Chapter 1 : Introduction and Number Theory 1-1 to 1-57

1.1 Concept Building - Security - What is it really? 1-2
1.2 Concept Building - Information Security Concepts 1-4
1.2.1 Confidentiality 1-4
1.2.2 Integrity 1-5
1.2.3 Availability 1-5
1.3 Concept Building - Security Threats and Vulnerabilities 1-6
1.3.1 Security Threats 1-6
1.3.1(A) Comparison between Security Threats 1-8
1.3.2 Security Vulnerabilities 1-8
1.4 Concept building - Access Control and Attacks 1-9
1.4.1 STRIDE Model 1-10
1.5 OSI Model 1-11
1.5.1 The OSI Security Architecture 1-12
1.5.2 Security Services 1-13
1.5.3 Security Mechanisms 1-13
1.5.4 Placement of Security Services and Mechanisms 1-14
1.6 Network Security Model 1-16
1.7 Types of Security Attacks 1-17
1.7.1 Active Attacks 1-17
1.7.2 Passive Attacks 1-19
1.7.3 Comparison between Active and Passive Attacks 1-20
1.8 Concept Building – Information Secrecy 1-21
1.9 Concept Building - Introduction to Cryptography 1-22
1.10 Classical Encryption Techniques 1-23
1.10.1 Substitution Cipher 1-23
1.10.1(A) Vignere Cipher 1-24
1.10.1(B) Playfair Cipher 1-26
1.10.1(C) Hill Cipher 1-35
1.10.1(D) Affine Cipher 1-36
1.10.2 Transposition Techniques 1-39
1.10.2(A) Keyed Transposition Cipher 1-40
1.10.2(B) Keyless Transposition Cipher 1-41

Scanned with CamScanner

1.11 Methods of Encryption 1-41
1.11.1 Symmetric Key Encryption 1-42
1.11.2 Asymmetric Key Encryption 1-43
1.11.3 Comparison between Symmetric and Asymmetric Keys 1-45
1.12 Modular Arithmetic and Number Theory 1-46
1.13 Greatest Common Divisor (GCD) 1-48
1.13.1 Euclid's or Euclidean Algorithm 1-48
1.13.2 Extended Euclidean Algorithm 1-49
1.13.3 Multiplicative Inverse using Extended Euclidean Algorithm 1-53

Chapter 2 : Symmetric and Asymmetric Key Cryptography and Key Management **2-1 to 2-30**

2.1 Concept Building - Types of Symmetric Algorithms (Ciphers) 2-2
2.1.1 Block Ciphers 2-2
2.1.2 Stream Ciphers 2-2
2.1.3 Comparison between Block and Stream Cipher 2-3
2.1.4 Block Cipher Principles 2-3
2.2 Data Encryption Standard (DES) 2-4
2.2.1 Block Diagram and Internals of DES 2-5
2.2.2 Block Cipher Modes of Operation (for DES and other Block Ciphers in General) 2-7
2.2.3 Comparison between Modes of Operation 2-8
2.2.4 Double DES 2-9
2.2.5 3DES or Triple DES 2-10
2.3 Advanced Encryption Standard (AES) 2-10
2.3.1 Block Diagram and Internals of AES 2-11
2.3.2 Comparison between DES and AES 2-12
2.4 Concept Building - Attacks on Cryptosystems 2-12
2.4.1 Comparison between Differential and Linear Cryptanalysis 2-13
2.5 Public Key Cryptography 2-13
2.5.1 Principles of Public Key Cryptosystems 2-14
2.5.2 RSA Algorithm 2-14
2.5.3 Knapsack Algorithm 2-17
2.6 Key Management Techniques 2-18
2.6.1 What is a Key? 2-18
2.6.2 Key States 2-19
2.6.3 Cryptoperiod (Key lifetime) 2-19

Scanned with CamScanner

2.6.4 Key Management Principles 2-20
2.6.5 Symmetric Key Distribution 2-20
2.6.5(A) Symmetric Key Distribution using Symmetric Encryption 2-20
2.6.5(B) Symmetric Key Distribution using Asymmetric Encryption 2-21
2.6.6 Asymmetric Key Distribution (Distribution of Public Keys) 2-22
2.6.7 Key Management using Diffie Hellman Key Exchange 2-23
2.7 Digital Certificate - X.509 2-26
2.8 Public Key Infrastructure (PKI) 2-27
2.8.1 Components of PKI 2-27
2.9 Needham-Schroeder protocol 2-29
2.10 Kerberos : Kerberos Authentication protocol 2-29

Chapter 3 : Cryptographic Hash Functions ..... 3-1 to 3-14

3.1 Concept Building – Information Accuracy 3-1
3.2 Message Authentication Functions 3-2
3.3 Cryptographic Hash Functions 3-3
3.3.1 Introduction 3-3
3.3.2 How does this Work? 3-3
3.3.3 Characteristics / Properties of Secure Hash Functions 3-4
3.4 Hash Functions (Algorithms) 3-6
3.4.1 SHA-1 3-7
3.4.2 SHA-3 3-8
3.4.3 MD5 3-9
3.4.3(A) Major Attributes of MD5 3-9
3.4.3(B) MD5 Algorithm Details 3-9
3.4.4 Comparison between SHA and MD5 3-10
3.5 MAC (Message Authentication Code) 3-10
3.5.1 HMAC 3-11
3.5.2 CBC-MAC 3-12
3.5.3 CMAC 3-12
3.5.4 Comparison between Hash and MAC 3-13
3.5.5 Comparison between HMAC, CBC-MAC and CMAC 3-13
3.6 Security of Hash Functions and MAC 3-13
3.6.1 Security of Hash Functions 3-13
3.6.2 Attacks on Hash Functions and MAC 3-14

Scanned with CamScanner

Chapter 4 : Authentication Protocols and Digital Signature Schemes 4-1 to 4-18

4.1 User and Entity Authentication 4-1
4.1.1 Types of Authentication Methods 4-2
4.1.2 Comparison between the Authentication Types 4-6
4.2 Factors of Authentication 4-6
4.3 One-way and Mutual Authentication Schemes 4-7
4.3.1 Needham Schroeder Authentication Protocol (Challenge Response Based Authentication) 4-7
4.3.1(A) The Needham–Schroeder Symmetric Key Based Authentication Protocol 4-7
4.3.1(B) The Needham–Schroeder Asymmetric Key Based Authentication Protocol 4-8
4.3.2 Kerberos Authentication Protocol 4-9
4.3.2(A) Problems Addressed by Kerberos 4-10
4.3.2(B) Components of Kerberos 4-10
4.3.2(C) How does it work? 4-10
4.3.2(D) Limitations of Kerberos 4-12
4.4 Digital Signature 4-12
4.4.1 How does this work? 4-12
4.4.2 Application and use of Digital Signature 4-13
4.4.3 Properties of Digital Signature 4-13
4.5 Attacks on Digital Signature 4-13
4.6 RSA Digital Signature Scheme 4-13

Chapter 5 : Network Security and Applications 5-1 to 5-39

5.1 Network Security Basics 5-2
5.2 TCP/IP Vulnerabilities (Layer Wise) 5-2
5.2.1 Packet Sniffing 5-3
5.2.2 ARP Spoofing 5-4
5.2.3 Port Scanning 5-5
5.2.3(A) Port Scanning Techniques 5-6
5.2.4 IP Spoofing 5-7
5.2.5 Denial of Service (DoS) and Distributed Denial of Service (DDoS) 5-7
5.2.5(A) Botnet 5-8
5.2.5(B) Types of DDoS Attacks 5-9
5.2.5(C) Preventing DDoS Attacks 5-10
5.3 Internet Security Protocols 5-10

Scanned with CamScanner

5.3.1 Secure Socket Layer (SSL) 5-11
5.3.1(A) Overview of SSL Protocol 5-12
5.3.1(B) SSL Record Layer Protocol 5-13
5.3.1(C) SSL Change Cipher Spec Protocol 5-14
5.3.1(D) SSL Alert Protocol 5-14
5.3.1(E) SSL Handshake Protocol 5-15
5.3.2 Transport Layer Security (TLS) 5-17
5.3.3 Internet Protocol Security (IPSec) 5-19
5.3.3(A) Security Association 5-20
5.3.3(B) Modes of Operation 5-21
5.3.3(C) Applications / Benefits / Usage of IPSec 5-21
5.3.3(D) How does IPSec work? 5-22
5.3.3(E) Security Protocols/ Services Provided by IPSec 5-22
5.4 Secure Email 5-26
5.4.1 Pretty Good Privacy (PGP) 5-26
5.4.1(A) PGP Services 5-28
5.4.1(C) PGP Algorithms 5-30
5.4.2 S/MIME 5-30
5.4.2(A) S/MIME Services 5-30
5.4.2(B) S/MIME Algorithms 5-31
5.4.2(C) S/MIME Cryptographic Message Syntax (CMS) 5-31
5.4.3 Firewalls 5-31
5.4.3(A) Components of a Firewall Rule 5-32
5.4.3(B) Classification of Firewalls 5-32
5.4.3(C) Challenges in Managing and Deploying Firewalls 5-35
5.4.4 Intrusion Detection Systems (IDS) 5-35
5.4.4(A) Introduction 5-36
5.4.4(B) Need for IDS 5-36
5.4.4(C) Types of IDS 5-36
5.4.4(D) Limitations and Challenges of IDS 5-37

Chapter 6 : System Security **6-1 to 6-37**

6.1 Software Vulnerabilities 6-1
6.2 Buffer Overflow 6-3
6.2.1 Causes of Buffer Overflow and Suggested Protection Mechanism 6-6

Scanned with CamScanner

6.2.1(A) Low-level Programming Languages 6-7
6.2.1(B) Unsafe Libraries 6-7
6.2.1(C) Poor Coding Practices 6-8
6.2.1(D) Lack of Secure Code Review and Analysis 6-8
6.2.1(E) Lack of Penetration Testing 6-10
6.2.1(F) No Specific Buffer Overflow Protection in Place 6-10
6.2.2 Specific Protection Mechanisms for Buffer Overflow 6-10
6.2.2(A) Operating System Based Buffer Overflow Protection Mechanisms 6-11
6.2.2(B) Compiler Based Buffer Overflow Protection Mechanisms 6-13
6.3 Types of Buffer Overflow 6-17
6.3.1 Based on Area of Memory 6-17
6.3.1(A) Stack Overflow 6-17
6.3.1(B) Heap Overflow 6-18
6.3.2 Based on Exploit 6-19
6.3.2(A) Off-By-One 6-19
6.3.2(B) Format String 6-19
6.3.2(C) Integer 6-21
6.4 SQL Injection (SQLi) 6-21
6.4.1 How does SQLi Work? 6-22
6.4.2 Types of SQLi 6-24
6.4.2(A) Based on the Order of Attack 6-24
6.4.2(B) In-band 6-25
6.4.2(C) Inferential (blind) 6-26
6.4.2(D) Out-of-band 6-27
6.4.3 Preventing SQLi 6-28
6.5 Malware (or Malicious Code) 6-30
6.5.1 Concept Building-Activities related to Malware 6-30
6.5.2 Types of Malware 6-30
6.5.2(A) Virus and Worms 6-32
6.5.3 General Guidelines for Preventing Malware 6-36

Scanned with CamScanner

1 Introduction and Number Theory

Syllabus

At the end of this unit, you should be able to understand and comprehend the following syllabus topics :

- The OSI Model
  - Security Architecture
  - Security Services
  - Security Mechanisms
  - Network Security Model
- Types of Security Attacks
- Classical Encryption techniques
  - Substitution techniques
    - Mono-alphabetic cipher
    - Poly-alphabetic cipher
      - Vignere cipher
      - Playfair cipher
      - Hill cipher
- Transposition techniques
    - Keyed transposition ciphers
    - Keyless transposition ciphers
- Symmetric cipher
- Modular Arithmetic and Number Theory
  - Euclid's algorithm
  - Prime numbers
  - Fermat's and Euler's theorem

Scanned with CamScanner

1.1 Concept Building - Security - What is it really?

Before we begin with understanding information security and its related concepts, let's talk.

Let me ask you a question	:	"How do you manage your Debit Card and its PIN? Do you leave your Debit Card unattended and with PIN information available to everyone?"
Your Response (Laughingly)	:	"Of course, not. I keep my Debit Card with me all the time and never share my PIN with anyone."
My Response	:	"Oh, that's nice. But, why do you need to do that?"
Your Response	:	"Because, I need to ensure that my money is safe, and no one takes it out except me. I don't trust everyone with my money these days, you know."
My Response	:	"Got it. You are a security champion."

If you followed our conversation, you already know what security is. Our job is easy now. Let us define some terms around our conversation above.

1. **Assets** : You were trying to protect your money, isn't it? It is called Assets. Money is your Asset in our conversation that you were trying to protect.

✍ ***Definition :*** *Assets are something that has value and is worth protecting.*

Security is all about ensuring that the assets are kept protected all the time as much as possible within your capabilities or means.

2. **Controls (or Countermeasures)** : So, how did you actually safeguard your money? You didn't leave the Debit Card around and you memorized your PIN. Isn't it?

✍ ***Definition :*** *Any countermeasures or actions that you take to safeguard an asset are called Controls.*

So, in our *conversation*, you have put two controls in place to safeguard your money – first is to keep your Debit Card with you and second is to memorize your PIN. You are a security champion!

3. **Threat** : Hey, you told me that you don't trust everyone with your money, isn't it? That unknown everyone who can do evil to you or can harm you is called a Threat.

✍ ***Definition :*** *A threat is a person or an entity that can exploit an asset bypassing your controls (if they are weak controls and not enough to safeguard your asset).*

You knew there are threats around your money, and you protected it so well. You are a security champion.

4. **Vulnerability** : What if you left your Debit Card and PIN on the table for anyone to get hold of them and use? I hear you scream, "Come on, why would I do that to myself?". Exactly, You would not want to create a situation in which your assets can be harmed. This is precisely called addressing (or avoiding) a Vulnerability.

✍ ***Definition :*** *Vulnerability is the weakness or lack of controls around assets.*

I am happy that you have put two good controls (keeping your Debit Card safe and memorizing your PIN) and you avoided the vulnerability around your money (asset).

(Copyright No. - L82548/2019)

Scanned with CamScanner

5. **Risk :** So far, you would agree that leaving Debit Card and PIN unattended poses a likelihood that someone might just grab them and use them.

✍ ***Definition :*** *That likelihood of a harm occurring to an asset is called Risk.*

It is this Risk that you want to reduce by applying controls around your assets. Remember one thing here, Risk can NEVER be 0 (zero). Someone can steal your Debit Card from your wallet and force you at gunpoint to tell your PIN.

The core thing that you need to ensure when dealing with Risk is "to reduce it to an acceptable level". Never aim to make anything (or any asset more precisely speaking) risk free because that's not possible, really.

6. **Exposure :** Someday suppose you do accidently leave your Debit Card behind and your PIN was known to someone, you could actually lose some or all your money. That particular day or rather that particular situation of you forgetting your Debit Card behind could lead to an exposure.

✍ ***Definition :*** *Exposure is an instance of being harmed.*

So, if you got exposed anytime, immediately change your PIN and take a lesson in security to apply controls always around assets so that you do not have future exposures. I am sure you won't have exposures because you are a security champion already, aren't you?

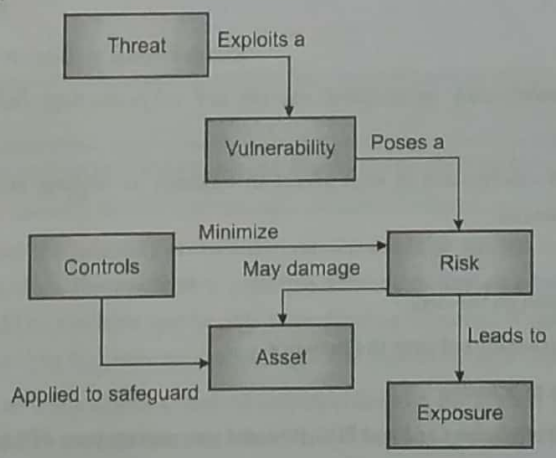


Fig. 1.1.1

Let me summarize the above terms in a simple block diagram.

If you are feeling good about what we talked about so far, believe me, you have started your security journey on a high note. The several chapters and topics that you read in this book (howsoever complex or dry they look at first) are all written to help you effectively do just ONE thing – **Safeguard your Assets**.

If you know what you are ;

- Trying to protect
- And from whom
- And how

You understand security. There is nothing else to learn.

Scanned with CamScanner

1.2 Concept Building - Information Security Concepts

Q.	Define the goals of security.	MU - Dec. 15, 2 Marks
Q.	Enlist security goals. Discuss their significance.	MU - May 19, 5 Marks

Now that you have a general understanding of security, let's set some context about Information Security. When we say information security – what exactly are we protecting? What is the asset? The asset here is "Information" or more precisely "Digital Information". The information could be about your Facebook user account, Online bank account, OS password, email or pretty much anything that touches a computer system.

There are 3 tenets (or pillars) of security :

1. Confidentiality
2. Integrity
3. Availability

These tenets in short are also called as the CIA triad or any other combination of the first letters in their words. These are also sometimes called **goals of security**.

Let's dive deeper into each one of them.

1.2.1 Confidentiality

✍ ***Definition :*** *Confidentiality can be defined as, an act of protecting information from unauthorized disclosure to an entity.*

It ensures that the protected information is kept secret throughout its lifetime and is made available only to the authorized entities as and when needed.

The information should be,

1. **Protected at Rest :** When stored on the disk.
2. **Protected in Motion :** When transmitted over the network.
3. **Protected during Use :** When processing.

Remember our conversation from Debit Card and PIN? How did you protect your PIN and provide confidentiality to it?

1. **Protected at Rest :** You didn't write it down. You kept it in your mind. No one could know or use it except you.
2. **Protected in Motion :** You physically moved to an ATM (carrying your mind and the protected PIN there) instead of revealing it to anyone.
3. **Protected during use :** You watch out if someone is looking at your fingers as you punch the PIN on the ATM keyboard.

In terms of digital information, confidentiality is enforced using several mechanisms :

1. Encryption
2. Access control
3. Data classification

We would be studying them at depth in later chapters.

Scanned with CamScanner

1.2.2 Integrity

✍ ***Definition :*** *Integrity can be defined as, an act of protecting information from unauthorized modification by an entity.*

It ensures that the information remains intact and no unauthorized entity can modify it. Any modification to the information is allowed only if the entity is authorized to do so. The information requires to maintain its integrity throughout its lifetime.

For example, during criminal investigations, any evidence that you collect is protected from touching or any modifications to ensure that those evidences can be used during court proceedings. If an evidence is tampered, it is not admissible in the court and cannot be used. Another example is email. If I send you an email and someone changes it before you read it, you might get wrong information, or it could be severely damaging to our relations.

In terms of digital information, integrity is enforced using several mechanisms :

1. Hashing
2. Access Control
3. Data Classification
4. Input and output sanitization

We would be studying them at depth in later chapters.

1.2.3 Availability

✍ ***Definition :*** *Availability can be defined as, an act of protecting information from unauthorized destruction by an entity.*

It ensures that the information is adequately protected to remain available when it is needed. Any unauthorized entity should not be able to destroy it. Also, the availability principle extends to any equipment such as computers, network devices and printers. These should be available and be able to perform as expected. If someone can get access to them and then prevent you from using these then that impacts availability of the system for your use.

For example, your Windows or Linux systems track all activities done on the system via log files. If I do some mischiefs around your computer and then delete the log files, you would have no way to prove that I did something to your computer. The availability of log files is crucial to ensure that the system is adequately monitored and protected from any security mishaps.

Availability is generally enforced using several mechanisms :

1. Access control
2. Isolation
3. Back up
4. Disaster recovery
5. Business continuity processes

We would be studying them at depth in later chapters.

Scanned with CamScanner

Let's summarize the above 3 security principles with the help of diagrams.

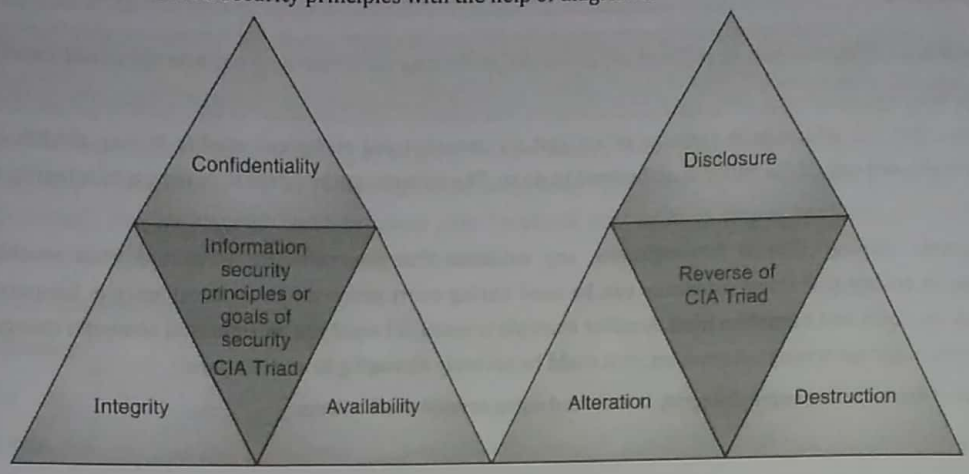


Fig. 1.2.1 : Security Principles

Confidentiality, Integrity and Availability are the 3 core principles of security. Ensuring that you understand the objectives behind these principles is crucial to your success in the information and cybersecurity domain.

1.3 Concept Building - Security Threats and Vulnerabilities

From our previous discussion on Debit Card and PIN, you now understand what security threat and vulnerabilities mean. Threats can exploit your assets and vulnerabilities are situations that could possibly lead to such an exploit.

Let us review some of the security threats and vulnerabilities commonly found in the context of information security.

1.3.1 Security Threats

There are several security threats to an information system. Some of them are briefed as following.

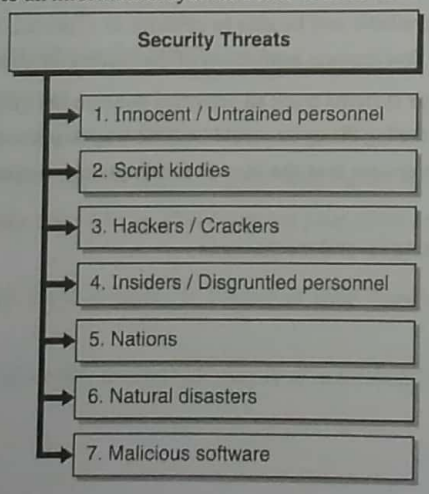


Fig. 1.3.1 : Security Threats

Scanned with CamScanner

1. **Innocent / Untrained personnel :** These could be your employees, household members or any person who does not understand intricate complexities of security. These people believe the information presented to them and often are soft targets of several frauds. These can get easily convinced and can be pushed to do harm to your organization (say by sharing critical details) or to any other critical asset. As a countermeasure, you should provide security training time to time and enforce the idea that security is everyone's responsibility.
2. **Script kiddies :** These are just exploiting the systems for fun. They have a lot of free time and can go around the internet to find systems that have weak controls. Once a system is found, they can play games, watch movies, download other software or just send some random messages on the screen. These do not have sophisticated skills to exploit weaknesses themselves and usually depend on attack tools or software. As a countermeasure, test your website and software against general attack tools and ensure that any weaker controls are sufficiently addressed.
3. **Hackers / Crackers :** These are people who have sophisticated computer security skills. They have a deep understanding of how various protocols, services, operating systems, drivers, network equipment etc. work and can thus launch sophisticated attacks on such information systems. They usually hide their presence and activities to ensure that they are unnoticed and can exploit the systems for a long time without getting detected. As a countermeasure, invest in penetration testing of your website and software and ensure that all the security findings are adequately addressed.
4. **Insiders / Disgruntled personnel :** These people are on your side, but they have malicious intent to impact your systems. They might have grudge on you or the organization and typically exploit the systems to take revenge. Insider threats are extremely hard to detect since you might believe that their actions are part of their job and may not suspect them or monitor them very closely. As a countermeasure, use access control to provide least possible permissions required to carry out one's job. You should evaluate the permissions time to time and ensure that those permissions are still relevant to the job done by the person.
5. **Nations :** Many a times, nations spy on each other and want to steal information related to country defence, forces, arms, and other intellectual property and confidential information that can severely damage the reputation of the country or its economics. These attacks are highly sophisticated but have huge impact on nations. For example, you might have heard of Russian involvement in the US elections. As a countermeasure, nations typically protect the sensitive information by limiting information sharing only amongst the high-ranking officials. They deploy top notch security solutions, processes and continuously monitor their operations to detect any unauthorized activities.
6. **Natural disasters :** Natural disasters such as flood, earthquake, lightning, etc. can severely damage the information systems (remember availability as one of the tenets of security?) and could impact information availability. As a countermeasure, you invest in backup, business continuity processes and disaster recovery solutions that can quickly bring back the systems and information to avoid large impact on your business.
7. **Malicious software :** These are software programs written with malicious intent. The purpose of these programs is to harm the information systems or extract useful information in an unauthorized manner. As a countermeasure, you install such software detection tools. These tools could be anti-virus, anti-malware, anti-spyware, intrusion detection system, intrusion prevention system, etc. We would explore this in depth in the subsequent section.

Scanned with CamScanner

1.3.1(A) Comparison between Security Threats

Table 1.3.1 : Comparison between Security Threats

Threat	Skills required	Impact	Detection Possibility
Innocent / Untrained personnel	None	Low - High	Low
Script Kiddies	Medium	Low - High	High
Hackers / Crackers	High	High	Low
Insiders	Low	High	Low
Nations	High	High	Low
Natural Disasters	None	High	Low
Malicious software	Medium	High	High

1.3.2 Security Vulnerabilities

As you recall, vulnerability is the lack of controls around assets. Let us see some of the common security vulnerabilities.

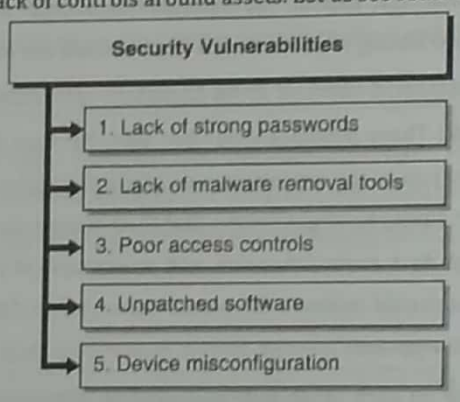


Fig. 1.3.2 : Security Vulnerabilities

1. **Lack of strong passwords :** Passwords are everywhere. For quick and easy recall, you tend to choose simple passwords such as your name, date of birth, school name, etc. People, who know you, are generally aware of this information as well and can try to use this information as your password. Additionally, if you tend to choose simple English words as your passwords, someone can do a dictionary-based attack (more on that later) and can find out your password. As a countermeasure, always choose a complex password that is the combination of uppercase letters, lowercase letters, digits and special characters. Change your passwords regularly and do not use old passwords.
2. **Lack of malware removal tools :** There are several malwares that can be installed on your device as you browse through various websites on the internet. If you do not have a good malware removal tool, overtime, your device might be compromised and impacted. As a countermeasure, install a reputed malware removal tool and update its definitions time to time.
3. **Poor access controls :** Do you allow everyone to be an administrator on your system? Can anyone access anything? If your answers are yes, you probably have not thought the "Least Privilege" principle yet. You should only grant enough permissions as required by the job at hand. No more and no less. Understand the different permissions required and assign them accordingly.

Scanned with CamScanner

4. **Unpatched software :** Vendors release security patches (software bundles that fix something in the software installed previously) time to time to fix security vulnerabilities found. If you do not install these patches, your system could be prone to exploit because the required security fix to stop the exploit is not installed. As a countermeasure, install software updates as released by the vendor specially the ones that carry security fixes.

5. **Device misconfiguration :** Quite a few times, we configure devices for maximum ease and minimum security. For example, have you locked your phone with a password or a PIN? If you get a security warning, do you click ok even without reading and understanding what the warning is about? When installing an app on your phone, do you ignore to review device permissions that the app would have? Such ignorant behaviour and poor device configuration could weaken the controls that the vendor has put out of factory. As a countermeasure, carefully review your device settings and ensure that they are tuned to provide adequate security.

1.4 Concept building - Access Control and Attacks

Let's understand the basics of access control mechanisms. These mechanisms are at the core of designing security in information systems and are often the target of attacks.

1. Identification

✍ ***Definition :*** *Identification (in short ID) is defined as, a way to claim an entity's presence with respect to the process being carried out.*

This means that during a process, your presence (or your consent) is ascertained (or established). For example, when you try to login to your Facebook account, you provide your Email or Phone number to establish your presence during the login process. There are several other forms of identification that we use today such as Aadhar Card, PAN Card, Voter ID, Debit Card, Admit card, etc. All of these identification methods bring a sense of credibility that you are present, or you give your consent to complete a particular process.

