in a manner prescribed by its definition using XML-based messages conveyed by Internet protocols.

1.1.2 The entities involved:

The atomic web service involves three entities requestor(client),the provider(or server) and a registry as shown below in fig 25.1

- Providers register or publish their services in a public registry.
- Requesters discover services by querying the registry for services that match certain criteria.
- Once a requester has identified a provider whose services it need, it binds to and invokes the service of that provider.

Entities involved in a web service

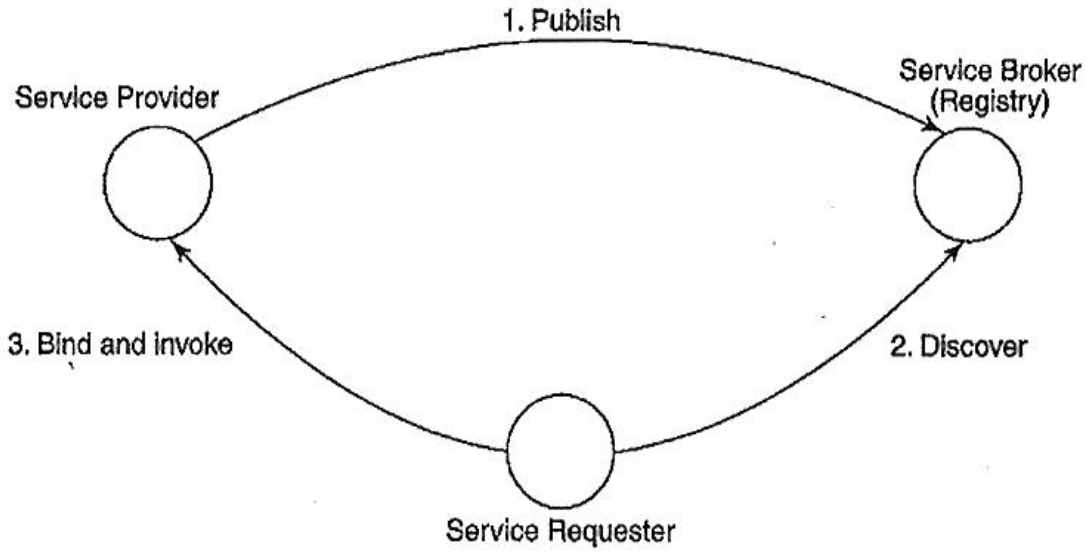


Figure 25.1 *Entities involved in a web service*

- The technologies to support web services are all based on XML-Extended Markup Language, which has become lingua franca for electronic document.

1.2 Technologies for web services.

1.2.1 XML

- XML is a markup language .
- Uses **tags** as a mechanism to identify structures in a document or to specify presentation style /format.
- Example chapter in a text book made up of one or more sections.

Each section in turn is made up of zero or more subsections and each subsection is made up of one or more paragraphs. These facts may be represented using a markup language as follows

```
<chapter>
    <section>
        <subsection>
            <paragraph>
            </paragraph>
        </subsection>
    </section>
</chapter>
```

- XML tags are used to describe the structure of the data .
- XML is a Meta language and provides a facility to define tag sets in diverse fields such as business, medicine, mathematics and law.
- **Element:** is the most basic markup found in an XML document. The start of an element tag within a document is indicated by start tag which contains name of the element within the angular bracket.

Figure 25.2(a) shows a Purchase Order in XML. The tag on Line 3 of the document indicates the start of element *shipTo*. The end-tag, </ *shipTo*> on Line 9 in Fig. 25.2(a) indicates the end of the element, *shipTo*. An *end-tag* can be recognized by the "/" to the immediate right of the opening angular bracket.

- An Element may contain data or it may contain other sub elements or it may contain both data and other sub elements.

In Fig. 25.2(a), the element, *shipTo* contains sub-elements *name*, *street*, *city*, *state* and *PIN*. The sub-elements, in this case. contain only text. For example, the name element (Line 4) contains the customer's name.

- An Element may contain zero or more attributes. An **Attribute** is a name value pair which appears after the element name in the element's start tag

For example *shipTo* (Line 3) contains a single attribute whose name is *country* and whose value is INDIA.

```
1   <?xml version="1.0"?>

2   <purchaseOrder orderDate="2009-01-05">

3      <shipTo  country="INDIA" >
4         <name>  Kiran Kumar  </name>
5         <street> 63 M.G. Road </street>
6         <city>  New Chicago  </city>
7         <state> Gujarat </state>
8        <PIN> 123456 </PIN>
9      </shipTo>

10     <items>

11        <item partNum="129BZ" >
12           <productName> Electric Toaster </productName>
13           <quantity> 1 </quantity >
14           <UnitPrice> 1412.00 </UnitPrice >
15        </item>

16        <item partNum = "798RD" >
17           <productName>  Dinner Set </productName>
18           <quantity> 1 </quantity>
19           <UnitPrice> 2142 </UnitPrice>
20        </item>

21     </items>

22  </purchaseOrder>
```

Purchase order conforming to schema definition in (b)

Figure 25.2(a) *Purchase order in XML*

The Purchase Order in Fig. 25.2(a) is highly structured – the name and address of the person receiving the shipment occurs first. This is followed by the items ordered. Each item includes the *productName*, *quantity*, and *UnitPrice* in sequence. The correct sequencing and nesting of elements is necessary since computers are expected to process such documents. But how does a computer know what to expect in a document such as a Purchase Order?

A *Document Type Definition* or the more recently standardized *XML schema* contains rules to interpret the document's content. The rules include information such as

- What is the element type, e.g., string, decimal, complex type, etc.?
- Is an element optional? If not, how many times should it occur (once, one or more times, etc.)?
- Does an element have any attributes? If so, what are their names and types?
- What is the content of an element (other sub-elements or text)?
- What is the sequence of elements and how are they nested?

Figure 25.2(b) shows the XML schema representation of a purchase order. The purchase order document of Fig. 25.2(a) is an instance of this schema. We consider here a toy example: a real-world schema of a purchase order may include hundreds of elements and attributes.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://xyz.org/po.xsd"
xmlns="http://tempuri.org/po.xsd" elementFormDefault="qualified">

 <xs:element name="purchaseOrder" type="PurchaseOrderType"/>

 <xs:element name="comment" type="xs:string"/>

 <xs:complexType name="PurchaseOrderType">
  <xs:sequence>
   <xs:element name="shipTo" type="Address"/>
   <xs:element ref="comment" minOccurs="0"/>
   <xs:element name="items"  type="Items"/>
  </xs:sequence>
  <xs:attribute name="orderDate" type="xs:date"/>
 </xs:complexType>

 <xs:complexType name="Address">
  <xs:sequence>
   <xs:element name="name"   type="xs:string"/>
   <xs:element name="street" type="xs:string"/>
   <xs:element name="city"   type="xs:string"/>
   <xs:element name="state"  type="xs:string"/>
   <xs:element name="PIN"    type="xs:decimal"/>
  </xs:sequence>
  <xs:attribute name="country" type="xs:NMTOKEN"/>
 </xs:complexType>

 <xs:complexType name="Items">
  <xs:sequence>
   <xs:element name="item" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="productName" type="xs:string"/>
      <xs:element name="quantity">
       <xs:simpleType>
        <xs:restriction base="xs:positiveInteger">
         <xs:maxExclusive value="200"/>
        </xs:restriction>
       </xs:simpleType>
      </xs:element>
      <xs:element name="UnitPrice"    type="xs:decimal"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
 </xs:complexType>

</xs:schema>
```

XML schema definition of a purchase order

Figure 25.2(b) *Purchase order in XML*

BMSIT&M, Dept of CSE

Chethana C

chapter 6 public key cryptography & RSA (1)

RSA operations

- generate a public key/private key pair
- one time operations, unless compromised

steps

1) choose prime p & q
   $n = p \times q$ → modulus and $\phi(n)$
2) choose encryption key e, such that gcd $(e, \phi(n)) = 1$ (coprime)
   ∴ publickey = pair (e, n) pair of integers
3) compute decryption key $d = e^{-1} \bmod \phi(n)$
   d = privatekey

note: $\phi(n) = pq - 1 - (p-1) - (q-1) = (p-1)(q-1)$

procedure: Encryption

Let m = message
|m| = length of m

* message is split into multiple blocks, each of size b except possibly for the last block.
* the size of last block is |m| mod b if different from 0.
  where $b = \lceil \log_2 n \rceil$ → no of bits used to represent
  n is referred to as the key size, b
* for each block m_i, we find corresponding ciphertext c_i as

$$c_i = m_i^e \bmod n$$

Decryption: For the given block of ciphertext c_i, the corresponding plaintext is

$$m_i = c_i^d \bmod n$$

Ex: Let $p=3$, $q=11$, $m = 00111011$

$\therefore n = p \cdot q = 33$

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

$= 2 \times 10 = 20$

* choose publickey e, such that $\gcd(e, \phi(n)) = 1$

ex: Let $e=2$, $\gcd(2,20) \neq 1$

$e=3$, $\gcd(3,20) = 1$ ✓

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

$d = 3^{-1} \bmod 20$

$d = 7$

$\therefore$ private key = 7

$e^{-1} = 1 \bmod 20$

~~$d^{-1} = 1 \bmod 20$~~

$3e^{-1} = 1 \bmod 20$

$3 \times 7 e^{-1} = 7 \bmod 20$ *by 7

$21 e^{-1} = 7 \bmod 20$

$1 \cdot e^{-1} = 7 \bmod 20$

$3^{-1} = 7$

* Let $b=6$, Key Size (minimum no of bit in n)
* Given $m = 001110|11$

block1 = 001110 = m_1

block2 = 000011 = m_2

$m_1 = 001110$ $m_2 = 000011$ (padded with zero to the left) (=3 in decimal)

$m_1 = 14$ $m_2 = 03$

Encryption

(1) $c_1 = m_1^e \bmod n$

$c_1 = 14^3 \bmod n$

$= 14 \times 14 \times 14 \bmod 33$

$= 196 \times 14 \bmod 33$

$= (196 - 33 \times 6) \times 14 \bmod 33$

$= -2 \times 14 \bmod 33$

$= -28 \bmod 33$

$c_1 = 5$

Reduction

note: $33 \times 6 = 198$

Decryption

(1) $m_1 = c_1^d \bmod n$

$m_1 = 5^7 \bmod n$

$= 5 * 5 * 5 * 5 * 5 * 5 * 5 \bmod 33$

$= 25 * 25 * 25 * 5 \bmod 33$

$= (-8) * (-8) * (-8) * 5 \bmod 33$

$= 64 * (-40) \bmod 33$

$= (64 - 33 \times 2) * (-40) \bmod 33$

$= -2 * (-40) \bmod 33$

$= 80 \bmod 33$

$= (80 - 33 \times 2) \bmod 33$

$= 80 - 66 \bmod 33$

$m_1 = 14$

Reduction modulo

This may be done at one or more intermediate steps

* helps to maintain a bound on the size of the numbers involved

$(25 - 33 \times 1)$

$= -8$

② Encryption

$m_2 = 3$

$c_2 = 3^3 \bmod 33$

$= 27 \bmod 33$

$c_2 = 27$

Decryption $(c_2^d \bmod n$

$m_2 = 27^7 \bmod 33$

$= 27 \times 27 \times 27 \times 27 \times 27 \times 27 \times 27 \bmod 33$

$= 729 \times 729 \times 729 \times 27 \bmod 33$

$= (729 - 33*22) \times (729 - 33*22) \times (729 - 33*22) * 27 \bmod 33$

$= 3 \times 3 \times 3 \times 27 \bmod 33$

$= 27 \times 27 \bmod 33$

$= (27-33)(27-33) \bmod 33$

$= (-6)(-6) \bmod 33$

$= 36 \bmod 33$

$= (36-33) \bmod 33$

$= 3 \bmod 33$

$m_2 \equiv 3$

or

$729 \bmod 33$

$(729 - 33*22) \bmod 33$

$3 \bmod 33$

$= 3 = m_2$

Note: In real applications, the modulus n and decryption key d are hundreds of digits long

software performance

Java has no of APIs of relavance to cryptography.

↳ Key generation

↳ encryption/decryption

Java.security package & sub packages

Practical issues

1) Generating Primes
2) Side channel and other attacks
3) s

↳ modulus factorization

↳ Small Exponent attack.

↳ Side channel attack.

6.2 How does RSA Works

Let the i-th message block have value = m_i. Let its corresponding ciphertext be c_i. Decrypting the ciphertext using Eq. (6.2) yields

$$\begin{aligned} c_i^d \bmod n &= (m_i^e \bmod n)^d && \bmod n && \text{from Eq. (6.1)} \\ &= m_i^{e \times d} && \bmod n && \text{laws of modulo arithmetic} \\ &= m_i^{1 + k \times \Phi(n)} && \bmod n && \text{for some integer } k \end{aligned}$$

The last step follows from the fact that d was chosen to be the inverse of e modulo $\Phi(n)$. So, $e \times d$ and 1 differ by an integral multiple of $\Phi(n)$.

We next show that $m_i^{1 + k \times \Phi(n)} \bmod p = m_i$

$$\begin{aligned} m_i^{1 + k \times \Phi(n)} \bmod p &= m_i \, m_i^{k \times \Phi(n)} \bmod p \\ &= m_i \, (m_i^{k \times (q-1)})^{(p-1)} \bmod p \end{aligned}$$

(a) Suppose $\gcd(m_i, p) = 1$. Then $\gcd(m_i^{k \times (q-1)}, p) = 1$.
From Fermat's Little Theorem,

$$(m_i^{k \times (q-1)})^{(p-1)} \bmod p = 1$$

So

$$m_i^{1 + k \times \Phi(n)} \bmod p = m_i \bmod p$$

(b) On the other hand, suppose $\gcd(m_i, p) > 1$. This implies that m_i is an integral multiple of p. In that case

$$m_i^{1 + k \times \Phi(n)} \bmod p = m_i \bmod p = 0$$

We conclude that, regardless of the value of the message m_i,

$$m_i^{1 + k \times \Phi(n)} \bmod p = m_i \bmod p \tag{6.3}$$

In a similar manner, it can be shown that regardless of the value of the message m_i,

$$m_i^{1 + k \times \Phi(n)} \bmod q = m_i \bmod q \tag{6.4}$$

From Eq. (6.3),

$$m_i^{1 + k \times \Phi(n)} = m_i + c_1 p \qquad \text{for some integer } c_1 \tag{6.5}$$

From Eq. (6.4),

$$m_i^{1 + k \times \Phi(n)} = m_i + c_2 q \qquad \text{for some integer } c_2 \tag{6.6}$$

Equating the RHS of Eqs (6.5) and (6.6),

$$m_i + c_1 p = m_i + c_2 q$$

So

$$c_1 p = c_2 q$$

Since p and q are prime, it follows that c_1 should have q as factor.
So

$$c_1 p = c_3 q \, p \qquad \text{for some integer } c_3$$

Substituting in Eq. (6.5), we get

$$m_i^{1 + k \times \Phi(n)} = m_i + c_3 pq = m_i + c_3 n$$

So

$$c_i^d \bmod n = m_i^{1 + k \times \Phi(n)} \bmod n = m_i$$

6.3 Performance

6.3.1 Time Complexity

Encryption

- Both Encryption and Decryption involve repeated Multiplications (modulo n) of n bit numbers.
- The encryption key is usually a small integer (relative to n).
- So encryption involves a small constant number of modulo n multiplications. The time complexity of encryption is O (b^2)

Decryption

- Involves raising a b-bit number to the power of d.
- A naïve implementation of decryption involves d multiplication.
- Since d is of the same order as n complexity of a decryption is O(nb^2).

6.3.2 Speeding up RSA

1. **Square and Multiply**
   - This approach first computes the squares followed by the products
   - We can speed up the decryption of ciphertext C by computing C, C^2, C^4, C^8 etc up to a maximum number of b terms
   - Each element in a series is the square of preceding element.
   - Then we multiply elements in his series whose positions correspond to 1's in the binary representation of the decryption Key d.
     - Each multiplication is a modulo n multiplication so the intermediate products are never more than b bits wide.
     - Example as follows below.

Suppose the decryption key is 57.
Then, naive decryption would involve a total of 56 modulo n multiplications.
However, the "square and multiply" approach involves computing

$$c^2,\ c^4,\ c^8,\ c^{16} \text{ and } c^{32} \qquad \text{each reduced modulo } n.$$

Since the binary representation of 57 is 111001, we selectively multiply c, c^8, c^{16}, *and* c^{32} to obtain the original plaintext (see Fig. 6.2). Thus, we now perform decryption using only *five square operations* and *three multiplications*, a considerable savings compared to 56 *multiplications* without "square and multiply."

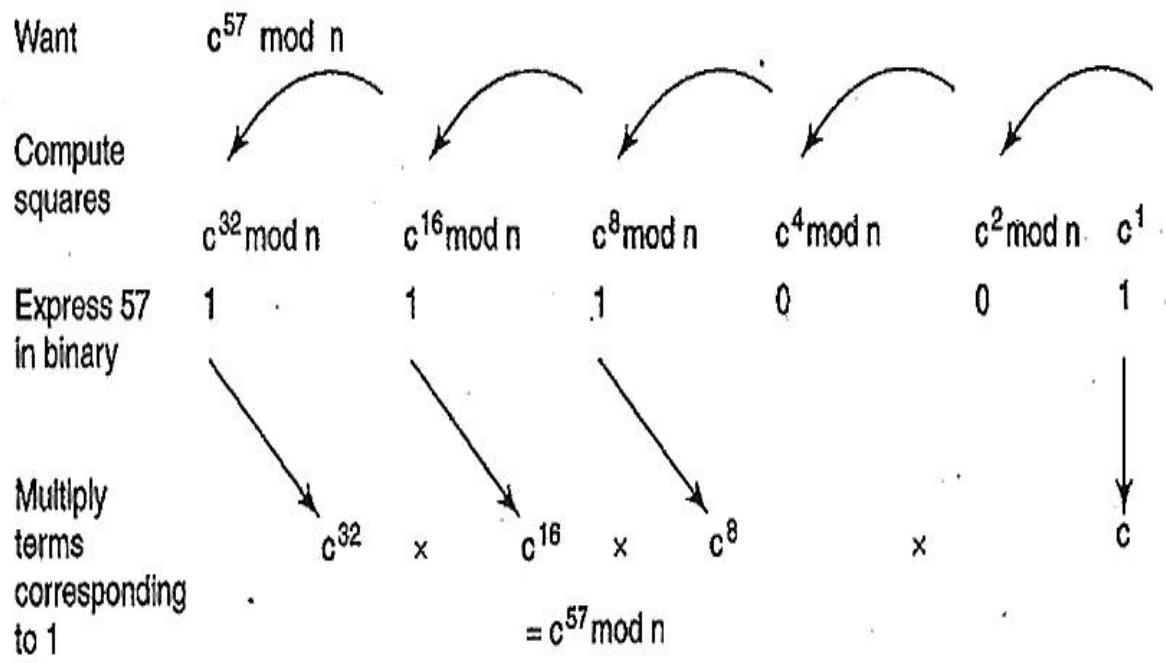


Figure 6.2 *Use of "square and multiply" in RSA*

- In general Decryption involves b-1 squre opearions and at most b-1 multiplication.
- Each square and multiplication is followed by reduction modulo n.

2. **Key Size:** The choice of Key size represents a tradeoff between security and performance.
   - A larger the key size provide the greater security.
   - But time for encryption and decryption increases.
   - Doubling the key size increases the time complexity for encryption by roughly a factor of 4 and decryption by a factor of 8 according to asymptotic notations.

6.3.3 Software Performance

- The Java programming language has a number of APIs of relevance to cryptography
- APIs for
  - Key Generation.
  - Encryption / Decryption,
  - Message digests,
  - Digital signature.
- These are contained in java.security.package and its various sub packages.
- Java also permits the import of classes created by various third parties that implement cryptographic algorithms
- Bouncy Castle is an example for third party provider whose API is available for use in both Java and C++ programs.

An example of the use of the Java APIs for key generation, encryption, and decryption is shown in Fig. 6.3.