2. Authentication

✍ ***Definition :*** *Authentication is defined as, a way to ensure that the entity is indeed what it claims to be.*

This means that providing just the ID is not enough. You must additionally prove that the ID belongs to you. For example, even if I know your Facebook email address or phone, I cannot login as you until I also know the password. Thus, knowing just the ID is not enough. We need to prove that the ID belongs to us and that is what is precisely called authentication. It is for this reason that you need to additionally sign when you submit Aadhar card or PAN card as an ID proof to ensure that someone didn't just use the photocopy of those IDs without your permission (or consent). Some of the ways to authenticate an ID are passwords, biometric (like your Aadhar fingerprints or phone sensor), PIN (like for Debit Card), or OTP (SMS that you get to confirm transaction).

3. Authorization

✍ ***Definition :*** *Authorization is defined as, a way to determine what resource an entity can access.*

Once you have provided your ID and have been successfully authenticated, the next step is authorization where the system determines if you have the permission to access the desired object. For example, even if you have a valid voter ID card but if your name is not on the electoral list at a particular area booth, you won't be allowed to vote. Having authenticated ID is one thing and getting access to the resource is another. Just because you have an authenticated ID, does not mean that you have automatically access to the resources. So, authenticated ID is a must for authorization but that does not always guarantee that you would be allowed access.

Scanned with CamScanner

4. Accountability

Definition : *Accountability is defined as, a way to record your actions.*

Suppose, you used a system to take print outs. That system logs this action (pretty much like you record attendance in lab or classroom) to build a trace (evidence or proof) that you used the printer. If you were not supposed to use the printer, the evidence can be used to find you accountable for using it without permissions and could result in particular consequences. Accountability is a key determinant of how securely a system is operating. The logs generated are continuously monitored and necessary alarms are raised if any entry is found to be suspicious.

Let's summarize the 4 access control steps with the help of Fig. 1.4.1.

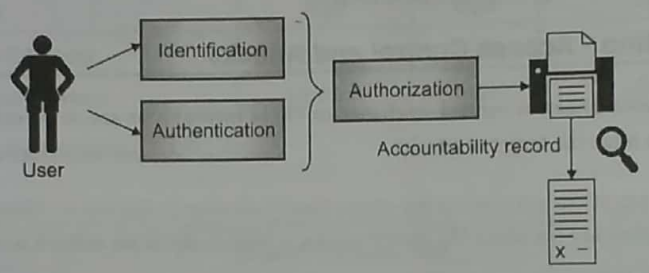


Fig. 1.4.1 : Access Control Steps

5. Non-repudiation

Definition : *Non-Repudiation is defined as, a way to prove your actions.*

It is used in conjunction with accountability and the CIA triad. Non-repudiation provides an assurance that someone cannot deny their actions later on. For example, if I sent you an email, I cannot later deny that I did not. To send an email, I must have used my email ID and password and then sent it over to you over a secure network where no one could change the email body. If you can establish all of these facts truthfully, you have proven that I sent that email and thus established non-repudiation.

Now that you understand the access control steps and security pillars (CIA), let us understand security attacks and what these attacks are actually targeting.

1.4.1 STRIDE Model

STRIDE model is often cited as a reference when designing security for information systems.

Table 1.4.1 : Attacks on Information Systems

Attack Category	Security Property that is attacked
Spoofing	Authentication
Tampering	Integrity
Repudiation	Non-repudiation
Information Disclosure	Confidentiality
Denial of service	Availability
Elevation of Privilege	Authorization

(Copyright No. - L82548/2019)

Scanned with CamScanner

1. **Spoofing :** When someone tries to steal your identity, it is called spoofing. Spoofing is also called as impersonating someone or something. For example, if I try to login to your Facebook account using your email address and password guesses, I am trying to spoof or impersonate your identity. Spoofing can also be used for fake news, fake websites, fake or malicious files or anything else that could be mistaken for being real. The target of spoofing attacks is to crack authenticity (of person, file, website, news or anything real).
2. **Tampering :** When someone tries to do an unauthorized modification to something, it is called tampering. You would have heard of cricketers tampering with the ball to suit their requirements. It is a punishable offense. Similarly, in information systems, if you try to tamper with say files, emails, or any other information in an unauthorized way, it is called tampering. Basically, you are trying to attack the integrity of the object by making such alterations.
3. **Repudiation :** In this, you are trying to falsely claim that you didn't carry out a particular action. For example, you didn't send an email or didn't visit a website. Remember your childhood days when you broke something, and you try to escape the punishment saying that you didn't actually break it and then your parents finding proofs that it was indeed you who broke it. That's precisely what is non-repudiation. In repudiation attacks, you are trying to destroy evidences that someone can use to falsify your denial claims.
4. **Information Disclosure :** This pertains to unauthorized revealing of any confidential information. In these attacks, the attacker wishes to know the confidential information and tries to crack controls around it to get hold of such confidential information.
5. **Denial of service :** In this type of attacks, the purpose of attack is to make the information system or its services unstable or unavailable to perform its assigned activities. For example, if you can succeed to bring Flipkart website temporarily down, it might mean severe loss of business for Flipkart and the customers may go to a different website for placing their urgent orders.
6. **Elevation of Privileges :** In this category of attack, the attacker tries to get elevated (higher) privileges (permissions / authority) over resources. For example, if you are only a user on a system and you try to attack the system to become an administrator on the system, it is called an elevation of privileges attack.

1.5 OSI Model

Note : Discussing OSI Model in-depth is beyond the scope of this book. It is assumed that you have covered it in detail in your subjects on networking. A general high-level overview is presented here as a refresher.

✍ ***Definition :*** *The Open Systems Interconnection Model (OSI Model) is a conceptual model that characterizes and standardizes the communication functions of networked communications without diving into complexities of protocols, architecture and the underlying technologies.*

The OSI model consists of 7 layers. Each layer interacts with the layer above and below it and passes on the respective protocol data units encapsulated into their respective headers.

Table 1.5.1 : Layers of OSI model

Layer Number	Layer Name	Protocol Data Unit	Function
7	Application	Data	Application Interface – APIs, UIs
6	Presentation	Data	Data translation between networking and application
5	Session	Data	Manage communication sessions between sender and receiver

Scanned with CamScanner

Layer Number	Layer Name	Protocol Data Unit	Function
4	Transport	Segment, Datagram	Reliable data transmission
3	Network	Packet	Network packet addressing and routing
2	Data Link	Frame	Transmission of data between two nodes
1	Physical	Binary	Actual communication over physical media

Note that each OSI layer protocol adds its own information to the data packet.

✍ ***Definition :*** *The process of adding layer specific information and passing on the data to the next layer below is called encapsulation.*

Encapsulation happens top – down (From Application to Physical Layer).

✍ ***Definition :*** *The process of removing layer specific information and passing on the data to the next layer above is called decapsulation.*

Decapsulation happens bottom – up (from Physical to Application Layer).

The Table 1.5.2 shows a quick reference summary for various protocols at the respective OSI Layers.

Table 1.5.2 : Protocols used is Various OSI Layers

Layer Name	Protocols Used
Application	HTTP, FTP, SMTP, etc.
Presentation	JPEG, MPEG, TIFF, ASCII, etc.
Session	NFS, RPC, etc.
Transport	TCP, UDP, SSL, etc.
Network	IP, ICMP, OSPF, etc.
Data Link	ARP, PPP, Ethernet, etc.
Physical	ISDN, DSL, 10Base-T, etc.

1.5.1 The OSI Security Architecture

The objective of the OSI model is to permit the interconnection of heterogeneous computer systems so that useful communication between application processes may be achieved. At the various OSI layers, the security controls must be established in order to protect the information exchanged between the application processes (or the connected computers or devices). Such controls make it difficult to obtain the information in any unauthorized way.

✍ ***Definition :*** *The OSI Security Architecture identifies the basic security services and mechanisms and their appropriate placement at the various layers.*

OSI security functions are concerned only with the OSI layers involved in the communications path. It does not include other security controls such as securing the operating system or the application process itself. Let's learn about the various security services and mechanisms placed at the OSI layers.

Scanned with CamScanner

1.5.2 Security Services

Q. Define authentication and non repudiation and show with examples how each one can be achieved.

MU - Dec. 16, 5 Marks

Definition : *Security Services are safeguard controls recommended to be placed at the various OSI layers.*

The various security services are listed as shown in Fig. 1.5.1.

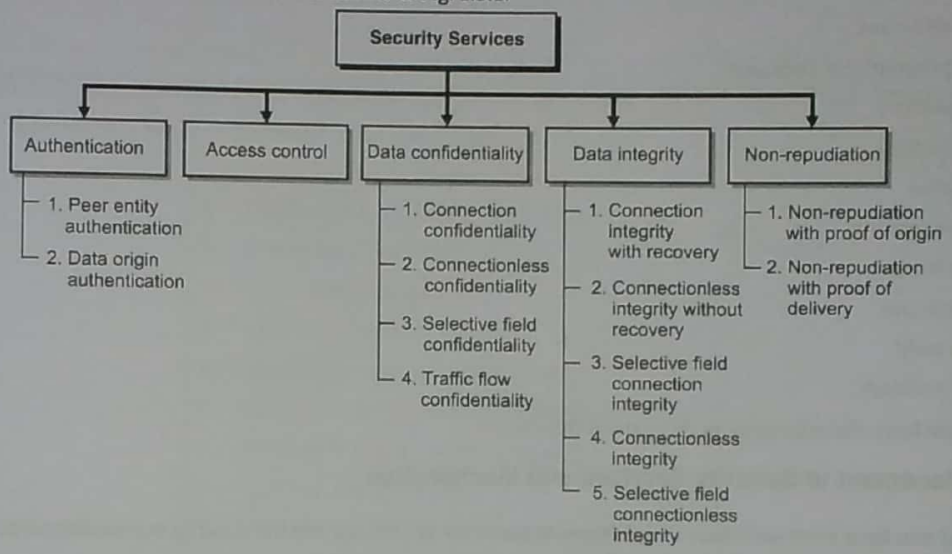


Fig. 1.5.1 : Security Services

1.5.3 Security Mechanisms

Q. Specify mechanisms to achieve each goal. **MU - Dec. 15, 3 Marks**

Q. List with examples the different mechanisms to achieve security. **MU - May 16, 5 Marks**

Definition : *Security mechanisms are various techniques recommended to provide security services at the various OSI layers.*

The various security mechanisms that can be applied are as following.

- Encipherment (Encryption)
  - Symmetric
  - Asymmetric
- Digital Signature
  - Signing a data unit
  - Verifying a data unit
- Access control
  - Passwords
  - Time of access
  - Duration of access
  - Access route

Scanned with CamScanner

- Data integrity
  - Sent quantity of data
  - Received quantity of data
  - Sequencing of data units
  - Time stamping
- Authentication
  - Handshaking
  - Cryptographic techniques
- Traffic padding
- Routing control
- Notarization
- Pervasive security
- Security labels
- Event detection
- Security audit
- Security recovery

You would learn these techniques throughout this course.

1.5.4 Placement of Security Services and Mechanisms

Now that you have a fair understanding of the various security services and the security mechanisms that can be used at the various OSI layers, let us see recommended placement for them.

Table 1.5.3 : Layerwise Security Services and Mechanisms

OSI Layer	Security Service	Security Mechanism
Physical Layer	• Connection Confidentiality • Traffic Flow Confidentiality	Encipherment
Data Link Layer	• Connection Confidentiality • Connectionless Confidentiality	Encipherment
Network Layer	• Peer Entity Authentication • Data Origin Authentication • Access Control service • Connection Confidentiality • Connectionless Confidentiality • Traffic Flow Confidentiality • Connection Integrity without recovery • Connectionless Integrity	Authentication Encipherment Digital Signature Access control Routing control Traffic Padding Data integrity
Transport Layer	• Peer Entity Authentication • Data Origin Authentication	Authentication Encipherment

Scanned with CamScanner

OSI Layer	Security Service	Security Mechanism
	• Access Control service • Connection Confidentiality • Connectionless Confidentiality • Connection Integrity with recovery • Connection Integrity without recovery • Connectionless Integrity	Digital Signature Access control Data integrity
Session Layer	No security services are provided in the session layer	Not Applicable
Presentation Layer	• Connection Confidentiality • Connectionless Confidentiality • Selective Field Confidentiality • Traffic Flow Confidentiality • Peer Entity Authentication • Data Origin Authentication • Connection Integrity with Recovery • Connection Integrity without Recovery • Selective Field Connection Integrity • Connectionless Integrity • Selective Field Connectionless Integrity • Non-repudiation with Proof or Origin • Non-repudiation with Proof of Delivery	Encipherment Digital Signature Data integrity Notarization
Application Layer	• Peer Entity Authentication • Data Origin Authentication • Access Control Service • Connection Confidentiality • Connectionless Confidentiality • Selective Field Confidentiality • Traffic Flow Confidentiality • Connection Integrity with Recovery • Connection Integrity without Recovery • Selective Field Connection Integrity • Connectionless Integrity • Selective Field Connectionless Integrity • Non-repudiation with Proof of Origin • Non-repudiation with Proof of Delivery	Encipherment Access Control Digital Signature Data integrity Traffic Padding Notarization

Scanned with CamScanner

1.6 Network Security Model

Definition : *Network Security Model (NSM) is a seven-layer model that divides the task of securing a network infrastructure into seven manageable sections.*

It is similar to the seven OSI layers. The model is generic and can apply to all security implementation and devices. NSM provides a unified way of securing networks. It is easier to pinpoint issues at the respective NSM layers and address the gaps, if any.

The Table 1.6.1 lists the NSM layers and how they align with the OSI layers. It is important to understand that like the OSI layers, each NSM layer builds on top of the previous layer. If any layer is compromised, the layers above it are disrupted as well. For example, if there is an attack at NSM layer 2, it would disrupt layers above it (3, 4, 5, 6 and 7). Let's learn about each of the NSM layers.

Table 1.6.1 : Layers in Network Security Model

Network Security Model (NSM)	OSI Model (inverted)
Physical	Physical
VLAN	Data Link
ACL	Network
Software	Transport
User	Session
Administrative	Presentation
IT Department	Application

1. **NSM Layer 1 – Physical :** It works at the physical layer. It ensures to safeguard the physical aspects of network. For example, physical access to the routers, switches or any other networking equipment. There could be several physical forms of physical security such as security alarms, security guards and CCTV.
2. **NSM Layer 2 – VLAN :** VLAN stands for Virtual Local Area Network. At this layer, the network is segmented (partitioned) into smaller network chunks to safeguard them individually and to also manage them effectively. It ensures that only authorized devices connect to the provided networks. You could create VLANs department wise, region wise or in any other suitable grouping mechanism based on your site requirements.
3. **NSM Layer 3 – ACL :** ACL stands for Access Control List. ACLs are created to allow or deny access based on the network layer from the OSI layer. For example, certain IP ranges (say finance department) might be restricted for access by other devices on the network. ACLs can be created on routers, firewalls and switches and can effectively control the network access as designed and intended.
4. **NSM Layer 4 – Software :** The software layer is focused on keeping the device software up to date with the latest upgrades and patches in order to mitigate any known software vulnerabilities. At this layer, the patches are installed to ensure that the software running on the device cannot be exploited. For example, you install security patches on your operating system or update applications on your phone to ensure that you are running the secure version of the software and it does not have any known exploits.
5. **NSM Layer 5 – User :** This layer deals with the user access and management. The user layer focuses on the user's training and knowledge about security on the network. The user should understand the basic concepts of network security and should be capable of applying security related judgement. For example, users should be aware of which software to run on the system and which not.

Scanned with CamScanner

6. **NSM Layer 6 – Administrative :** The administrative layer focuses on the training of administrative users. It works very similar to the user layer but focuses primarily on the administrative staff. It provides guidance to the layers below it to adequately protect the network. For example, it can dictate which software is allowed for user consumption.

7. **NSM Layer 7 – IT Department :** The IT department layer deals directly with the maintenance of all layers and making sure that the entire network works correctly from NSM model. It has several professionals in the team that know to architect and operate a secure network.

1.7 Types of Security Attacks

Q. List and explain various types of attacks on encrypted message. **MU - May 18, 5 Marks**

At a high level, there are two broad categories of security attacks carried over the network – active attacks and passive attacks are shown in Fig. 1.7.1.

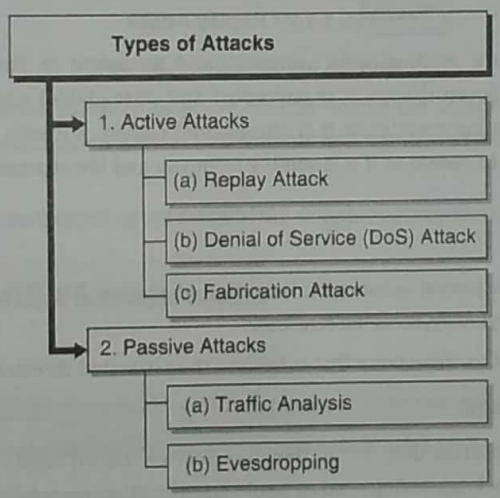


Fig. 1.7.1 : Types of Security Attacks

1.7.1 Active Attacks

✍ ***Definition :*** *An Active attack is defined as, an attack where the attacker actively participates in the communication or the attack mechanism and disrupts the systems by sending several manipulated inputs.*

In a nutshell, the attacker intercepts (captures) the communication channel, and manipulates the communication going over it. Another variation of the active attack is when the attacker continuously disrupts the ability of the system to process the information correctly. Let us expand on some of the examples of active attacks.

1. Replay Attack

- This is like replaying a song. The attacker captures the real and specific communication packets, stores them with herself, and then sends it at a later point in time as though she is sending the information for the first time like an authentic information.

Scanned with CamScanner

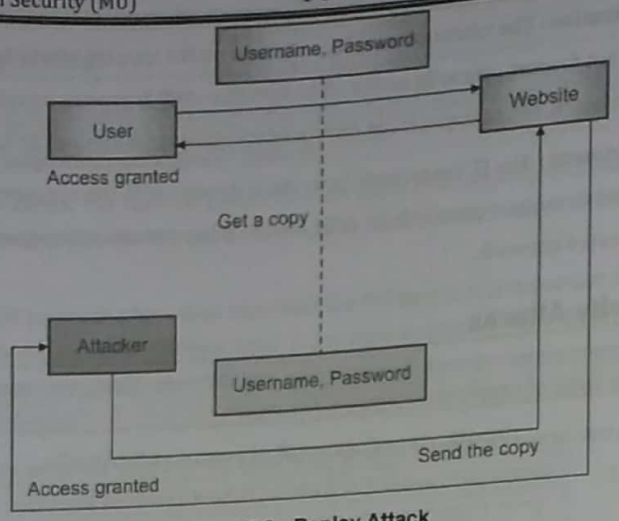


Fig. 1.7.2 : Replay Attack

- **For example :** Suppose you are sending your username and password to Facebook. Someone can capture that information over the network (even though it is encrypted and unreadable) and without even knowing the actual content of the captured information ever (since it is unreadable) can store it with herself. At a later point in time, the same captured information can be resent as if it is coming from you and the attacker might get access to your Facebook account.

Countermeasures

- You can use timestamps and sequence numbers (also called as session ID). If the message comes with a sequence number that is already used previously, it can be rejected.
- Similarly, if a message comes with a timestamp that is beyond the estimated threshold, it can be rejected.

2. Denial of Service (DoS) Attack

- Denial of Service (DoS) or its variation Distributed Denial of Service (DDoS) refers to a category of attacks that can be aimed at various layers of network, operating system, application or other parts of information systems.

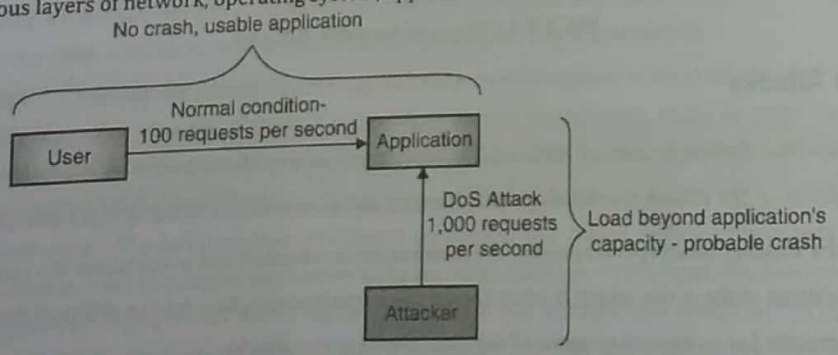


Fig. 1.7.3 : Denial of Service (DoS) Attack

- In this type of attack, the attacker overloads the system beyond its capacity such that the system or controls around it fail. Once the system fails, it is no more available for performing its assigned activities.
- **For example :** An application might be capable of processing a maximum of 100 requests per second. Attacker would typically send over 1,000 requests per second such that the application fails to cope up with it and crashes. The sole motivation behind DoS attacks is to bring down the availability of the system.

Scanned with CamScanner

Countermeasures

- Some of the countermeasures to protect from DoS are firewall, application limit, white listing networks, etc.
- Firewall can be used to drop network connections that come from a particular location or based on other networking parameters (a list of allowed IP addresses, etc.).
- Application limits can protect application from crashing when the rate of requests goes beyond a set limit.

3. Fabrication Attacks

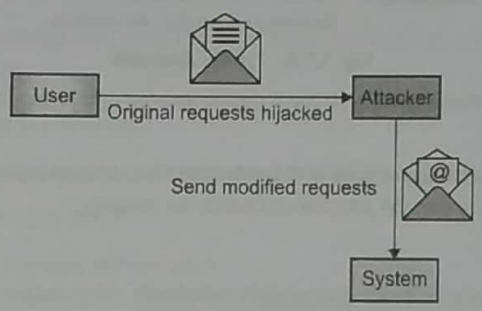


Fig. 1.7.4 : Fabrication Attacks

- Fabrication attack is again a broad category of active attacks where the attacker deliberately modifies messages, parameters, properties, etc. of information system components and try to alter the behaviour of the system often by passing security controls.
- SQL injection, masquerade and email spoofing are some of the examples of fabrication attacks.

Countermeasure :

- Hashing, redundancy checks, and input and output validation.

1.7.2 Passive Attacks

✍ ***Definition :*** *A passive attack is defined as, an attack where the attacker does not alter the behaviour of the information system and silently performs her malicious activities.*

Unlike active attacks, passive attacks are predominantly used to learn information such as number of systems, how the system operates and behaves under various circumstances and general operational characteristics such as the name and version of the operating system running on the targeted machines. Once the information is gathered, the attacker launches complex and more impactful attacks. Let us expand on some of the examples of passive attacks.

1. Traffic Analysis

- In traffic analysis, the network traffic and its patterns are watched out over a period of time to infer important information and guess possible activities.
- **For example :** Following is some information that can be possibly guessed just by knowing the traffic volume and patterns.
  - Long communication : It can denote some emergency.
  - Short communications : It can denote planning, checking, and negotiation.
  - No communication : It can indicate a lack of activity.
  - Time of communication : It can indicate who works when.

Scanned with CamScanner

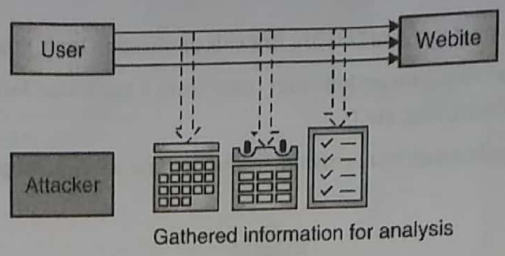


Fig. 1.7.5 : Traffic Analysis

- Such an information gathering exercise can give you a decent amount of information about your target to launch more sophisticated attacks.
- Some of the countermeasures to traffic analysis is to randomize the communication or send fake traffic time to time to degrade the quality of information that the attacker can gather for analysis.

2. Eavesdropping

- Eavesdropping is very similar to over hearing someone's telephonic conversation taking advantage of your proximity to the person.
- You can be standing close enough to the person to hear what she is speaking over phone or also hear what the other party on the phone is saying, especially in calm areas. Similarly, in digital communication, you can wire tap and get a copy of information being communicated and spy on it.
- Spying on such communication by tapping the information as it is being sent or received is called eavesdropping.

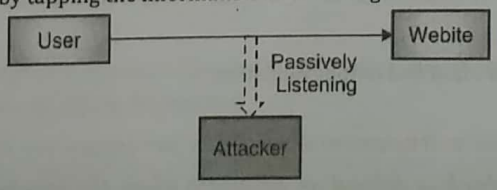


Fig. 1.7.6 : Eavesdropping

- Some of the countermeasures to eavesdropping is sending noise time to time or using random channels of communication.

1.7.3 Comparison between Active and Passive Attacks

Table 1.7.1 : Comparison between Active and Passive Attacks

Comparison Attribute	Active Attack	Passive Attack
Complexity	High	Low
Impact	High	Low
Detection Possibility	High	Low
Prevention Possibility	High	Low
Duration of attack	Short	Long
System Behaviour	Modified	Unaffected
Original Information	Modified	Unaffected
Purpose	Harm the ecosystem	Learn about the ecosystem

Scanned with CamScanner

1.8 Concept Building – Information Secrecy

Consider a scenario. You and your friend are in different cities and are chatting over phone using an application. You are discussing a new idea to found a startup and have some cool plans that can change the world and make the business very profitable.

But wait, I have two questions for you;

1. How do you know that your conversation (you and your friend sending and receiving text from each other) is not available for anyone else to read?

   [**Goal :** Information not available to everyone].

2. Don't you think that your business plan is confidential and the conversation between you and your friend should somehow remain readable only to you both?

   [**Goal :** Information available to intended entities only].

If you realize the importance of protecting the digital confidential information, you so much wish that there was a way in which you could ensure that the information is available only to you both – blink your eyes and wish granted – Welcome to the world of Cryptography!!

The dictionary meaning of cryptography is "secret writing".

Digitally speaking,

✍ ***Definition :*** *Cryptography is a science and a method of storing and transmitting information in a form that only those it is intended for can read and process its.*

In a nutshell, the information is available in a readable form only to those who are authorized.

Let's take a different example to understand a related concept. You might have used coupon codes in online shopping. Based on the offers running on the website, the coupon code (a set of characters) discounts the price on your chosen items. You apply the coupon code to your cart and based on various terms and conditions, the discount amount is calculated. Sometimes, you get a flat off and sometimes a percentage of the cart value.

In this scenario, we have two interesting concepts;

- The coupon code (a unique set of alphabets and digits).
- The coupon code processing terms and conditions (the algorithm (rules) that determines how to apply the coupon code and how much discount to actually give you).
- In a different example, you and your friend might have some words (or codes) that are only understood by you both. When you use that word, your friend knows what exactly that really means even if you say it loud and clear and others hear it.

For example : Suppose you both have decided that when you say, "I eat banana", that would actually mean "Let's bunk college today". Now, when you say it, your friend gets the real meaning whereas everyone else thinks that you were referring to a fruit.

So, an important concept to understand here is that when the real information is hidden within what is being communicated, the communicated information can be stored or transmitted without disclosing the actual meaning and we don't care if someone gets the communicated information because the real information is hidden. So, real information can freely move around hiding within the communicated information.

Scanned with CamScanner

1.9 Concept Building - Introduction to Cryptography

Now that you understand some of the scenarios around information secrecy, let's define some the basic terms that we would be using throughout the chapter.

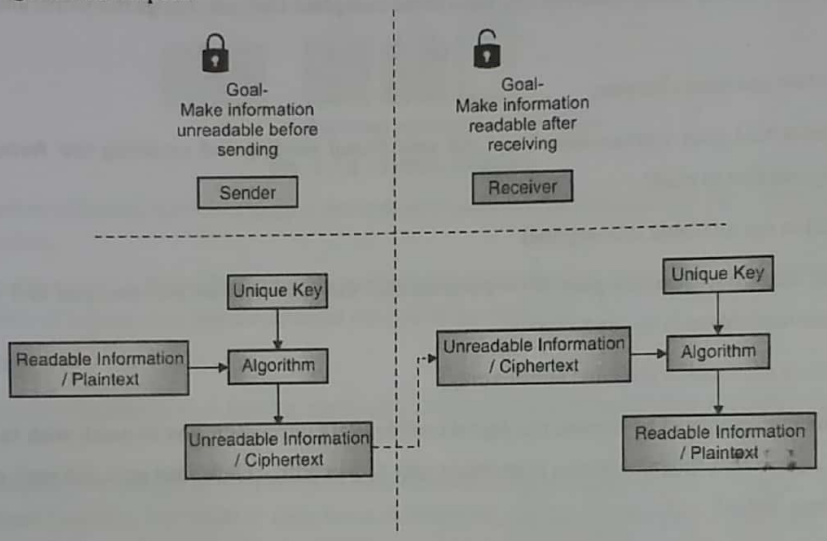


Fig. 1.9.1 : How Cryptography Works

1. **Goal of cryptography :** As you understand, the core goal of cryptography is to hide confidential information. So, cryptography majorly provides Confidentiality out of the CIA triad (Confidentiality, Integrity and Availability).
2. **Information/ Data :** This is the asset that is being protected (provide confidentiality using cryptography in this case). An asset is something that has value and is worth protecting. The sender ensures that the information is only readable by the intended receiver even if it is captured/ seen by anyone else.
3. **Unique Key :** This is a set of numbers (like your coupon code) that helps to make the information secret. It is like your usual key that helps to lock and unlock the door. Note here that like various bikes could have the same locking mechanism, a key is unique to a bike even if the bikes are of same model.
4. **Algorithm :** Algorithm is a process or set of rules to be followed to make the information secret. There are various algorithms (like various types of locks) available that make the information secret in different ways using the keys. In cryptography, such algorithms are also called ciphers.
5. **Plaintext :** The information that is readable and understandable is called plaintext.
6. **Ciphertext :** The information that is not-understandable even if you can read it is called ciphertext.
7. **Encryption (or encipher) :** Encryption is a method of converting plaintext into ciphertext by using an algorithm and a key.

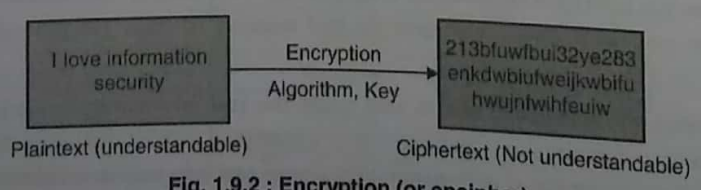


Fig. 1.9.2 : Encryption (or encipher)

Scanned with CamScanner

8. **Decryption (or decipher) :** Decryption is a method of converting ciphertext into plaintext by using an algorithm and a key.

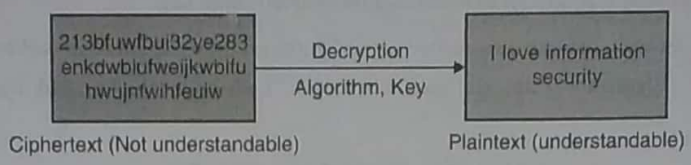


Fig. 1.9.3 : Decryption (or decipher)

Crypto Entities	Is Secret?	Is understandable?
Plaintext	No (But, need to)	Yes
Ciphertext	No	No
Key	Yes	No
Algorithm	No	Yes

Note : The rest of the sections in this chapter heavily build upon the concepts and the introduction you got. Please take some time to make yourself familiar and comfortable with the terms before reading further. Classical Encryption Techniques.

1.10 Classical Encryption Techniques

Q. What are traditional ciphers? **MU - May 19, 2 Marks**

Encryption typically involves two operations as shown in Fig. 1.10.1.

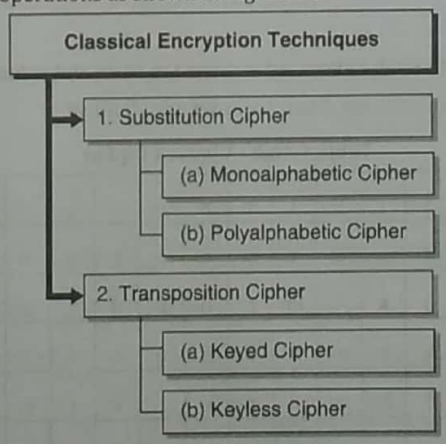


Fig. 1.10.1 : Classical Encryption Techniques

1.10.1 Substitution Cipher

Q. Define the following with example :
(i) Substitution cipher (ii) Polyalphabetic cipher **MU - Dec. 15, 4 Marks**

Q. With the help of examples compare and contrast mono-alphabetic ciphers and poly-alphabetic ciphers? **MU - Dec. 17, 5 Marks**

Q. Discuss any one substitution cipher with example. List their merits and demerits. **MU - May 19, 4 Marks**

Scanned with CamScanner

- In this operation, one character is replaced by another (like substitutes in games). For example, A can be substituted by D and B can be substituted by E and so on based on a chosen substitution key.
- Let's take an example. Assume that our substitution key is "shift next by 3". Recall from earlier discussion that the key is preserved secretly. Our algorithm (rule) is simple substitution. Let's apply the key and algorithm and encrypt some plaintext.

Table 1.10.1 : Simple Substitution Table

Plaintext	Ciphertext
I love cybersecurity	L oryhfbehuvhfxulwb
Apple	Dssoh
23456	56789

- The Table 1.10.1 is a simple substitution table where each character in plaintext is moved by 3 characters. Characters such as "y" when shifted, take the form of y→z, z→ a, a→b.
- The above example is a classical substitution cipher called **Caesar cipher** named after Julius Caesar. This type of substitution cipher is also referred to as a **"monoalphabetic substitution cipher"** because it uses only one character at a time.
- Another type of substitution cipher is called **"polyalphabetic substitution cipher".** In this, more than one alphabet is used at a time for encrypting plaintext. Let's learn a few polyalphabetic substitution ciphers.

1.10.1(A) Vignere Cipher

Table 1.10.2 is the Vignere table where alphabets are arranged in rows and columns. It is simple to draw. Just write a-z skipping one alphabet from left, at a time, in a row. First two rows are identical.

Table 1.10.2 : Vignere Cipher

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a
c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b
d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c
e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d
f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e
g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f
h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g
i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h
j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i
k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j

Scanned with CamScanner

l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k
m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l
n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m
o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n
p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q
s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r
t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s
u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t
v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u
w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v
x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x
z	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

Let's take an example:

Plaintext : Apple

Security Key : Snow

Algorithm : Vignere Cipher

Let's try to find out ciphertext given the information above. You will need to superimpose the security key above the plaintext you wish to encrypt. You would find the ciphertext value for each plaintext character at the intersection of column (from security key value) and row (from plaintext value).

Security Key →	s	n	o	w	s	n	o	w ← column
Plaintext →	a	p	p	l	e			← row

Ciphertext would be :

- First letter (ciphertext of plaintext a) → intersection of column s and row a →value is s
- Second letter (ciphertext of plaintext p) → intersection of column n and row p →value is c
- Third letter (ciphertext of plaintext p) → intersection of column o and row p → value is d
- Fourth letter (ciphertext of plaintext l) → intersection of column w and row l → value is h
- Fifth letter (ciphertext of plaintext e) → intersection of column s and row e → value is w

Scanned with CamScanner

Table 1.10.3 : Vignere Cipher Example

Plaintext \ Security Key	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
b	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a
c	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b
d	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c
e	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d
f	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e
g	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f
h	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g
i	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h
j	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i
k	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j
l	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k
m	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l
n	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m
o	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n
p	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
q	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
r	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q
s	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r
t	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s
u	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t
v	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u
w	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v
x	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
y	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x
z	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y

So, the ciphertext for plaintext "apple" with security key "snow" and using algorithm "Vignere cipher" is "*scdhw*".

1.10.1(B) Playfair Cipher

✍ ***Definition :*** *The Playfair cipher is a symmetric-key based encryption technique that uses digraph substitution cipher.*

Scanned with CamScanner

This technique encrypts the pairs of alphabets (digraphs), instead of single alphabets as in the case of simple substitution ciphers like Caesar cipher. The Playfair cipher is thus significantly harder to break. It involves 625 combinations of alphabet pairs instead of just 26 in the case of single alphabets. Hence, the regular cryptanalysis techniques such as the frequency analysis are harder to perform.

Algorithm

1. Start by creating a 5 x 5 key square by choosing a key and filling rest of the places by the remaining alphabets such that any alphabet occurs only once in the 5 x 5 square. 5 x 5 will cover up only 25 alphabets. Hence, i and j are combined and treated as 1 position. This would then cover all the 26 alphabets in the 5 x 5 square.
2. Take the plaintext and remove any punctuations, special characters, numbers, etc. such that only alphabets remain in the plaintext. Then make pairs of the alphabets in the plaintext. If you have just one alphabet left out, use X to make a pair. Any pairs that have the same alphabets are also replaced by a X.
3. Use the pairs in plaintext to substitute with the key square positions. Locate the alphabets in the key square and follow the substitution rules :
   (a) If the alphabets appear on the same row of the key square, replace them with the alphabets to their immediate right respectively (If the alphabet is in the rightmost corner, wrap around to take the leftmost alphabet of the row).
   (b) If the alphabets appear on the same column of the key square, replace them with the alphabets immediately below respectively (If the alphabet is at the bottom most position, wrap around to the take the topmost alphabet of the column).
   (c) If the alphabets are in different rows and columns, replace the pair with the alphabets on the same row respectively but at the corners of the rectangle defined by the original pair.

Ex. 1.10.1 : Use Playfair cipher to encrypt the word "greet" using the key "moon mission".

Soln. :

Step 1 : Construct the key square.

- Unique alphabets from the given key "moon mission" are → *m, o, n, i* and *s*. *i* and *j* are combined into one cell in the key square. Rest of the alphabets are filled serially such that the alphabets already in the key square (the key in the first row) are not repeated. This gives the following 5 x 5 key square.

m	**o**	**n**	**i/j**	**s**
a	b	c	d	e
f	g	h	k	l
p	q	r	t	u
v	w	x	y	z

Step 2: Arrange plaintext.

- The plaintext to encrypt is "greet". It is ordered as the following pairs.

  "gr", "ex", "et"
- The 2nd occurrence of *e* in *ee* is replaced with *x* to give "*ex*".

Scanned with CamScanner

Step 3 : Apply substitution based on the key square and the plaintext pairs.

"*gr*" is encrypted as following.

m	o	n	i/j	s
a	b	c	d	e
f	g	h	k	l
p	q	r	t	u
v	w	x	y	z

- The alphabets to encrypt are in different rows and hence they form a rectangle. You need to pick the corners.
- The alphabet in the corner of the rectangle of the same row as *g* is *h*.
- The alphabet in the corner of the rectangle of the same row as *r* is *q*.
- Hence, "*gr*" is encrypted as "*hq*".
- Now, pick the next plaintext pair "*ex*".

m	o	n	i/j	s
a	b	c	d	e
f	g	h	k	l
p	q	r	t	u
v	w	x	y	z

- The alphabets to encrypt are in different rows and hence they form a rectangle. You need to pick the corners.
- The alphabet in the corner of the rectangle of the same row as *e* is *c*.
- The alphabet in the corner of the rectangle of the same row as *x* is *z*.
- Hence, "*ex*" is encrypted as "*cz*".
- Now, pick the next plaintext pair "*et*".

m	o	n	i/j	s
a	b	c	d	e
f	g	h	k	l
p	q	r	t	u
v	w	x	y	z

- The alphabets to encrypt are in different rows and hence they form a rectangle. You need to pick the corners.
- The alphabet in the corner of the rectangle of the same row as *e* is *d*.
- The alphabet in the corner of the rectangle of the same row as *t* is *u*.
- Hence, "*et*" is encrypted as "*du*".
- Hence, the plaintext "*greet*" is encrypted as "*hqczdu*" using Playfair cipher using the key "*moon mission*".

Scanned with CamScanner

Ex. 1.10.2 : Use Playfair cipher to encrypt the plaintext "Why, don't you?" using the key "keyword".

Soln. :

Step 1 : Construct the key square.

k	e	y	w	o
r	d	a	b	c
f	g	h	i/j	l
m	n	p	q	s
t	u	v	x	z

Step 2 : Arrange the plaintext into pairs.

(remove all punctuations).

"wh", "yd", "on", "ty", "ou"

Step 3 : Apply substitution for each plaintext pair.

k	e	y	w	o
r	d	a	b	c
f	g	h	i/j	l
m	n	p	q	s
t	u	v	x	z

"wh" is *"yl"*.

k	e	y	w	o
r	d	a	b	c
f	g	h	i/j	l
m	n	p	q	s
t	u	v	x	z

"yd" is *"ea"*.

k	e	y	w	o
r	d	a	b	c
f	g	h	i/j	l
m	n	p	q	s
t	u	v	x	z

Scanned with CamScanner

"*on*" is "*es*".

k	e	y	w	o
r	d	a	b	c
f	g	h	i/j	l
m	n	p	q	s
t	u	v	x	z

"*ty*" is "*vk*".

k	e	y	w	o
r	d	a	b	c
f	g	h	i/j	l
m	n	p	q	s
t	u	v	x	z

"*ou*" is "*ez*".

Hence, "Why, don't you?" is encrypted as "yieaesvkez" using the key "keyword".

Ex. 1.10.3 : Encrypt "The key is hidden under the door" using Playfair cipher with keyword "domestic".

MU - Dec. 15, 5 Marks

Soln. :

Step 1 – Construct the 5 x 5 key square.

i and j are combined into one cell in the key square. Rest of the alphabets are filled serially such that the alphabets already in the key square (the key in the first row) are not repeated. This gives the following 5 x 5 key square.

d	o	m	e	s
t	i/j	c	a	b
f	g	h	k	l
n	p	q	r	u
v	w	x	y	z

Step 2 – Arrange the plaintext into pairs.

(remove all punctuations).

"th", "ek", "ey", "is", "hi", "dd", "en", "un", "de", "rt", "he", "do", "or"

Step 3 – Apply substitution for each plaintext pair.

"th" is encrypted as following.

d	o	m	e	s
t	i/j	c	a	b
f	g	h	k	l
n	p	q	r	u
v	w	x	y	z

Scanned with CamScanner

The alphabets to encrypt are in different rows and hence they form a rectangle. You need to pick the corners.

- The alphabet in the corner of the rectangle of the same row as *t* is *c*.
- The alphabet in the corner of the rectangle of the same row as *h* is *f*.

Hence, "th" is encrypted as "cf".

"ek" is encrypted as following.

d	o	m	e	s
t	i/j	c	a	b
f	g	h	k	l
n	p	q	r	u
v	w	x	y	z

The alphabets appear on the same column of the key square. You need to replace them with the alphabets immediately below respectively (If the alphabet is at the bottom most position, wrap around to the take the topmost alphabet of the column).

- The alphabet below *e* is *a*
- The alphabet below *k* is *r*

Hence, "*ek*" is encrypted as "*ar*".

"ey" is encrypted as following.

d	o	m	e	s
t	i/j	c	a	b
f	g	h	k	l
n	p	q	r	u
v	w	x	y	z

The alphabets appear on the same column of the key square. You need to replace them with the alphabets immediately below respectively (If the alphabet is at the bottom most position, wrap around to the take the topmost alphabet of the column).

- The alphabet below *e* is *a*
- The alphabet below *y* is *e*

Hence, "*ey*" is encrypted as "*ae*".

"is" is encrypted as following.

d	o	m	e	s
t	i/j	c	a	b
f	g	h	k	l
n	p	q	r	u
v	w	x	y	z

The alphabets to encrypt are in different rows and hence they form a rectangle. You need to pick the corners.

Scanned with CamScanner

- The alphabet in the corner of the rectangle of the same row as *i* is *b*.
- The alphabet in the corner of the rectangle of the same row as *s* is *o*.

Hence, "*is*" is encrypted as "*bo*".

"hi" is encrypted as following.

d	o	m	e	s
t	i/j	c	a	b
f	g	h	k	l
n	p	q	r	u
v	w	x	y	z

The alphabets to encrypt are in different rows and hence they form a rectangle. You need to pick the corners.

- The alphabet in the corner of the rectangle of the same row as *h* is *g*.
- The alphabet in the corner of the rectangle of the same row as *i* is *c*.

Hence, "*hi*" is encrypted as "gc".

"dd" is encrypted as "dx" as following.

d	o	m	e	s
t	i/j	c	a	b
f	g	h	k	l
n	p	q	r	u
v	w	x	y	z

The alphabets to encrypt are in different rows and hence they form a rectangle. You need to pick the corners.

- The alphabet in the corner of the rectangle of the same row as *d* is *m*
- The alphabet in the corner of the rectangle of the same row as *x* is *v*

Hence, "*dd*" is encrypted as "*mv*".

"en" is encrypted as following.

d	o	m	e	s
t	i/j	c	a	b
f	g	h	k	l
n	p	q	r	u
v	w	x	y	z

The alphabets to encrypt are in different rows and hence they form a rectangle. You need to pick the corners.

- The alphabet in the corner of the rectangle of the same row as *e* is *d*.
- The alphabet in the corner of the rectangle of the same row as *n* is *r*.

Scanned with CamScanner

Hence, "*en*" is encrypted as "*dr*".

"un" is encrypted as following.

d	o	m	e	s
t	i/j	c	a	b
f	g	h	k	l
n	p	q	r	u
v	w	x	y	z

The alphabets appear on the same row of the key square. You need to replace them with the alphabets to their immediate right respectively (If the alphabet is in the rightmost corner, wrap around to take the leftmost alphabet of the row).

- The alphabet to the right of *u* is *n*.
- The alphabet to the right of *n* is *p*.

Hence, "*un*" is encrypted as "np".

"de" is encrypted as following.

d	o	m	e	s
t	i/j	c	a	b
f	g	h	k	l
n	p	q	r	u
v	w	x	y	z

The alphabets appear on the same row of the key square. You need to replace them with the alphabets to their immediate right respectively (If the alphabet is in the rightmost corner, wrap around to take the leftmost alphabet of the row).

- The alphabet to the right of *d* is *o*.
- The alphabet to the right of *e* is *s*

Hence, "*de*" is encrypted as "*os*".

"rt" is encrypted as following.

d	o	m	e	s
t	i/j	c	a	b
f	g	h	k	l
n	p	q	r	u
v	w	x	y	z

Scanned with CamScanner

The alphabets to encrypt are in different rows and hence they form a rectangle. You need to pick the corners.

- The alphabet in the corner of the rectangle of the same row as *r* is *n*
- The alphabet in the corner of the rectangle of the same row as *t* is *a*

Hence, "*rt*" is encrypted as "*na*".

"he" is encrypted as following.

d	o	m	e	s
t	i/j	c	a	b
f	g	h	k	l
n	p	q	r	u
v	w	x	y	z

The alphabets to encrypt are in different rows and hence they form a rectangle. You need to pick the corners.

The alphabet in the corner of the rectangle of the same row as *h* is *k*

The alphabet in the corner of the rectangle of the same row as *e* is *m*

Hence, "*he*" is encrypted as "*km*".

"do" is encrypted as following.

d	o	m	e	s
t	i/j	c	a	b
f	g	h	k	l
n	p	q	r	u
v	w	x	y	z

The alphabets appear on the same row of the key square. You need to replace them with the alphabets to their immediate right respectively (If the alphabet is in the rightmost corner, wrap around to take the leftmost alphabet of the row).

The alphabet to the right of *d* is *o*

The alphabet to the right of *o* is *m*

Hence, "*do*" is encrypted as "*om*".

"or" is encrypted as following.

d	o	m	e	s
t	i/j	c	a	b
f	g	h	k	l
n	p	q	r	u
v	w	x	y	z

The alphabets to encrypt are in different rows and hence they form a rectangle. You need to pick the corners.

- The alphabet in the corner of the rectangle of the same row as *o* is *e*

Scanned with CamScanner

- The alphabet in the corner of the rectangle of the same row as *r* is *p*

Hence, "*or*" is encrypted as "*ep*".

Hence, the plaintext "The key is hidden under the door" using Playfair cipher with keyword "domestic" is encrypted as "cfarae bogc mv dr np osna km om ep".

1.10.1(C) Hill Cipher

Definition : *The Hill cipher is a polygraphic substitution cipher based on linear algebra.*

By polygraphic, we mean that it can work on substitution for up to 3-alphabets at a time. It arranges the key and the plaintext into a matrix format and their multiplication undergoes the mod 26 operation to find the resultant ciphertext.

Algorithm

1. Arrange the key and the plaintext in a matrix format. Use the Table 1.10.4 for assigning numbers to alphabets for matrix operations. Note that you need to create the plaintext matrix according to the given key matrix such that multiplication is possible. The number of columns in the key matrix must be equal to the number of rows in the plaintext matrix. If the plaintext is larger, break it up into multiple matrices and apply further steps on each of the plaintext matrix using the key. To fill the matrix, in case you are short of matrix elements, you may use zeroes.

Table 1.10.4 : Alphabet Positions

Alphabet	Number	Alphabet	Number
A	0	N	13
B	1	O	14
C	2	P	15
D	3	Q	16
E	4	R	17
F	5	S	18
G	6	T	19
H	7	U	20
I	8	V	21
J	9	W	22
K	10	X	23
L	11	Y	24
M	12	Z	25

2. Carry out multiplication of the key and the plaintext matrix.
3. Perform mod 26 operation on the resultant multiplication.
4. Use the Table 1.10.4 again to convert numbers back to alphabets. These alphabets represent the ciphertext.

Scanned with CamScanner

Ex. 1.10.4 : Encrypt the message "Exam" using the Hill cipher with the key $\begin{bmatrix} 9 & 4 \\ 5 & 7 \end{bmatrix}$.

Soln. :

Plaintext "*Exam*" when converted into a number matrix would be $\begin{bmatrix} 4 & 0 \\ 23 & 12 \end{bmatrix}$.

Multiply the Key and Plaintext matrices.

$$\begin{bmatrix} 9 & 4 \\ 5 & 7 \end{bmatrix} \times \begin{bmatrix} 4 & 0 \\ 23 & 12 \end{bmatrix} = \begin{bmatrix} 128 & 48 \\ 181 & 84 \end{bmatrix}$$

Perform mod 26 operation on the result.

$$\begin{bmatrix} 128 & 48 \\ 181 & 84 \end{bmatrix} \times \text{mod } 26 = \begin{bmatrix} 24 & 22 \\ 25 & 6 \end{bmatrix}$$

Converting the numbers from mod operation back to alphabets you get "*YZWG*". Hence, encrypting the plaintext "*Exam*" using the given key gives ciphertext "*YZWG*".

Ex. 1.10.5 : Encrypt the message "DEF" using the Hill cipher with the key $\begin{bmatrix} 2 & 4 & 5 \\ 9 & 2 & 1 \\ 3 & 8 & 7 \end{bmatrix}$.

Soln. :

Plaintext "*DEF*" when converted into a number matrix would be

Multiply the Key and Plaintext matrices.

$$\begin{bmatrix} 2 & 4 & 5 \\ 9 & 2 & 1 \\ 3 & 8 & 7 \end{bmatrix} \times \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix} = \begin{bmatrix} 47 \\ 40 \\ 76 \end{bmatrix}$$

Perform mod 26 operation on the result.

$$\begin{bmatrix} 47 \\ 40 \\ 76 \end{bmatrix} \times \text{mod } 26 = \begin{bmatrix} 21 \\ 14 \\ 24 \end{bmatrix}$$

Converting the numbers from mod operation back to alphabets you get "*VOY*". Hence, encrypting the plaintext "*DEF*" using the given key gives ciphertext "*VOY*".

1.10.1(D) Affine Cipher

✍ ***Definition :*** *Affine cipher is a type of monoalphabetic substitution cipher, wherein each letter in an alphabet is mapped to its numeric equivalent, encrypted using a simple mathematical function, and converted back to a letter.*

TechKnowledge Publications

Scanned with CamScanner

Each letter is encrypted using the Affine function *(Ax + B) mod 26*. Values (A, B) are called coefficients of Affine function. You use the position value of each alphabet and use the Affine function to calculate the position value of the corresponding encrypted letter. Then, you substitute each plaintext letter with the corresponding encrypted letter from the table.

Let's see an example.

Ex. 1.10.6 : Encrypt the plaintext message "SECURITY" using affine cipher with the key pair (3, 7). Decrypt to get back original plaintext. **MU - May 19, 10 Marks**

Soln. :

Here the coefficients of Affine function are (3, 7). So, A = 3 and B = 7.

Create a table of all alphabets using the formula *(Ax + B) mod 26*. Use A = 3 and B = 7 for this computation.

Position Value of Alphabet	Plaintext Alphabet	Affine Calculation (A*Position + B) mod 26	Encrypted Alphabet (Position from Affine calculation)
0	A	(3 * 0 + 7) mod 26 = 7	H
1	B	(3 * 1 + 7) mod 26 = 10	K
2	C	(3 * 2 + 7) mod 26 = 13	N
3	D	(3 * 3 + 7) mod 26 = 16	Q
4	E	(3 * 4 + 7) mod 26 = 19	T
5	F	(3 * 5 + 7) mod 26 = 22	W
6	G	(3 * 6 + 7) mod 26 = 25	Z
7	H	(3 * 7 + 7) mod 26 = 2	C
8	I	(3 * 8 + 7) mod 26 = 5	F
9	J	(3 * 9 + 7) mod 26 = 8	I
10	K	(3 * 10 + 7) mod 26 = 11	L
11	L	(3 * 11 + 7) mod 26 = 14	O
12	M	(3 * 12 + 7) mod 26 = 17	R
13	N	(3 * 13 + 7) mod 26 = 20	U
14	O	(3 * 14 + 7) mod 26 = 23	X
15	P	(3 * 15 + 7) mod 26 = 0	A
16	Q	(3 * 16 + 7) mod 26 = 3	D
17	R	(3 * 17 + 7) mod 26 = 6	G
18	S	(3 * 18 + 7) mod 26 = 9	J
19	T	(3 * 19 + 7) mod 26 = 12	M
20	U	(3 * 20 + 7) mod 26 = 15	P
21	V	(3 * 21 + 7) mod 26 = 18	S
22	W	(3 * 22 + 7) mod 26 = 21	V
23	X	(3 * 23 + 7) mod 26 = 24	Y
24	Y	(3 * 24 + 7) mod 26 = 1	B
25	Z	(3 * 25 + 7) mod 26 = 4	E

Scanned with CamScanner

Once the table is ready, you can substitute each letter in the plaintext with its corresponding encrypted letter.

Position Value of Alphabet	Plaintext Alphabet	Affine Calculation (A*Position + B) mod 26	Encrypted Alphabet (Position from Affine calculation)
0	A	(3 * 0 + 7) mod 26 = 7	H
1	B	(3 * 1 + 7) mod 26 = 10	K
2	C	(3 * 2 + 7) mod 26 = 13	N
3	D	(3 * 3 + 7) mod 26 = 16	Q
4	E	(3 * 4 + 7) mod 26 = 19	T
5	F	(3 * 5 + 7) mod 26 = 22	W
6	G	(3 * 6 + 7) mod 26 = 25	Z
7	H	(3 * 7 + 7) mod 26 = 2	C
8	I	(3 * 8 + 7) mod 26 = 5	F
9	J	(3 * 9 + 7) mod 26 = 8	I
10	K	(3 * 10 + 7) mod 26 = 11	L
11	L	(3 * 11 + 7) mod 26 = 14	O
12	M	(3 * 12 + 7) mod 26 = 17	R
13	N	(3 * 13 + 7) mod 26 = 20	U
14	O	(3 * 14 + 7) mod 26 = 23	X
15	P	(3 * 15 + 7) mod 26 = 0	A
16	Q	(3 * 16 + 7) mod 26 = 3	D
17	R	(3 * 17 + 7) mod 26 = 6	G
18	S	(3 * 18 + 7) mod 26 = 9	J
19	T	(3 * 19 + 7) mod 26 = 12	M
20	U	(3 * 20 + 7) mod 26 = 15	P
21	V	(3 * 21 + 7) mod 26 = 18	S
22	W	(3 * 22 + 7) mod 26 = 21	V
23	X	(3 * 23 + 7) mod 26 = 24	Y
24	Y	(3 * 24 + 7) mod 26 = 1	B
25	Z	(3 * 25 + 7) mod 26 = 4	E

(Copyright No. - L82548/2019)

Scanned with CamScanner

So,

$S \rightarrow J$

$E \rightarrow T$

$C \rightarrow N$

$U \rightarrow P$

$R \rightarrow G$

$I \rightarrow F$

$T \rightarrow M$

$Y \rightarrow B$

So, the plaintext SECURITY, when encrypted using Affine cipher using the Affine coefficients of (3, 7), give JTNPGFMB as the encrypted text.

Note here that if you have to decrypt the ciphertext using Affine cipher, you follow the same approach. You first create the table using the Affine coefficients and then substitute the encrypted letters with their corresponding plaintext letters.

Difference between Monoalphabetic and Polyalphabetic Ciphers

Note a key difference between monoalphabetic cipher and polyalphabetic cipher;

- For repeated characters, in monoalphabetic cipher, ciphertext is same (for example, plaintext y is ciphertext b as per example we chose earlier) whereas;
- For polyalphabetic cipher, repeated plaintext characters need not lead to the same ciphertext (for example, there are two instances of p in plaintext word apple in the polyalphabetic cipher example. One is encrypted as c and another is encrypted as d).

So, polyalphabetic ciphers are stronger than monoalphabetic since they usually give different ciphertext values for repeated characters in plaintext and hence are less prone to frequency analysis attack. Frequency analysis attack tries to find a correlation between plaintext and ciphertext and determine the security key. For example, the attacker might guess that y is encrypted as b in the monoalphabetic cipher, so it could mean the security key is "shift next by 3". Once the attacker determines the key, converting any ciphertext back to plaintext is a trivial (very simple) task.

For Cryptography to be successful, keeping the key secret is very important.

1.10.2 Transposition Techniques

Q. Explain with examples, keyed and keyless transposition ciphers.	MU - May 16, 5 Marks
Q. Discuss any one transposition cipher with example. List their merits and demerits.	MU - May 19, 4 Marks

In this operation, the position of characters is jumbled up (mixed up) like a letter arranging game. For example, the plaintext "apple" could be transposed into ciphertext as "elpap". Note that all the characters in plaintext are also present in the ciphertext but at different positions. For example, position of "a" in plaintext is 1 whereas in ciphertext it is 4. Note that it is a very simplistic example. Various complex mathematical transposition algorithms are usually used in cryptography. Transposition can be carried out using two techniques – Keyed and Keyless. Let us learn about them.

Scanned with CamScanner

1.10.2(A) Keyed Transposition Cipher

Definition : *In keyed transposition, a random key is used to describe the transposition sequence and carry out the transposition.*

This is also called Columnar Transposition Cipher.