```
KeyPairGenerator kpg = KeyPairGenerator.getInstance("RSA", "BC");
kpg.initialize(1024);
KeyPair kp = kpg.generateKeyPair( );
Cipher c = Cipher.getInstance ("RSA/ECB/PKCS1Padding", "BC");

String plainText = "Hello World!";
c.init(Cipher.ENCRYPT_MODE, kp.getPublic( ));
byte [] encryptedText = c.doFinal(plainText.getBytes( ));

c.init(Cipher.DECRYPT_MODE, kp.getPrivate( ));
byte [] decryptedText = c.doFinal(encryptedText);
String recoveredText = new String(decryptedText);
```

Figure 6.3 *Illustrating Java APIs for RSA encryption/decryption*

Figure 6.4 shows the time to perform encryption and decryption of a 60-byte integer for various key sizes. Execution times are on a Pentium 3.2 GHz machine with 512 MB RAM.

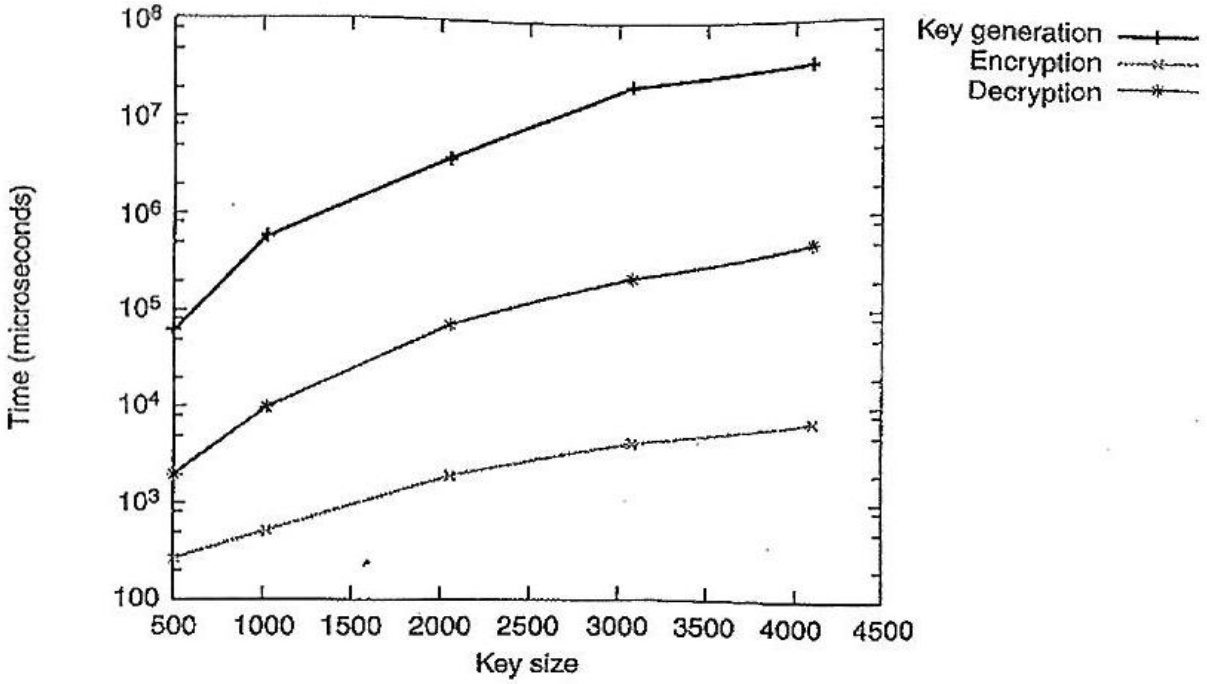


Figure 6.4 *Time for RSA key operations as a function of key size*

6.4 Applications

1. Message Confidentiality: one of application of Public key cryptography.

- Achieved through encryption.
- Suppose A wants to send a confidential message to B.
- Public key Cryptography:
  - With Public key encryption A needs to use B's public Key.
  - Computationally intensive and more expensive.
  - Each entity must store only its private key securely.
- Secret Key Cryptography
  - With secret key cryptography A and B need to share a secret .
  - Each pair of the entities would have agreed up on a secret key (using offline mode).
  - Then securely shares the secret Keys- one per entity it wishes to communicate with.
- Hoe does A obtains B's public key.
- This is done through a digital certificate.
- B's digital certificate is an authentic electronic document from which one can extract the public key of B.

To combine the speed of secret key cryptography and the convenience of public key cryptography, a *session key* is often employed. Here's how it works.

The sender

- chooses a fresh random number, *s*, as the secret key. This is referred to as a session key
- encrypts the message with the session key ($E_s(m)$)
- encrypts the session key with the recipient's public key ($E_{B,pu}(s)$)
- sends the encrypted message and the encrypted session key in the same message (see Fig. 6.5).

The receiver

- uses his private key to decrypt the part of the message containing the encrypted session key
- uses the session key to decrypt the message (see Fig. 6.5).

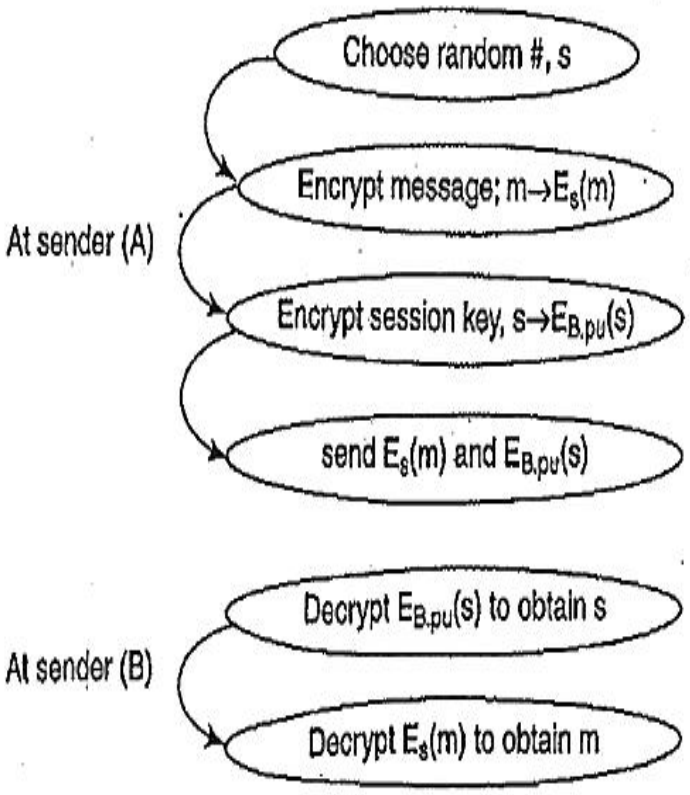


Figure 6.5 *Encrypted message with encrypted session key*

- The Session key is used to encrypt /decrypt the remaining messages in that session.
- The Session key is valid for only the duration of the session and is destroyed thereafter.

2. **Message Integrity and authentication:** Public key Cryptography can be used to generate digital Signature that provides message Integrity and authentication together with non-repudiation.

6.5 Practical Issues

6.5.1 Generating Primes

1. Naïve Methods

- To check that the number is prime or not, we could examine divisibility by all integers less than $\sqrt{p}$.
- The reason for stopping at $\sqrt{p}$ is that if p is composite (non-prime) then at least one of its factors must be less than $\sqrt{p}$
- Another optimization technique could b divisible by odd integers only.

Disadvantage:

- Do not easily scale up
- Not feasible in primality testing of integers that are hundreds of digit long.

2. Miller Rabin Test

- It is an probabilistic method.
- This asserts that the number is prime with some probability, $1-\varepsilon$, ε can be made arbitrarily small.(at the rate of greater computational time).
- This test uses Fermat's theorem.
- It rejects the Hypothesis that an integer p if for an arbitrary integer, i<p,

$$i^{p-1} \neq 1 \pmod{p}$$

3. The AKS Test

- It is an deterministic test for primality .
- Known after the originators Mahindra Agarwal, Neeraj Kayal and Nitin Saxena.
- Its time complexity is $O(\log^{12} p)$.
- Its claims to fame that it is polynomial in log p and that holds unconditionally for all candidate integers not just those with specific characteristics.
- There have been improvements in AKS that run in $O(\log^{6} p)$ time.

6.5.2 Side channel and Other Attacks: Several ways in which RSA can be attacked.

- Modulus Factorization
- Small Exponent attack
- Side Channel Attack.

A. Modulus Factorization

- Attack on the mathematical foundation of RSA.
- One way of attacking RSA is to find factorization of modulus n that is obtaining its prime factor p and q.
- Using p, q attacker can find ϕ (n) and decryption key d (using extended Euclid's algorithm).

1. Pollard Rho Algorithm

- Integers modulo n are randomly selected .Let these numbers be denoted by $r_1, r_2 \ldots$
- For each new integer r_i selected $\gcd(r_i - r_j, n)$ is computed for each $j<i$.
- Generating integers will be stoped when $\gcd(r_i - r_j, n) > 1$ for some j. This happens when $r_i - r_j$ is a multiple of p or q, which occurs when $r_i \bmod p = r_j \bmod p$.
- We need to select on an average about O(sort(p)).
- Pollardo rho uses loop that involves only two gcd computations per iteration with O(1) per storage. The average number of iterations is O (sort (p)).
- This algorithm is reasonable choice for factorizing RSA moduli that are tens of digits long.
- But what about real –world moduli that are hundreds of digit long.

It turns out that, thus far, *no polynomial time algorithm* has been devised for factoring an arbitrarily large integer that is itself the product of two very large integers (hundreds of digits long) of comparable size. The best known factorization algorithms together with their running times are

Quadratic sieve	$O\left(e^{(1+o(1))\sqrt{(\ln n)(\ln \ln n)}}\right)$
Elliptic curve	$O\left(e^{(1+o(1))\sqrt{(2 \ln p)(\ln \ln p)}}\right)$
General Number Field Sieve (GNFS)	$O\left(e^{(1.92+o(1))\sqrt[3]{(\ln n)^{1/3}(\ln \ln n)^{2/3}}}\right)$

Note: MIP-years: One MIP year is the amount of processing power made available by one machine running continuously for a year and executing 1 million instructions per second.

2. Parallel Processing

- With today's best known factorization algorithm the horse power to factorize a 600 bit modulus is about 800 MIP-years.
- This translate to a completion time of 20 years on today's mid-range desktop.
- One option is to use Parallel Processing. Parallelize the factorization algorithm and employ tens or hundreds of high end machines to obtain results in few weeks.

B. Small Exponent attack

- Consider the scenario of person wishes to send message m to three persons.
- Assume each persons has same key =3.
- RSA moduli of three persons would almost certainly be different.
- Let these be n1,n2 and n3 and N=n1*n2*n3

Now suppose an attacker eavesdrops upon the ciphertexts, c_1, c_2, and c_3. These are related to the message, m, by

$$c_1 = m^3 \bmod n_1$$
$$c_2 = m^3 \bmod n_2$$
$$c_3 = m^3 \bmod n_3$$

- Since the prime factors of n1, n2, n3 are different, it follows that n1, n2, n3 are relatively prime.
- Hence knowing the residues m3 mod n1, m^3mod n2, m^3 mod n3, Chinese reminder theorem can be used to reconstruct m^3 mod N.

Since $m < n_1$, $m < n_2$ and $m < n_3$, $m^3 < N$.

- Hence, $m^3 \bmod N = m^3$ and so $m = (m^3 \bmod N)^{1/3}$.
- A more obvious attack with an encryption key e=3 occurs if an attacker knows or guesses that the message $m<N^{1/3.}$
- In this case, the operation cube root modulo N on the cipher text reduces to the regular algebraic cube root of an integer.

C. Side Channel attack

1. Based on monitoring timing and power measurements of a cryptographic algorithm on a device.

- Successful in leaking sensitive information like secret/private keys.
- This is especially the case for embedded devices such as smart card.
- It is generally not possible for the attacker to inspect the contents of registers and RAM during smart card operation.
- However there is inexpensive, off the shelf equipment available that enables him/her to connect a smart via probes to equipment that can accurately monitor variables such as timing and power consumption.
- Since these attacks access such a channels they are referred to as side-channel attacks.
- Side-channels from embedded devices are less noisy since cryptographic operations are usually performed with little interference from other operations or processes within the embedded device.

The leakage of key information in a side channel attack is not due to a poor cryptographic algorithm but due to the peculiarities of its implementation. Consider the snippet of code in Fig. 6.6 that performs RSA decryption using the "*Square and Multiply*" technique. The decryption key, d, is assumed to be a k–bit integer with the most significant bit = 1.