Algorithm

1. Arrange the plaintext in a column under the given key.
2. Rearrange the plaintext column-wise in key's alphabetic order.

Ex. 1.10.7 : Use the key "ENCRYPT" to encrypt the plaintext "Save the king from attack" using transposition cipher.

Soln. : Draw a table and arrange the key and the plaintext under the key column-wise. Mark the alphabets in the key in their order alphabetically. For example, for the given key, the alphabet "*C*" comes first in the order of 26 alphabets. Then comes "*E*" and hence marked 2. Likewise mark all the alphabets in the key according to their occurrence in the alphabets.

E	N	C	R	Y	P	T
2	3	1	5	7	4	6
s	a	v	e	t	h	e
k	i	n	g	f	r	o
m	a	t	t	a	c	k

Read the columns in order.

- Take column C marked as 1st in order : vnt
- Take column E marked as 2nd in order : skm
- Take column N marked as 3rd in order : aia

Follow likewise to get the ciphertext as "*vntskmaiahrcegteoktfa*".

Ex. 1.10.8 : Use the key "SORROW" to encrypt the plaintext "Demonetization tonight" using transposition cipher.

Soln. :

Draw a table and arrange the key and the plaintext as following. Mark the order of the columns from left to right in case of repeated key characters. Pad the columns with "x" to fill the columns if the plaintext does not fill the table completely.

S	O	R	R	O	W
5	1	3	4	2	6
d	e	m	o	n	e
t	i	z	a	t	i
o	n	t	o	n	i
g	h	t	x	x	x

The resulting ciphertext is "*eimhntnxmzttoaoxdtogeiix*".

Scanned with CamScanner

1.10.2(B) Keyless Transposition Cipher

✍ ***Definition :*** *In keyless transposition, a transposition sequence is described without a random key.*

One such example of Keyless Transposition Cipher is Railfence Cipher. Railfence cipher uses the rail size as a key and does not use a random key as such. It can be easily broken.

Algorithm

1. Based on the rail size, arrange the plaintext.
2. Rearrange the plaintext row-wise to get the ciphertext.

Ex. 1.10.9 : Encrypt the plaintext "Save the king from attack" using Railfence cipher. Assume a suitable rail size.

Soln. :

Assuming a rail size of 3. All it means is that it would have 3 rows. Arrange the plaintext in rails one alphabet at a time.

rail 1 →	s				t				i				r				t				k
rail 2 →		a		e		h		k		n		f		o		a		t		c	
rail 3 →			v				e				g				m				a		

Rearrange the plaintext rail-wise (row-wise) to get the ciphertext. Start from rail 1, then rail 2 and finally rail 3. Here, the ciphertext would be "*stirtkaehknfoatcvegma*".

Ex. 10.7.10 : Encrypt the plaintext "Demonetization tonight" using Railfence cipher. Assume the rail size of 4.

Soln. : Arrange the plaintext in 4 rails (rows).

rail 1 →	d						t						o						g		
rail 2 →		e				e		i				i		n				i		h	
rail 3 →			m		n				z		t				t		n				t
rail 4 →				o						a						o					

The resulting ciphertext is "*dtogeeiinihmnzttntoao*".

1.11 Methods of Encryption

As you know, encryption is primarily driven by two components – Keys and Algorithms. Based on the number of keys used in the encryption and decryption process, the encryption methods can be classified as,

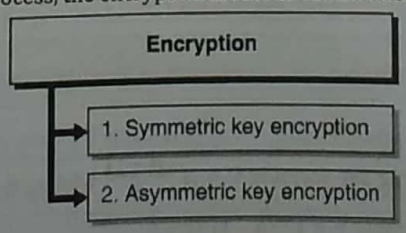


Fig. 1.11.1 : Encryption

Scanned with CamScanner

1.11.1 Symmetric Key Encryption

Symmetric means same.

Definition : *In Symmetric Key Encryption, the key used for encryption is same as the key used for decryption.*

The keys are identical. The sender as well as the receiver must have exactly the same key to encrypt or decrypt. As an example, symmetric key is like your regular lock key. The same key can be used for locking the door as well as unlocking the door.

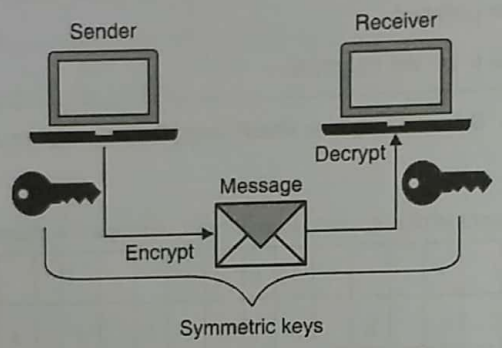


Fig. 1.11.2 : Symmetric Key Encryption

As you understand, a symmetric key is unique between a sender and a receiver. If there are more entities involved and each require to secretly communicate with the another, you end up having multiple keys. Let's take an example, suppose there are 4 friends – A, B, C and D and each one of them require communicating with one another secretly. It is obvious that you cannot share the keys between a pair of friends with another pair of friends. So, if A and B share a key, B and C cannot share the same key because C would also know the secret key between A and B and can then decrypt communication between A and B. So, you would require several keys to ensure that each pair of sender and receiver have a unique key. So, how many keys would you need?

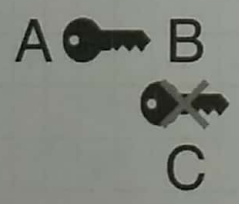


Fig. 1.11.3 : Symmetric keys between a pair should not be shared

You would need below keys (one for each pair of sender and receiver) :

1. A <> B
2. A <> C
3. A <> D
4. B <> C
5. B <> D
6. C <> D

This can be effectively calculated using the formula $K = N \frac{(N-1)}{2}$ where, N is the number of entities that need to secretly communicate. So, in the above example, K = 4 * (4 – 1) /2 = 6.

Scanned with CamScanner

If we have 100 entities, it would require 100 * (100 – 1) /2 = 4,950 keys! What if the entire world wants to communicate with each other? Can you imagine? I will come back to it later on and answer that for you.

Another problem here is how does A send the key she is using to B? If the sender and receiver have to use the same key, there should be a way to securely transfer the key. Isn't it? For example, if you have to give your house keys to your friend, you probably exchange hands in person. You don't leave the key somewhere that could potentially be picked / looked by someone else other than your friend. I will answer this question as well later on.

Some of the examples of symmetric key algorithm are DES, AES and Blowfish.

Advantages of Symmetric Keys

1. Computationally faster than the asymmetric keys.
2. Hard to break if the key used is long.

Disadvantages of Symmetric Keys

1. Requires a secure mechanism to exchange keys.
2. Each pair of sender and receiver require a unique key.
3. Provides only confidentiality but not authenticity and non-repudiation.

1.11.2 Asymmetric Key Encryption

Unlike symmetric keys,

✍ ***Definition :*** *In Asymmetric Key Encryption, there are two keys that are mathematically related. If one is used for encryption, then only the corresponding other key can be used for decryption.*

Asymmetric means not equal. In the asymmetric system, two mathematically related keys work as key pair. If you use one key for encryption, then you need the other key in the pair for decryption. The encrypting key cannot be used for decrypting.

Note : Asymmetric Keys based Cryptography is also called as Public Key Cryptography.

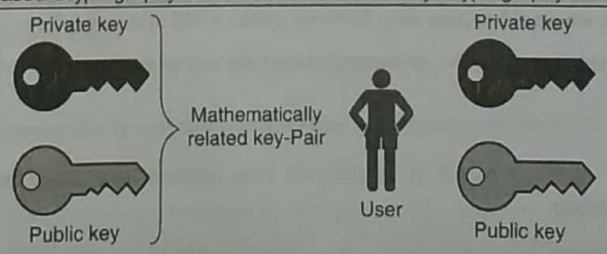


Fig. 1.11.4 : Asymmetric Key Encryption

Let's call one of the keys as the Public Key and its counterpart in the pair as the Private Key. A user owns both the keys. The public key is known to the world while the private key is known only to the user.

Key Used	Corresponding Key Required	Security Service Provided
Encryption – Public Key	Decryption – Private Key	Confidentiality
Private Key	Public Key	Authentication and Non-repudiation

Let us understand these two use cases of the asymmetric keys.

Scanned with CamScanner

Use Case 1 - User A wants to send a secret message to User B

- User A knows : User A's Public Key, User A's Private Key, User B's Public Key.
- User B knows : User A's Public Key, User B's Public Key, User B's Private Key.

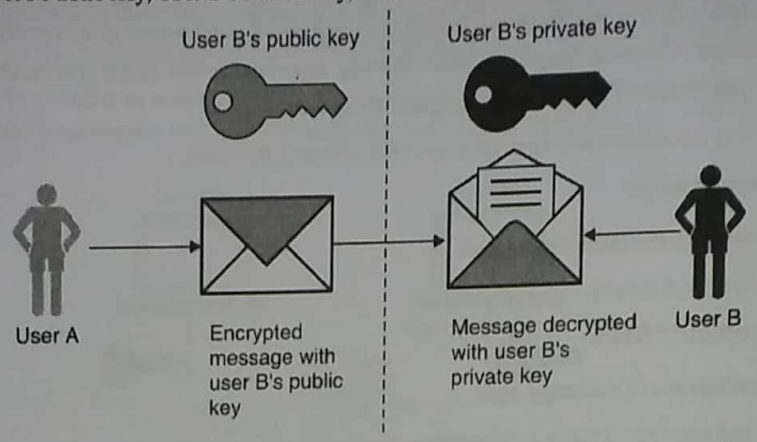


Fig. 1.11.5 : Encryption and Decryption using Public Key Cryptography

- User A wants to send an encrypted message such that only User B can read it. User A encrypts the message with User B's Public Key. Now, because User A used User B's Public Key, she is sure that only User B can decrypt the message as decryption would require the corresponding Private Key and only User B knows about her Private Key.
- Hence, in this scenario you find that asymmetric keys as well can be used to send encrypted messages as you saw in the case of symmetric keys. One core difference to note here is that User A need not know User B's Private Key to send her an encrypted message. A separate key distribution problem does not exist as Public Keys are known to the world and only the user needs to know and protect her Private Keys.
- You also see that you require only two keys per entity for encrypted communication. So, for 100 people to send each other encrypted messages, we would require only 200 keys unlike 4,095 symmetric keys (recall from our discussion on symmetric keys in the previous section)! So, asymmetric keys help you to overcome two of the limitations of symmetric keys :
  1. Key distribution
  2. Number of keys to manage
- Hence, I answer the question for you that I asked in the previous section – how do we manage keys if the whole world wants to communicate with each other? The answer is using asymmetric keys!

Use Case 2 - Proving authenticity and non-repudiation

- The second use case of asymmetric keys is for proving authenticity of an entity. This is highly used today for server validation. Have you seen "https" in front of website address? That is one of the examples of this use case.
- Suppose, you want to do an online transaction. How do you ensure that the bank's website address you are interacting with is the right one? That's precisely what asymmetric keys help you solve as well.

Scanned with CamScanner

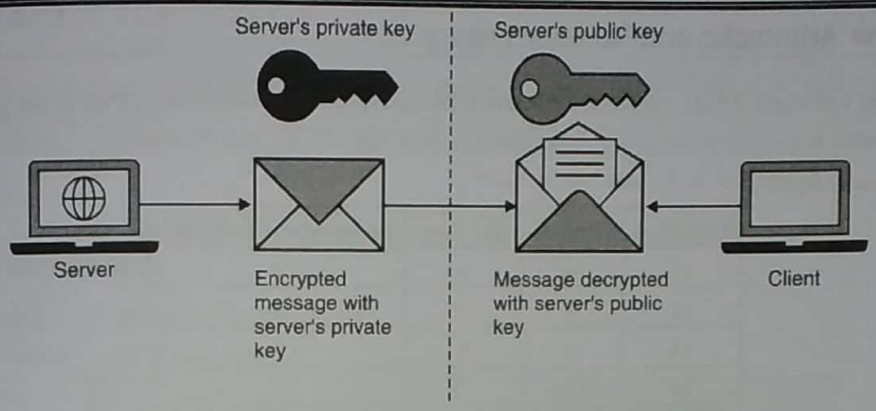


Fig. 1.11.6 : Non-repudiation using Public Key Cryptography

- Client wants to authenticate the server before it begins the transaction. It sends the server a plaintext message and asks it to encrypt it with its Private Key. The server encrypts the message that the client sent with its Private Key and sends it back to the client.
- Client uses the world-known Public Key of the server and decrypts the message it received from the server. If the message gets successfully decrypted and it matches with what the client sent earlier to the server to encrypt, the client has now validated that it is indeed interacting with the authentic server because except the authentic server, no one else would have known server's Private Key. The client is satisfied, and it begins the secure transaction after having established the server's authenticity.
- Hence, you find that asymmetric keys could effectively be used for authentication and non-repudiation. Some examples of algorithms that use asymmetric keys are RSA, ECC, Diffie-Hellman, and El Gamal.

Advantages of Asymmetric Keys

1. Easy key distribution.
2. Less number of keys to manage.
3. Can also be used for providing authentication and non-repudiation.

Disadvantages of Asymmetric Keys

1. Slower than symmetric keys.
2. Requires significant CPU power due to complex mathematical relation between the keys.

1.11.3 Comparison between Symmetric and Asymmetric Keys

Comparison Attribute	Symmetric Keys	Asymmetric Keys
Speed	High	Low
Complexity	Low	High
Number of keys	High	Low
Key Distribution	Problematic	Easier
Security Services	Confidentiality	Confidentiality, Authenticity, Non-repudiation

Scanned with CamScanner

1.12 Modular Arithmetic and Number Theory

Definition : *The modular arithmetic deals with operations on integers specifically around remainders from division.*

Let's take a few examples.

Dividend	Divisor	Quotient	Remainder (Modulus)
15	5	3	0
15	4	3	3
15	3	5	0
15	2	7	1
15	1	15	0

So, for example, number 15 when divided by 2, gives you Quotient of 7 and Remainder of 1. This can be mathematically written as 15 mod 2.

Note : Before you proceed, try finding out a few mods (remainders from division) for numbers of your choice.

1. Congruence Property

- Two numbers are said to be in congruence modulo, if they give out the same mod.

  For example,

  15 mod 2 = 1

  17 mod 2 = 1

- Hence, 15 is congruent to 17 *modulo* 2 i.e. 15 and 17 when undergo mod operation with 2, they both give same result of 1. They can be mathematically denoted as,

  $$15 \equiv 17 \pmod 2$$

2. Modular Addition

(A + B) mod C = (A mod C + B mod C) mod C

Let, A = 12, B = 15 and C = 5

Left Hand Side	Right Hand Side
= (12 + 15) mod 5 = 27 mod 5 = 2	= (12 mod 5 + 15 mod 5) mod 5 = (2 + 0) mod 5 = 2 mod 5 = 2

3. Modular operation on Negative numbers

- If you come across modular operation on a negative number, make the number positive by repetitively adding mod until it becomes positive.
- **For example :** If you have to find −13 mod 5, keeping adding 5 to −13 until you get a positive number. So,

  −13 + 5 = −8 + 5 = −3 + 5 = 2.

  Now, do mod on the positive number you got. In this case, −13 mod 5 becomes 2 mod 5. Hence, the answer is 2.

4. Modular Subtraction

(A − B) mod C = (A mod C − B mod C) mod C

Scanned with CamScanner

Let, A = 12, B = 15 and C = 5

Left Hand Side	Right Hand Side
= (12 – 15) mod 5 = –3 mod 5 = 2	= (12 mod 5 – 15 mod 5) mod 5 = (2 – 0) mod 5 = 2 mod 5 = 2

5. Modular Multiplication

(A * B) mod C = (A mod C * B mod C) mod C

Let, A = 12, B = 15 and C = 5

Left Hand Side	Right Hand Side
= (12 * 15) mod 5 = 180 mod 5 = 0	= (12 mod 5 * 15 mod 5) mod 5 = (2 * 0) mod 5 = 0 mod 5 = 0

6. Modular Inverse

- Modular arithmetic does not have division operation. However, it has inverse. Inverse of a general number is 1 divided by that number.
- **For example :** Inverse of 2 is ½. In other words, a number when multiplied by its inverse would give 1. So, 2 multiplied by its inverse ½ would give 2 * ½ = 1.
- So, modular inverse of A mod C is the value of B such that when A is multiplied by B and the mod C operation is carried out, it gives 1. Mathematically, it can be written as,

$$A * B \equiv 1 \pmod{C}$$

Note : To understand the above, refer to the congruency equation. mod C operation on A*B should be same as mod C operation on 1.

Let's take an example.

Suppose, you have to find modular inverse of 12 mod 5. Here, A = 12 and C = 5. Let's assume value of B from 0 onwards until we find (A*B) mod C = 1.

Value of B	Operation	Result
0	(12 * 0) mod 5	0
1	(12 * 1) mod 5	2
2	(12 * 2) mod 5	4
3	**(12 * 3) mod 5**	**1**

Hence, modular inverse of 12 mod 5 is 3.

7. Prime Numbers

- These are whole numbers which are greater than 1 and are only divisible by 1 and itself.
- **For example :** 2, 3, 5, 7, 11 and various others.

Scanned with CamScanner

8. Coprime Numbers

- Two integers a and b are said to be relatively prime, mutually prime, or coprime (also written co-prime) if the only positive integer (factor) that divides both of them is 1.
- **For example :** 5 and 7 are coprime because their common factor is only 1 whereas 14 and 18 are not coprime because their common factor can also be 2.

9. Discrete Logarithm

- If a is an arbitrary integer relatively prime to n and g is a primitive root of n, then there exists exactly one number μ such that $a = g^{\mu} \pmod n$.
- The number μ is then called the discrete logarithm of a with respect to the base g modulo n and is denoted

$$\mu = \text{ind}_g a \pmod n.$$

- Discrete logarithms are quickly computable in a few special cases. However, no efficient method is known for computing them in general. Several important algorithms in public-key cryptography use discrete logarithms.

1.13 Greatest Common Divisor (GCD)

Greatest Common Divisor (GCD) of two positive integers is the largest integer that can fully divide both the integers. For example, GCD (5, 10) is 5. GCD of (11, 13) is 1. GCD is usually found out by finding the factors of the respective integers and then choosing the common highest factor.

For example, to find GCD (24, 70)

- Factors of 24 = 2 * 2 * 2 * 3 → Factors could be 2, 3, 4, 6, 8, 12, 24
- Factors of 70 = 2 * 5 * 7 → Factors are only 2, 5, 7

Hence, the largest common factor is 2. Hence, GCD (24, 70) = 2.

1.13.1 Euclid's or Euclidean Algorithm

Finding GCD for smaller numbers is quite straight forward. But, when it comes to finding GCD of large numbers, it might be a complex task. This is precisely where Euclid's (or Euclidean) algorithm helps.

Euclid's algorithm states that,

$$\gcd(a,b) = \gcd(a \bmod b, b) \quad \text{if } a > b$$

$$\gcd(a,b) = \gcd(a, b \bmod a) \quad \text{if } b > a$$

Ex. 1.13.1 : Find gcd(50, 65) using Euclidean algorithm.

Soln. :

$$\begin{aligned}
\gcd(50,65) &= \gcd(50, 65 \bmod 50) \quad \text{[because 65 is greater than 50]} \\
&= \gcd(50,15) \\
&= \gcd(50 \bmod 15, 15) \quad \text{[because 50 is greater than 15]} \\
&= \gcd(5,15) \\
&= \gcd(5, 15 \bmod 5) \quad \text{[because 15 is greater than 5]} \\
&= \gcd(5,0) \quad \text{[stop here once the mod of a term becomes 0]} \\
&= 5 \quad \text{[is the gcd(50,65)]}
\end{aligned}$$

Scanned with CamScanner

Ex. 1.13.2 : Find gcd(464, 238) using Euclidean algorithm.

Soln. :

$$\begin{aligned}
\gcd(464,238) &= \gcd(464 \bmod 238, 238) \\
&= \gcd(226,238) \\
&= \gcd(226, 238 \bmod 226) \\
&= \gcd(226,12) \\
&= \gcd(226 \bmod 12, 12) \\
&= \gcd(10,12) \\
&= \gcd(10, 12 \bmod 10) \\
&= \gcd(10,2) \\
&= \gcd(10 \bmod 2, 2) \\
&= \gcd(0,2) \\
&= 2
\end{aligned}$$

Ex. 1.13.3 : Find gcd(105, 80).

Soln. :

$$\begin{aligned}
\gcd(105,80) &= \gcd(105 \bmod 80, 80) \\
&= \gcd(25, 80 \bmod 25) \\
&= \gcd(25 \bmod 5, 5) \\
&= \gcd(0,5) \\
&= 5
\end{aligned}$$

1.13.2 Extended Euclidean Algorithm

The Extended Euclidean Algorithm can be used to find the gcd of two numbers, and also to simultaneously express the gcd as a linear combination of these numbers. It helps to find values of coefficients x and y such that to satisfy the following equation :

$$ax + by = \gcd(a,b)$$

In the extended algorithm, the computation involves several entities. First, let's define them:

I = index : This would just be used to iterate (repeat) the process.

Note : If b is greater than a, then assume a = b and b = a. Swap the values to avoid negatives.

- **r = remainder :** Temporary placeholder variable for a and b.

  r would be calculated as $r_{i+1} = r_{i-1} - q_i r_i$

- q would be calculated as $q_{i+1} = r_{i-1} / r_i$
- s = Temporary placeholder for values while deriving coefficient x.

  s would be calculated as $s_{i+1} = s_{i-1} - q_i s_i$

- t = Temporary place holder for values while deriving coefficient y.

  t would be calculated as $t_{i+1} = t_{i-1} - q_i t_i$

- x would be $s_{i+1} = \dfrac{b}{\gcd(a, b)}$

  y would be $t_{i+1} = \dfrac{a}{\gcd(a, b)}$

Scanned with CamScanner

Ex. 1.13.4 : For a = 161 and b = 42, calculate gcd(a, b) and also the values of x and y to satisfy the extended Euclidean algorithm.

Soln.:

$r_0 = 161$ and $r_1 = 42$

i starts with 0

First draw the initial table.

Index i	quotient q for i	Remainder r for i	S for i	t for i
0		161	1	0
1		42	0	1

Start steps.

Index i	quotient q for i	Remainder r for i	S for i	t for i
0		161	1	0
1		42	0	1
2	**161 / 42 = 3**	**161 – 3 * 42 = 35**	**1 – 3 * 0 = 1**	**0 – 3 * 1 = –3**

For the highlighted row,

- i = 1 (we are just doing the calculation for 2nd row, hence the value of i is still 1)
- q_1 can we written as q_i where i = 1
  - So, $q_1 = r_{i-1} / r_i = r_0 / r_1 = 161 / 42 = 3$
- r_2 can be written as r_{i+1} where i = 1
  - So, $r_2 = r_{i-1} - q_i r_i$ which means $r_2 = r_0 - q_1 * r_1$
  - $r_2 = 161 - 3 * 42$

    $r_{i-1} = r_0$ which is 161

    $r_i = r_1$ which is 42
- Similarly, s_2 can be written as s_{i+1} where i = 1
  - So, $s_2 = s_{i-1} - q_i s_i = s_0 - q_1 s_1 = 1 - 3*0 = 1$
- Similarly, t_2 can be written as t_{i+1} where i = 1
  - So, $t_2 = t_{i-1} - q_i t_i = t_0 - q_1 t_1 = 0 - 3*1 = -3$
- Similarly proceed to the next step.

Index i	quotient q for i	Remainder r for i	S for i	t for i
0		161	1	0
1		42	0	1
2	161 / 42 = 3	161 – 3 * 42 = 35	1 – 3 * 0 = 1	0 – 3 * 1 = –3
3	**42 / 35 = 1**	**42 – 1 * 35 = 7**	**0 – 1 * 1 = –1**	**1 – 1 * –3 = 4**

Here again :

Scanned with CamScanner

$q_i = q_2$ = where i = 2

- So, $q_2 = r_{i-1} / r_i = r_1 / r_2 = 42 / 35 = 1$

$r_3 = r_1 - q_2 * r_2 = 42 - 1 * 35 = 7$

$s_3 = s_1 - q_2 * s_2 = 0 - 1 * 1 = -1$

$t_3 = t_1 - q_2 * t_2 = 1 - 1 * -3 = 4$

Index i	quotient q for i	Remainder r for i	S for i	t for i
0		161	1	0
1		42	0	1
2	161 / 42 = 3	161 – 3 * 42 = 35	1 – 3 * 0 = 1	0 – 3 * 1 = –3
3	42 / 35 = 1	42 – 1 * 35 = 7	0 – 1 * 1 = –1	1 – 1 * –3 = 4
4	35 / 7 = 5	35 – 5 * 7 = 0	Do not calculate	Do not calculate

$q_i = q_3$ where i = 3

- So, $q_3 = r_2 / r_3 = 35 / 7 = 5$

$r_4 = r_2 - q_3 * r_3 = 35 - 5 * 7 = 0$

- Do not calculate s and t once you get r = 0
- Last calculated s and t become x and y respectively
- Last calculated r becomes gcd

So, according to extended Euclidean algorithm, for numbers 161 and 42

gcd(161, 42) = 7

In the equation ax + by = gcd(161,42)

x = –1

y = 4

You can verify the answer by putting the values in the equation.

Left-hand side:

$$\begin{aligned} ax + by &= 161*-1 - 42*4 \\ &= -161 + 168 \\ &= 7 \text{ [which is equal to gcd(161,42)]} \end{aligned}$$

Ex. 1.13.5 : For a = 256 and b = 5004, calculate gcd(a, b) and also the values of x and y to satisfy the extended Euclidean algorithm.

Soln. :

First note that b > a. Hence, lets swap the values to make the calculations simple. We would re-swap them in the final answer.

So, assume a = 5004 and b = 256

Scanned with CamScanner

Index i	quotient q for i	Remainder r for i	S for i	t for i
0		5004	1	0
1		256	0	1

Index i	quotient q for i	Remainder r for i	s for i	t for i
0		5004	1	0
1		256	0	1
2	5004 / 256 = 19	5004 – 19 * 256 = 140	1 – 19*0 = 1	0 – 19*1 = –19
3	256 / 140 = 1	256 – 1 * 140 = 116	0 – 1*1 = –1	1 – 1*–19 = 20
4	140 / 116 = 1	140 – 1 * 116 = 24	1 – 1*1 = 2	–19 – 1*20 = –39
5	116 / 24 = 4	116 – 24 * 4 = 20	–1 – 4*2 = –9	20 – 39*4 = 176
6	24 / 20 = 1	24 – 20 * 1 = **4**	2 – 9*1 = **11**	–39 – 1*176 = **–215**
7	20 / 4 = 5	20 – 4* 5 = 0	Do not calculate	Do not calculate

gcd(256, 5004) = 4

We originally swapped the value. So, re-swap it.

Hence, x = –215 and y = 11

Putting it in the equation, you get,

Left-Hand Side :

$$ax + by = 256 * -215 + 5004 * 11$$
$$= -55{,}040 + 55{,}044$$
$$= 4 \text{ [which is equal to right hand side = gcd(256,5004)]}$$

Ex. 1.13.6 : For a = 86 and b = 14, calculate gcd(a, b) and also the values of x and y to satisfy the extended Euclidean algorithm.

Soln. :

Index i	quotient q for i	Remainder r for i	s for i	t for i
0		86	1	0
1		14	0	1
2	$\frac{86}{14} = \mathbf{6}$	86 – 6*14 = **2**	1 – 6*0 = **1**	0 – 6*1 = **–6**
3	$\frac{14}{2} = 7$	14 – 2*7 = 0	Do not calculate	Do not calculate

gcd(86,14) = 2

x = 1, y = –6

To verify, let's put the above values in the equation:

$$ax + by = 86*1 - 14*6$$
$$= 86 - 84$$
$$= 2 \text{ [this is the gcd value that we got for 86,14]}$$

Scanned with CamScanner

Ex. 1.13.7 : For a = 999 and b = 9, calculate gcd(a, b) and also the values of x and y to satisfy the extended Euclidean algorithm.

Soln. :

Index i	quotient q for i	Remainder r for i	s for i	t for i
0		999	1	0
1		**9**	**0**	**1**
2	999 / 9 = 111	999 – 111*9 = 0	Do not calculate	Do not calculate

gcd(999,9) = 9

x = 0, y = 1

Let's put those values in the equation:

ax + by= 999 * 0 + 1*9

= 9 [which matches the gcd value we got for 999,9]

✍ ***Exam Tip :*** *It would perhaps be easy for you to calculate the first two columns q and r until you get 0 in r. That way you are focusing on one set of calculation at a time. Once you have q and r computed in the first two columns, calculate s and t by just substitution the values of q and r in the respective equations.*

1.13.3 Multiplicative Inverse using Extended Euclidean Algorithm

If you recall our discussion from the previous section on modular inverse, you understand what inverse operation is. One application of the extended Euclidean Algorithm is to find out multiplicative inverse.

✍ ***Definition :*** *A modular multiplicative inverse of an integer a is an integer x such that the product ax is congruent to 1 with respect to the modulus m.*

In modular arithmetic, it can be written as $ax \equiv 1 \pmod m$

According to the extended Euclidean Algorithm,

$$ax + my = \gcd(a,m) = 1$$

$$ax + my = 1$$

$$ax - 1 = (-y)m$$

Dividing both sides by (mod m)

$$ax \pmod m - 1 \pmod m = (-y)m \pmod m$$

$$ax \pmod m - 1 \pmod m = 0$$

$$ax \equiv 1 \pmod m$$

Note : The multiplicative inverse of a modulo m exists if and only if a and m are coprime (i.e., if gcd(a, m) = 1)

So, if you come across a question where it is asked to calculate multiplicative inverse such that gcd(a, m) is not 1, do not attempt to solve the problem. Just calculate the gcd and show that the numbers are not coprime and hence the multiplicative inverse does not exist.

Scanned with CamScanner

Ex. 1.13.8 : Find multiplicate inverse of 24140 mod 40902.

Soln. :

$$
\begin{aligned}
\gcd(24140, 40902) &= \gcd(24140, 40902 \bmod 24140) \\
&= \gcd(24140, 16762) \\
&= \gcd(24140 \bmod 16762, 16762) \\
&= \gcd(7378, 16762) \\
&= \gcd(7378, 16762 \bmod 7378) \\
&= \gcd(7378, 2006) \\
&= \gcd(7378 \bmod 2006, 2006) \\
&= \gcd(1360, 2006) \\
&= \gcd(1360, 2006 \bmod 1360) \\
&= \gcd(1360 \bmod 646, 646) \\
&= \gcd(68, 646 \bmod 68) \\
&= \gcd(68 \bmod 34, 34) \\
&= \gcd(0, 34) \\
&= 34
\end{aligned}
$$

Here you find that gcd(24140, 40902) = 34. Hence, multiplicative inverse of 24140 mod 40902 does NOT exist.

Note : You can also calculate gcd using tabular method as you learnt in the extended Euclidean algorithm section to avoid repeating the gcd steps to calculate x and y if gcd = 1 does exist.

Ex. 1.13.9 : Find multiplicate inverse of 8 mod 11.

Soln. :

Since 8 < 11, let's swap the values for simplicity.

Index i	quotient q for i	Remainder r for i	s for i	t for i
0		11	1	0
1		8	0	1

Calculate next steps.

Index i	quotient q for i	Remainder r for i	s for i	t for i
0		11	1	0
1		8	0	1
2	11 / 8 = 1	11 – 8*1 = 3	1 – 1*0 = 1	0 – 1*1 = –1
3	8 / 3 = 2	8 – 3*2 = 2	0 – 2*1 = –2	1 – 2*–1 = 3
4	3 / 2 = 1	3 – 2*1 = 1	1 – 1*–2 = 3	–1 – 1*3 = –4
5	2 / 1 = 2	2 – 1*2 = 0	Do not calculate	Do not calculate

Let's re-swap the values.

Hence, x = – 4 and y = 3

Putting the values in the extended Euclidean algorithm, you get

$$ax + by = 1$$

Scanned with CamScanner

$8(-4) + 11(3) = 1$

Since, you have to find multiplicative inverse in mod 11, divide both sides by mod 11.

$8(-4) \bmod 11 + 11(3) \bmod 11 = 1 \bmod 11$

$8(-4) \bmod 11 + 0 = 1$

Recall our discussion on calculating mod for negative numbers. You need to keep adding mod until the number turns positive and then calculate mod on the positive number you got.

$-4 + 11 = 7$

$7 \bmod 11 = 7$

Hence, $8(7) \bmod 11 = 1$

So, multiplicative inverse of 8 mod 11 is 7.

Note : You can also test your solution. If there are mistakes, re-visit the steps you took. In this example, 8*7 = 56 and 56 mod 11 = 1. Hence, you find that 7 is indeed multiplicative inverse of 7 in mod 11.

Ex. 1.13.10 : Find multiplicate inverse of 1234 mod 4321.

Soln. :

Since 1234 < 4321, let's swap the values for simplicity.

Index i	quotient q for i	Remainder r for i	s for i	t for i
0		4321	1	0
1		1234	0	1
2	4321 / 1234 = 3	4321 – 3*1234 = 619	1 – 3*0 = 1	0 – 3*1 = –3
3	1234 / 619 = 1	1234 – 1*619 = 615	0 – 1*1 = –1	1 – 1*–3 = 4
4	619 / 615 = 1	619 – 1*615 = 4	1 – 1*–1 = 2	–3 – 1*4 = –7
5	615 / 4 = 153	615 – 4*153 = 3	–1 – 153*2 = –307	4 – 153*–7 = 1075
6	4 / 3 = 1	4 – 1*3 = **1**	2 – 1*–307 = **309**	–7 – 1*1075 = **–1082**
7	3 / 1 = 3	3 – 3*1 = 0	Do not calculate	Do not calculate

Let's re-swap the values.

Hence, $x = -1082$ and $y = 309$

Putting it in the equation,

$ax + by = 1$

$1234(-1082) + 4321(309) = 1$

Dividing both sides by mod 4321, you get

$1234(-1082) \bmod 4321 + 4321(309) \bmod 4321 = 1 \bmod 4321$

$1234(-1082) \bmod 4321 + 0 = 1$

Convert –1082 to positive

$-1082 + 4321 = 3239$

$3239 \bmod 4321 = 3239$

Hence,

$1234(3239) \bmod 4321 = 1$

Or 3239 is multiplicative modular inverse of 1234 in mod 4321.

Scanned with CamScanner

Review Questions

Here are a few review questions to help you gauge your understanding of this chapter. Try to attempt these questions and ensure that you can recall the points mentioned in the chapter.

[A] Computer Security Concepts

Q. 1 Explain the terms Assets, Controls, Threats, Vulnerabilities, Risk and Exposure with examples. **(8 Marks)**

Q. 2 Draw a block diagram depicting the relation between Assets, Controls, Threats, Vulnerabilities, Risk and Exposure. **(6 Marks)**

Q. 3 With many security controls, you can reduce the risk to zero. Comment. **(4 Marks)**

Q. 4 Describe the three pillars of security with examples. **(8 Marks)**

Q. 5 Describe the three goals of security with examples. **(8 Marks)**

Q. 6 Write a short note on confidentiality. **(4 Marks)**

Q. 7 Write a short note on integrity. **(4 Marks)**

Q. 8 Write a short note on availability. **(4 Marks)**

Q. 9 Explain the terms Identification, Authentication, Authorisation, Accountability and Non-repudiation with example. **(8 Marks)**

Q. 10 Draw a block diagram depicting the relation between Identification, Authentication, Authorisation and Accountability. **(6 Marks)**

[B] OSI Model

Q. 11 Draw a chart for various Security Services provided at various OSI layers. **(6 Marks)**

Q. 12 What is OSI Model? List a few Security Services and Mechanisms for each layer. **(8 Marks)**

Q. 13 Write a short note on Network Security Model. **(6 Marks)**

[C] Types of Security Attacks

Q. 14 Classify security attacks and briefly explain each attack. **(8 Marks)**

Q. 15 Describe Replay Attack with a block diagram. **(8 Marks)**

Q. 16 Describe Denial of Service (DoS) Attack with a block diagram. **(8 Marks)**

Q. 17 Describe Fabrication Attack with a block diagram. **(4 Marks)**

Q. 18 Write a short note on traffic analysis. **(6 Marks)**

Q. 19 Write a short note on eavesdropping. **(6 Marks)**

Q. 20 Compare Active and Passive attacks. **(8 Marks)**

[D] Classical Encryption Techniques

Q. 21 With an example, explain how you can use substitution for encryption. **(6 Marks)**

Q. 22 With an example, explain how you can use transposition for encryption. **(6 Marks)**

(Copyright No. - L82548/2019)

Scanned with CamScanner

Q. 23 Take an example of your choice and work it through explaining Vignere Cipher. **(8 Marks)**

Q. 24 Take an example of your choice and work it through explaining Playfair Cipher. **(8 Marks)**

Q. 25 Take an example of your choice and work it through explaining Hill Cipher. **(8 Marks)**

Q. 26 Take an example of your choice and work it through explaining Keyed Transposition Cipher. **(8 Marks)**

Q. 27 Take an example of your choice and work it through explaining Keyless Transposition Cipher. **(8 Marks)**

Q. 28 Write a short note explaining the difference between monoalphabetic and polyalphabetic cipher. **(6 Marks)**

Q. 29 Write a short note on steganography. **(4 Marks)**

Q. 30 Compare cryptography with steganography. **(4 Marks)**

[F] Methods of Encryption

Q. 31 Describe Symmetric Key Encryption. **(6 Marks)**

Q. 32 Write advantages and disadvantages of Symmetric Key Encryption. **(4 Marks)**

Q. 33 Describe Asymmetric Key Encryption. **(6 Marks)**

Q. 34 Write advantages and disadvantages of Asymmetric Key Encryption. **(4 Marks)**

Q. 35 Compare symmetric and asymmetric encryption. **(6 Marks)**

❑❑❑

Scanned with CamScanner

2 Symmetric and Asymmetric Key Cryptography and Key Management

Syllabus

At the end of this unit, you should be able to understand and comprehend the following syllabus topics :

- Block cipher principles
- Data Encryption Standard (DES)
  - Double DES
  - Triple DES
  - Block cipher modes of operation
- Advanced Encryption Standard (AES)
- Stream Cipher: RC4 algorithm
- Public key cryptography
  - Principles of public key cryptosystems
  - RSA Algorithm
  - Knapsack Algorithm
  - Key management techniques
  - Using symmetric and asymmetric algorithms and trusted third party
    - Symmetric key agreement: Diffie Hellman
    - Symmetric Key Distribution
      - Needham-Schroeder protocol
      - Kerberos: Kerberos Authentication protocol
  - KDC
    - Public key Distribution
    - Digital Certificate: X.509
    - PKI

Scanned with CamScanner

2.1 Concept Building - Types of Symmetric Algorithms (Ciphers)

Symmetric key based algorithms (ciphers) can work either on blocks of bits (characters) or one bit at a time.

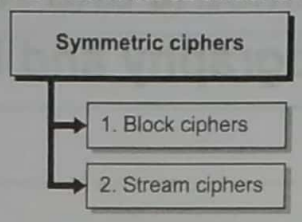


Fig. 2.1.1 : Symmetric Ciphers

2.1.1 Block Ciphers

✍ ***Definition :*** *The algorithms that work on blocks are called block ciphers.*

In block ciphers, the information that needs to be encrypted is broken into smaller and equal block sizes. Then, the encryption operation (substitution and transposition) is applied to each block. The resultant ciphertext from each block is then combined to produce the encrypted information.

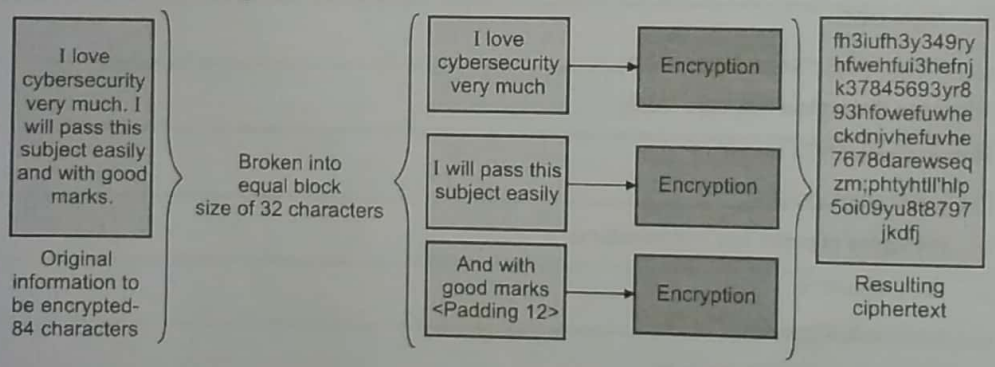


Fig. 2.1.2 : Block Ciphers

This is a highly simplified block diagram of how a block cipher works. The information is broken into equal size blocks and then the encryption operation is carried out on each block. If the block size has lesser number of characters than required to form a block, then padding is done to fill the block. Padding is just filling some temporary information to form a block. Finally, the resulting encrypted information from each block is combined to get the overall encrypted message.

DES and AES are two of the examples of Symmetric Block Ciphers.

2.1.2 Stream Ciphers

✍ ***Definition :*** *The algorithms that work on one-bit at a time are called stream ciphers.*

Unlike block ciphers, stream ciphers work on one bit of plaintext at a time. Each bit of plaintext is combined with the bits of security key and then XORed to get ciphertext.

Note : If you recall from your logical design classes, the Table 2.1.1 is truth table of XOR. For result to be 1, both the inputs should be different.

(Copyright No. - L82548/2019)

Scanned with CamScanner

Table 2.1.1 : XOR Truth Table

Sr. No.	Input X	Input Y	Output Z (XOR X, Y)
1.	0	0	0
2.	0	1	1
3.	1	0	1
4.	1	1	0

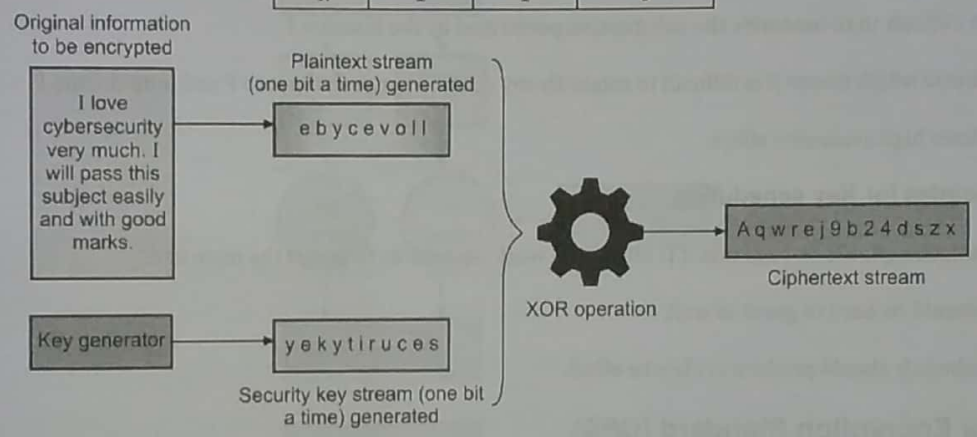


Fig. 2.1.3 : Stream Ciphers

RC4 is an example of stream cipher.

2.1.3 Comparison between Block and Stream Cipher

Q. Compare and contrast - Block and stream ciphers. **MU - Dec. 16, 3 Marks**

Sr. No.	Comparison Attribute	Block Cipher	Stream Cipher
1.	Security	High	Low
2.	Speed	Low	High
3.	Application	Non-real time such as documents	Real time data such as Voice
4.	Commonly used	Yes	No

2.1.4 Block Cipher Principles

There are three critical components in designing a block cipher.

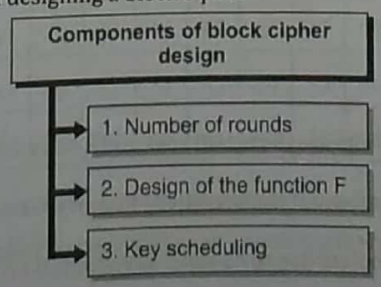


Fig. 2.1.4 : Components of Block Cipher Design

Following are the principles around each of these.

Scanned with CamScanner

Design Principles for Number of Rounds in the Block Cipher Algorithm

1. The greater the number of rounds, the more difficult it is to perform cryptanalysis.
2. The number of rounds is chosen such that a known cryptanalysis takes a greater effort compared to brute-force attack.

Design Principles for function F (Feistel network) in the Block Cipher Algorithm

1. It must be difficult to re-assemble the substitution performed by the function F.
2. F is non-linear which means it is difficult to establish any relation between input to F and output from F.
3. F should have high avalanche effect.

Design Principles for Key scheduling

1. Subkey selection should be such that it is difficult to work backwards to derive the main key.
2. Subkeys should be hard to guess as well.
3. The key schedule should produce avalanche effect.

2.2 Data Encryption Standard (DES)

Q. Explain working of DES detailing the Fiestel structure.	**MU - Dec. 15, 10 Marks**
Q. Explain working of DES.	**MU - May 16, 10 Marks**
Q. Explain DES, detailing the Fiestel structure and S-block design.	**MU - May 17, 10 Marks**
Q. What is the purpose of S-boxes in DES? Explain the avalanche effect?	**MU - May 18, 5 Marks**

✍ ***Definition :*** *Data Encryption Standard (DES) is a symmetric key based block cipher standard used for encryption and decryption.*

It came into existence and usage around Nov 1976 and was predominantly used in the industry until 2002.

Major attributes of DES

- It is a symmetric key based algorithm.
- It works as a block cipher.
- It uses 64-bit blocks.
- It uses a key size of 64-bits in which 56-bits are the actual keys and 8-keys are used for error detection.
- It uses 16 rounds of operation (substitution and transposition) to convert a block of plaintext into ciphertext.
- DES is now considered insecure and obsolete due to its short key-size (56-bits only).

Scanned with CamScanner

2.2.1 Block Diagram and Internals of DES

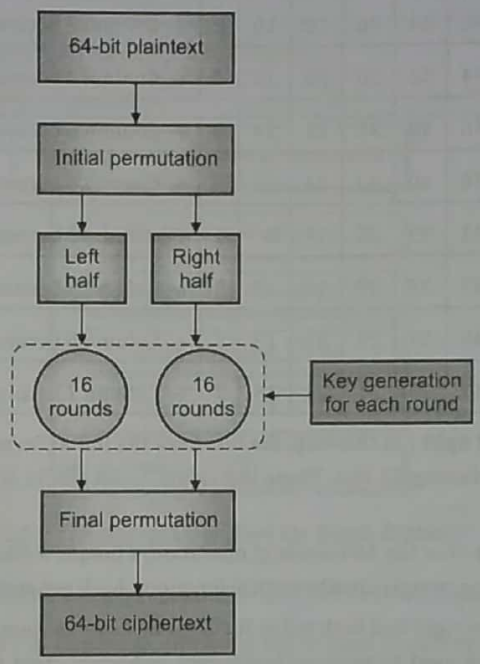


Fig. 2.2.1 : Block diagram of DES

This is very simplistic view of DES. Let us understand what happens at each stage.

Step 1 - Creation of 64-bit blocks : In this step, the plaintext information to be encrypted is broken into 64-bit blocks. DES is a block cipher and block creation is similar to as explained in the earlier section.

Table 2.2.1 : 64 bits of Plaintext

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	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

Step 2 - Initial Permutation : In this step, the 64 bits in the plaintext blocks are re-arranged (transposed). This is done as per the diffusion property of the cipher to ensure that any small variance in plaintext produces a large variance in the ciphertext.

Scanned with CamScanner

Table 2.2.2 : Initial Permutation (Re-arrange bits of Plaintext)

58	50	42	34	26	18	10	2	← Column 2 becomes 1st row
60	52	44	36	28	20	12	4	← Column 4 becomes 2nd row
62	54	46	38	30	22	14	6	← Column 6 becomes 3rd row
64	56	48	40	32	24	16	8	← Column 8 becomes 4th row
57	49	41	33	25	17	9	1	← Column 1 becomes 5th row
59	51	43	35	27	19	11	3	← Column 3 becomes 6th row
61	52	45	37	29	21	13	5	← Column 5 becomes 7th row
63	55	47	39	31	23	15	7	← Column 7 becomes 8th row

- **Step 3 - Left Half and Right Half Split :** In this step, the bits from the Initial Permutation stage are split into two parts – left half and right half each containing 32-bits. These individual 32-bit blocks are then continuously worked through the 16 rounds of operation.
- **Step 4 - Subkey Key Generation :** For the 16 rounds of operation, a unique subkey is derived for each round from the 56-bit key. The key is derived using complex mathematical functions. Each generated subkey is 48-bit long.
- **Step 5 - Rounds :** Left half and the right half both individually go through 16 rounds of encryption operation. In each of the rounds, the derived subkey is used to produce temporary ciphertext. This temporary ciphertext produced after each round is used in the next round until the final round is complete. Each round consists of substitutions and successive permutations.
- **Step 6 - Final Permutation :** In the last stage, we need to bring the bits back to their respective positions. The bit positions were changed at the initial permutation stage.

Table 2.2.3 : Final Permutation (Re-arrange bits of Ciphertext)

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

- **Step 7 - Final Ciphertext :** Once all the steps are done, you get the final ciphertext for the plaintext given the security key of your choice via DES.

✍ ***Exam Tip :*** *IF you hear that an algorithm is broken or is insecure, it means that it is computationally feasible to find out the key used for encryption. Note that cryptography heavily depends upon our understanding of mathematics and the computation power available today. What is secure and infeasible today, could be insecure and feasible to crack in future.*

Scanned with CamScanner

2.2.2 Block Cipher Modes of Operation (for DES and other Block Ciphers in General)

Q.	Explain with examples the CBC and ECB modes of block ciphers.	MU - Dec. 16, 3 Marks
Q.	Discuss in detail block cipher modes of operation.	MU - May 19, 10 Marks

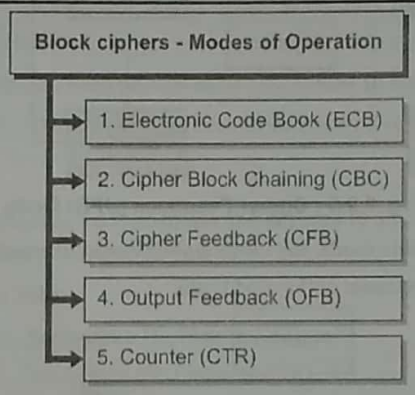


Fig. 2.2.2 : Modes of Operation for Block Ciphers

DES and other block ciphers can potentially work in several modes. Let's review them carefully.

1. **Electronic Code Book (ECB) Mode :** In this mode, the same key is used to encrypt all the blocks. Key derivatives or subkeys are not used. Additionally, each block is treated separately and the ciphertext of previous block does not influence successive blocks.

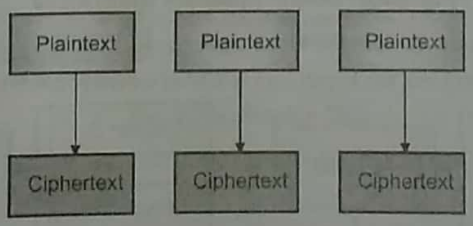


Fig. 2.2.3 : Electronic Code Book (ECB) Mode

2. **Cipher Block Chaining (CBC) Mode :** In this mode, the ciphertext of previous block is used with the next plaintext block. The two blocks (ciphertext of previous block and plaintext of next block) are XORed and then passed through the encryption operation. This generates a lot more randomness in the final ciphertext.

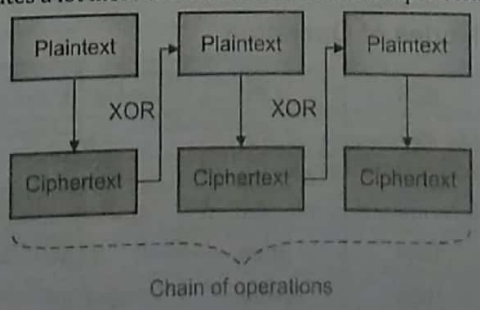


Fig. 2.2.4 : Cipher Block Chaining (CBC) Mode

Scanned with CamScanner

3. **Cipher Feedback (CFB) Mode :** In this mode, the block cipher works like a stream cipher. The ciphertext from the previous block is XORed with the key (keystream) for the next block. This way the key increasingly becomes random and brings more randomness in the overall encryption process.

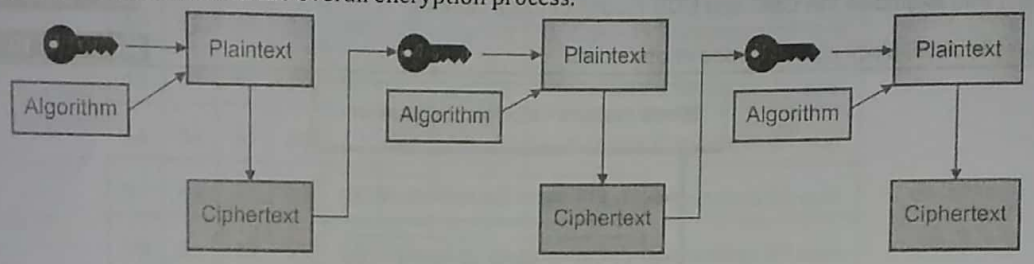


Fig. 2.2.5 : Cipher Feedback (CFB) Mode

4. **Output Feedback (OFB) Mode :** In this mode, the block cipher works like a stream cipher. The keystream used in the previous block is XORed with the keystream of the next block.

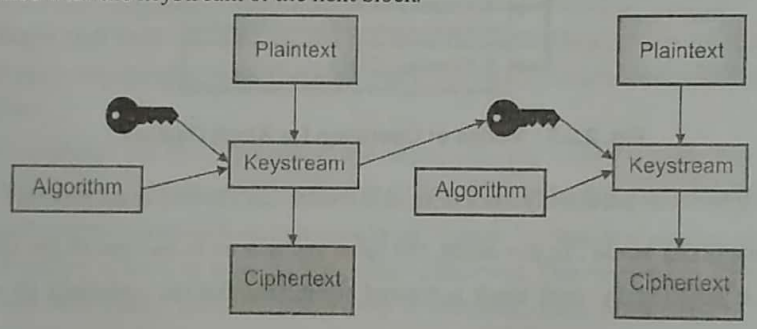


Fig. 2.2.6 : Output Feedback (OFB) Mode

5. **Counter (CTR) Mode :** In this mode as well, the block cipher works like a stream cipher. The key is converted into keystream (as used in stream cipher) and the keystream is XORed with a counter that increases for every block.

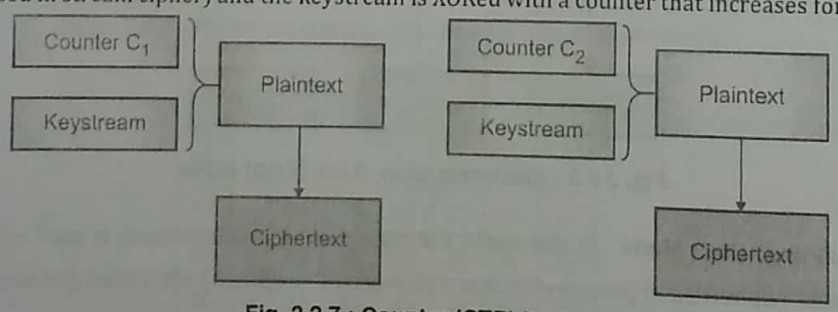


Fig. 2.2.7 : Counter (CTR) Mode

2.2.3 Comparison between Modes of Operation

Sr. No.	Mode	ECB	CBC	CFB	OFB	CTR
1.	In-Parallel block encryption	Yes	No	No	No	Yes
2.	Suited for	Small Information	Any size of information	Small Information	Small Information	Any size of information
3.	Security and randomness	Low	High	High	High	High
4.	Speed	High	Medium	Medium	Medium	High
5.	Complexity	Low	High	High	High	Low
6.	Works like stream cipher?	No	No	Yes	Yes	Yes

(Copyright No. - L82548/2019)

Scanned with CamScanner

Weakness in DES

1. **Small key size :** 56-bits of keys have a keyspace (possible values) of 2^{56}. While that might seem a lot, it is actually not given the compute power we have today. In 1990s, the compute power we had was significantly lower and hence was considered secure at that time.

✍ ***Definition :*** *The type of attack where each combination is tried in an attempt to find the right combination is also called as brute force attack.*

2. **Prone to linear cryptanalysis :** DES has been proven to be susceptible to linear cryptanalysis.
3. **Prone to differential cryptanalysis :** DES has been proven to be susceptive to differential cryptanalysis.

2.2.4 Double DES

In order to strengthen DES, it was considered to increase the key size to 112-bits effectively. The way it was chosen to do so was to use 2 keys of 56-bits each. Let's call them K_1 and K_2.

Mathematically, it can be denoted as below :

Ciphertext C = Encryption (K_2, Encryption (K_1,Plaintext P))

Plaintext P = Decryption (K_1, Decryption (K_2,Ciphertext C))

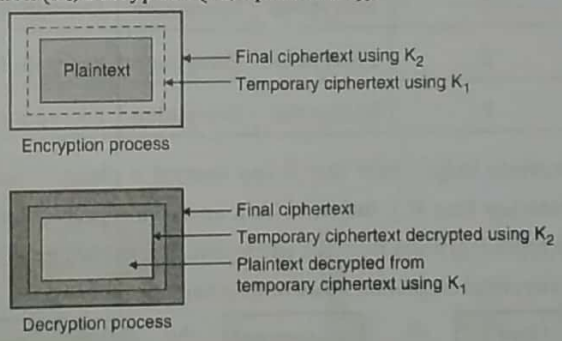


Fig. 2.2.8 : Double DES

- During the encryption process, first the plaintext is encrypted with Key K_1 and then the result is again encrypted with Key K_2 to get the final ciphertext for plaintext.
- During the decryption process, first the Key K_2 is used to decrypt to get the ciphertext that Key K_1 can decrypt to get the plaintext.
- However, Double DES was proven to be ineffective. Meet in the middle attack was shown to reduce the complexity to just 2^{57}(2^{56} attempts made twice, hence $2x2^{56} = 2^{57}$) instead of 2^{112} as originally thought.