```
Given d, n, and c
Want: c^d mod n

// k ≡ log2 n is the key size
x = c
for (i = k – 2; i ≥ 0; i--)
    x = x^2 mod n;              // Square operation
if (di == 1)
    x = x × c  mod  n;          // Multiply operation
return (x);
```

Figure 6.6 *Implementation of square and multiply with conditional multiply*

In Fig. 6.6, the *square operation* is executed in each iteration. However, the *multiply operation* is skipped if the corresponding bit in the decryption key, d, is a 0. Conditional execution of this type may be exploited by an attacker to deduce, for example, the number and positions of 1's in d. Here are some possible strategies.

- The attacker may carefully monitor the power consumed by the smart card over the duration of the decryption.
- If the power consumption characteristics of the square and multiply operation are dissimilar then the attacker can identify the iterations during which the multiply operation is skipped.
- From this he/she can readily deduce the positions of 1's in the decryption key, d.as shown in the Fig 6.7.

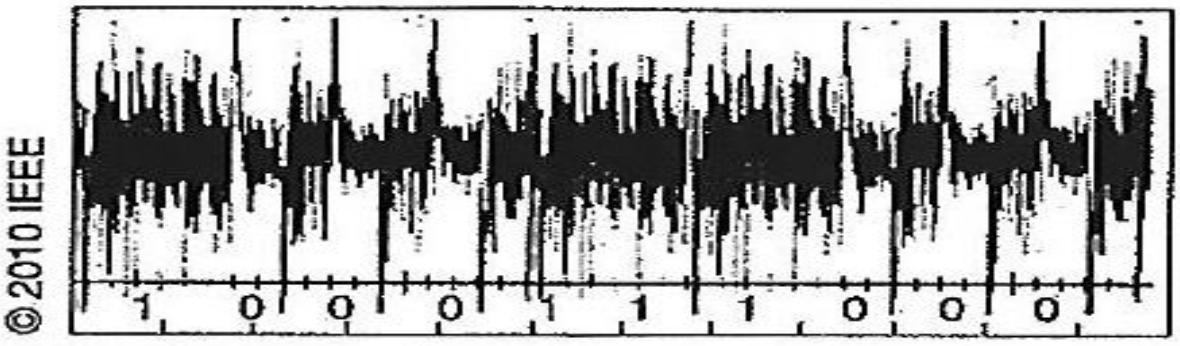


Figure 6.7 *Power profile of decryption operation using implementation of Fig. 6.6**

- These operations involve modulo n reduction. If the result obtained after

performing the multiplication or square operation is less than n, then no reduction is required.

- Hence the time for multiplication and squaring are not constants but depend on the input C.
- The attacker may experiment with different inputs c and also with different decryption keys which provides further insights into timing and power requirements.

Solution

- To thwart the side channel attacks the implementation of fig. 6.8 may be employed.

```
Given:  d, n, and c
Want:   c^d mod n

// k ≡ log2 n is the key size
x = c
for ( i = k – 2; i ≥ 0; i-- ) {
    x = x^2 mod n          // Square operation
    y = x × c mod n        // Unconditional
                           // Multiply operation
    if (di == 1)
      x = y
}
return (x)
```

Figure 6.8 *Implementation of square and multiply with unconditional multiply*

- The multiplication operation is performed in each iterations regardless of the value of the bit in d inspected during that iterations.
- Thus the attacker will be unable to launch a successful side channel attack based on timing and power measurements.

2. Side channel attack by inducing transient faults.

- Another class of side channel attack by inducing transient faults into the chip in a smart card.
- The radioactive particles produced by heavy metals such as uranium and thorium caused electronic hardware to malfunction.
- These metals were present in very tiny quantities in the package material around chip and caused bits in a processor to randomly flip.

- Since the sophisticated techniques including those using highly focused laser beams have been used to target very specific parts of an embedded processor at specific points during program execution of a given operation.

3. Other technique at injecting faults manipulate the voltage supply or the clock to a smart card

- Most smart card require external voltage supply and an external clock input.
- Glitches in execution may occur when very high or low clock frequencies are applied or when the spikes in the voltage supply are introduced.
- The effect of such input may cause instructions to be skipped, data to be corrupted etc.

4. How does the induction of a fault help the attacker to deduce the Key?

To answer this question, we turn to Fig. 6.8, which shows the implementation of the Square and Multiply algorithm with the unconditional multiply operation. Now suppose the attacker injects a fault in the system during the multiply step of a certain iteration. Then the result of the multiply operation will be erroneous. If the bit of the decryption key during that iteration is a 0, then the multiply instruction is superfluous and will not affect the final decrypted output. However, if the bit of the decryption key during that iteration is a 1, then the multiply instruction is necessary and an error in it will lead to a faulty decrypted value.

To obtain the decryption key, this experiment would have to be repeated *d* times. Each time a fault is injected in the multiply instruction of a different iteration. The result of the decryption is compared to the correct value (without faults). If the result does not match the correct value, the attacker infers that the bit of the decryption key is a 1.

- For all embedded systems there is a need to optimize the design for speed, chip area power requirements etc.

6.6 Public Key Cryptography standard (PKCS).

- A solution to the problems with small encryption keys is to pad the message with non- zero random bits before performing encryption.
- The number and position of these random bits has been standardized so that the receiver does not misinterpret the random bits of the data.
- Padding is also important if the message contains data that can be guessed.

- An attacker could guess the plain text the encrypt with public key and verify whether its encrypted version coincides with the ciphertext sent.
- He/she could repeat this sequence of guess –encrypt- verify until a match is found between his/her encrypted value and the ciphertext sniffed by him/her.

The *Public Key Cryptography Standard* (PKCS # 1) specifies, among other things, the format of each block to be encrypted by RSA. As shown in Fig. 6.9, the bytes of the block from left (most significant) should be 00 followed by the byte 02 (in hexadecimal) followed by *at least eight* random non-zero bytes and another 00. The rest of the block is composed of data [Fig. 6.9(a)]. The 00 to the right of the random byte string indicates the start of the data section in a block.

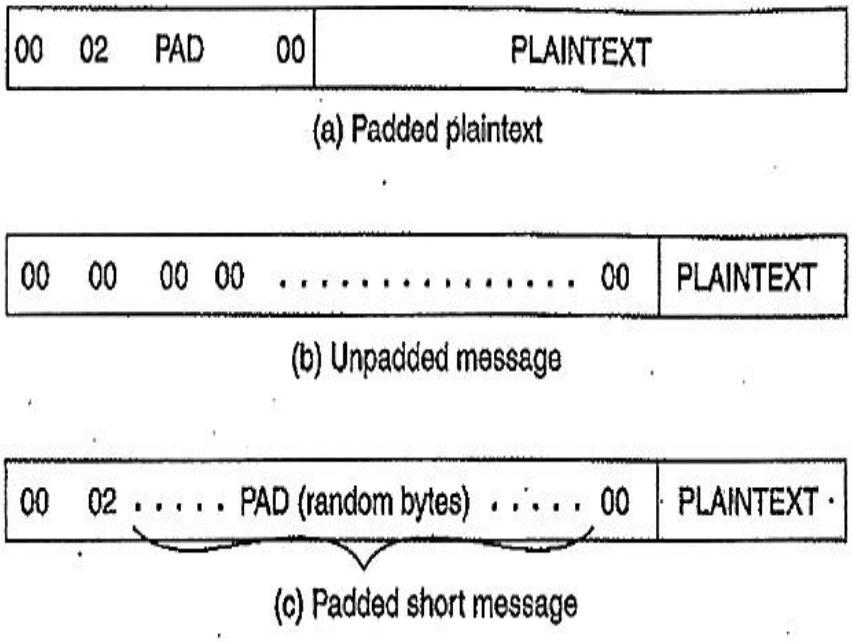


Figure 6.9 *Use of plaintext padding in RSA encryption.*

By padding a short message with random bytes in a block of plaintext, the ciphertext will be a function of not just the short message as in Fig. 6.9(b), but also a function of the large sequence of random bytes [Fig. 6.9(c)]. To be successful, an attacker will have to guess not merely the message but also the random bytes. The problems discussed in the last section due to the use of the small exponent such as $e = 3$ are also solved by padding the block of plaintext.

- PKCS#1is one of a set 15 standards for public key cryptography developed by RSA Laboratories.
- The PKCS standard includes algorithm-independent syntax for many of the artefacts like digital signatures, digital envelopers, extended certificates etc.

Other Applications

1. EL Gamal Encryption

- Uses a very large prime number p.
- Uses generator g in $<z_p^*, *_p>$.
- EL Gamal encryption private key is an integer a, $1<a<p-1$
- The corresponding public key is the triplet (p,g,α) where α is the encryption key calculated from $\alpha = g^a \bmod p$.

Let (p, g, α) be the public key of A. To encrypt a message $m < p-1$, to be sent to A, B does the following:

- He chooses a random number r, $1 < r < p-1$ such that r is relatively prime to $p-1$.
- He computes

$$C_1 = g^r \bmod p$$

and uses α from A's public key to compute

$$C_2 = (m * \alpha^r) \bmod p$$

- He sends the ciphertext (C_1, C_2) to A.

To decrypt the ciphertext (C_1, C_2), A uses her private key, a and computes

$$(C_1^{-a}) * C_2 \bmod p.$$

To see why decryption does indeed recover the original message, observe that

$$(C_1^{-a}) * C_2 \bmod p = (g^r \bmod p)^{-a} * (m*\alpha^r \bmod p)$$ using the definitions of C_1 and C_2 above

$$= (g^{-ar} * g^{ar} * m) \bmod p$$ using the definition of the El Gamal public key

$$= m$$

- Cipher text computed using this method is twice the size of the original plaintext.

Example

- Let p =131, g=2.
- Let A private key a =97
- So A's public key is, $\alpha = g^a \bmod p = 2^{97} \bmod 131 \equiv 14$.
- Let the message to be send is m = 75.
- Let the Sender B chooses the random number r = 33.

B then computes

$$C_1 = g^r \bmod p$$
$$= 2^{33} \bmod 131$$
$$\equiv 103$$

and

$$C_2 = (m * \alpha^r) \bmod p$$
$$= 75 * 14^{33} \bmod 131$$
$$\triangleq 51$$

To decrypt the message, A computes

$$(C_1^{-a}) * C_2 \bmod p = 103^{-97} * 51 \bmod 131$$
$$= 75$$

- A Recovers the plaintext or original message using her private key.

Advantages.

- Knowing the value of C_1, g, p it is computationally infeasible to find the value of r.
- The strength of this algorithm is closely related to the difficulty of solving discrete logarithm problem for large values of p (several hundred digits long).

Precaution to be taken in this algorithm (Known Plain Text attack)

- The same random number should not be used again.
- If message m and m' are encrypted using the same random r,
- The cipher text corresponding to the first message is

$$(C_1, C_2) = (g^r \bmod p, m^*\alpha^r \bmod p).$$

- The cipher text corresponding to the second message is

$$(C_1', C_2') = (g^r \bmod p, m'^*\alpha^r \bmod p).$$

- Consider the eavesdropper having both the cipher text pairs,.
- If eavesdropper also happens to have first message m, the he/she can obtain the value of the second message m' as follows

$$m * C_2' * C_2^{-1} \bmod p = m * (m' * \alpha^r \bmod p) * (m^*\alpha^r \bmod p)^{-1} \bmod p$$
$$= m' * m * m^{-1} * \alpha^r * \alpha^{-r} \bmod p$$
$$= m'$$

El Gamal Signatures

- Let a and (p,g,α) be a private and public key of A. To sign a message m A does the following
  1. She computes the hash h(m) of the message
  2. She chooses a random number r, 1<r<p-1 such that r is relatively prime to p-1.
  3. She computes $x = g^y \bmod p$
  4. She computes $y = (h(m)-ax)r^{-1} \bmod (p-1)$
  5. The signature is the pair(x,y).
- To verify the signature, the following check is performed.

$$\alpha^x * x^y \bmod p \stackrel{?}{=} g^{h(m)} \bmod p$$

- To prove that a valid signature satisfies the above equation, we start with the expression in step 4.

$$y = (h(m) - ax)r^{-1} \bmod (p-1)$$
$$ry = (h(m) - ax) + k(p-1) \quad \text{where } k \text{ is an integer}$$

- Raising both the sides to the power of g and reducing modulo p we get

$$g^{ry} = g^{h(m)} g^{-ax} \bmod p \quad \text{(note that } g^{k(p-1)} = 1 \bmod p \text{ from Fermat's Theorem)}$$

or $$g^{ax} g^{ry} = g^{h(m)} \bmod p$$

So $$\alpha^x * x^y = g^{h(m)} \bmod p \quad \text{(since } \alpha = g^a \bmod p \text{ and } x = g^r \bmod p\text{)}$$

- El Gamal Signature on a document is not unique
- He/ She uses a different random number to sign the same document, the signature he/she produces each time will be different.
- One approach to forge a signature would be to proceed to complete step 1 through 3 in the signature generation procedure. Step 4 however requires knowledge of the private key a. Without a, it is not possible to complete the computation of digital signature.

Related signature scheme

- El Gamal Signatures comprised of two integers each of abot 1000 bits.
- So El Gamal Signatures occupies 2000 bits.

1. **Schnorr Scheme**
   - Helps in reducing the size of the signature to less than 400 bits with no loss of security.
   - Choose a large prime p (about 1000 bits) so that the following holds: q is a prime about 160 bits wide that divides (p-1)
   - Let g be the q^{th} root of 1 mod p. So $g^q = 1 \bmod p$
   - Let a be an integer $1 \le a \le q-1$.(As before a is the private key)
   - Let $\alpha = g^a \bmod p$.
   - Let r be the random number $1 \le r \le q-1$

Then the *Schnorr signature* is the pair, (x, y) where

$$x = h(m \,\|\, g^r \bmod p) \quad \text{and}$$
$$y = (r + ax) \bmod q$$

Signature verification is performed by computing the value of x using the signer's public key and checking whether it equals the value of x received as shown below.

$$x \stackrel{?}{=} h(m \,\|\, g^y \alpha^{-x} \bmod p)$$

We note that

$$\begin{aligned} g^y \alpha^{-x} \bmod p &= g^y (g^a)^{-x} && \bmod p \\ &= g^{y-ax} && \bmod p \\ &= g^{r+kq} && \bmod p, \text{ where } k \text{ is an integer.} \\ & && \text{This follows from the definition of } y. \\ &= g^r && \bmod p \\ & && \text{since } g \text{ is the } q\text{-th root of 1} \end{aligned}$$

   - The signer computed the values of x and y as a function of the message, a nonce and private key.
   - The verifier computes the x as a function corresponding public key.
   - If the computed value of x matches the received value , the verifier can be sure that the signer has correctly used the genuine private key (corresponding to her public key).
   - Thus her signature is authentic.

2. Digital signature algorithm which is used in Digital signature standard (DSS).
3. Infeasibility of discrete logarithm problem in multiplicative group of or a binary field GF(2^n), apart from Z_p*or in a prime subgroups of Z_p*.

SOAP

• Simple Object Access Protocol standardised by W3C.
• For exchanging structured information over internet.
• SOAP can be used over any transport protocol such as TCP, HTTP, SMTP
• SOAP defines a model for processing individual, one-way messages

- A soap message fits snugly inside the body of an HTTP request or response packet.
- The MIME type field in the HTTP header of a SOAP message is set equal to text/xml.

SOAP Message Format

• SOAP message is an XML document made up of :

– SOAP Envelope

- SOAP Header (optional)
- SOAP Body (Mandatory)

SOAP Header:

- The header is used to extend the message and may include security meta information such as encryption algorithm used, digital signature computed on the message etc.
- Header is optional

SOAP Body:

• Most of the information in the message is contained in its body.

- In lieu of the document style message format the body of the SOAP message may contain remote procedure calls (RPCs) in XML format.
- The example below shows a SOAP request message from a client to a provider the current stock prices.
- It is encapsulated in HTTP request packet.

```
POST /InStock HTTP/1.1
Content-Type: application/soap+xml; charset=utf-8
. . .
<?xml version="1.0"?>
<soap:Envelope
 xmlns:soap = "http://www.w3.org/2001/12/soap-envelope . . .>
  <soap:Body xmlns:X="http://www.stockQuote.com/price">
    <X:GetPrice xmlns:X = "http://www. . . " >
      <X:StockName> MyStartUp </X:StockName>
    </X:GetPrice>
  </soap:Body>
</soap:Envelope>
```

(a) SOAP message in HTTP POST request

- The SOAP response message is encapsulated in HTTP response packet

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
. . .
<?xml version="1.0"?>
<soap:Envelope
 xmlns:soap = "http://www.w3.org/2001/12/soap-envelope . .
  <soap:Body xmlns:X="http://www.stockQuote.com/price">
    <X:GetPriceResponse xmlns:X = "http://www. . . " >
      <X:Price> 3847 </X:Price>
    </X:GetPriceResponse>
  </soap:Body>
</soap:Envelope>
```

(b) SOAP message in HTTP response

- The mapping between soap message and an underlying transport protocol is referred as SOAP binding.
- Soap may run on top of HTTP or SMTP but most commonly used over HTTP.
- In case of HTTP binding, either a Get or POST request may be used.
- In the later case , the soap message may be encapsulated in the body of the HTTP post packet as well as in HTTP response packet.
- A SOAP message between two endpoints may be routed through several intermediaries.
- A node in the SOAP message path may require that a particular header element say <block1> be processed by the ultimate destination node or its immediate successor node.
- This information is conveyed by the two attributes **role** and **mustUnderstand** that are included in the <block1>.
- Processing Header might involve modifying values within the given header, removing the header or inserting a new header.