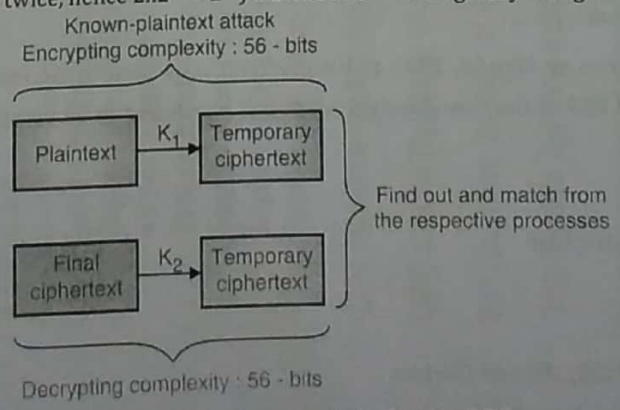


Fig. 2.2.9 : Complexity in Double DES

Scanned with CamScanner

- So, using K_1 if you could derive temporary ciphertext using encryption process and using K_2 if you could also derive the same temporary ciphertext using decryption process, you have found a match and the keys you chose (K_1 and K_2) are now known to you. Hence, you could effectively find both the keys and break Double DES without original thought of complexity of 112 bits.
- Hence, Double DES was not adopted in the industry and is not used.

2.2.5 3DES or Triple DES

Q. Write short notes on 3DES. **MU - May 19, 4 Marks**

Finding that Double DES was ineffective, Triple DES or 3DES was conceived. 3DES uses 48 rounds of operation and can work in the following modes using two or three keys.

Table 2.2.4 : 3DES or Triple DES

Mode	Number of keys	Key 1	Key 2	Key 3
DES-EEE3	3	Encryption	Encryption	Encryption
DES-EDE3	3	Encryption	Decryption	Encryption
DES-EEE2	2	Encryption	Encryption	Encryption Using Key 1
DES-EDE2	2	Encryption	Decryption	Encryption Using Key 1

You might wonder how decryption helps? Note that if you encrypt a plaintext using a key (say K_1) and run the decryption process using a different key (say K_2), the text (from encryption process using K_1) becomes more random. The use of a different key in the decryption process from the encryption process brings added randomness and hence helps to make attacks such as linear or differential cryptanalysis extremely hard.

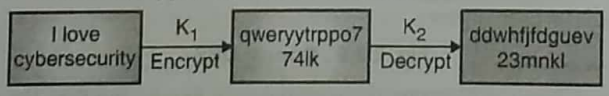


Fig. 2.2.10 : Decryption Process

2.3 Advanced Encryption Standard (AES)

Definition : *Advanced Encryption Standard (AES) is a symmetric key based block cipher standard used for encryption and decryption.*

The standard became effective on May 26, 2002 and is predominantly used in the industry today due to its strong cipher properties. AES replaced DES as the new standard when DES was found to be insecure and vulnerable to various attacks.

Major attributes of AES

- It is a symmetric key based algorithm.
- It works as a block cipher.
- It uses 128-bit blocks.
- It can work with key sizes of 128, 192 and 256 bits.
- Number of rounds of operation depends upon the key size.

Scanned with CamScanner

- 128-bit key undergoes 10 rounds
- 192-bit key undergoes 12 rounds
- 256-bit key undergoes 14 rounds
- AES is considered highly secure due to its long key sizes and is used in the industry today.

2.3.1 Block Diagram and Internals of AES

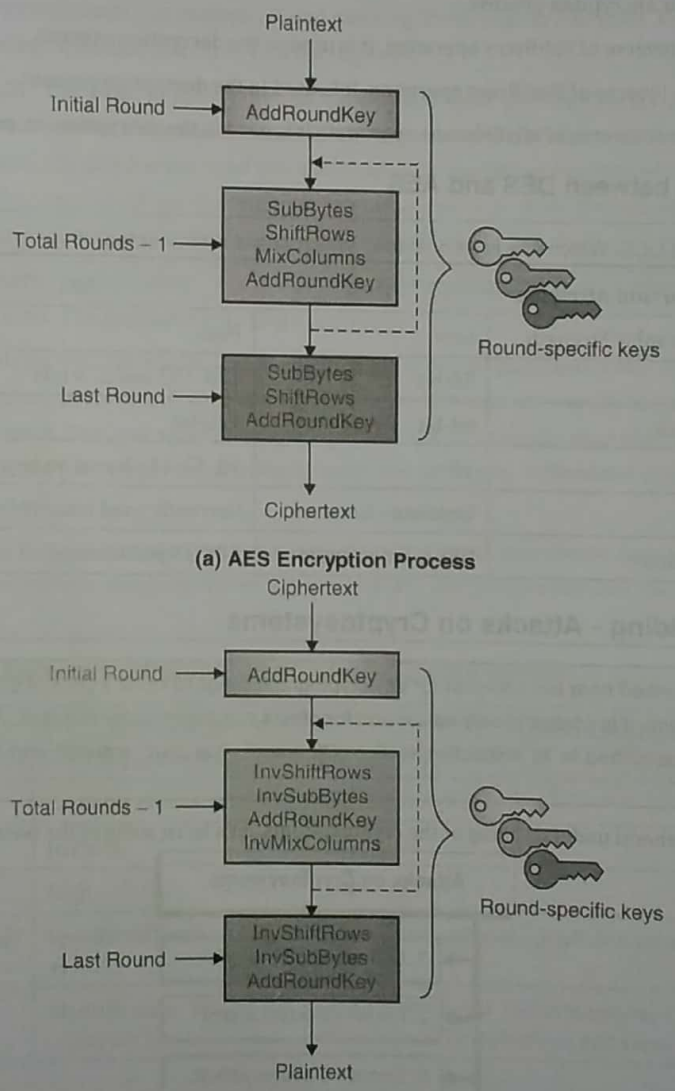


Fig. 2.3.1 : Block diagram of AES

Let's understand the blocks.

1. **AddRoundKey** : In this transformation step, a round key is generated and XORed with the intermediate (temporary) ciphertext. This block is used in both encryption as well as decryption process.

Scanned with CamScanner

2. **SubBytes :** In this transformation step, the intermediate ciphertext undergoes various substitution operations. It is used for encryption process.
3. **ShiftRows :** In this transformation step, the intermediate ciphertext undergoes various row-wise transposition operations. It is used for encryption process.
4. **MixColumns :** In this transformation step, the intermediate ciphertext undergoes various column-wise transposition operations.It is used for encryption process.
5. **InvSubBytes :** This is inverse of SubBytes operation. It is used in the decryption process.
6. **InvShiftRows :** This is inverse of ShiftRows operation. It is used in the decryption process.
7. **InvMixColumns :** This is inverse of MixColumns operation. It is used in the decryption process.

2.3.2 Comparison between DES and AES

Q. Compare AES and DES. Which one is bit oriented? Which one is byte oriented? **MU - May 19, 5 Marks**

Comparison Attribute	DES	AES
Cryptographic Strength	Low	High
Key Size	56-bit	128, 192 and 256 bits
Block Size	64-bit	128-bit
Rounds	16	10, 12, 14 - based on key size
Usage	Obsolete - Not used	Currently used industry standard
Orientation	DES is bit oriented	AES is byte oriented

2.4 Concept Building - Attacks on Cryptosystems

Note : The attacks described here are common for any cryptographic algorithm be it DES, AES, RSA or any other. While for some algorithms it is comparatively easier and for others it is theoretically possible. Any specific attack against an algorithm is described in its respective section. Otherwise, you could mention and elaborate on the following attacks.

Now that you have a general understanding of the cryptosystems, let's learn some of the possible attacks on them.

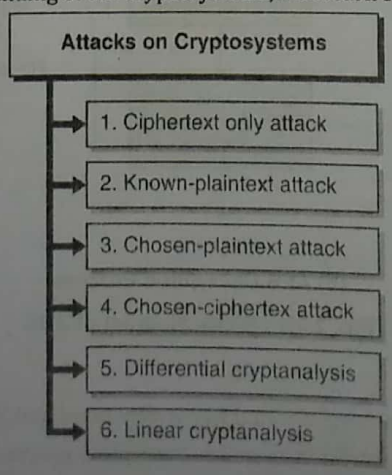


Fig. 2.4.1 : Attacks on Cryptosystems

Scanned with CamScanner

1. **Ciphertext-only attack :** In this type of attack, the attacker has ciphertext of several messages. The algorithm is known, and the goal of the attack is to find out the key used in encryption. Once the key is found out, it is possible to decrypt messages that were encrypted using the key.
2. **Known-Plaintext attack :** The attacker knows the plaintext partially and the corresponding ciphertext. The goal is to find out the key. Once the key is known, the attacker can then use the key to decrypt ciphertext for which she does not know the plaintext. For example, you might be using a fixed greeting in your messages (For example, "Dear Friend") or you might be sending same message (for example, "Good morning") everyday to someone. The attacker could know this and also the corresponding ciphertext and try to find out the key.
3. **Chosen-Plaintext attack :** The attacker knows the exact plaintext and the corresponding ciphertext. The goal is to find out the key. Once the key is known, the attacker can then use the key to decrypt ciphertext for which she does not know the plaintext. For example, the attacker can send you a message that she knows that you will definitely forward to your friends. While forwarding, you might encrypt the message using your key. Now, the attacker can grab the ciphertext that you sent, and she already knows the plaintext message she sent you earlier.
4. **Chosen-Ciphertext attack :** In this attack, the attacker choses the ciphertext that she wants to be decrypted and know the corresponding plaintext. The goal again is to find out the key.
5. **Differential Cryptanalysis :** In this attack, the attacker choses a pair of plaintexts and follows through each stage in their respective encryption process and compare the difference between the results at each stage. The key used in encrypting the pair is same. The goal again is to figure out the key by carefully studying the differences in results at various stages between the pair. Since, the attacker chooses the plaintexts, differential analysis is considered to be a type of chosen-plaintext attack.
6. **Linear Cryptanalysis :** The attacker carries out a known-plaintext attack and tries to figure out the key. She evaluates the input and output at various stages of the encryption process and tries to find out the probability of specific key values.

2.4.1 Comparison between Differential and Linear Cryptanalysis

Comparison Attribute	Differential Cryptanalysis	Linear Cryptanalysis
Plaintext selection	Carefully chosen	Any random plaintext
Plaintext used	In Pairs	One by one
Complexity of attack	High	Low
Mathematical relation between plaintexts used	Specific differences (such as XOR)	Linear approximation (such as a series of XOR operations)
Goal of the attack	Identify some bits of the unknown key	Identify the linear relation between some bits of the plaintext, some bits of the ciphertext and some bits of the unknown key

2.5 Public Key Cryptography

Q. Elaborate the steps of key generation using RSA algorithm. **MU - May 16, 5 Marks**

Q. Briefly define idea behind RSA and also explain :

(i) Give public key and private key.

(ii) Describe security in this system. **MU - May 17, 5 Marks**

(Copyright No. - L82548/2019)

Scanned with CamScanner

You learnt about the use of asymmetric keys earlier. Recall and revise that section before you proceed.

Public key cryptography relies on the use of such asymmetric keys for providing various cryptographic services such as encryption and digital signature.

✍ ***Definition :*** *Public key cryptography is a cryptographic scheme that uses two mathematically related keys, a public key and a private key, for providing various cryptographic services.*

2.5.1 Principles of Public Key Cryptosystems

Following are some basic principles of public key-based cryptosystems.

1. Public key cryptosystems require the use of two keys – a public key and a private key.
2. Public keys are widely known.
3. Private key is kept secret with its owner.
4. The two keys are mathematically related and form a key pair.
5. One key in the key pair cannot be used to derive the other key in the key pair.
6. Any key in the key pair can be used for encryption. The other key then must be used for decryption.
7. Sender and the receiver both must have their own key pairs.

In this section, you are going to learn about various asymmetric key based algorithms.

2.5.2 RSA Algorithm

RSA, named after its inventors Ron Rivest, Adi Shamir and Leonard Adleman, is an asymmetric key based algorithm. As you understand, RSA, or any other asymmetric key based algorithms can be used for confidentiality [encryption, decryption], authentication and non-repudiation.

RSA is based on finding prime factors for very large numbers. The length of numbers that we are referring to here is around 500 digits!

RSA Key Length	Number of digits
1024-bit	309
2048-bit	617
4096-bit	1233

Let's understand how RSA derives public and private keys and how does encryption and decryption process work based on the derived keys.

1. Choose two random large prime numbers, p and q.
2. Multiply the numbers. $n = p*q$.
3. Choose a random integer to be encryption key e such that e and $(p - 1)(q - 1)$ are relatively prime.
4. Decryption key is computed as $d = e^{-1} \bmod ([p - 1]*[q - 1])$.
5. The public key = (n, e).
6. The private key = (n, d).
7. For encrypting message M with public key (n, e), you get ciphertext $C = M^e \bmod n$.
8. For decrypting ciphertext with private key (n, d), you get plaintext $M = C^d \bmod n$.

Scanned with CamScanner

Ex. 2.5.1 : Perform encryption and decryption using RSA algorithm with p = 7, q = 11, e = 17 and M = 8.

Soln. :

$$n = p * q$$
$$n = 7 * 11 = 77$$
$$r = (p - 1) * (q - 1)$$
$$r = 6*10 = 60$$
$$d = e^{-1} \bmod r$$
$$ed \equiv 1 \bmod 60$$
$$17d \equiv 1 \bmod 60$$

Let's calculate modulo inverse using extended Euclidean algorithm (swapping 17 and 60)

Index i	quotient q for i	Remainder r for i	s for i	t for i
0		60	1	0
1		17	0	1
2	60 / 17 = 3	60 – 3*17 = 9	1 – 3*0 = 1	0 – 3*1 = –3
3	17 / 9 = 1	17 – 1*9 = 8	0 – 1*1 = –1	1 – 1*–3 = 4
4	9 / 8 = 1	9 – 1*8 = **1**	1 – 1*–1 = **2**	–3 – 1*4 = **–7**
5	8 / 1 = 8	8 – 1*8 = 0	Do not calculate	Do not calculate

Let's re-swap the values.

$$x = -7$$
$$y = 2$$

Putting the values in the extended Euclidean algorithm, you get

$$ax + by = 1$$
$$17(-7) + 60(2) = 1$$

Since, you have to find multiplicative inverse in mod 60, divide both sides by mod 60.

$$17(-7) \bmod 60 + 60(2) \bmod 60 = 1 \bmod 60$$
$$17(-7) \bmod 60 + 0 = 1$$

Recall our discussion on calculating mod for negative numbers. You need to keep adding mod until the number turns positive and then calculate mod on the positive number you got.

$$-7 + 60 = 53$$
$$53 \bmod 60 = 53$$

Hence, $17(53) \bmod 60 = 1$

Hence, value of decrypting key, $d = 53$

Now, you have all the values needed for encryption and decryption.

As per RSA,

$$C = M^e \bmod n$$

Scanned with CamScanner

$$C = 8^{17} \bmod 77$$

$$\mathbf{C = 57}$$

$$M = C^d \bmod n$$

$$M = 57^{53} \bmod 77$$

$$\mathbf{M = 8}$$

Ex. 2.5.2 : In an RSA system the public key (e, n) of user A is defined as (7,119). Calculate $\phi(n)$ and private key d. What is the ciphertext when you encrypt message m =10, using the public key? **MU - Dec. 15, 10 Marks**

Soln. :

Since, n is 119, assuming p and q were 7 and 17 respectively (by factorizing 119).

$$r = (p-1)*(q-1)$$

$$r = 6*16 = 96$$

phi(n) can be calculated using the formula

$$\text{phi}(n) = n*\left(1-\frac{1}{7}\right)*\left(1-\frac{1}{17}\right) \text{ [where 7 and 17 are prime factors of 119)}$$

Hence, $\text{phi}(n) = 96$

$$d = e^{-1} \bmod r$$

$$ed \equiv 1 \bmod 96$$

$$7d \equiv 1 \bmod 96$$

Let's calculate modulo inverse using extended Euclidean algorithm (swapping 7 and 96)

Index i	quotient q for i	Remainder r for i	s for i	t for i
0		96	1	0
1		7	0	1
2	96 / 7 = 13	96 – 7* 13 = 5	1 – 0*13 = 1	0 – 1*13 = – 13
3	7 / 5 = 1	7 – 5 * 1 = 2	0 – 1*1 = –1	1 – (– 13)*1 = 14
4	5 / 2 = 2	5 – 2* 2 = 1	1 – (– 1)*2 = 3	– 13 – 14*2 = **– 41**
5	2 /1 = 2	2 – 1*2 = 0	Do not calculate	Do not calculate

Let's re-swap the values.

$$x = -41$$

$$y = 3$$

Putting the values in the extended Euclidean algorithm, you get

$$ax + by = 1$$

$$7*(-41) + 96*(3) = 1$$

Since, you have to find multiplicative inverse in mod 96, divide both sides by mod 96.

$$7*(-41) \bmod 96 + 96*(3) \bmod 96 = 1 \bmod 96$$

$$7*(-41) \bmod 96 + 0 = 1 \bmod 96$$

$$7*(-41) \bmod 96 = 1 \bmod 96$$

You need to keep adding mod until the number turns positive and then calculate mod on the positive number you got.

Scanned with CamScanner

$$-41 + 96 = 55$$

$$55 \bmod 96 = 55$$

Hence, multiplicative inverse is 55.

Therefore, the value of decrypting key, $d = 55$.

Now, you have all the values needed for encryption and decryption.

As per RSA,

$$C = M^e \bmod n$$

$$C = 10^7 \bmod 119$$

$$\boldsymbol{C = 73} \text{ [encrypted message]}$$

$$M = C^d \bmod n$$

$$M = 73^{55} \bmod 119$$

$$\boldsymbol{M = 10} \text{ [decrypted message]}$$

Attacks on RSA

1. **Brute-force attack :** Here the attacker tries to find factors of n by trying out various possibilities.
2. **Common Modulus :** To avoid generating a different modulus n = p*q for each user one may wish to fix n for all the users. It might seem like deriving the decrypting key d is not possible for every encrypting key e by any other user since encrypting key e is a randomly chosen value. But, the problem with this approach is that a particular user who knows her pair of e and d can successfully use her own pair to find the factors for common modulus. Once the factors are known, since the encrypting key e is known to everyone (because it is public key), the decrypting key d could be found out. Hence, you should not be using common modulus to generate keys for multiple users.
3. **Choosing smaller numbers :** The security of the algorithm comes from the fact that factoring large numbers is computationally intensive. Sometimes, to improve system performance, smaller numbers can be chosen which can significantly enhance the performance but at the cost of making the algorithm weaker. Hence, you should always choose large numbers to maintain the strength of the algorithm.
4. **Man in the middle attack :** The attacker could collect all the ciphertext coming out from the user's system (that is encrypted with her private key) and try to find the private key. The information known to the attacker is the ciphertext and public key.

2.5.3 Knapsack Algorithm

✍ ***Definition :*** *The Merkle-Hellman Knapsack algorithm is an asymmetric key based algorithm used for encryption and decryption.*

It was invented by Ralph Merkle and Martin Hellman in 1978.

Major attributes of the Knapsack Algorithm

- It is based on public key cryptography meaning that it requires two keys – public and private.
- Unlike RSA, it is one way. Public key is used for encryption and private key is used for decryption.
- Hence, it cannot be used for digital signature and non-repudiation.
- It is outdated and is not used anymore.

Scanned with CamScanner

Algorithm

- Knapsack means a bag. The basic Knapsack problem aims at maximizing the weight of the knapsack taking values from a set of available weights.

✍ ***Definition :*** *Given a Knapsack of a maximum capacity of W and N items each with its own value and weight, choose items to place inside the Knapsack such that the final contents in the knapsack has the maximum value.*

In Merkle-Hellman Knapsack Algorithm, the keys are two knapsacks.

- The public key is a 'hard' knapsack A, and
- The private key is an 'easy', or super-increasing, knapsack B.

Two numbers, a multiplier and a modulus, can be used to convert the super-increasing knapsack B into the hard knapsack A.

So,

- Given a set of numbers *A* and a number *b*.
- Find a subset of *A* which sums to *b*.

So, if you have a set of weights of 1, 4, 6, 8 and 15, and you want to get a weight of 29, you could thus use 6, 8 and 15 (6 + 8 + 15 = 29). So, the code would become 00111 (represented by not picking '1', not picking '4', picking '6', picking '8' and picking '15').

Plaintext	10001	11001	00001	00101
Knapsack	1,4,6,8,15	1,4,6,8,15	1,4,6,8,15	1,4,6,8,15
Ciphertext	1+15 = 16	1+4+15 = 20	15	6+15 = 21

2.6 Key Management Techniques

Now that you understand various cryptographic methods and procedures, let us focus on the keys. As you know, keys play the critical role in delivering several cryptographic services. Without adequate and protected methods of key generation, distribution, and disposal, the entire cryptosystem could collapse, and you might not be able to use any cryptographic services.

2.6.1 What is a Key?

What is a Key? As per National Institute of Standards and Technology (NIST) :

✍ ***Definition :*** *A key is a parameter used in conjunction with a cryptographic algorithm that determines its operation in such a way that an entity with knowledge of the key can reproduce, reverse or verify the operation, while an entity without knowledge of the key cannot.*

Let us recall the keys used in various cryptographic services assuming that there are two parties involved in the communication and they both wish to use cryptographic services.

Cryptographic Service	Keys used
Symmetric Key Encryption / Decryption	One symmetric key
Asymmetric Key Encryption / Decryption	Two Public Private Key Pairs
Hashing	No Keys
MACs	One symmetric Key
Digital Signature	Two Public Private Key Pairs

Scanned with CamScanner

2.6.2 Key States

It is important for you to understand that the keys could be in various states and can transition from one state to another state based on circumstances and scenarios.

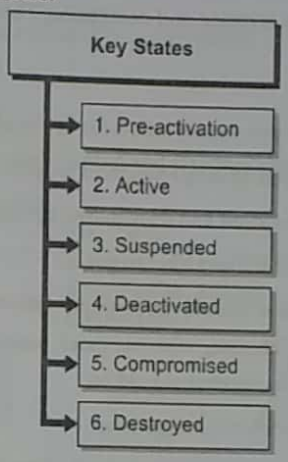


Fig. 2.6.1 : Key States

1. **Pre-activation state :** In this state, the key is generated but is not yet in use. This state is reached as soon as the key is generated.
2. **Active state :** In this state, the key can be used to provide cryptographic services.
3. **Suspended state :** The key in this state is restricted temporarily for use. The key can be transitioned to the active, de-active or destroyed state based on various criteria.
4. **Deactivated state :** In this state, the key is not used to carry out new cryptographic services. However, it may be used for previously consumed cryptographic services. For example, a deactivated key may not be used for encrypting new information but can still be used for decrypting previous information that it was used to encrypt when it was in the active state.
5. **Compromised state :** In this state, the key is exposed, and its secrecy can no more be guaranteed. Once a key is determined to be in this state, it is strongly advisable to suspend its use, carry out investigation to find the root cause of exposure and then destroy it.
6. **Destroyed state :** In this state, the key is safely disposed. The key is no more available to carry out any form of cryptographic services.

2.6.3 Cryptoperiod (Key lifetime)

All cryptographic keys come with a validity period. Some keys are short-lived, and some are long-lived.

✍ ***Definition :*** *A cryptoperiod is the time span during which a specific key is authorised for use by legitimate entities, or the keys for a given system will remain in effect.*

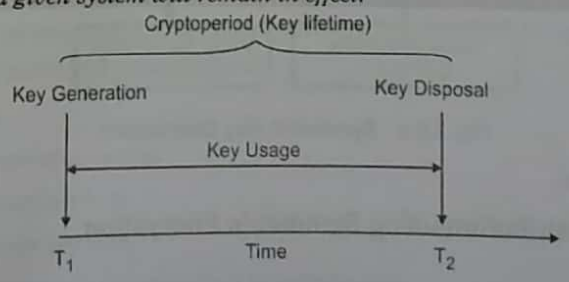


Fig. 2.6.2 : Cryptoperiod (Key Lifetime)

Scanned with CamScanner

- Master Keys are usually long-lived. They are not used for providing cryptographic services themselves. Instead, they are used to periodically generate session keys.
- Session Keys are usually short-lived (up to a session). These are generated from the master keys and are used for providing cryptographic services such as encryption and decryption. These are frequently generated and destroyed.

2.6.4 Key Management Principles

Following are some general principles that you should follow for key management.

1. **Adequate key length :** The key length should be long enough to provide the necessary level of protection.
2. **Secure transmission :** The keys should be stored and transmitted by secure means.
3. **Difficult to guess :** The keys should be random, and hard to guess.
4. **Adequate Cryptoperiod :** The key's lifetime should correspond with the sensitivity of the data it is protecting. Low sensitive data may allow for a longer key lifetime, whereas more sensitive data might require a shorter key lifetime. Also, the more the key is used, the shorter its lifetime should be.
5. **Backup :** The keys should be backed up in case of emergencies.
6. **Adequate key disposal :** The keys should be properly destroyed when they expire.

2.6.5 Symmetric Key Distribution

Symmetric key distribution requires that the communication parties have the exact same key. The same key is used for cryptographic services between the parties.

Ways to distribute symmetric keys

Assume that there are two communicating parties – *A* and *B*. The symmetric key can be distributed in the following ways :

1. A and B can physically generate and distribute the key amongst each other.
2. Any third party C can physically generate and distribute the key to both A and B.
3. A and B can get connected to a key distribution center and get the key.

In reality, distributing the keys physically is infeasible. Hence, a third party, generally known as the Key Distribution Center, is tasked to distribute the keys amongst the communicating parties.

The symmetric key distribution can be achieved in two ways.

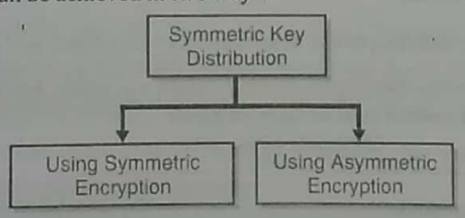


Fig. 2.6.3 : Symmetric Key Distribution

Let's learn about both of them.

2.6.5(A) Symmetric Key Distribution using Symmetric Encryption

Let's assume the following.

- *A* and *B* are the two communicating parties.

Scanned with CamScanner

- There is a **trusted third party** known as Key Distribution Center (KDC) denoted by *K*.
- *A* and *K* have a shared master key *Ka*.
- Similarly, *B* and *K* have a shared master key *Kb*.
- *Na* is a random number called nonce generated by *A* to avoid replays.
- *Ida* is the unique identifier for the party *A*.
- *IDb* is the unique identifier for the party *B*.
- *Ks* is the desired session key that needs to reach both *A* and *B* so that they can securely communicate.

The steps for key distribution are as following.

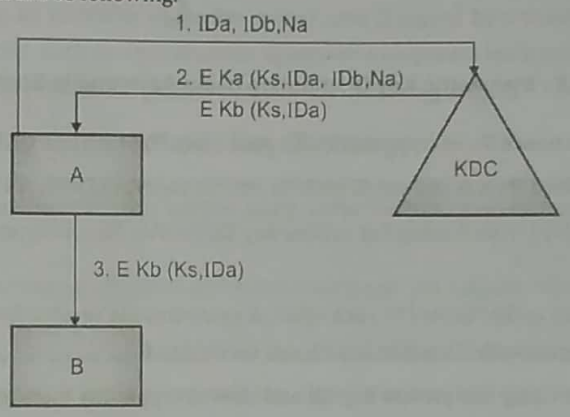


Fig. 2.6.4 : Symmetric Key Distribution using Symmetric Encryption

1. The communicating party A initiates a request with KDC by identifying itself as IDa and identifying the target party as IDb. It sends a nonce Na to ensure that the request is protected against the replay attacks.
2. The KDC generates the desired session key Ks and encrypts it with the master secret shared between it and A, Ka. It also sends back the original request from A to ensure that the response matches with the initial request. It also sends another copy of session key Ks and IDa encrypted using the master secret shared between it and B, Kb. This copy can only be decrypted by B.
3. A verifies the information received from KDC, stores the session key Ks and forwards the copy intended for B.
   B decrypts the copy, stores the session key Ks and gets to know that the communicating party is A with whom the received session key is to be used.

   This way, both *A* and *B* receive the session key Ks and can then begin secure communication.

2.6.5(B) Symmetric Key Distribution using Asymmetric Encryption

Using symmetric key encryption to distribute symmetric keys requires KDC. Using asymmetric encryption for key distribution requires the use of public and private key pairs.

Let's assume the following.

- *A* and *B* are the two communicating parties.
- A's public key is Pa and private key is Sa.
- B's public key is Pb and private key is Sb.
- *Na* is a random number called nonce generated by *A* to avoid replays.
- *Nb* is a random number called nonce generated by *B* to avoid replays.

Scanned with CamScanner

- *IDa* is the unique identifier for the party *A*.
- *IDb* is the unique identifier for the party *B*.
- The session key to be distributed is Ks.

The steps for key distribution are as following.