WSDL

• Web services Definition language is a language for describing web services .

- It exposes the operations and communication protocols used by web service.

A complete wsdl service description will include definitions of various elements such as types, messages, operation, port types, and bindings.

- Port Type: Specifies one or more operations within its scope.
- Operation: is an abstract definition of an action.
    - Involves one or more messages.

- For example an operation can receive a message that needs no response or it can send a notification message-one that expects no response.
- More commonly an operation receives a request and sends a response.
- Web service developer is permitted to indicate the specific communication protocols to be used in support of each operation. This is referred to as bindings.
- Permissible bindings includes SOAP, HTTP POST,HTTP GET.

• Message: abstract definition of data being exchanged as a part of operation.

- A message may have multiple parts. Each parts have an associated type.

Figure 25.4 shows a portType, which comprises a single operation involving two messages: a request and a response.

```
<message name="message1">
   <part name=" . . . " type=" . . . "/>
</message>

<message name="message2">
   <part name=" . . . " type=" . . . "/>
</message>

<portType name="portType1">
  <operation name=" . . . ">
      <input message="message1"/>
      <output message="message2"/>
  </operation>
</portType>
```

Figure 25.4 *Port type that includes one operation comprising two messages*

UDDI

- Universal description discovery and integration is a registry or catalogue that allows businesses across the globe to list themselves on the internet.
- By using SOAP messages the user queries the registry for specific services.
- In response they are provided the access to the WSDL that describes the operations, messages and protocols for the desired web service.
- The registry includes the equivalent of white, yellow and green pages of a telephone directory.
- White pages provide address and contact information of a service.
- Yellow pages provide an industrial categorization of the services and the green pages provide information about services that the business exposes.

WS security

- Token Types
  - Web security addresses basic problems in securing messages used in web services.
  - Its main functions are:
    - It defines XML elements that are used to communicate *security tokens* (defined below) in the header of a SOAP message.
    - Together with the XML Encryption Standard it defines the syntax and processing rules used to *encrypt* one or more parts of a SOAP message.
    - Together with the XML Signature Standard, it defines the syntax and processing rules used to create and represent a *digital signature* on one or more parts of a SOAP message.
  - The security related information is contained within a <Security> element in a SOAP header.
  - It specifies what operations are performed and in what order.(signing, encryption etc).
  - The <Security> element includes the security tokens, keys signatures, timestamp and security meta –information.
  - A security claim is a statement made about a subject's privilages etc.
  - A claim may be made by subject himself or by another party on behalf of the subject.
  - One or more claims is/are represented by a security token.
  - Common examples of security tokens are a
    - username+password
    - an X.509 certificate or
    - A Kerberos ticket.
  - username+password is the mostly used security tokens.
  - The default is to send the password in the clear but this is not a very secure option.
  - Alternatively the password (pw), a nonce (n) and the timestamp(t) may be concatenated and then hahed using a cryptographic hash function such as SHA-1

SHA-1 *(n, t, pw)*

The user name, hash, nonce, and timestamp are sent in a <UsernameToken>

```
< UsernameToken >
        < Username > John < /Username >
        < Password Type = "PasswordDigest" >
                4u%h&+q:L
        < /Password >
        < Nonce > . . . < /Nonce >
        < Created > . . . </Created >
< /UsernameToken>
```

- Security tokens containing usernames are pure text. Some Security tokens may contain binary data such as signatures or keys.
- The **BinarySecurityToken** element is used in that case.
- Examples of binary security tokens include X.509 certificates and Kerberos tickets.
- The binary content in such a token is rendered readable by encoding it using BASE 64 encoding or using hexadecimal notation.

The following is the representation of an X.509 certificate within the <Security> element of a SOAP header.

```
< Security >
...
        < BinarySecurityToken
                ValueType = " ... X509v3"
                EncodingType = " ... Base64Binary" >
                        Lp9rba4Pc7G ...
            < / BinarySecurityToken >
...
< /Security >
```

The first version of the WS-Security standard was created by the Organization for the Advancement of Structured Information Standards (OASIS) in 2004 with input from IBM, Microsoft and Verisign. Version 1.1 (the latest version) was released in February 2006.

WS Security

2. XML Encryption

• It defines XML elements for representing encrypted data and keys used for encryption.

• Allows Encryption at different levels of granularity

- An entire document
- A complete XML element within a document
- Content of an XML element.

▪ The standard permits any combination of elements within the body and/or the header of the SOAP message to be encrypted.

• <EncryptedData> element contains: used to represent encrypted data in SOAP messages.

```
24  < S11:Body wsu:Id = "#Id1" >
25      <xenc:EncryptedData
26      Type = "http://www.w3.org/2001/04/xmlenc#Element"
27            Id="Id2">
28        <xenc:EncryptionMethod
29              Algorithm = "http://www.w3.org/xmlenc#aes256-cbc"/>
30        <xenc:CipherData>
31            <xenc:CipherValue>
32                    tdaqUsjXipJ09jlkjh5oinlkdsn...
33            </xenc:CipherValue>
34        </xenc:CipherData>
35    </xenc:EncryptedData>

                  This is text sent in the clear. The previous segment
                  of text and the next segment of text are both
                  encrypted with the key contained in the header.

36    <xenc:EncryptedData
37            Type = "http://www.w3.org/2001/04/xmlenc#Element"
38            Id = "Id3">
39        <xenc:EncryptionMethod
40             Algorithm = "http://www.w3.org/xmlenc#aes256-cbc"/>
41        <xenc:CipherData>
42            <xenc:CipherValue>
43                    tdaqUsjXipJ09jlkjh5oinlkdsn...
44            </xenc:CipherValue>
45        </xenc:CipherData>
46    </xenc:EncryptedData>

47  </ S11:Body >
```

- The actual ciphertext of each encrypted elements is enclosed in a **<Cipher Value >** subelements (line 32,34).
- The encryption algorithm used is mentioned inside <EncryptionData>, AES in cbc mode.(line 29,40).

- Information on the Key used for encryption may be included in <EncryptionData>.
- Alternatively the key used may be included in the header.

```
4       <S11:Header>
5          <wsse:Security>
6            <xenc:EncryptedKey>
7                  <xenc:EncryptionMethod Algorithm =
8                          "http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
9                  <ds:KeyInfo>
10                   <ds:KeyName> CN= Rajiv Singhvi, C=IN </ds:KeyName>
11                 </ds:KeyInfo>
12                 <xenc:CipherData>
13                     <xenc:CipherValue>
14                         aKLj89qwhMZsaRiDutagbx78bigxb...
15                     </xenc:CipherValue>
16                 </xenc:CipherData>
17                 <xenc:ReferenceList>
18                     <xenc:DataReference URI="#Id2"/>
19                     <xenc:DataReference URI="#Id3"/>
20                 </xenc:ReferenceList>
21           </xenc:EncryptedKey>
22         < / wsse:Security>
23      </ S11:Header>
```

- The encryption key may be pre-shared secret between the communicating parties.
- It may be short term key or session key chosen by the sender.
- In this case of session key it, should be transmitted in encrypted form. For this purpose it is enclosed in an <EncryptedKey> element and placed in the SOAP header(line 6 to 21).
- In case if many segments of SOAP messages are encrypted using short term key. In this case <ReferenceList> containing a manifest of the encrypted segments is included in <EncryptedKey> (line 18 to 19).
- Line 8 indicates the algorithm used to encrypt the key is RSA.

- The <KeyInfo> elements (line 9-11) identifies the key used to encrypt the short term Key.
- Line 14 we can see the cipher text of the encrypted short term Keys.
- In line 10, we can see that encrypted short term key can be decrypted by the private key corresponding to the certificate of belonging to Rajiv singhvi.

3. XML Signatures

- Xml signature standard was developed jointly by W^3C and IETF in 2002.
- It specifies the syntax for signature and signature keys while offering a rich set of options for signing XML documents.
  - For example parts of a document can be signed by an entity.
  - One or more intermediaries may attach their signatures to the document.
  - Two entities may sign overlapping or disjoint parts of the document.
- XM signature also involve computing the hash of a document followed by encryption using signer's private key.

XML allows a lot of leeway in syntax.

- Extraneous white spaces are liberally permitted
- Two documents may be syntactically identical despite superficial differences in appearance resulting in different binaries.(UNICODE/Base 64 representation.)
- Cryptographic hash applied separately to two different documents are always distinct.
- Thus syntactically identical documents signed by same individual may have different digital signature.

Canonicalization

- The parts of the document that need to be signed are first transformed into a canonical form before computing their hashes.
- This guarantees that the syntactically identical documents produce the same serialized representations.

Signatures are included in the header of the SOAP message containing document. More specifically they are contained in a<Signature> element in the header. The major elements and sub elements contained in **<Signatue>** is as shown in fig 25.6 below

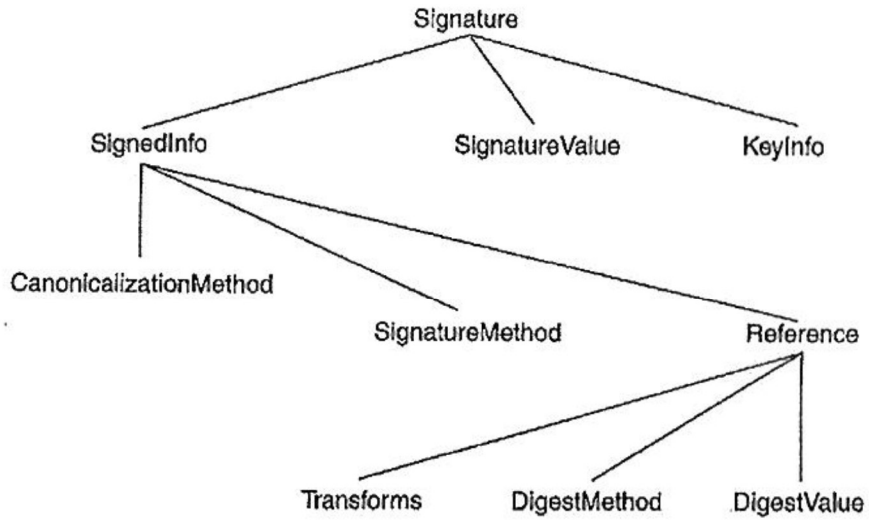


Figure 25.6 *Tree for signature element*

<SignedInfo> Within this element is included information about the *canonicalization algorithm* and *signature algorithm* employed. An example of the signature algorithm is RSA-SHA1, i.e., the use of SHA-1 to perform the hash followed by an RSA private key operation on the hash value. A "signature" in the form of a Message Authentication Code (MAC) is also supported.

A single signature is computed over possibly multiple elements in the document. Some of the elements may be in the header and others may be in the body of the SOAP message. *References* to all these are included within <SignedInfo>. Each reference is followed by the digest algorithm used in computing its digest and the digest value.

<SignatureValue > This element contains the *digital signature*. Note that the signature is computed on the entire <Signature> element. This is done by canonicalizing the <Signature> element using the canonicalization algorithm specified in the <SignedInfo> sub-element. The signature is then generated using the signature algorithm specified in the <SignedInfo> sub-element.

<KeyInfo > This element typically includes reference to *key* material that is needed for *verifying the signature* at the receiver end. For example, it may reference an X.509 certificate. The public key in the certificate would then be used to verify the signature.

The following points are worthy of note regarding the digital signature of figure 25.7

- The digital signature covers Three elements.
  - Two of these are timestamps in the SOAP header
    - Line 6: document creation date/Time.
    - Line 9: document expiration date/Time.
  - The third element is in the body of the SOAP Envelop: Line 52 &53.
- The Three elements are reffered to within the <SignedInfo> subelement by their Ids: ID1,ID2,ID3.(Line 20,27,34)
- The three elements are cananicalized using the canonicalization algorithm specified on lines 22,29,36.
- The digest of the three elements appear on lines 25, 32, 39 using digest algorithm specified on lines 24,31 and 38.
- The entire <signature> element is then canonicalized using the canonicalization algorithm specified on line 18.
- Finally the canonicalized <signature> element is signed.
- Line 19- signature algorithm used is RSA- SHA1.
- To perform signature verification the receiver needs to know the public key corresponding to the private key used for signing. The <KeyInfo> element in line 43 contains this information. It contains the reference to an element with ID =DigCert which is a BinarySecurityToken on line 13.
- The ValueType of this token indicates that it is an X.509 certificate.
- The certificate is encoded in Base64 and is attached.

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <S11:Envelope xmlns:S11="..." xmlns:wsse="..." xmlns:wsu="..."
                                  xmlns:xenc="..." xmlns:ds="...">
3   <S11:Header>
4       <wsu:Timestamp>
5         <wsu:Created  wsu:Id="Id1" >
6               20090418T15:35:27Z
7         </wsu:Created>
8         <wsu:Expires wsu:Id="Id2" >
9            20090418T15:37:00Z
10        </wsu:Expires>
11      </wsu:Timestamp>
```

```
12      <wsse:Security>
13        <wsse:BinarySecurityToken
                ValueType = "...#X509v3"
                wsu:Id = "DigCert"
                EncodingType = "...#Base64Binary">
14            ysG7FeWSQ3lpK9JhNMN...
15        </wsse:BinarySecurityToken>
16        <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
17          <SignedInfo>
18            <CanonicalizationMethod
                Algorithm="http://www.w3.org/2001/10/xmlexcc14n#" />
19            <SignatureMethod
                Algorithm="http://www.w3.org/2000/xmldsig#rsasha1" />

20            <Reference URI="#Id1">
21              <Transforms>
22                <TransformAlgorithm =
                    "http://www.w3.org/2001/xmlexcc14n#" />
23              </Transforms>
24              <DigestMethodAlgorithm=
                    "http://www.w3.org/2000/09/xmldsig#sha1"/>
25              <DigestValue> Yha1 . . . pKl </DigestValue>
26            </Reference>

27            <Reference URI="#Id2">
28              <Transforms>
29                <Transform Algorithm=
                        "http://www.w3.org/2001/10/xmlexcc14n#" />
30              </Transforms>
31              <DigestMethodAlgorithm=
                    "http://www.w3.org/2000/09/xmldsig#sha1"/>
32              <DigestValue> ts7Q . . . 0KB </DigestValue>
33            </Reference>
```

```
34              <Reference URI="#Id3">
35                 <Transforms>
36                    <Transform Algorithm=
                              "http://www.w3.org/2001/10/xmlexcc14n#" />
37                 </Transforms>
38                 <DigestMethod Algorithm=
                      "http://www.w3.org/2000/09/xmldsig#sha1"/>
39                 <DigestValue>  m5hl . . .  xTv  </DigestValue>
40              </Reference>
41           </SignedInfo>
42           <SignatureValue>  hMB...uaW  </SignatureValue>
43           <KeyInfo>
44                <wsse:SecurityTokenReference>
45                   <wsse:Reference URI ="#DigCert" />
46                </wsse:SecurityTokenReference>
47            </KeyInfo>

48         </Signature>
49      </ wsse:Security>
50   </ S11:Header>

51   < S11:Body wsu:Id = "#Id3" >
52         Include the body too in the computation
53         of the digital signature.
54   </ S11:Body >
55 </ S11:Envelope >
```

Figure 25.7 *Illustrating XML signatures*

SAML

- **Motivation**
  - Consider the long term key client C of a service provider SP1.
  - Each time C request service form SP1 he needs to be authenticated.
    - Can be done using login name password.
    - Sp1 can store cookie in C's browser (encrypted form), which would be transparently dispatched to SP1 when C visits SP1's website.
    - Relevant information about C can be also stored in cookies or at the server.
  - Now if C wishes the service from another provider SP2, the cookies at C browser which was created by SP1 can be read by SP2 to trust C, if SP1 and SP2 share a trust relationship.
  - The cookie could include the information such as "SP1 trusts C".
  - If Sp2 Knows SP1 that trust C, the Sp2 might also be willing to trust C.

Disadvantage

- Browser do not allow cookies created by one server to be dispatched to a server in a different domain.

- **Assertion Types**
  - SAML is the XML based standard developed in May 2002 by OASIS.
  -
  - SAML provides XML schema for expressing assertions about a principal. For example,

example, SP1 might make the following assertion:

SP1 authenticated C
using password-based authentication
on 1st February 2010
at 09:25:15 hours.

  - In above example Sp1 is the asserting party, performs the role of an Identity Provider (I).
  - SP2 is a consumer of assertions and is referred to as the relying party.

➢ **SAML defines three types of assertions.**

1. Authentication Statement: is an assertion by Identity provider I, that it authenticate the principal C by using authentication method at a particular point of time.

2. Attributes statement: is an assertion by Identity provider I, that the value of the attribute A for principal C is a.

3. An Authorization statement: is an assertion by Identity provider I that a principal C is permitted to perform an action or operation O on resource R.

- In the example, SAML assertion has authentication statement containing the identities of issuer and Principal.
- A URL is used to identify the issuer and an email address is used to identify the principal (line 2 and 5).
- The statement indicates the date/time at which the principal was authenticated. (line 12).
- It asserts that the principal was authenticated using a password transmitted across a protected channel (using SSL).

- It also includes an explicit condition that the authentication is valid for the next 26 hours.