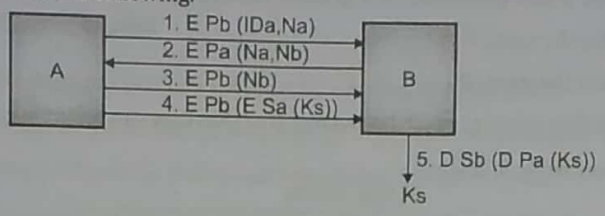


Fig. 2.6.5 : Symmetric Key Distribution using Asymmetric Encryption

1. A sends its identity IDa and a nonce Na encrypted with B's public key Pb.
2. B decrypts the message received from A and sends back Na and Nb encrypted with A's public key.
3. A decrypts the message received from B using her private key Sa, verifies Na, encrypts Nb with Pb and sends it back to B.
4. At this point, both A and B are authenticated to each other. A generates the session key Ks encrypts it with her private key Sa and then encrypts it again with B's public key Pb and sends it to B.
5. B first decrypts the message using her private key Sb and then decrypts the message again using A's public key Pa. B now gets the session key Ks and knows that only A could have used her private key to send the session key to her.

2.6.6 Asymmetric Key Distribution (Distribution of Public Keys)

Asymmetric keys involve public keys and private keys. Public keys are known in general. Private keys are kept secure and are in constant possession of their respective owners. The goal of asymmetric key distribution is to distribute public keys only. Private keys are not distributed.

Ways to distribute asymmetric keys

Asymmetric keys can be distributed using the following ways.

Fig. 2.6.6 : Asymmetric Key Distribution

1. **Broadcast :** Broadcast or public announcement of public keys is the simplest of all the distribution techniques. In this, the public keys are constantly broadcasted (shared with the larger user community). As new users get into the community, they hear such key broadcast messages and store the public keys to use if and when required.

Scanned with CamScanner

2. **Public directory :** A centralized and publicly open directory (very much like an old telephone directory) could be established to register and distribute public keys. Users can register new keys, change previous keys or delete the keys if required. The company maintaining the directory could exercise control to ensure that the directory is highly available, and the key information contained there-in is not destroyed or maligned.
3. **Public-key authority :** In Public-key authority, the central authority itself has a public and private key pair. When a user asks for the public key of the desired target user, the authority encrypts the public key information of the desired user with its own private key to securely distribute the public key.
4. **Certificates :** Certificate is the most widely used technique to distribute public keys in the industry today. A certificate typically consists of a public key, an identifier of the key owner, and is signed by a trusted third party. Typically, the third party is a certificate authority, such as a government agency or a corporate institution, that is trusted by the user community. You will learn about certificates in greater detail in the upcoming section.

2.6.7 Key Management using Diffie Hellman Key Exchange

Q. Explain how a key is shared between two parties using Diffie Hellman key exchange algorithm. What is the drawback of this algorithm? **MU - Dec. 15, 10 Marks**

Q. Explain "Diffie-Hellmen" key exchange algorithm with suitable example. Also explain the problem of MIM attack in it. **MU - May 17, 10 Marks**

If you recall, one of the challenges with using symmetric key algorithms was key distribution. How could sender and the receiver agree upon the key that would be used for encryption and decryption (recall that the symmetric key based algorithms use the SAME key for both encryption as well as decryption)? I asked you this question in the symmetric key section and here I am again to help you with the answer. Diffie-Hellman algorithm is one of the answers to the question.

✍ ***Definition :*** *The Diffie-Hellman algorithm provides a way of generating a shared secret between the sender and the receiver in such a way that the secret need not be exchanged or transferred over the communication medium.*

Basically, the sender and the receiver create the key together at their respective ends at the same time. The key, that they create at their respective ends, is mathematically computed to be the same. Hence, the key distribution need not happen, and the sender and the receiver can confidentially communicate using the key they created. You should note here that the Diffie-Hellman algorithm is NOT used for actual encryption and decryption process. It is used only for key generation.

Let's understand the steps involved in generating the shared key. Assume that there are two users Alex and Bobby who need to generate a shared key for securely communicating with each other.

1. Alex chooses two prime numbers g and p and also a secret number a. He calculates value of A such that $A = g^a \bmod p$. He then sends g, p and A to Bobby. Note here that Alex does not share the secret number a with Bobby.
2. Similarly, Bobby chooses a secret number b and computes the value of B such that $B = g^b \bmod p$. She then sends B to Alex.
3. Alex computes the shared key at his end as Shared Key, $S = B^a \bmod p$
4. Bobby computes the shared key at her end as Shared Key, $S = A^b \bmod p$

Scanned with CamScanner

The values of the shared key, S, derived in the step 3 and 4 are equal due to mod operation.

- $(g^a \bmod p)^b \bmod p = g^{ab} \bmod p$
- $(g^b \bmod p)^a \bmod p = g^{ba} \bmod p$

It does not matter which step you do earlier. Both the keys created would be equal and can be used with any of the algorithms such as DES and AES to encrypt the information and communicate confidentially.

Let's understand the algorithm by solving a question.

Ex. 2.6.1 : Calculate shared key between two users if the initial chosen prime numbers are 5 and 7.

Soln. :

$$g = 5$$
$$p = 7$$

Assume that the user Alex chooses a secret number $a = 2$

$$A = g^a \bmod p$$
$$A = 5^2 \bmod 7$$
$$A = 25 \bmod 7$$
$$A = 4$$

Alex sends g, p and A to Bobby.

Assume that the user Bobby chooses a secret number $b = 3$

$$B = g^b \bmod p$$
$$B = 5^3 \bmod 7$$
$$B = 125 \bmod 7$$
$$B = 6$$

Bobby sends B to Alex

Now, both the users compute the shared key, S, at their respective ends.

Alex calculates it as,

$$S = B^a \bmod p$$
$$S = 6^2 \bmod 7$$
$$S = 36 \bmod 7$$
$$S = 1$$

Bobby calculates it as,

$$S = A^b \bmod p$$
$$S = 4^3 \bmod 7$$
$$S = 64 \bmod 7$$
$$S = 1$$

So, the shared key, that can be used between Alex and Bobby, is $S = 1$.

Note : The example we took here involves very small numbers to make it easy for you to understand the steps involved in the key generation. Practically, the numbers used in the shared key generation are extremely large, possibly containing around 500 digits!

Scanned with CamScanner

Ex. 2.6.2 : Users A and B use the Diffie-Hellman key exchange technique with a common prime 71 and primitive root 7. Show that 7 is primitive root of 71. If user A has private key x = 5, what is A's Public Key R_1? If user B has private key y =12, what is B's public key R_2? What is the shared secret key? **MU - May 19, 10 Marks**

Soln. :

- If the multiplicative order of a number r modulo n is equal to Euler Totient Function $\Phi(n)$, then it is a primitive root.
- $\Phi(71) = 70$ (all the 70 numbers before 71 are coprime with 71).
- Let's assume that 7 is a primitive root of 71. Then, it should be true that

  $\Phi(71)^7 \bmod 71 = \Phi(71)$.
- Calculate $70^7 \bmod 71$. This gives 70 which is equal to $\Phi(71)$. Hence, 7 is indeed a primitive root of 71.

  $g = 7$

  $p = 71$

For user A

- Private key x = 5 which means

  $A = g^x \bmod p$

  $A = 7^5 \bmod 71$

  $A = 51$
- Hence, the public key of user A is 51.

For user B

- Private key Y = 12 which means

  $B = g^y \bmod p$

  $B = 7^{12} \bmod 71$

  $B = 4$
- Hence, the public key of user B is 4.
- Now, both the users compute the shared key, S, at their respective ends.
- User A calculates it as,

  $S = B^x \bmod p$

  $S = 4^5 \bmod 71$

  $S = 30$
- User B calculates it as,

  $S = A^y \bmod p$

  $S = 51^{12} \bmod 71$

  $S = 30$
- Hence, the shared key between User A and User B is 30.

(Copyright No. - L82548/2019)

Scanned with CamScanner

2.7 Digital Certificate - X.509

Q. Give the format of X.509 digital certificate and explain the use of a digital signature in it. **MU - Dec. 15, 5 Marks**

Q. What is a digital certificate? How does it help to validate the authenticity of a user? Explain the X.509 certificate format. **MU - Dec. 17, 10 Marks**

Q. Write short notes on X.509. **MU - May 19, 4 Marks**

X.509 is a standard that defines the requirements for public key certificates.

Definition : *A certificate is a signed data structure that binds a public key to a person, computer, or organization.*

The certificate uniquely identifies the owner of a public key. Certificates are issued by Certification Authorities (CAs) such as Verisign, Global Sign etc. In a secure communication, certificates are used to identify the systems and establish trust amongst the communicating parties. Since its inception in 1998, three versions of the X.509 public key certificate standard have evolved. The most commonly used version of X.509 is version 3. X.509 certificates are used for secure communication such as establishing a https connection.

Table 2.7.1 : Contents of a X.509 version 3 certificate

Fields	Purpose	Example
Version	Identifies the version of the certificate	V3
Serial Number	Unique number for the certificate	79ad16a14aa0a5ad4c7358f407132e65
Signature algorithm	Algorithm used to create digital signature for certificate	sha1RSA
Signature hash algorithm	Algorithm used to calculate hash of certificate	sha1
Issuer	Name of certificate issuer	CN = Microsoft Root Certificate Authority DC = Microsoft DC = com
Valid from	Date from which certificate is valid	Thursday, May 10, 2001 4:49:22 AM
Valid to	Date until which certificate is valid	Monday, May 10, 2021 4:58:13 AM
Subject	Name of certificate owner	CN = Microsoft Root Certificate Authority DC = Microsoft DC = com
Public key	Public key	a5 94 ef 15 14 89 fd 4b 73 ... (4096 bits)
Issuer unique ID	ID of issuing Certificate Authority (CA)	03475573948593hgfyuerwe327e7e52513fc2ae3
Subject unique ID	ID of subject	0eac826040562797e52513fc2ae10a539559e4a4
Extensions	Optional Information	Thumbprint, Friendly Name, Key Usage, etc.

Note : If you use Microsoft® Windows® OS, you can go to (Windows Key + R) Run and type certmgr.msc. It would open up certificate manager where you can browse various certificates stored on your system. Double-click on any certificate and examine the various fields it has.

Scanned with CamScanner

2.8 Public Key Infrastructure (PKI)

Asymmetric cryptography has been widely adopted to provide various cryptographic services such as encryption, digital signature, authentication and non-repudiation. Asymmetric cryptography is critically dependent on the availability of an ecosystem of public private key-pairs. In this ecosystem, the key-pairs need to be securely generated, transported, distributed, verified, used and disposed. Public Key Infrastructure (PKI) provides such an ecosystem of operation.

✍ ***Definition :*** *Public Key Infrastructure (PKI) consists of programs, data formats, procedures, communication protocols, security policies, and public key cryptographic mechanisms working in a comprehensive manner to operationalize asymmetric cryptography based cryptographic services.*

It supports the distribution, revocation and verification of public keys used for public key encryption and enables linking of identities with public key certificates.

Note : Do not confuse between Public Key Cryptography and Public Key Infrastructure. Public Key Cryptography refers to the use of asymmetric keys for cryptographic services whereas Public Key Infrastructure is about managing the entire lifecycle of those keys. It is more about infrastructure management around those keys.

2.8.1 Components of PKI

PKI has several components that work together in a comprehensive manner to operationalize the ecosystem of public-private keys. The major components of PKI are as following.

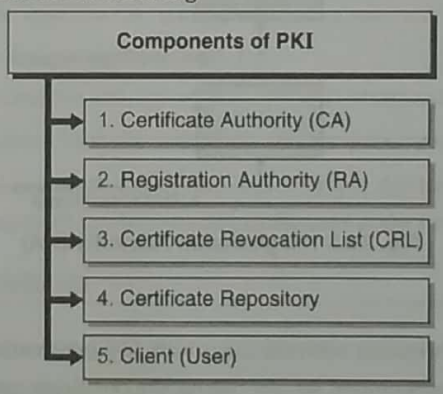


Fig. 2.8.1 : Components of PKI

1. Certificate Authority (CA)

✍ ***Definition :*** *The Certificate Authority (CA) is a trusted third party that provides the root of trust (highest level of trust) for all PKI certificates and provides services that can be used to authenticate the issued certificates.*

When the CA signs the certificate, it binds the certificate's identity to the public key, and the CA takes liability for the authenticity of that certificate. It is this trusted third party (the CA) that allows people who have never met to authenticate to each other and to communicate securely. A CA is a trusted organization (or server) that maintains and issues digital certificates. Some examples of CAs are Verisign, Comodo and Geotrust.

Scanned with CamScanner

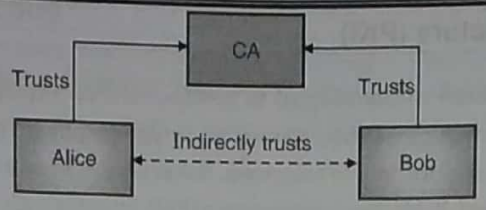


Fig. 2.8.2 : Certificate Authority (CA)

2. Registration Authority (RA)

Definition *: The Registration Authority (RA) performs the certification registration duties. The RA establishes and confirms the identity of a certificate requester, initiates the certificate generation process with the CA, and manages the certificate life-cycle.*

The RA cannot issue certificates itself. It acts as a middle-man between the certificate requester and the associated CA. RA verifies all the necessary information before approving the request to get a certificate from the CA.

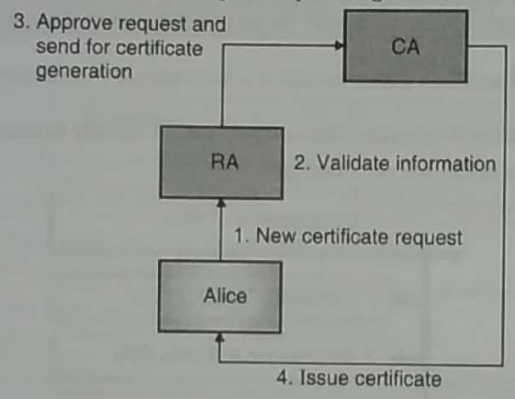


Fig. 2.8.3 : Registration Authority (RA)

3. Certificate Revocation List (CRL)

Certificates hold the public key information whereas the private key information should be kept secure with the user. In scenarios where the private key is compromised for any reason, the certificate cannot be trusted any further and hence it is crucial to let the world know that the certificate should not be trusted. Such a mechanism to notify the voidance of the certificate is called Certificate Revocation List (CRL).

Definition : *A Certificate Revocation List (CRL) is a list of digital certificates that have been revoked by the issuing Certificate Authority (CA) before their scheduled expiration date and should no longer be trusted.*

CRLs provide the blacklist of certificates and are used by PKI clients to verify a certificate's validity and trustworthiness.

4. Certificate Repository

This is a generic database that stores both the valid as well as the invalid certificates. This database stores information about all the issued certificates. In addition to the certificate itself, the database includes validity period and status of each certificate. Certificate revocation is done by updating this database and marking the certificates as unfit for use.

Scanned with CamScanner

5. Client

These are users, machines or other end point entities that use PKI infrastructure. They could request for new certificates, request certificates for other users (to get their public key) or carry out other certificate operations such as checking the CRL to confirm a certificate's validity.

Steps Involved in a Digital Certificate Generation

Following diagram provides the high-level steps involved in certificate generation.

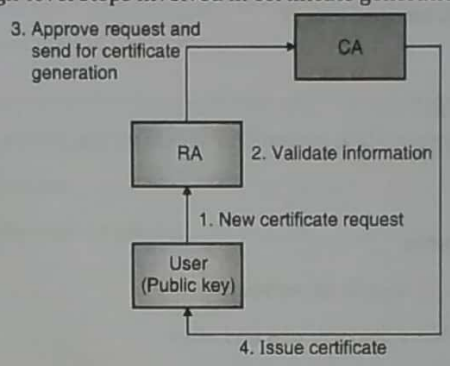


Fig. 2.8.4 : Steps involved in a Digital Certificate Generation

1. Any user, desiring to get a certificate from the CA, sends her information such as name, address, etc. along with her public key to the RA.
2. The RA verifies all the submitted information and approves the certificate request.
3. The RA forwards the information received from the user along with her public key to the CA.
4. The CA computes the hash value of the user's public key and encrypts the hash value with its own private key to generate a digital signature. It then sends all this information to the user in the form of a digital certificate.

2.9 Needham-Schroeder protocol

This is covered in Unit 4. Please refer Section 4.3.1.

2.10 Kerberos : Kerberos Authentication protocol

KDC and this topic is covered in Unit 4. Please refer Section 4.3.2.

Review Questions

Here are a few review questions to help you gauge your understanding of this chapter. Try to attempt these questions and ensure that you can recall the points mentioned in the chapter.

[A] Block Ciphers

Q. 1 Write a short note on block ciphers. **(6 Marks)**

Q. 2 List the design principles of block ciphers. **(8 Marks)**

Q. 3 What is Data Encryption Standard (DES)? List its major attributes. **(6 Marks)**

Q. 4 Draw a block diagram of DES and explain how it works. **(8 Marks)**

Scanned with CamScanner

Q. 5	Write a short note on Electronic Code Book (ECB) Mode of Block Cipher operation.	(4 Marks)
Q. 6	Write a short note on Cipher Block Chaining (CBC) Mode of Block Cipher operation.	(4 Marks)
Q. 7	Write a short note on Cipher Feedback (CFB) Mode of Block Cipher operation.	(4 Marks)
Q. 8	Write a short note on Output Feedback (OFB) Mode of Block Cipher operation.	(4 Marks)
Q. 9	Write a short note on Counter (CTR) Mode of Block Cipher operation.	(4 Marks)
Q. 10	Compare various modes of Block Cipher operations.	(8 Marks)
Q. 11	Why is DES not recommended to be used today?	(4 Marks)
Q. 12	List the weaknesses in DES.	(4 Marks)
Q. 13	Write a short note on Double DES.	(6 Marks)
Q. 14	Write a short note on Triple DES.	(6 Marks)
Q. 15	Write a short note on 3DES.	(6 Marks)
Q. 16	What is AES? List its major attributes.	(4 Marks)
Q. 17	Draw the block diagram of AES and explain its working.	(8 Marks)
Q. 18	Compare AES and DES. Which one would you use and why?	(8 Marks)
Q. 19	Describe RC5 Algorithm.	(8 Marks)
Q. 20	Describe the various attacks possible on cryptosystems.	(8 Marks)
Q. 21	Compare differential and linear cryptanalysis.	(8 Marks)

[B] Public Key Cryptography

Q. 22	List the principles of public key cryptosystems.	(6 Marks)
Q. 23	Describe the various key states.	(8 Marks)
Q. 24	Write a short note on Cryptoperiod (Key lifetime).	(4 Marks)
Q. 25	List the Key Management Principles.	(6 Marks)
Q. 26	Describe Symmetric Key Distribution using Symmetric Encryption.	(8 Marks)
Q. 27	Describe Symmetric Key Distribution using Asymmetric Encryption.	(8 Marks)
Q. 28	Describe the ways to distribute asymmetric keys.	(6 Marks)
Q. 29	Describe the ways to distribute public keys.	(6 Marks)

[C] Digital Certificate - X.509

Q. 30	What is a X.509 Certificate? Why is it used?	(4 Marks)
Q. 31	What is a X.509 Certificate? Describe its contents.	(8 Marks)
Q. 32	What is a X.509 Certificate? Describe the steps involved in its generation.	(8 Marks)

[D] Public Key Infrastructure (PKI)

Q. 33	What is PKI? List its components.	(8 Marks)

□□□

Scanned with CamScanner

3 Cryptographic Hash Functions

Syllabus

At the end of this unit, you should be able to understand and comprehend the following syllabus topics :

- Cryptographic hash functions
  - Properties of secure hash functions
  - SHA
  - MD5
  - MAC
  - HMAC
  - CMAC

3.1 Concept Building – Information Accuracy

So far, in the earlier chapters, you learnt about information secrecy – which was all about keeping the message secret. The focus of this chapter is information accuracy which is about ensuring the message integrity.

✍ ***Definition :*** *Information or message accuracy is a process to ensure that the received message is exactly the same as it was sent.*

What does this mean? This means that;

- The message has not been altered in anyway;
  - No addition
  - No modification
  - No deletion
- The message is actually sent by the sender (proof of sender's identity).
- The order or the sequence of the messages is not changed.
- The messages are sent and received within an expected time frame.

Let's consider a day to day scenario. You go to a shop to buy bread and in-exchange you give the shopkeeper a Rupees 100 currency note. The shopkeeper happily takes the note from you, examines it briefly, keeps it in the drawer and return you some change. You take the change, examine the returned currency briefly, slide it down in your wallet and move. What just happened? Why did the shopkeeper take the note you gave without any hesitation? Why did you take the change without any hesitation?

Scanned with CamScanner

What did the shopkeeper examine when you gave the note and why? What did you examine when you received the change and why? Are you telling me, *come on*, this is child simple? Are you trying to explain me the following?

- There were two parties to the transaction – you and shopkeeper.
- Shopkeeper wanted to accurately determine that the note you gave was genuine and not fake.
- Shopkeeper looked at some of the known properties of the note (that were attached to the note itself) and verified it accurately.
- You wanted to accurately determine that the change currency that you got was genuine and not fake.
- You looked at some of the known properties of the note that you got and verified the note accurately.

So, you understand that in this world, where the faith is based on the principle of "trust but verify", you need a mechanism to accurately determine the accuracy of the information that you get.

Have you ever trusted a fake news or video clip and later laughed at yourself or felt bad about trusting the information without checking it genuinely? Don't you feel that there should be *someone* who could just tell you if the information you got was accurate or not? Did you really receive the email that you were sent, or someone modified it on its way ? Did someone modify your file? If all these problems worry you, I request you not to worry, you have a way out. Please heartily welcome Cryptographic Hash functions!

3.2 Message Authentication Functions

At a high level, the message authentication can be performed using three mechanisms :

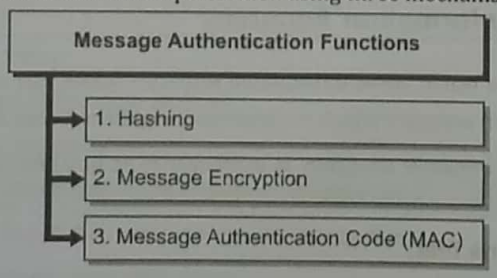


Fig. 3.2.1 : Message Authentication Functions

1. **Hashing :** It deals with producing a unique hash value of the message that can be computed at the sender's and the receiver's end. If the hashes match at both the ends, the message is verified.
2. **Message encryption :** In this, the ciphertext of the entire message can be used to serve as its authenticator. The message is encrypted at the sender's and the receiver's end. If they both get the same ciphertext using the same key, it verifies the message. Note here that the focus is not to encrypt the message but to get the resulting ciphertext to serve as a way to verify the authenticity of the message.
3. **Message Authentication Code (MAC) :** MAC is very similar to hash. It uses a key to calculate the hash value of the message. The MAC is calculated at both the ends (sender's and receiver's) using the same key. If the MAC value matches at both the ends, the message is verified. You will learn about it in detail in the subsequent sections.

Scanned with CamScanner

3.3 Cryptographic Hash Functions

Q. What is the role of a hash function in security ? **MU- Dec. 17, 3 Marks**

Q. What are the requirements of the cryptographic hash functions? State real world applications of hash functions. **MU- May 19, 3 Marks**

As encryption provides message confidentiality, hashing provides message integrity and authentication.

3.3.1 Introduction

Definition : *Hashing is the process of taking any length of input information and finding a unique fixed length representation of that input information.*

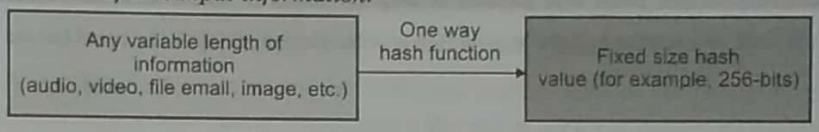


Fig. 3.3.1 : Example of Hash Function

Hashing is the process of finding a unique message digest (or hash value) that corresponds to the input information. The length of input could be just one character or a huge video file. Hashing always produces a fixed size representation of the information. Note here that hashing does not make the information unreadable. It is not same as encryption. Hashing is just a way to attach some additional information to the original information that could later prove that the information is not modified. Remember our discussion from the concept building Section 3.1. The shopkeeper can see Rupees 100 written clearly but verifies the currency note using other properties "attached" to it that proves its originality and authenticity.

3.3.2 How does this Work?

At the source of the information (it could be sender, website, company or anything else where the information is created) the hash value is calculated. This hash value along with the original information is sent to the receiver as shown in Fig. 3.3.2.

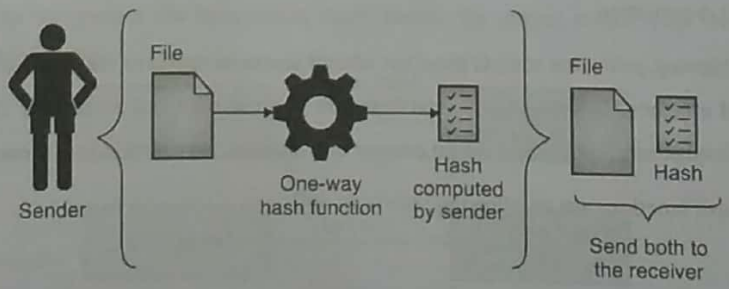


Fig. 3.3.2 : Working of Hash Algorithms

At the destination of the information (it could be a receiver, website, email program, or anything else where the information is received to process further), the hash value is calculated again and matched with the hash value that came with the information from source. If the two hash values (at source and at destination) match, the information is determined to be unmodified and is consumed. But, if the two hash values do not match, it proves that the information is altered and is often rejected as shown in Fig. 3.3.3.

Scanned with CamScanner

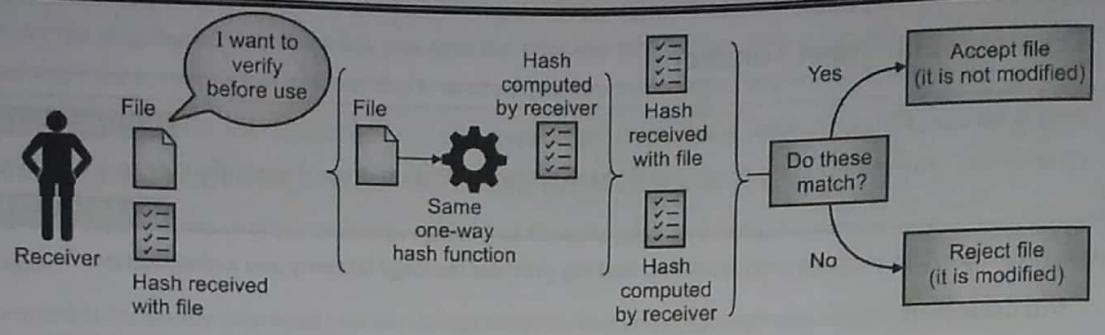


Fig. 3.3.3 : Hashing used for Proving Data Integrity

I want to again stress on the point that hashing is only for integrity checking of the information. The original information may or may not be encrypted. If the information is to be encrypted, following could be one of the sequences for integrity checking.

1. Create information
2. Calculate its hash value
3. Encrypt information
4. Send encrypted information and hash value
5. Receive encrypted information and hash value
6. Decrypt information
7. Calculate its hash value at the receiving end
8. Match source and destination hash
9. If matched, process information
10. If not matched, reject information

Now, you might be thinking, yeah, this sounds good but what if someone captures the message and the hash, alters the message and creates and attaches the new hash to match the altered message and sends it to the receiver. How would receiver ever know? Great question, I must say. I would answer that later on. Let's park it at the moment.

Real world examples of hash functions are as following :

1. Digital Signature
2. Verifying the integrity of downloaded files

3.3.3 Characteristics / Properties of Secure Hash Functions

Q.	What are the properties of hash functions?	MU- Dec. 17, 3 Marks
Q.	What characteristics are needed in secure hash function?	MU- May 18, 4 Marks

Scanned with CamScanner

Unlike encryption, hash functions have some unique properties. Let's learn about them.

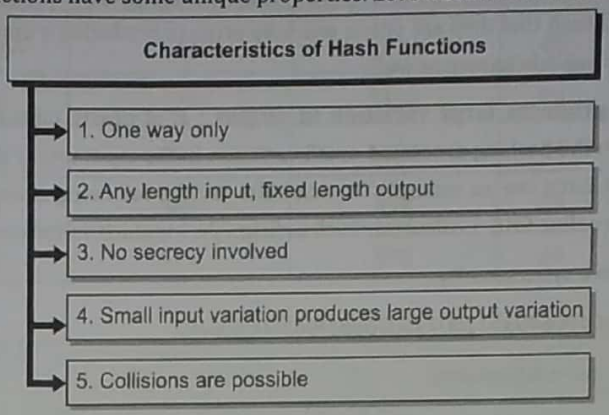


Fig. 3.3.4 : Characteristics of Hash Function

1. **One way only :** This means that it is easy to calculate hash value from information and nearly impossible to convert the hash value back to the original information. Understand it like this. It is easy to convert milk into cheese, but can you convert cheese back to the original milk? Sounds weird and impossible right? That's how hash functions are. One way only.