```
1   <saml:Assertion xmlns:saml = ...  Version = "2.0"
                                      IssueInstant = "2010-02-01T08:25:15Z">

2       <saml:Issuer Format= ... :entity>
                    http://www.admin.iitb.ac.in
3       </saml:Issuer>

4       <saml:Subject>
5           <saml:NameID Format = " ... :emailAddress">
                rajeshX@cse.iitb.ac.in
6           </saml:NameID>
7       </saml:Subject>
```

-

```
8       <saml:Conditions
9           NotBefore = "2010-02-01T08:26:00Z"
10          NotOnOrAfter = "2010-02-02T10:30:00Z"
11      </saml:Conditions>

12      <saml:AuthnStatement AuthnInstant = "2010-02-01T08:25:15Z"
                                            SessionIndex = "1234">
13          <saml:AuthnContext>
14              <saml:AuthnContextClassRef>
                    ...:PasswordProtectedTransport
15              </saml:AuthnContextClassRef>
16          </saml:AuthnContext>
17      </saml:AuthnStatement>
18  </saml:Assertion>
```

Figure 25.8 *SAML assertion – Authentication statement*

- **Creating/Communicating assertions**
  - Useful application of SAML is in single sign on.
  - Let us consider a usage scenario of a single sign on over the web.

A user, Sandeep, visits the website of his familiar travel agent, SmartTravels in order to book a ticket to say, Rio. Sandeep logs in and authenticates himself at the SmartTravels site. He indicates travel preferences including date/time of departure, budgetary constraints, etc. He is presented with a choice of airlines satisfying his requirements. After clicking on his preferred airline, Jet Air, he is seamlessly directed to the website of that airline where he makes a reservation.

For the sake of completeness, we enumerate all the steps involved in Sandeep's transaction.

1. Sandeep logs in to the SmartTravels website and is *authenticated*. He indicates the destination city, date and time of travel, and price of ticket he is willing to pay.
2. SmartTravels determines that Sandeep is a gold customer and presents a list of airlines that satisfy Sandeep's requirements.
3. Sandeep clicks on the airline of interest, say JetAir.
4. SmartTravels creates *SAML assertions* indicating that
   a. Sandeep has been authenticated using a login name–password mechanism (authentication assertion)
   b. Sandeep is a gold customer (attribute assertion)
5. SmartTravels creates an HTML form with two hidden inputs. The first, named *SAMLResponse*, contains the signed SAML assertion. The second hidden input, called RelayState, contains the *URL of the resource* required by Sandeep. The relevant portion of the form is

```
<html>
<body onload = "document.forms.SAMLform.submit( );">
<form name = SAMLform method = post
        action = "www.JetAir.com/Reservations/SAMLConsumer">
<input type = hidden name = "SAMLResponse"
                        value = " . . . signed assertion . . . "/>
<input type = hidden name = "RelayState"
                        value = "encodedTargetURL"/>
</form>
</body>
</html>
```

   This form is sent to Sandeep's browser.
6. When Sandeep's browser receives the form, it is immediately re-directed to www.JetAir.com/Reservations/SAMLConsumer.
7. The assertions are consumed by the JetAir web server. It is now aware that Sandeep is a gold customer of its business partner – SmartTravels. It returns Sandeep a page containing information on its travel schedules and special fares.

- **Other Standards**

➢ **WS trust**

- Two end points of a web service may have never interacted with each other.
- To build trust between themselves they could use an intermediary known to both parties who would create a SAML token on behalf of the party that needs to be authenticated.
- This just is the one way of establishing trust.
- We need a framework that is more powerful and general. Here is the concise with list.

- Our framework should encompass different kinds of trust – direct trust and indirect trust brokered by one or more intermediaries.
- Besides SAML tokens, we need to support other token types – simple password-based tokens, digital certificates, Kerberos tickets, etc.
- Our framework should be able to define messages for requesting security tokens from a Security Token Service. We also need to define messages that communicate the security tokens. Finally, we need to validate tokens received from a client.

The WS-Trust standard is exactly the response to our wish list.

Before the importer and exporter can transact securely, they need to establish a relationship of mutual trust. Basically, I needs to securely communicate its credentials to E. The credentials are security tokens acceptable to E (i.e., verifiable by E). WS-Trust could be used in the following manner:

- I authenticates himself to IB.
- Since IB and CB trust each other, IB requests CB to issue a security token to be used by I.
- CB creates a security token and includes information such as the maximum credit amount extended to I. CB acts as a Security Token Service Provider. CB communicates this token to IB who forwards it to I.
- I dispatches the token to E.
- E requests EB to validate the token. EB may be able to validate the token on its own. If not, it sends it to CB for validation.
- The success or failure of the validation process is communicated by CB to E through EB.

Example 25.1

Consider an importer, I, who wishes to import goods from an exporter, E, in another country. I and E are not known to each other and so need to establish a trust relationship with each other. I and E have trust relationships with their "local" banks, IE and IB, respectively. IB and EB do not have a direct trust relationship with each other. They each do, however, have a trust relationship with an intermediary – a well-known international bank called the Correspondent Bank, CB (see Fig. 25.9). The trust relationships are summarized below.

$$I \leftrightarrow IB \quad IB \leftrightarrow CB$$
$$E \leftrightarrow EB \quad EB \leftrightarrow CB$$

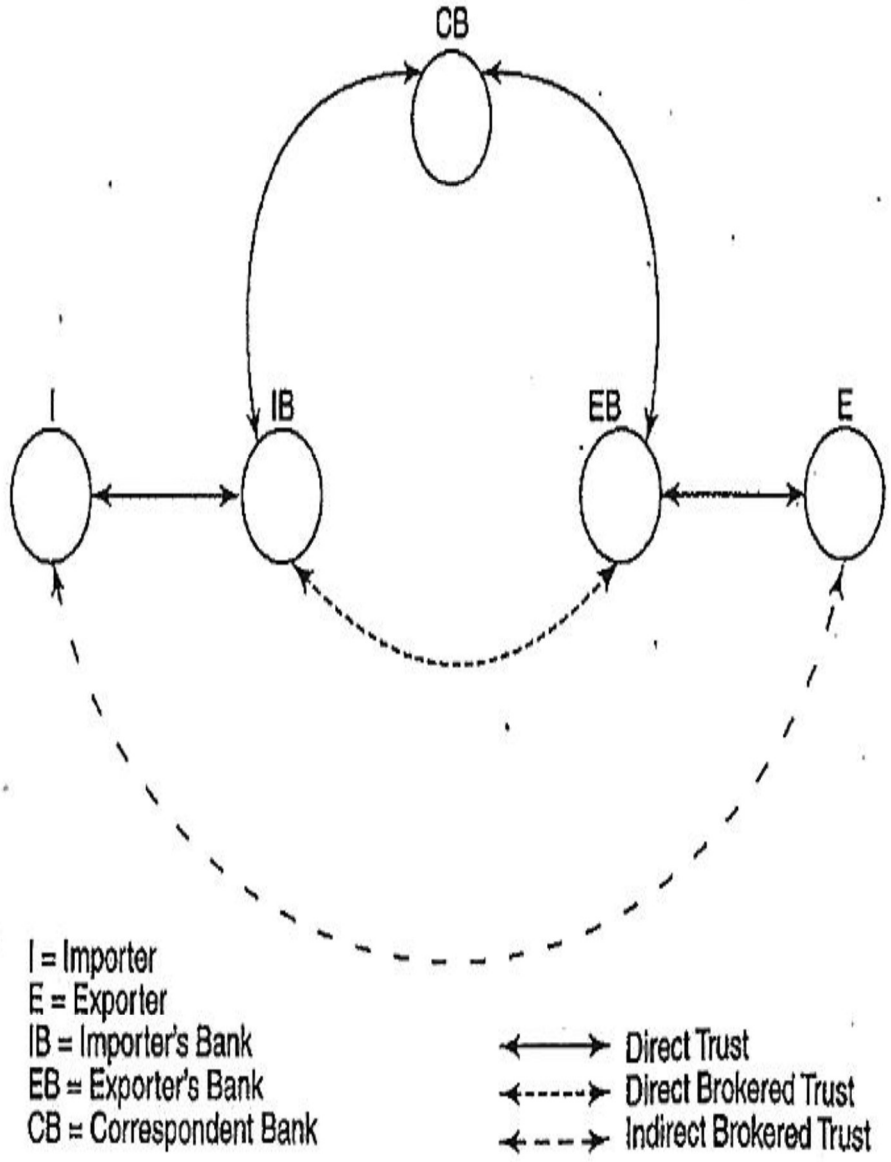


Figure 25.9 *Trust relationship between entities involved in international trade*

➢ **WS security policy**

- Enables a web service to specify the security tokens it will accept for authentication and access control.
- For example it might state that it accepts either X.509 certificates or Kerberos tickets.
- It conveys information about whether it requires all or part of the client messages to be encrypted.
- If the encryption is required it will indicate the encryption algorithm supported and the key size.
- Also the security policy may require that all or part of the messages be integrity protected. In this case It will also specify the integrity check algorithm.
- WS security policy assertions are communicated as a part of the web service's WSDL.
- Alternatively it may be included in entries in the UDDI registry related to the web service provider.

Returning to the example of Fig. 25.9, the policies regarding authentication laid out by the relevant entities may be as follows:

- An importer must authenticate himself to his local bank via a login name and password.
- A local bank must authenticate itself to the global bank using a challenge-response protocol in conjunction with a digital certificate.
- An importer must authenticate himself to the given exporter through a SAML token signed by a global bank.

BMS
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
Avalahalli, Doddaballapur Main Road, Bengaluru – 560064
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Course Name	Cryptography Network Security and Cyber Law
Course Code	17CS61

Module-5

- IT act aim and objectives
- Scope of the act
- Major Concepts
- Important provisions
- Attribution, acknowledgement, and dispatch of electronic records
- Secure electronic records and secure digital signatures
- Regulation of certifying authorities: Appointment of Controller and Other officers
- Digital Signature certificates
- Duties of Subscribers
- Penalties and adjudication
- The cyber regulations appellate tribunal
- Offences, Network service providers not to be liable in certain cases
- Miscellaneous Provisions.

INFORMATION TECHNOLOGY ACT 2000

Is to provide legal recognition for transactions carried out by means of electronic data interchange and other means of electronic communication, commonly referred to as "electronic commerce", which involve the use of alternatives to paper-based methods of communication

Companies rely on IT for fast communications, data processing and market intelligence. IT plays an integral role in every industry, helping companies improve business processes, achieve cost efficiencies, drive revenue growth and maintain a competitive advantage in the marketplace.

Why do we need information technology?

Using computers and software, businesses use **information technology** to ensure that their departments run smoothly. ... They purchase software packages and hardware that helps them get their job done. Larger businesses have their own **information technology** department designed to upkeep the software and hardware.

WHY IT Act is required?
The **Act** provides a legal framework for electronic governance by giving recognition to electronic records and digital signatures. It also defines cybercrimes and prescribes penalties for them.

27.1 IT ACT: AIM AND OBJECTIVES

- The information technology act, 2000 is an important law related to Indian cyber law.

- The act strives to achieve the following objectives:

1. To give legal recognition to ***transactions*** done by electronic way or by use of the internet.

2. To grant legal recognition to ***digital signature*** for accepting any agreement via computer.

3. To provide facility of ***filling documents online.***

4. To authorize any undertaking to ***store their data in electronic storage***.
5. To prevent ***cybercrime*** by imposing high penalty for such crimes and protect privacy of internet users.
6. To keep legal recognition for keeping ***books of account by bankers*** and undertaking in electronic form.

27.2 SCOPE OF THE ACT

- The act attempts to address the following issues:

a. *Legal recognition of electronic documents.*

b. *Legal recognition of digital signatures.*
c. *offences and Contraventions*
d. *Justice dispensation systems for cybercrimes.*

27.3 MAJOR CONCEPTS

1. ***"Access":*** implies ***gaining entry into, instructing or communicating*** with the logical, arithmetical, or memory function resources of a computer, computer system or computer network;

2. ***"Addressee": is*** a person who is intended by the originator to ***receive the electronic record*** but does not include any intermediary;

3. ***"Adjudicating officer":*** means an adjudicating officer appointed.

4. ***"Affixing digita0lsignature" :*** means adoption of any methodology or procedure by a person for the purpose of authenticating an electronic record by means of digital signature.
5. ***"Appropriate Government":*** means any matter, —

   - Enumerated in the list II of the seventh schedule to the constitution
   - Relating to any State law enacted under List III of the Seventh Schedule to the Constitution, the state government and in any other case the central government.

6. ***"Asymmetric crypto system"*** is a system of a ***secure key pair*** consisting of a private key for creating a digital signature and a public key to verify the digital signature;

7. ***"Certifying Authority"*** means a person who has been ***granted a license to issue a Digital Signature Certificate*** under section 24;

8. ***"Certification practice statement"*** means a statement issued by a Certifying Authority to specify the practices that the Certifying Authority employs in issuing Digital Signature Certificates;

9. ***"computer"*** means any electronic magnetic, optical or other high- speed data processing device or system which performs logical, arithmetic, and memory functions by manipulations of electronic, magnetic or optical impulses, and includes all input, output, processing, storage, computer software, or communication facilities which are connected or related to the computer in a computer system or computer network;

10. ***"Computer network"*** means the interconnection of one or more computers through—
    - The use of satellite, microwave, terrestrial line or other communication media; and

    - Terminals or a complex consisting of two or more interconnected computers whether or not the interconnection is continuously maintained.

11. ***"computer system"*** means a device or collection of devices, including input and output support devices and excluding calculators which are not programmable and capable of being used in conjunction with external files, which contain computer programmes, electronic instructions, input data and output data, that performs logic, arithmetic, data storage and retrieval, communication control and other functions.
12. ***"data"*** means a representation of information, knowledge, facts, concepts or instructions which are being prepared or have been prepared in a formalized manner, and is intended to be processed, is being processed or has been processed in a computer system or computer network, and may be in any form (including computer printouts magnetic or optical storage media, punched cards, punched tapes) or stored internally in the memory of the computer.\
13. "***digital signature"*** means ***authentication of any electronic record*** by a subscriber by

means of an electronic method or procedure in accordance with the provisions of section 3;

14. ***"electronic form"*** with reference to information means any information generated, sent, received or stored in media, magnetic, optical, computer memory, micro film, computer generated micro fiche or similar device;

15. ***"Electronic Gazette"*** means the Official Gazette published in the electronic form.

16. ***"Electronic record"*** means data, record or data generated, image or sound stored, received or sent in an electronic form or micro film or computer generated micro fiche.

17. ***"Information"*** includes data, text, images, sound, voice, codes, computer programmes, software and databases or micro film or computer generated micro fiche.

18. ***"Intermediary"*** with respect to any particular electronic message means any person who on behalf of another person receives, stores or transmits that message or provides any service with respect to that message.

19. ***"key pair",*** in an asymmetric crypto system, means a private key and its mathematically related public key, which are so related that the public key can verify a digital signature created by the private key.

20. "**Originator**" refers to a person who sends generates, stores, or transmits any electronic message or causes any electronic message to be sent, generated, stored, or transmitted to any other person, but does not include any intermediary.

21. "**Private Key"** refers to the key of a key pair used to create a digital signature.

22. **"Public key"** refers to the key of a key pair used to verify a digital signature which is listed in the digital signature certificate.

Secure systems refer to computer hardware, software and procedure that

1. Is reasonably secure from unauthorized access and misuse.
2. Provides a reasonable level of reliability and correct operations.
3. Is reasonably suited to performing the intended functions and
4. Adheres to generally accepted security procedures.

27.4 IMPORTANT PROVISIONS

27.4.1 Digital Signature: Authentication of Electronic Records.

This is an way to ensures that an electronic record or document is authentic. The Act contains the following provisions in relation to digitalsignature.

1. Any subscriber may authenticate an electronic record by ***affixing his digital signature.***
2. The authentication of the electronic record shall be effected by the use of ***asymmetric crypto system and hash function*** which envelop and transform the initial electronic record into another electronic record.

Explanation*:* "hash function" means an algorithm mapping or translation of one sequence of bits into another, generally smaller, set known as ***"hash result"*** such that an electronic record yields the same hash result every time the algorithm is executed with the same electronic record as its input making it computationally infeasible—

a. to derive or reconstruct the original electronic record from the hash result produced by the algorithm.

b. that two electronic records can produce the same hash result using the algorithm.

3. Any person by the use of a ***public key of the subscriber*** can verify the electronic record.

4. The private key and the ***public key are unique to the subscriber*** and constitute a functioning key pair.

27.4.2 Electronic Governance: Legal recognition of electronic records.

E Governance is the public sector's use of information and communication technologies (ICT) with the aim of improving information and service delivery, encouraging citizen participation in the decision process and making government more accountable, transparent and effective.

The three main target groups that can be distinguished in government concepts are

- Government.
- Citizens
- Business/interest group.

Generally, four basic models of E-Governance are available.

- Government to- citizen (customer)
- Government to employees
- Government to Government
- Government to Business (intergovernmental).

Where any law provides that information or any other matter shall be in writing or in the typewritten or printed form, then, notwithstanding anything contained in such law, such requirement shall be deemed to have been satisfied if such information or matter is—

(a) Rendered or made available in an electronic form; and

(b) Accessible so as to be usable for a subsequent reference.

27.4.3 Legal recognition of digital signatures

- Digital signature affixed to a digital document establishes the origin of that digital document.
- Digital signatures are considered much more secure and fool-proof compared to physical signature.
- The IT Acts provides the legal sanctity for using digital signatures.

➢ Where any law provides that information or any other matter shall be authenticated by affixing the signature or any document shall be signed or bear the signature of any person (then, notwithstanding anything contained in such law, such requirement shall be deemed to have been satisfied, if such information or matter is authenticated by means of digital signature affixed in such manner as may be prescribed by the Central Government.

27.4.4 Use of electronic records and digital signatures in Government and its agencies.

Government have passed laws and regulations encouraging the usage of digitally signed electronic documents rather than the paper documents. For example, Income tax returns and corporate returns etc. are to be digitally signed and uploaded electronically.

1. Where any law provides for—

✓ the filing of any form. application or any other document with any office, authority, body or agency owned or controlled by the appropriate Government in a particular manner;

✓ the issue or grant of any licence, permit, sanction or approval by whatever name called in a particular manner;

✓ the receipt or payment of money in a particular manner, then, notwithstanding anything contained in any other law for the time being in force, such requirement shall be deemed to have been satisfied if such filing, issue, grant, receipt or payment, as the case may be, is effected by means of such electronic form as may be prescribed by the appropriate Government.

2. The appropriate Government may, for the purposes of sub-section (1), by rules, prescribe—

✓ the manner and format in which such electronic records shall be filed, created or issued;

✓ the manner or method of payment of any fee or charges for filing, creation or issue any electronic record under clause *(a).*

27.4. 5 Retention of electronic records.

1. Where any law provides that documents, records or information shall be retained for any specific period, then, that requirement shall be deemed to have been satisfied if such documents, records or information are retained in the electronic form, if—
   (a) the information contained therein remains accessible so as to be usable for a subsequent reference;
   (b) the electronic record is retained in the format in which it was originally generated, sent or received or in a format which can be demonstrated to represent accurately the information originally generated, sent or received;
   (c) the details which will facilitate the identification of the origin, destination, date and time of despatch or receipt of such electronic record are available in the electronic record.
2. Nothing in this section shall apply to any law that expressly provides for the retention of documents, records or information in the form of electronic record.

27.4.6 Publication of rule, regulation, etc., in Electronic Gazette.

Where any law provides that any rule, regulation, order, bye-law, notification or any other matter shall be published in the Official

- Gazette, then, such requirement shall be deemed to have been satisfied if such rule, regulation, order, bye-law, notification or any other matter is published in the Official Gazette or Electronic Gazette:

- Provided that where any rule, regulation, order, bye-law, notification or any other matter is published in the Official Gazette or Electronic Gazette, the date of publication shall be deemed to be the date of the Gazette which was first published in any form.

27.4.7 Power to make rules by Central Government in respect of digital signature.

- The Central Government may, for the purposes of this Act, by rules, prescribe—

1. The type of digital signature;

2. The manner and format in which the digital signature shall be affixed;

3. The manner or procedure which facilitates identification of the person affixing the digital signature;

4. Control processes and procedures to ensure adequate integrity, security and confidentiality of electronic records or payments; and
5. Any other matter which is necessary to give legal effect to digital signatures.

27.5 ATTRIBUTION, ACKNOWLEDGEMENT, AND DISPATCH OF ELECTRONIC RECORDS

27.5.1 Attribution of electronic records.

An electronic record shall be attributed to the originator—

1. If it was sent by the ***originator*** himself;

2. By a person who had the ***authority to act on behalf of the originator*** in respect of that electronic record;

3. By an information system programmed by or on behalf of the originator to operate automatically.

27.5.2 Acknowledgment of receipt.

1. Where the originator has not agreed with the addressee that the acknowledgment of receipt of electronic record be given in a particular form or by a particular method, an acknowledgment may be given by—

    (a) Any communication by the addressee, automated or otherwise; or

    (b) Any conduct of the addressee, sufficient to indicate to the originator that the electronic record has been received.

2. Where the originator has stipulated that the electronic record shall be binding only on receipt of an acknowledgment of such electronic record by him, then unless

acknowledgment has been so received, the electronic record shall be deemed to have been never sent by the originator.

3. Where the originator has not stipulated that the electronic record shall be binding only on receipt of such acknowledgment, and the acknowledgment has not been received by the originator within the time specified or agreed or, if no time has been specified or agreed to within a reasonable time, then the originator may give notice to the addressee stating that no acknowledgment has been received by him and specifying a reasonable time by which the acknowledgment must be received by him and if no acknowledgment is received within the aforesaid time limit he may after giving notice to the addressee, treat the electronic record as though it has never been sent.

27.5.3 Time and place of despatch and receipt of electronic record.

1. UNLESS as otherwise agreed to between the originator and the addressee, the dispatch of an electronic record occurs when it enters a computer resource outside the control of the originator.

2. UNLESS as otherwise agreed between the originator and the addressee, the time of receipt of an electronic record shall be determined as follows, namely: —

   a) if the addressee has designated a computer resource for the purpose of receiving electronic records—

      - receipt occurs at the time when the electronic, record enters the designated computer resource;

      - if the electronic record is sent to a computer resource of the addressee that is not the designated computer resource, receipt occurs at the time when the electronic record is retrieved by the addressee;

   b) if the addresses have not designated a computer resource along with specified timings, if any receipt occurs when the electronic record enters the computer resource of the addressee.

3. Unless as otherwise agreed to between the originator and the addressee, an electronic record is deemed to be dispatched at the place where the originator has his place of business, and is deemed to be received at the place where the addressee has his place of business.
4. The provisions of the subsection (2) shall apply notwithstanding that the place where the computer resource is located may be different from the place where the computer resource is located may be different from the place where the electronic record is deemed to have been received under subsection (3).
5. For the purpose of this section
   a) if the originator or the addressee has more than one place of business, the principal place of business, shall be the place of business;
   b) if the originator or the addressee does not have a place of business, his usual place of residence shall be deemed to be the place of business;
   c) "usual place of residence", in relation to a body corporate, means the place where it is registered.

27.6 SECURE ELECTRONIC RECORDS AND SECURE DIGITAL SIGNATURES

27.6.1 Secure electronic record.

Where any security procedure has been applied to an electronic record at a specific point of time. then such record shall be deemed to be a secure electronic record from such point of time to the time of verification.

27.6.2 Secure digital signature.

27.6.2.1 If, by application of a security procedure agreed to by the parties concerned, it can be verified that a digital signature, at the time it was affixed, was—

a. unique to the subscriber affixing it;

b. capable of identifying such subscriber;

c. created in a manner or using a means under the exclusive control of the subscriber and is linked to the electronic record to which it relates in such a manner that if the electronic record was altered the digital signature would be invalidated, then such digital signature shall be deemed to be a secure digital signature.

27.6.3 Security procedure.

- The Central Government for the purposes of this Act prescribe the security procedure having regard to commercial circumstances prevailing at the time when the procedure was used, including—
  a. the nature of the transaction;

  b. the level of sophistication of the parties with reference to their technological capacity;

  c. the volume of similar transactions engaged in by other parties;

  d. the availability of alternatives offered to but rejected by any party;

  e. the cost of alternative procedures; and

  f. the procedures in general use for similar types of transactions or communications.

27.7 REGULATION OF CERTIFYING AUTHORITIES

1. The Central Government may, by notification in the Official Gazette, appoint a Controller of Certifying Authorities for the purposes of this Act and may also by the same or subsequent notification appoint such number of Deputy Controllers and Assistant Controllers as it deems fit.

2. The Controller shall discharge his functions under this Act subject to the general control and directions of the Central Government.

3. The Deputy Controllers and Assistant Controllers shall perform the functions assigned to them by the Controller under the general superintendence and control of the Controller.

4. The qualifications, experience and terms and conditions of service of Controller, Deputy Controllers and Assistant Controllers shall be such as may be prescribed by the Central Government.

5. The Head Office and Branch Office of the office of the Controller shall be at such places as the Central Government may specify, and these may be established at such places as the Central Government may think fit.

There shall be a seal of the Office of the Controller.

27.7.1 Functions of Controller.

- The Controller may perform all or any of the following functions, namely:—
  1. exercising supervision over the activities of the Certifying Authorities;
  2. certifying public keys of the Certifying Authorities;
  3. laying down the standards to be maintained by the Certifying Authorities;
  4. specifying the qualifications and experience which employees of the Certifying Authorities should possess;
  5. specifying the conditions subject to which the Certifying Authorities shall conduct their business;
  6. specifying the contents of written, printed or visual materials and advertisements that may be distributed or used in respect of a Digital Signature Certificate and the public key
  7. specifying the form and content of a Digital Signature Certificate and the key,
  8. specifying the form and manner in which accounts shall be maintained by the Certifying Authorities;
  9. specifying the terms and conditions subject to which auditors may be appointed and the remuneration to be paid to them;
  10. facilitating the establishment of any electronic system by a Certifying Authority either solely or jointly with other Certifying Authorities and regulation of such systems;
  11. specifying the manner in which the Certifying Authorities shall conduct their dealings with the subscribers;
  12. resolving any conflict of interests between the Certifying Authorities and the subscribers;
  13. laying down the duties of the Certifying Authorities;
  14. maintaining a data base containing the disclosure record of every Certifying Authority containing such particulars as may be specified by regulations, which shall be accessible to public.
  15.

27.7.2 Recognition of foreign Certifying Authorities.

- the Controller may with the previous approval of the Central Government, and by notification in the Official Gazette, recognize any foreign Certifying Authority as a Certifying Authority for the purposes of this Act.

- Where any Certifying Authority is recognized under sub-section (1), the Digital Signature Certificate issued by such Certifying Authority shall be valid for the purposes of this Act.

- The Controller may, if he is satisfied that any Certifying Authority has contravened any of the conditions and restrictions subject to which it was granted recognition under sub-section (1) he may, for reasons to be recorded in writing, by notification in the Official Gazette, revoke such recognition.

27.7.3 Controller to act as repository.

1. The **Controller** shall be the repository of all Digital Signature Certificates issued under this Act.
2. The Controller shall—

   √ make use of hardware, software and procedures that are secure intrusion and misuse;

   √ observe such other standards as may be prescribed by the Central Government, to ensure that the secrecy and security of the digital signatures are assured.

3. The Controller shall maintain a computerised data base of all public keys in such a manner that such data base and the public keys are available to any member of the public.

27.7.4 Licence to issue Digital Signature Certificates.

1. Any person may make an application, to the Controller, for a licence to issue Digital Signature Certificates.

2. No licence shall be issued under sub-section (1), unless the applicant fulfills such requirements with respect to qualification, expertise, manpower, financial resources and other infrastructure facilities, which are necessary to issue Digital Signature Certificates as may be prescribed by the Central Government

3. A licence granted under this section shall—

   - be valid for such period as may be prescribed by the Central Government;

   - not be transferable or heritable;

   - be subject to such terms and conditions as may be specified by the regulations.

27.7.5 Application for licence

1. Every application for issue of a licence shall be in such form as may be prescribed by the Central Government.

2. Every application for issue of a licence shall be accompanied by—

   - ✓ a certification practice statement;
   - ✓ a statement including the procedures with respect to identification of the applicant;
   - ✓ payment of such fees, not exceeding twenty-five thousand rupees as may be prescribed by the Central Government;

   such other documents, as may be prescribed by the Central Government.

27.7.6 Renewal of licence.

➢ An application for renewal of a licence shall be—

✓ in the required form;

✓ accompanied by such fees, not exceeding five thousand rupees, as may be prescribed by the Central Government and shall be made not less than forty-five days before the date of expiry of the period of validity of the licence.

27.7.7 Procedure for grant or rejection of licence.

➢ The Controller may, on receipt of an application under sub-section (1) of section 21, after considering the documents accompanying the application and such other factors, as he deems fit, grant the licence or reject the application:

27 .7.8 Suspension of licence.

➢ The Controller may, if he is satisfied after making such inquiry, as he may think fit, that a Certifying Authority has,—

(a) made a statement in, or in relation to, the application for the issue or renewal of the licence, which is incorrect or false in material particulars;

(b) failed to comply with the terms and conditions subject to which the licence was granted;

27.7.9 Notice of suspension or revocation of licence.

1. Where the licence of the Certifying Authority is suspended or revoked, the Controller shall publish notice of such suspension or revocation, as the case may be, in the database maintained by him.

2. Where one or more repositories are specified, the Controller shall publish notices of such

suspension or revocation, as the case may be, in all such repositories:

3. Provided that the data base containing the notice of such suspension or revocation, as the case may be, shall be made available through a web site which shall be accessible round the clock:

27.7.10 Power to delegate.

The Controller may, in writing, authorize the Deputy Controller, Assistant Controller or any officer to exercise any of the powers of the Controller under this Chapter.

27.7.11 Power to investigate contraventions.

1. The Controller or any officer authorized by him in this behalf shall take up for investigation any contravention of the provisions of this Act, rules or regulations made thereunder.

2. The Controller or any officer authorized by him in this behalf shall exercise the like powers which are conferred on Income-tax authorities under Chapter XIII of the Income-tax Act, 1961 and shall exercise such powers, subject to such limitations laid down under that Act.

27.7.12 Access to computers and data.

1. Without prejudice to the provisions of sub-section (1) of section 69, the Controller or any person authorized by him shall, if he has reasonable cause to suspect that any contravention of the provisions of this Act, rules or regulations made there under has been committed, have access to any computer system, any apparatus, data or any other material connected with such system, for the purpose of searching or causing a search to be made for obtaining any information or data contained in or available to such computer system.

2. For the purposes of sub-section (1), the Controller or any person authorised by him may, by order, direct any person incharge of, or otherwise concerned with the operation of, the computer system, data apparatus or material, to provide him with such reasonable technical and other assistance as he may consider necessary.

27.7.13 Certifying Authority to follow certain procedures.

Every Certifying Authority shall, —

- make use of hardware, software and procedures that are secure from intrusion and misuse;

- provide a reasonable level of reliability in its services which are reasonably suited to the performance of intended functions;

- adhere to security procedures to ensure that the secrecy and privacy of the digital signatures are assured; and

- observe such other standards as may be specified by regulations.

27.7.14 Certifying Authority to ensure compliance of the Act, etc.

√ Every Certifying Authority shall ensure that every person employed or otherwise engaged by it complies, in the course of his employment or engagement, with the provisions of this Act, rules,

regulations and orders made thereunder

27.7.15 Display of licence

√ Every Certifying Authority shall display its licence at a conspicuous place of the premises in which it carries on its business.

27.7.16 Surrender of licence

1. Every Certifying Authority whose licence is suspended or revoked shall immediately after such suspension or revocation, surrender the licence to the Controller.

2. Where any Certifying Authority fails to surrender a licence under sub-section (1), the person in whose favour a licence is issued, shall be guilty of an offence and shall be punished with imprisonment which may extend up to six months or a fine which may extend up to ten thousand rupees or with both.

27.7.17 Disclosure.

1. Every Certifying Authority shall disclose in the manner specified by regulations—

   ✓ its Digital Signature Certificate which contains the public key corresponding to the private key used by that Certifying Authority to digitally sign another Digital Signature Certificate;

   ✓ any certification practice statement relevant thereto;

   ✓ notice of the revocation or suspension of its Certifying Authority certificate, if any; and any other fact that materially and adversely affects either the reliability of a Digital Signature Certificate, which that Authority has issued, or the Authority's ability to perform its services.

2. Where in the opinion of the Certifying Authority any event has occurred or any situation has arisen which may materially and adversely affect the integrity of its computer system or the conditions subject to which a Digital Signature Certificate was granted, then, the Certifying Authority shall—

   (a) use reasonable efforts to notify any person who is likely to be affected by that occurrence; or
   (b) act in accordance with the procedure specified in its certification practice statement to deal with such event or situation.

27.8 DIGITAL SIGNATURE CERTIFICATE

27.8.1 Certifying Authority to issue Digital Signature Certificate.

1. Any person may make an application to the Certifying Authority for the issue of a Digital Signature Certificate in such form as may be prescribed by the CentralGovernment
2. Every such application shall be accompanied by such fee not exceeding twenty- five thousand rupees as may be prescribed by the Central Government, to be paid to the Certifying Authority:
3. Every such application shall be accompanied by a certification practice statement or where there is no such statement, a statement containing such particulars, as may be specified by

regulations.

4. On receipt of an application under sub-section (1), the Certifying Authority may, after consideration of the certification practice statement or the other statement under sub- section (3)
5. and after making such enquiries as it may deem fit, grant the Digital Signature Certificate or for reasons to be recorded in writing, reject the application:
6. Provided that no Digital Signature Certificate shall be granted unless the Certifying Authority is satisfied that—

- the applicant holds the private key corresponding to the public key to be listed in the Digital Signature Certificate;
- the applicant holds a private key, which is capable of creating a digital signature;
- the public key to be listed in the certificate can be used to verify a digital signature affixed by the private key held by the applicant.

27.8.2 Representations upon issuance of Digital Signature Certificate.

- A Certifying Authority while issuing a Digital Signature Certificate shall certify that--

1. it has complied with the provisions of this Act and the rules and

   regulations made there under;
2. it has published the Digital Signature Certificate or otherwise made it available to such person relying on it and the subscriber has accepted it;
3. the subscriber holds the private key corresponding to the public key, listed in the Digital Signature Certificate;
4. the subscriber's public key and private key constitute a functioning key pair,
5. the information contained in the Digital Signature Certificate is accurate; and
6. it has no knowledge of any material fact, which if it had been included in the Digital Signature Certificate would adversely affect the reliability of the representations made in clauses (a) to (d).

27.8.3 Suspension of Digital Signature Certificate

1. Subject to the provisions of sub-section (2), the Certifying Authority which has issued a Digital Signature Certificate may suspend such Digital Signature Certificate,—

(a) on receipt of a request to that effect from—

I. the subscriber listed in toe Digital Signature Certificate; or

II. any person duly authorised to act on behalf of that subscriber,

2. if it is of opinion that the Digital Signature Certificate should be suspended in public interest
3. A Digital Signature Certificate shall not be suspended for a period exceeding fifteen days unless the subscriber has been given an opportunity of being heard in the matter.
4. On suspension of a Digital Signature Certificate under this section, the Certifying Authority shall communicate the same to the subscriber.

27.8.4 Revocation of Digital Signature Certificate.

1. A Certifying Authority may revoke a Digital Signature Certificate issued byit—
   (a) where the subscriber or any other person authorised by him makes a request to that effect;
   (b) upon the death of the subscriber, or
   (c) upon the dissolution of the firm or winding up of the company where the subscriber is a firm or a company.
2. Subject to the provisions of sub-section (3) and without prejudice to the provisions of sub- section (1), a Certifying Authority may revoke a Digital Signature Certificate which has been issued by it at any time, if it is of opinionthat—
   (a) a material fact represented in the Digital Signature Certificate is false or has been concealed;
   (b) a requirement for issuance of the Digital Signature Certificate was not satisfied;
   (c) the Certifying Authority's private key or security system was compromised in a manner materially affecting the Digital Signature Certificate's reliability;
   (d) the subscriber has been declared insolvent **or** dead or where a subscriber is a firm or a company, which has been dissolved, wound- up **or** otherwise ceased to exist
3. A Digital Signature Certificate shall not be revoked unless the subscriber has been given an opportunity of being heard in the matter.
4. On revocation of a Digital Signature Certificate under this section, the Certifying Authority

27.8.5 Notice of suspension or revocation.

1. Where a Digital Signature Certificate is suspended or revoked under section 37 or section 38, the Certifying Authority shall publish a notice of such suspension or revocation, as the case may be, in the repository specified in the Digital Signature Certificate for publication of such notice.
2. Where one or more repositories are specified, the Certifying Authority shall publish notices of such suspension or revocation, as the case may he. in all such repositories.

27.9 DUTIES OF SUBSCRIBERS

27.9.1 Generating key pair

- Where any Digital Signature Certificate, the public key of which corresponds to the private key of that subscriber which is to be listed in the Digital Signature Certificate has been accepted by a subscriber, then, the subscriber shall generate the key pair by applying the security procedure.

27.9.2 Acceptance of Digital Signature Certificate

1. A subscriber shall be deemed to have accepted a Digital Signature Certificate if he publishes or authorises the publication of a Digital Signature Certificate—

(a) to one or more persons;

(b) in a repository, or otherwise demonstrates his approval of the Digital Signature Certificate in any manner.

2. By accepting a Digital Signature Certificate the subscriber certifies to all who reasonably rely on the information contained in the Digital Signature Certificate that—
   (a) the subscriber holds the private key corresponding to the public key listed in the Digital Signature Certificate and is entitled to hold the same;
   (b) all representations made by the subscriber to the Certifying Authority and all material relevant to the information contained in the Digital Signature Certificate are true;
   (c) all information in the Digital Signature Certificate that is within the knowledge of the subscriber is true.

27.9.3 Control of private key

1. Every subscriber shall exercise reasonable care to retain control of the private key corresponding to the public key listed in his Digital Signature Certificate and take all steps to prevent its disclosure to a person not authorised to affix the digital signature of thesubscriber.
2. If the private key corresponding to the public key listed in the Digital

Signature Certificate has been compromised, then, the subscriber shall communicate the same without any delay to the Certifying Authority in such manner as may be specified by the regulations.

27.10 PENALTIES AND ADJUD1CATION

27.10.1 Penalty for damage to computer, computer system, etc.

- If any person without permission of the owner or any other person who is incharge of a computer, computer system or computer network, —

1. accesses or secures access to such computer, computer system or computer network;
2. downloads, copies or extracts any data, computer data base or information from such computer, computer system or computer network including information or data held or stored in any removable storage medium;
3. introduces or causes to be introduced any computer contaminant or computer virus into any computer, computer system or computer network;
4. damages or causes to be damaged any computer, computer system or computer network, data, computer data base or any other programmes residing in such computer, computer system or computer network;
5. disrupts or causes disruption of any computer, computer system or computer network;
6. denies or causes the denial of access to any person authorised to access any computer, computer system or computer network by anymeans;
7. provides any assistance to any person to facilitate access to a computer, computer system or computer network in contravention of the provisions of this Act, rules or regulations made thereunder;
8. charges the services availed of by a person to the account of another person by tampering with or manipulating any computer, computer system, or computer network,

- he shall be liable to pay damages by way of compensation not exceeding one crore rupees to the person so affected.
- "computer contaminant" means any set of computer instructions that are

designed— to modify, destroy, record, transmit data or programme residing within a computer, computer system or computer network; orby any means to usurp the normal operation of the computer, computer system, or computer network;

- "computer data base" means a representation of information, knowledge, facts, concepts or instructions in text, image, audio, video that are being prepared or have been prepared in a formalised manner or have been produced by a computer, computer system or computer network and are intended for use in a computer, computer system or computer network;
- "computer virus" means any computer instruction, information, data or programme that destroys, damages, degrades or adversely affects the performance of a computer resource or attaches itself to another computer resource and operates when a programme, data or instruction is executed or some other event takes place in that computer resource;
- "damage" means to destroy, alter, delete, add, modify or rearrange any computer resource by any means.

27.10.2 compensation for failure to protect data

- If a corporate handling any sensitive information in a computer resource owns, controls or operates in negligent in maintaining security which causes gain to other person.in such case the corporate shall be liable to pay damages to the aggrieved party.

27.10.3 Penalty for failure to furnish information return, etc.

- If any person who is required under this Act or any rules or regulations made thereunder to—

1. furnish any document, return or report to the Controller or? he Certifying Authority fails to furnish the same, he shall be liable to a penalty not exceeding one lakh and fifty thousand rupees for each such failure;
2. file any return or furnish any information, books or other documents within the time specified therefor in the regulations fails to file return or furnish the same within the time specified therefor in the regulations, he shall be liable to a penalty not exceeding five thousand rupees for every day during which such failure continues
3. maintain books of account or records, fails to maintain the same, he shall be liable to a penalty not exceeding ten thousand rupees for every day during which the failure continues.

27.10.4 Residuary penalty

- The contravention of which no penalty has been separately provided, shall be liable to pay a compensation not exceeding twenty-five thousand rupees to the person affected by such contravention or a penalty not exceeding twenty-five thousand rupees.

27.10.5 Power to adjudicate

1. For the purpose of adjudging under this Chapter whether any person has committed a contravention of any of the provisions of this Act or of any rule, regulation, direction or order made thereunder the Central Government shall, subject to the provisions of sub-section (3), appoint any officer not below the rank of a Director to the Government of India or an equivalent officer of a State Government to be an adjudicating officer' for holding an inquiry in the manner prescribed by the Central Government.
2. The adjudicating officer shall, after giving the person referred to in sub-section
    - a reasonable opportunity for making representation in the matter and if, on such inquiry, he is satisfied that the person has committed the contravention, he may impose such penalty or award such compensation as he thinks fit in accordance with the provisions of that section.

3. No person shall be appointed as an adjudicating officer unless he possesses such experience in the field of Information Technology and legal or judicial experience as may be prescribed by the Central Government.
4. Where more than one adjudicating officers are appointed, the Central Government shall specify by order the matters and places with respect to which such officers shall exercise their jurisdiction
5. Every adjudicating officer shall have the powers of a civil court which are conferred on the Cyber Appellate Tribunal under sub- section (2) of section 58, and—
   (a) all proceedings before it shall be deemed to be judicial proceedings within the meaning of sections 193 and 228 of the Indian Penal Code;
   (b) shall be deemed to be a civil court for the purposes of sections 345 and 346 of the Code of Criminal Procedure, 1973.

27.10.6 Factors to be taken into account by the adjudicating officer

1. While adjudging the quantum of compensation under this Chapter, the adjudicating officer shall have due regard to the following factors, namely:—
   1. the amount of gain of unfair advantage, wherever quantifiable, made as a result of the default
   2. the amount of loss caused to any person as a result of the default;
   3. the repetitive nature of the default.

27.11 THE CYBER REGULATIONS APPELLATE TRIBUNAL

27.11.1 Establishment of Cyber Appellate Tribunal.

1. The Central Government shall, by notification, establish one or more appellate tribunals to be known as the Cyber Regulations Appellate Tribunal.
2. The Central Government shall also specify, in the notification referred to in sub- section (1), the matters and places in relation to which the Cyber Appellate Tribunal may exercise jurisdiction.

27.11.2 Composition of Cyber Appellate Tribunal.

- A Cyber Appellate Tribunal shall consist of one person only (hereinafter referred to as the Residing Officer of the Cyber Appellate Tribunal) to be appointed, by notification, by the Central Government

27.11.3 Qualifications for appointment as Presiding Officer of the Cyber Appellate Tribunal.

A person shall not be qualified for appointment as the Presiding Officer of a Cyber Appellate Tribunal unless he—

1. is, or has been. or is qualified to be, a Judge of a High Court; or
2. is or has been a member of the Indian Legal Service and is holding or has held a post in Grade I of that Service for at least three years.

27.11.4 Term of office

- The Presiding Officer of a Cyber Appellate Tribunal shall hold office for a term of five years from the date on which he enters upon his office or until he attains the age of sixty- five years, whichever is earlier.

27.11.5 Salary, allowances and other terms and conditions of service of Presiding Officer.

- The salary and allowances payable to, and the other terms and conditions of service including

pension, gratuity and other retirement benefits of the Presiding Officer of a Cyber Appellate Tribunal shall be such as may be prescribed.

27.11.6 Filling up of vacancies.

- If, for reason other than temporary absence, any vacancy occurs in the office n the Presiding Officer of a Cyber Appellate Tribunal, then the Central Government shall appoint another person in accordance with the provisions of this Act to fill the vacancy and the proceedings may be continued before the Cyber Appellate Tribunal from the stage at which the vacancy is filled.

27.11.7 Resignation and removal

1. The Presiding Officer of a Cyber Appellate Tribunal may, by notice in writing under his hand addressed to the Central Government, resign his office:

Provided that the said Presiding Officer shall, unless he is permitted by the Central Government to relinquish his office sooner, continue to hold office until the expiry of three months from the date of receipt of such notice or until a person duly appointed as his successor enters upon his office or until the expiry of his term of office, whichever is the earliest.

2. The Presiding Officer of a Cyber Appellate Tribunal shall not be removed from his office except by an order by the Central Government on the ground of proved misbehavior or incapacity after an inquiry made by a Judge of the Supreme Court in which the Presiding Officer concerned has been informed of the charges against him and given a reasonable opportunity of being heard in respect of these charges.

3. The Central Government may, by rules, regulate the procedure for the investigation of misbehaviour or incapacity of the aforesaid Presiding Officer.

27.11.8 Orders constituting Appellate Tribunal to be final and not to invalidate its proceedings

- No order of the Central Government appointing any person as the Presiding Officer of a Cyber Appellate Tribunal shall be called in question in any manner and no act or proceeding before a Cyber Appellate Tribunal shall be called in question in any manner on the ground merely of any defect in the constitution of a Cyber Appellate Tribunal.

27.11.9 Staff of the Cyber Appellate Tribunal

1. The Central Government shall provide the Cyber Appellate Tribunal with such officers and employees as that Government may think fit
2. The officers and employees of the Cyber Appellate Tribunal shall discharge their functions under general superintendence of the Presiding Officer.
3. The salaries, allowances and other conditions of service of the officers and employees or' the Cyber Appellate Tribunal shall be such as may be prescribed by the Central Government.

27.11.10 Appeal to Cyber Appellate Tribunal

1. Save as provided in sub-section (2), any person aggrieved by an order made by Controller or an adjudicating officer under this Act may prefer an appeal to a Cyber Appellate Tribunal having jurisdiction in the matter.
2. No appeal shall lie to the Cyber Appellate Tribunal from an order made by an adjudicating officer with the consent of the parties.
3. Every appeal under sub-section (1) shall be filed within a period of tony-five days from the date on which a copy of the order made by the Controller or the adjudicating officer is received by the person aggrieved and it shall be in such form and be accompanied by such fee as may be prescribed:
   Provided that the Cyber Appellate Tribunal may entertain an appeal after the expiry of the

said period of tony-five days if it is satisfied that there was sufficient cause for not filing it within that period On receipt of an appeal under sub-section (1), the Cyber Appellate Tribunal may, after giving the parties to the appeal, an opportunity of being heard, pass such orders thereon as it thinks fit, confirming, modifying or setting aside the order appealed against.

4. The Cyber Appellate Tribunal shall send a copy of every order made by it to" the parties to the appeal and to the concerned Controller or adjudicating officer
5. The appeal filed before the Cyber Appellate Tribunal under sub-section (1) shall be dealt with by it as expeditiously as possible and endeavour shall be made by it to dispose of the appeal finally within six months from the date of receipt of the appeal.

27. 11.11. Procedure and powers of the Cyber Appellate Tribunal.

1. The Cyber Appellate Tribunal shall not be bound by the procedure laid down by the Code of civil Procedure, 1908 but shall be guided by the principles of natural justice and, subject to the other provisions of this Act and of any rules, the Cyber Appellate Tribunal shall have powers to regulate its own procedure including the place at which it shall have its sittings.
2. The Cyber Appellate Tribunal shall have, for the purposes of discharging its functions under this Act, the same powers as are vested in a civil court under the Code of Civil Procedure, 1908, while trying a suit, in respect of the following matters, namely:—
    a. summoning and enforcing the attendance of any person and examining him on oath;
    b. requiring the discovery and production of documents or other electronic records
    c. receiving evidence on affidavits;
    d. issuing commissions for the examination of witnesses or documents;
    e. reviewing its decisions;
    f. dismissing an application for default or deciding it *ex pane;*
    g. any other matter which may be prescribed.
3. Every proceeding before the Cyber Appellate Tribunal shall be deemed to be a judicial proceeding within the meaning of sections 193 and 228, and for the purposes of section 196 of the Indian Penal Code and the Cyber Appellate Tribunal shall be deemed to be a civil court for the purposes of section 195 and Chapter XXVI of the Code of Criminal Procedure, 1973.

27.11.12 Right to legal representation.

- The appellant may either appear in person or authorise one or more legal practitioners or any of its officers to present his or its case before the Cyber Appellate Tribunal.

27.11.13 Limitation.

- The provisions of the Limitation Act, 1963, shall, as far as may be, apply to an appeal made to the Cyber Appellate Tribunal.

27.11.14 Civil court not to have jurisdiction.

- No court shall have jurisdiction to entertain any suit or proceeding in respect of any matter which an adjudicating officer appointed under this Act or the Cyber Appellate Tribunal constituted under this Act is empowered by or under this Act to determine and
  no injunction shall be granted by any court or other authority in respect of any action taken or to be taken in pursuance of any power conferred by or under this Act.

27.11.15 Appeal to High Court

- Any person aggrieved by any decision or order of the Cyber Appellate Tribunal may file an appeal to the High Court within sixty days from the date of communication of the decision or order of the Cyber Appellate Tribunal to him on any question of fact or law arising out of such order.
- Provided that the High Court may, if it is satisfied that the appellant was prevented by sufficient cause from filing the appeal within the said period, allow it to be filed within a further period not exceeding sixty days.

27.11.16 Compounding of contraventions.

1. Any contravention under this Chapter may, either before or after the institution of adjudication proceedings, be compounded by the Controller or such other officer as may be specially authorised by him in this behalf or by the adjudicating officer, as the case may be, subject to such conditions as the Controller or such other officer or the adjudicating officer may specify:

   Provided that such a sum shall not, in any case, exceed the maximum amount of the penalty which may be imposed under this act for the contravention so compounded.
2. Nothing in sub-section (1) shall apply to a person who commits the same or similar contravention within a period of three years from the date on which the first contravention, committed by him was compounded.
3. Where any contravention has been compounded under sub-section (1), no proceeding or further proceeding, as the case may be, shall be taken against the person guilty of such contravention in respect of the contravention so compounded.

27.11.17 Recovery of penalty

- A penalty imposed under this Act, if it is not paid, shall be recovered as an arrear of land revenue and the licence or the Digital Signature Certificate, as the case may be, shall be suspended till the penalty is paid.

27.12 OFFENCES

27.12.1 Tampering with computer source documents

- Whoever knowingly or intentionally conceals, destroys or alters or intentionally or knowingly causes another to conceal, destroy or alter any computer source code used for a computer, computer programme, computer system or computer network, when the computer source code is required to be kept or maintained by law for the time being in force, shall be punishable with imprisonment up to three years, or with fine which may extend up to two lakh rupees, or with both.
- Explanation. — "computer source code" means the listing of programmes, computer commands, design and layout and programme analysis of computer resource in any form.

27.12.2 Hacking with computer system

(i) Whoever with the intent to cause or knowing that he is likely to cause wrongful loss or damage to the public or any person destroys or deletes or alters any information residing in a computer resource or diminishes its value or utility or affects it injuriously by any means, commits hack

(ii) Whoever commits hacking shall be punished with imprisonment up to three years, or with fine which may extend up to two lakh rupees, or with both.

27.12.3 punishment for receiving stolen computer resource or communication device

- Whoever dishonestly received or retains any stolen computer resource of communication device knowing 'on device or having reason to believe the same to be stolen computer resource or communication three ice, shall be punished with imprisonment of either description for a term which may extend to years or with fine which may extend to rupees one lakh or with both [Section 66B].

27.12.4 punishment for identity theft

- Whoever fraudulently or dishonestly make use of the electronic signature, password or any unique identification feature of any other person, shall be punished with

imprisonment of either description for a term which may extend to three years and shall also be liable to fine which may extend to rupees one lakh [Section 66B].

27.12.5 Punishment for cheating by personation by using computer resource

(iii) Whoever, by means for any communication device or computer resource cheats by personating, shall be punished with imprisonment of either description for a term which may extend to three years and shall also be liable to fine which may extend to one lakh rupees [Section 66D].

27.12.6 Punishment for violation of privacy

(iv) Whoever, intentionally or knowingly captures, publishes or transmits the image of a private area of any person without his or her consent, under circumstances violating the privacy of that person, shall be punished with imprisonment which may extend to three years or with fine not exceeding two lakh rupees, or with both [Section 66E].

27.12.7 Punishment for cyber terrorism

1. Whoever,

- With intent to threaten the unity, integrity, security of sovereignty of India or to strike terror in the people or any section of the people by-
  - (i) denying or cause the denial of access to any person authorized to access computer resource; or
  - (ii) attempting to penetrate or access a computer resource without authorization or exceeding authorized access; or
  - (iii) introducing or causing to introduce any computer contaminant, and by means of such conduct causes or is likely to cause death or injuries to persons or damage to or destruction of property or disrupts or knowing that it is likely to cause damage or disruption of supplies or services essential to the life of the community or adversely affect the critical information infrastructure specified under Section 70; or
- knowingly or intentionally penetrates or accesses a computer resource without authorization or exceeding authorized access, and by means of such conduct obtains access to information, data or computer database that is restricted; or any restricted information, data or computer database, with reasons to believe that such information, data or computer database so obtained may be used to cause or likely to cause injury to the interests of the sovereignty and integrity of India, the security of the State, friendly relations with foreign States, public order, decency or morality, or in relation to contempt of court, defamation or incitement to an offence, or to the advantage of any foreign nation, group of individuals, or otherwise, commits the offence of cyber terrorism.

2. Whoever commits or conspires to commit cyber terrorism shall be punishable with imprisonment which may extended to imprisonment for life [Section 66F]

27.12.7 Publishing of information which is obscene in electronic form

- Whoever publishes or transmits or causes to be published in the electronic form, any material which is lascivious or appeals to the prurient interest or if its effect is such as to tend to deprave and corrupt persons who are likely, having regard to all relevant circumstances, to read, see or hear the matter contained or embodied in it, shall be punished on first conviction with imprisonment of either description for a term which may extend to five years and with fine which may extend to one lakh rupees and in the event of a second or subsequent conviction with imprisonment of either description for a term which may extend to ten years and also with fine which may extend to two lakh rupees.

27.12.9 Power of Controller to give directions

(1) The Controller may, by order, direct a Certifying Authority or any employee of such Authority to take such measures or cease carrying on such activities as specified in the order if those are necessary to ensure compliance with the provisions of this Act, rules or any regulations made thereunder.

(2) Any person who fails to comply with any order under sub-section (1) shall be guilty of an offence and shall be liable on conviction to imprisonment for a term not exceeding three years or to a Fine not exceeding two lakh rupees or to both.

27.12.11 Protected system

(1) The appropriate Government may, by notification in the Official Gazette, declare that any computer, computer system or computer network to be a protected system.

(2) The appropriate Government may, by order in writing, authorise the persons who are authorized to access protected systems notified under sub-section (1).

(3) Any person who secures access or attempts to secure access to a protected system in contravention of the provisions of this section shall be punished with imprisonment of either description for a term which may extend to ten years and shall also be liable to fine.

27.12.12 Penalty for misrepresentation

- Whoever makes any misrepresentation to, or suppresses any material fact from, the Controller or the Certifying Authority for obtaining any licence or Digital Signature Certificate, as the case may be. shall be punished with imprisonment for a term which may extend to two years, or with fine which may extend to one lakh rupees, or with both.

27.12.13 Penalty for breach of confidentiality and privacy

- Save as otherwise provided in this Act or any other law for the time being in force, any person who, in pursuance of any of the powers conferred under this Act, rules or regulations made thereunder, has secured access to any electronic record, book, register, correspondence, information, document or other material without the consent of the person concerned discloses such electronic record, book. register, correspondence, information, document or other material to any other person shall be punished with imprisonment for a term which may extend to two years, or with fine which may extend to one lakh rupees, or with both.

27.12.14 Penalty for publishing Digital Signature Certificate false in certain particulars

(1) No person shall publish a Digital Signature Certificate or otherwise make it available to any other person with the knowledge that—

(a) the Certifying Authority listed in the certificate has not issued it; or

(b) the subscriber listed in the certificate has not accepted it; or

(c) the certificate has been revoked or suspended, unless such publication is for the purpose of verifying a digital signature created prior to such suspension or revocation.

(2) Any person who contravenes the provisions of sub-section (1) shall be punished with imprisonment for a term which may extend to two years, or with fine which may extend to one lakh rupees, or with both.

27.12.15Publication for fraudulent purpose

- Whoever knowingly creates, publishes or otherwise makes available a Digital Signature Certificate for any fraudulent or unlawful purpose shall be punished with imprisonment for a term which may extend to two years, or with fine which may extend to one lakh rupees, or with both.

27.12.16 Act to apply for offence or contravention committed outside India

(1) Subject to the provisions of sub-section (2), the provisions of this Act shall apply also to any offence or contravention committed outside India by any person irrespective of his nationality.

(2) For the purposes of sub-section (1), this Act shall apply to an offence or contravention committed outside India by any person if the act or conduct constituting the offence or contravention involves a computer, computer system or computer network located in India.

27.12.17 Confiscation

- Any computer, computer system, floppies, compact disks, tape drives or any other accessories related thereto, in respect of which any provision of this Act. rules, orders or regulations made there under has been or is being contravened, shall be liable to confiscation:
- However, where it is established to the satisfaction of the court adjudicating the confiscation that the person in whose possession, power or control of any such computer, computer system, floppies, compact disks, tape drives or any other accessories relating to is found is not responsible for the contravention of the provisions of this Act, rules, orders or regulations made there under, the court may, instead of making an order for confiscation of such computer, computer system, floppies, compact disks, tape drives or any other accessories related thereto, make such other order authorised by this Act against the person contravening of the provisions of this Act, rules, orders or regulations made there under as it may think fit.

27.12.18 Penalties or confiscation not to interfere with other punishments.

- No penalty imposed or confiscation made under this Act shall prevent the imposition of any other punishment to which the person affected thereby is liable under any other law for the time being in force.

27.12.19 Power to investigate offences.

- Notwithstanding anything contained in the Code of Criminal Procedure, 1973, a police officer not below the rank of Deputy Superintendent of Police shall investigate any offence under this Act.

27.13 NETWORK SERVICE PROVIDERS NOT TO BE LIABLE IN CERTAIN CASES

- For the removal of doubts, it is hereby declared that no person providing any service as a network service provider shall be liable under this Act, rules or regulations made there under for any third party information or data made available by him if he proves that the offence or contravention was committed without his knowledge or that he had exercised all due diligence to prevent the commission of such offence or contravention.

27.14 MISCELLANOUS PROVISIONS

27.14.1 Power of police officer and other officers to enter, search, etc.

1. Notwithstanding anything contained in the Code of Criminal Procedure, 1973, any police officer, not below the rank of a Deputy Superintendent of Police, or any other officer of the Central Government or a State Government authorised by the Central Government in this behalf may enter any public place and search and arrest without warrant any person found therein who is reasonably suspected or having committed or of committing or of being about to commit any

offence under this Act

***Explanation. —For* the purposes of this sub-section, the expression "public place" includes any public conveyance, any hotel, any shop or any other place intended for use by, or accessible to the public.**

2. Where any person is arrested under sub-section (1) by an officer other than a police officer, such officer shall, without unnecessary delay, take or send the person arrested before a magistrate having jurisdiction in the case or before the officer-in-charge of a police station.
3. The provisions of the Code of Criminal Procedure, 1973 shall, subject to the provisions of this section, apply, so far as may be, in relation to any entry, search or arrest, made under this section.

27.14.2 Act to have overriding effect

The provisions of this Act shall have effect notwithstanding anything inconsistent therewith contained in any other law for the time being in force.

27.14.3Controller, Deputy Controller and Assistant Controllers to be public servants

The Presiding Officer and other officers and employees of a Cyber Appellate Tribunal, the Controller, the Deputy Controller and the Assistant Controllers shall be deemed to be public servants within the meaning of section 21 of the Indian Penal Code.

27.14.4Power to give directions

The Central Government may give directions to any State Government as to the carrying into execution in the State of any of the provisions of this Act or of any rule, regulation or order made there under.

27.14.5 Protection of action taken in good faith

No suit, prosecution or other legal proceeding shall lie against the Central Government, the State Government, the Controller or any person acting on behalf of him, the Presiding Officer, adjudicating officers and the staff of the Cyber Appellate Tribunal for anything which is in good faith done or intended to be done in pursuance of this Act or any rule, regulation or order made there under.

27.14.6 Offences by companies

- Where a person committing a contravention of any of the provisions of this Act or of any rule, direction or order made thereunder is a company, every person who, at the time the contravention was committed, was in charge of, and was responsible to, the company for the conduct of business of the company as well as the company, shall be guilty of the contravention and shall be liable to be proceeded against and punished accordingly:
- Provided that nothing contained in this sub-section shall render any such person liable to punishment if he proves that the contravention took place without his knowledge or that he exercised all due diligence to prevent such contravention.
- Notwithstanding anything contained in sub-section (1), where a contravention of any of the provisions of this Act or of any rule, direction or order made thereunder has been committed by a company and it is proved that the contravention has taken place with the consent or connivance of, or is attributable to any neglect on the part of, any director,

manager, secretary or other officer of the company, such director, manager, secretary or other officer shall also be deemed to be guilty of the contravention and shall be liable to be proceeded against and punished accordingly.

27.14.7 Removal of difficulties

(1) If any difficulty arises in giving effect to the provisions of this Act, the Central Government may, by order published in the Official Gazette, make such provisions not inconsistent with the provisions of this Act as appear to it to be necessary or expedient for removing the difficulty:
(2) Every order made under this section shall be laid, as soon as may be after it is made, before each House of Parliament.

27.14.8Constitution of Advisory Committee

1. The Central Government shall, as soon as may be after the commencement of this Act, constitute a Committee called the Cyber Regulations Advisory Committee.
2. The Cyber Regulations Advisory Committee shall consist of a Chairperson and such number of other official and non-official members representing the interests principally affected or having special knowledge of the subject-matter as the Central Government may deem fit.
3. The Cyber Regulations Advisory Committee shall advise—

(a) the Central Government either generally as regards any rules or for any other purpose connected with this Act;

(b) the Controller in framing the regulations under this Act.

4. There shall be paid to the non-official members of such Committee such travelling and other allowances as the Central Government may fix.
5.

27.14.9 Special provisions as to evidence relating to electronic record

The contents of electronic records may be proved in accordance with the provisions of section 65B.

27.14.10 Admissibility of electronic records

- Any information contained in an electronic record which is printed on a paper, stored, recorded or copied in optical or magnetic media produced by a computer (hereinafter referred to as the computer output) shall be deemed to be also a document, if the conditions mentioned in this section are satisfied in relation to the information and computer in question and shall be admissible in any proceedings, without further proof or production of the original, as evidence of any contents of the original or of any fact stated therein of which direct evidence would be admissible.

27.14.11Presumption as to electronic records and digital signatures

1. In any proceedings involving a secure electronic record, the Court shall presume unless contrary is proved, that the secure electronic record has not been altered since the specific point of time to which the secure status relates
2. In any proceedings, involving secure digital signature, the Court shall presume unless the contrary is proved that—

(a) the secure digital signature is affixed by subscriber with the intention of signing or approving the electronic record;

(b) except in the case of a secure electronic record or a secure digital signature, nothing in this section shall create any presumption relating to authenticity and integrity of the electronic record or any digital signature.

27.14.12 Presumption as to Digital Signature Certificates

The Court shall presume, unless contrary is proved, that the information listed in a Digital Signature Certificate is correct, except for information specified as subscriber information which has not been verified, if the certificate was accepted by the subscriber.".

27.14.13.1.1 Presumption as to electronic messages

The Court may presume that an electronic message forwarded by the originator through an electronic mail server to the addressee to whom the message purports to be addressed corresponds with the message as fed into his computer for transmission; but the Court shall not make any presumption as to the person by whom such message was sent.