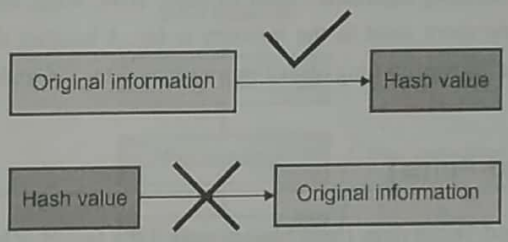


Fig. 3.3.5 : One-way only

2. **Any length input, fixed length output :** Hash functions can work on any length of input. It could work on a single character or a huge video file. It would always produce the same length output. Unlike encryption where the output size is more or less same as the input size, output from hashing process is only a unique representation of the original information and does not contain the information itself. Hence, the output length from hashing does not need to change with the length of input.

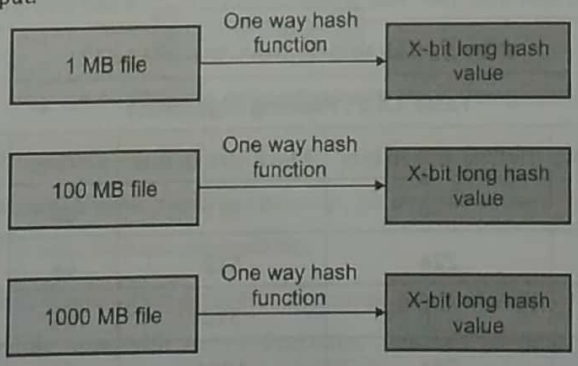


Fig. 3.3.6 : Any Length Input, Fixed Length Output

Scanned with CamScanner

3. **No secrecy involved :** Hashing process does not require a key and neither it requires any secret. The hashing algorithms are implemented such that they are only a one-way process producing a unique representation of the input information. Algorithms are publicly known as well.

4. **Small variation in input produces large variation in output :** It is nearly impossible to establish any relation between input and output in the hashing process. A small variation in the input totally changes the hash output. This is also called avalanche effect. Let's see an example. You can also try the following example at your end by calculating SHA-1 hash output using an online SHA-1 calculator such as http://www.sha1-online.com/.

   Original information : I love cybersecurity.

   SHA-1 Hash value : *653da04906493b11d0735020db1f94360cac64f0.*

   Original information : U love cybersecurity.

   SHA-1 Hash value : *8b9a7e5b4e5c8306c6ba678454e485356cb0aa24.*

Note : It does not matter which online calculator you use. You would get the same SHA-1 output for the given input.

5. **Collisions are possible :** While it is impossible to find original information from hash output, a collision condition is possible. Collision is a situation where two different inputs produce the same hash output. While this condition is extremely rare, some historical hashing algorithms such as MD2, MD4, MD5, SHA-1 have been shown to produce collision. These algorithms are no more used in the industry today. A hashing algorithm is considered strong if it provides high collision resistance. Newer hashing algorithms such as SHA-256 and SHA3-256 provide strong collision resistance.

3.4 Hash Functions (Algorithms)

National Institute of Standards and Technology (NIST) has specified the family of hashing algorithms that may or may not be fit for current use in the industry.

Table 3.4.1 : Family of Hashing Algorithms

Hash Family Name	List of Hashing Algorithms	Approved for use?
SHA-1	SHA-1	No
SHA-2	SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 and SHA-512/256	Yes
SHA-3	SHA3-224, SHA3-256, SHA3-384, SHA3-512	Yes

Table 3.4.2 : Hashing Algorithms

Hashing Algorithm	Output size in bits	Block size in bits	Rounds	First published
SHA-1	160	512	80	1995
SHA-224	224	512	64	2004
SHA-256	256	512	64	2001
SHA-384	384	1024	80	2001
SHA-512	512	1024	80	2001
SHA-512/224	224	1024	80	2012

Scanned with CamScanner

Hashing Algorithm	Output size in bits	Block size in bits	Rounds	First published
SHA-512/256	256	1024	80	2012
SHA3-224	224	1152	24	2015
SHA3-256	256	1088	24	2015
SHA3-384	384	832	24	2015
SHA3-512	512	576	24	2015

3.4.1 SHA-1

Q. Write note on SHA-1. **MU- May 17, 5 Marks**

Q. Explain the operation of secure hash algorithm on 512 bit block. **MU- May 18, 6 Marks**

✍ ***Definition :*** *Secure Hash Algorithm 1 is a cryptographic hash function that produces 160-bit long hash value from a given input.*

Collisions have been found in the algorithm and hence it is avoided in the industry wherever possible.

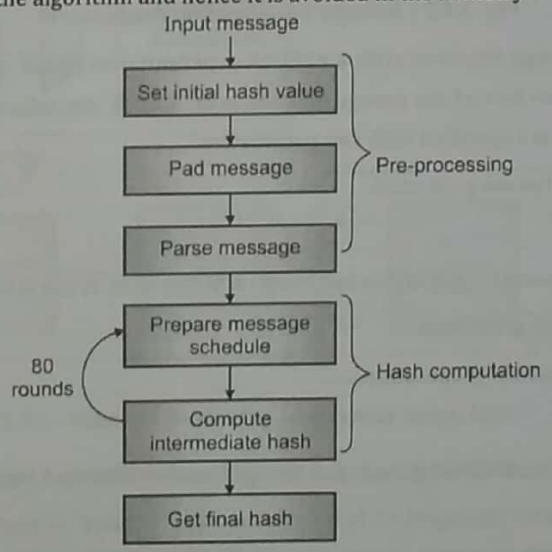


Fig. 3.4.1 : Steps in producing hash value using SHA-1

The algorithm has two stages are shown in Fig. 3.4.1.

1. **Pre-processing :** Pre-processing involves padding a message, parsing the padded message into equal sized blocks and then setting initial values to be used for hash computation.
2. **Hash computation :** The hash computation generates a message schedule (sequence of processing) from the padded message and uses that schedule along with various functions, constants and mathematical operations to generate a series of intermediate hash values per round. The final hash is value is computed after the last round and is 160-bit long.

Scanned with CamScanner

3.4.2 SHA-3

Definition : *Secure Hash Algorithm 3 is the newest addition to the hash function family. It is comprised of various hash functions that compute secure hash values.*

It is considered to be most secure against collision and is highly recommended to be used in the industry. SHA-3 is structurally quite different from earlier versions - SHA-1 and SHA-2. SHA-3 is a subset of the broader cryptographic primitive family KECCAK.

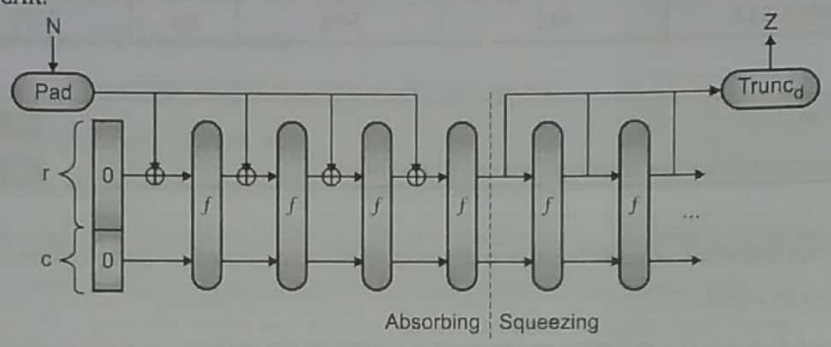


Fig. 3.4.2 : Sponge Construction Framework

KECCAK is the family of all sponge functions with a KECCAK-*p* permutation as the underlying function and multi-rate padding as the padding rule. The idea behind the sponge function is to "absorb" the information and "squeeze out" the hash value. The KECCAK-*p* permutations are specified with two parameters.

1. The fixed length of the strings that are permuted.
2. The number of rounds.

The sponge construction is a framework for specifying functions on binary data. It has three components.

1. Function *f* that works on fixed-length strings.
2. Parameter r that determines the rate of operation.
3. A padding rule denoted by pad.

The sponge function can be denoted as *SPONGE [f, pad, r]*. A sponge function takes two inputs :

1. A bit string that is denoted by N.
2. The bit length that is denoted by d.

So, the overall function is denoted as *SPONGE [f, pad, r](N, d)*.

In the sponge function, Absorb Phase involves :

1. XORing of message blocks into a subset of the state.
2. Using the permutation function for transferring the whole state.

In the sponge function, Squeeze Phase involves :

1. Reading output blocks from the same subset of state.
2. Replacing with state transformation function.

Scanned with CamScanner

3.4.3 MD5

Definition : *MD5 is a hashing algorithm.*

MD5 was designed by Ronald Rivest in 1991.

3.4.3(A) Major Attributes of MD5

- Block size - 512 bits
- Output size (hash size) - 128 bits
- Rounds - 4
- It is broken (collisions have been shown) and obsolete and is no more considered fit for cryptographic use.
- It is replaced by newer algorithms such as SHA-1 and SHA-2.

3.4.3(B) MD5 Algorithm Details

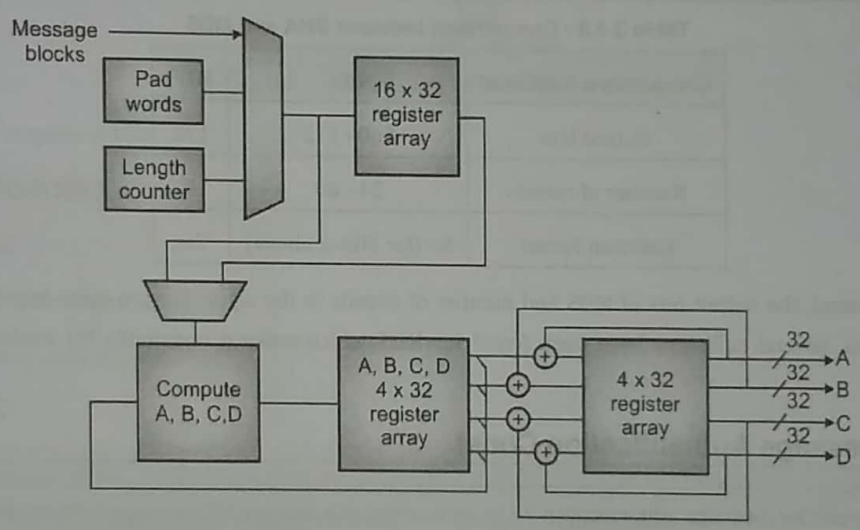


Fig. 3.4.3 : Steps in producing hash value using MD5

MD5 algorithm specifies the following 5 steps for computing the message digest (or the hash value).

1. **Append Padding Bits :** The message is "padded" (extended) so that its length (in bits) is congruent to 448, modulo 512. That is, the message is extended so that it is just 64 bits shorter of being a multiple of 512 bits long. Padding is always performed, even if the length of the message is already congruent to 448, modulo 512. Padding is performed as follows - a single "1" bit is appended to the message, and then "0" bits are appended so that the length in bits of the padded message becomes congruent to 448, modulo 512. In all, at least one bit and at most 512 bits are appended.
2. **Append Length :** A 64-bit representation of the length, of the message before the padding bits were added, is appended to the result of the previous step. After this step, the resulting message length (after padding with bits and with length specification) is an exact multiple of 512 bits.
3. **Initialize MD Buffer :** A four-word buffer (A,B,C,D) is used to compute the message digest. Here each of A, B, C, D is a 32-bit register. These registers are initialized.
4. **Process Message in 16-Word Blocks :** At this stage, the four auxiliary functions take as input three 32-bit words and produce as output one 32-bit word.

Scanned with CamScanner

The functions are;

(a) F(X,Y,Z) = X AND Y OR NOT(X) AND Z

(b) G(X,Y,Z) = X AND Z OR Y AND NOT(Z)

(c) H(X,Y,Z) = X XOR Y XOR Z

(d) I(X,Y,Z) = Y XOR (X OR NOT(Z))

5. **Output** : From these respective four functions, a message digest is produced and stored in the respective registers – A, B, C, D. The final output (message digest or the hash value) is given by concatenating (bringing together) these values.

3.4.4 Comparison between SHA and MD5

Q. Compare MD5 and SHA Hash functions. **MU- Dec. 15, Dec. 16, May 18, May 19, 5 Marks**

Table 3.4.3 : Comparison between SHA and MD5

Comparison Attribute	SHA	MD5
Output bits	160 - 512	128
Number of rounds	24 - 80	4
Collision Found	No (for SHA-2 above)	Yes

As you understand, the output bits of MD5 and number of rounds in the algorithm are quite less compared to SHA family of algorithms. Several collisions have been found for MD5 which make it unsuitable for modern data integrity requirements.

3.5 MAC (Message Authentication Code)

Q. What is the need for message authentication ? List various techniques used for message authentication. Explain any one. **MU- May 18, 10 Marks**

Hash values provide integrity. But what if someone alters the message and adds a new hash? How do you know that you received the original message that the sender had intended to send? Hash values are computed on the original message and usually sent along with the message. Creating hash values does not require any keys. Hence, hash values can only provide integrity but cannot provide any additional assurance that the message is authentic.

✍ ***Definition** : A Message Authentication Code (MAC) is a piece of information that can be used to authenticate a message.*

MAC confirms that the message came from the stated sender and has not been changed. The MAC value protects both a message's data integrity as well as its authenticity. MAC is sometimes also called as *keyed* hash.

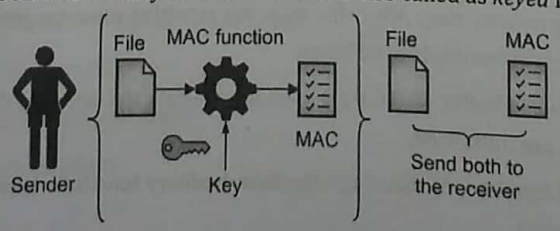


Fig. 3.5.1 : Sender's side (MAC Computed)

Scanned with CamScanner

The sender intends to send a file (or a message). He computes the MAC value for the file using the pre-shared key. He then forwards the message as well as the corresponding MAC value to the receiver.

At the receiver's end, the MAC value is again computed using the same pre-shared key. This computed MAC value is then compared with the received MAC. If these two match, the file (or the message) is not altered and is authenticated.

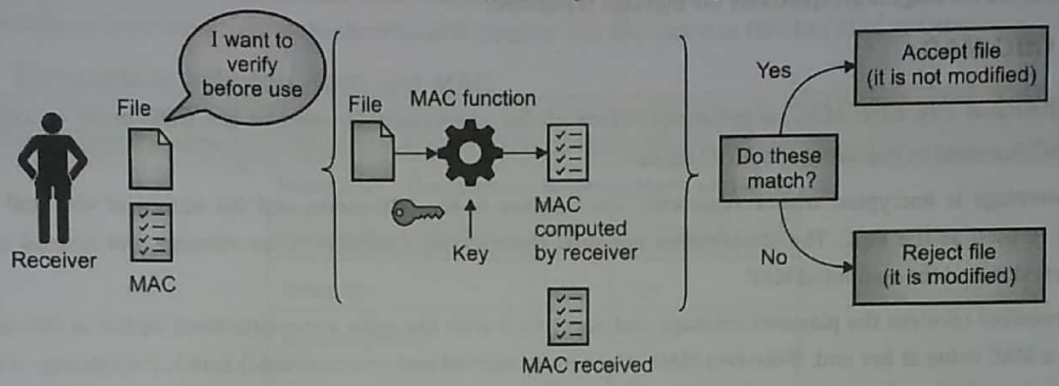


Fig. 3.5.2 : Receiver's side (MAC Verified)

There are three types of MAC :

1. HMAC (Hash MAC)
2. CBC-MAC
3. CMAC (Cipher-based MAC)

Let's learn about each of them.

3.5.1 HMAC

Definition : *In HMAC, a hashing function is used as a MAC function to calculate the MAC value.*

The hashing function could be general hash functions such as MD5, SHA-1, or SHA-2.

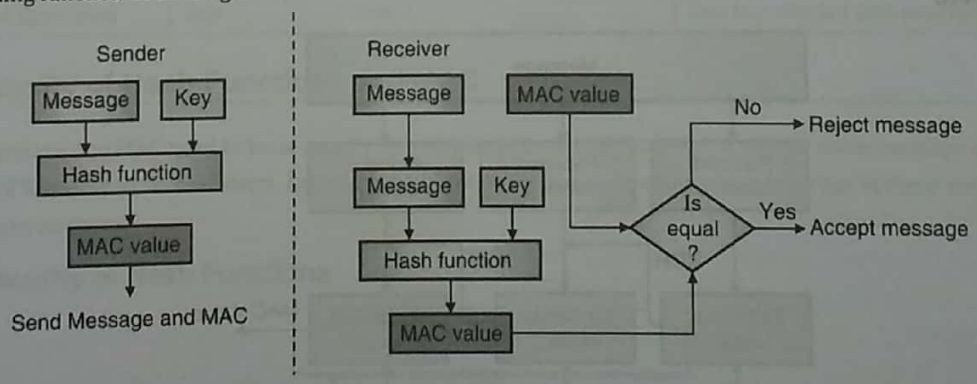


Fig. 3.5.3 : HMAC

On the sender's side, the message and a pre-shared key is passed together into a hash function to compute a MAC. The computed MAC along with the original message is sent to the receiver. The key is not sent.

Scanned with CamScanner

Note here that the key is not used to encrypt the message. The key just provides a random value that when passed with the message to a hashing function generates a MAC value. The key has no other role except to provide a random value.

On the receiver's side, the message is again passed through the same hashing function using the same pre-shared key. A MAC value is computed at the receiver's side and is matched with the MAC value that came from the sender. If the two MAC values match, the message is accepted else the message is rejected.

3.5.2 CBC-MAC

Definition : *In CBC-MAC, a symmetric block cipher encryption function in the CBC mode is used as the MAC function to calculate the MAC value.*

The message is encrypted with a symmetric block cipher in the CBC mode, and the output of the final block of ciphertext is used as the MAC. The sender does not send the encrypted version of the message, but instead sends the plaintext version and the computed MAC.

The receiver receives the plaintext message and encrypts it with the same symmetric block cipher in CBC mode and calculates a MAC value at her end. If the two MAC values (one received and one computed) match, the message is accepted else it is rejected. This method does not use a hashing algorithm as in HMAC.

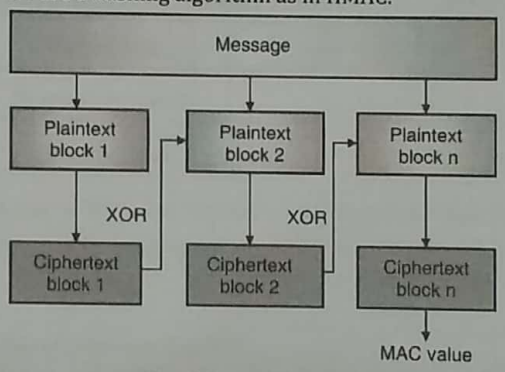


Fig. 3.5.4 : CBC-MAC

3.5.3 CMAC

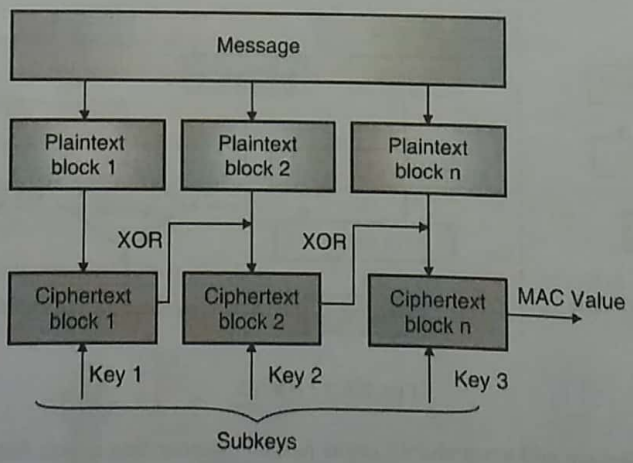


Fig. 3.5.5 : CMAC

CMAC (Cipher-based MAC) is very similar to CBC-MAC.

Scanned with CamScanner

***Definition** : In CMAC, a symmetric block cipher encryption function is used as the MAC function to calculate the MAC value.*

Note here that like CBC-MAC, a symmetric block cipher encryption function is used here as well but not in CBC mode. That's the major difference between CBC-MAC and CMAC. CMAC is typically calculated using AES-128 algorithm and provides strongest form of message authentication and integrity. It is also called as One-Key MAC or OMAC.

3.5.4 Comparison between Hash and MAC

The Table 3.5.1 summarizes the difference between Hash and MAC.

Table 3.5.1 : Comparison between Hash and MAC

Comparison Attribute	Hash	MAC
Integrity	Yes	Yes
Authentication	No	Yes
Non-repudiation	No	No
Keys Used	None	Symmetric Keys

3.5.5 Comparison between HMAC, CBC-MAC and CMAC

The Table 3.5.2 summarises the key differences between HMAC, CBC-MAC and CMAC.

Table 3.5.2 : Comparison between HMAC, CBC-MAC and CMAC

Comparison Attribute	HMAC	CBC-MAC	CMAC
MAC generation function	Hash Function	Symmetric cipher in CBC mode	Symmetric cipher
Speed of MAC generation	Highest	Lowest	Medium
Strength of MAC	High	Very High	Very High
Number of Keys used	One	One	One key divided into multiple sub-keys

3.6 Security of Hash Functions and MAC

Hash functions and MAC need to be secure by themselves to be useful. You cannot rely on these message authentication mechanisms if it is easy to "break" them. Let's understand some of the desired security properties of these mechanisms and possible attacks on them.

3.6.1 Security of Hash Functions

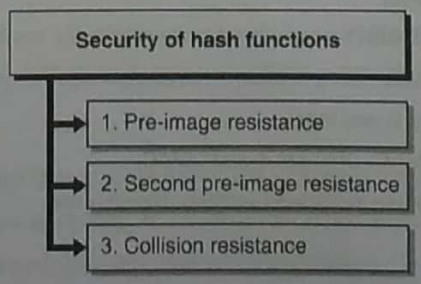


Fig. 3.6.1 : Security of Hash Functions

Scanned with CamScanner

The three desired security properties of hash functions are as following.

1. **Pre-image resistance :** This property ensures the one-way only security criteria of hash functions. It states that for a given hash value h, it should be impossible to find any message m such that, h = hash (m). So, in a nutshell, you cannot find the original message from its hash value.
2. **Second pre-image resistance :** This property states that given a message m_1, it is hard to find another message m_2, such that hash(m_1) = hash(m_2). So, looking at a message it should be hard to find another message which could produce the same hash. This property is also called weak collision resistance.
3. **Collision resistance :** This property states that it should be hard to find any two messages pairs m_1 and m_2 such that hash(m_1) = hash(m_2). This property is also called strong collision resistance.

3.6.2 Attacks on Hash Functions and MAC

There are two possible ways to attack hash functions and MAC.

1. **Brute-force attack :** In this, the attacker repeatedly tries to find various combinations of messages so as to produce a collision. The brute-force attack is more difficult to carry out on MAC than hash functions.
2. **Cryptanalysis :** Similar to finding weakness in the encryption algorithms, the attackers tries to find a weakness in the hash functions and exploit that weakness to carry out further attacks.

Review Questions

Here are a few review questions to help you gauge your understanding of this chapter. Try to attempt these questions and ensure that you can recall the points mentioned in the chapter.

[A] Cryptographic Hash Function

Q. 1 What is Hashing? Describe its properties. **(8 Marks)**

Q. 2 With a block diagram, explain how can you use hashing for ensuring message integrity? **(8 Marks)**

Q. 3 Write a short note on avalanche effect in hashing. **(4 Marks)**

Q. 4 Write a short note on collision in hashing. Which hashing algorithm would you suggest using today and why? **(8 Marks)**

Q. 5 With a block diagram, describe SHA-1. **(8 Marks)**

Q. 6 With a block diagram, describe SHA-3. **(8 Marks)**

Q. 7 With a block diagram, describe MD5. **(8 Marks)**

Q. 8 What is MAC (Message Authentication Code)? Why is it used over hashing? **(6 Marks)**

Q. 9 What is MAC (Message Authentication Code)? How does it work? **(8 Marks)**

Q. 10 With a block diagram, explain HMAC. **(8 Marks)**

Q. 11 With a block diagram, explain CBC-MAC. **(8 Marks)**

Q. 12 With a block diagram, explain CMAC. **(8 Marks)**

Q. 13 Provide a comparison between Hash and MAC. **(8 Marks)**

Q. 14 Provide a comparison between HMAC, CBC-MAC and CMAC. **(8 Marks)**

Q. 15 Explain the desired security properties that hash functions should have. **(6 Marks)**

□□□

Scanned with CamScanner

4 Authentication Protocols and Digital Signature Schemes

Syllabus

At the end of this unit, you should be able to understand and comprehend the following syllabus topics :

- User Authentication and Entity Authentication
  - Password Based
  - Challenge Response Based
    - Needham Schroeder Authentication protocol
    - Kerberos Authentication protocol
- Digital Signature
  - Attacks on Digital Signature
  - Digital Signature Scheme: RSA

4.1 User and Entity Authentication

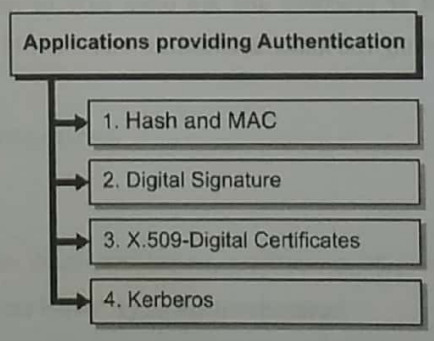


Fig. 4.1.1 : Applications provide Authentication

Authentication is a way to build trust in communicating parties or systems. Authentication serves as the crucial protection mechanism towards securing a communication – ensuring that the right set of parties are involved in the communication and no one else.

Broadly speaking, there are four ways that provide major authentication services today. You have learnt about the first three so far. Let's learn the fourth one - Kerberos. But before that, let's learn about some remote user authentication principles and the ways in which various authentication mechanisms work.

Scanned with CamScanner

Remote User Authentication Principles

Overtime, various authentication methods have evolved to address :

- The ease of authentication.
- Make it hard to break authentication.
- Make authentication techniques suitable for various devices.

At a high level, authentication methods are categorised as following.

4.1.1 Types of Authentication Methods

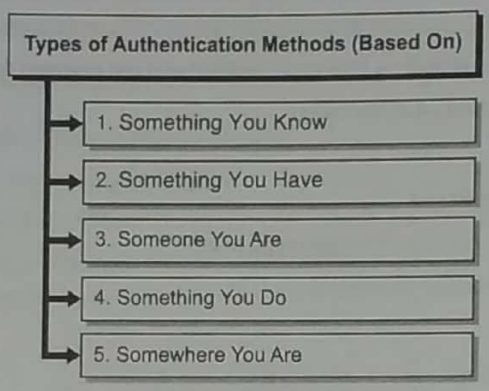


Fig. 4.1.2 : Types of Authentication Methods

1. Something You Know

✍ ***Definition** : The authentication methods based on "Something You Know" rely upon your secret knowledge about something.*

These secrets could be memorised or practiced and are often easy to recall for you. Some of the examples of authentication methods based on "Something You Know" are;

- Passwords
- PIN
- Passphrases
- Secret questions
  - Mother's maiden name
  - First pet name
  - First car purchase year
  - First school name
  - City in which you were born
  - Or other secret questions whose answers would be likely known just to you
- Lock key combination

Scanned with CamScanner

Advantages of "Something You Know" type authentication

- Easy to implement by developers in the product (OS, Applications or Websites)
- Easy to recall for the user
- Easy to authenticate for the user
- Easy to change for the user
- Chances of errors are low
- Can be used by non-person entities (hardware devices, applications, OS, programs, processes, etc.)

Disadvantages with "Something You Know" type authentication

Easy to crack (or break)

2. Something You Have

✍ ***Definition :*** *The authentication methods based on "Something You Have" rely upon your possession of something.*

You could possess something that could let you authenticate using it. You often don't require to remember anything while using it for authentication purpose. Some of the examples of authentication methods based on "Something You Have" are,

- Physical keys
- Badge
- Swipe Card (for example, Debit or Credit Card)
- Digital Certificates
- Security Keys (for example, Private Key)
- OTP (that you get via SMS)
- Tokens

Advantages of "Something You Have" type authentication

- Easy to use
- Does not often require remembering secrets
- Chances of errors are low
- Not easy to crack
- Can be used by non-person entities (hardware devices, applications, OS, programs, processes, etc.)

Disadvantages of "Something You Have" type authentication

- Difficult to change
- Possibility of loss or theft
- Requires distribution methods (provisioning) to reach to the user securely.

3. Someone You Are

✍ ***Definition :*** *The authentication methods based on "Someone You Are" rely upon your physical characteristics.*

Scanned with CamScanner

Your body has several physical characteristics that can be used to uniquely identify and authenticate you. These characteristics do not much change over time (as you age) and can serve authentication purpose for near lifetime. These characteristics are called Static Biometrics.

Definition : *Static Biometrics are physical characteristics that can be used for authentication.*

Generally speaking,

Definition : *The measurement and analysis of unique physical or behavioural characteristics is called biometrics.*

However, there can be scenarios where re-provisioning (re-calibrating) your biometric details might be required. For example, if you use a particular finger for fingerprint and your that particular finger is damaged permanently, you might have to choose another finger or another type of biometrics for authentication. Some of the examples of authentication methods based on "Someone You Are" are;

- Fingerprint
- Palm Scan
- Hand Geometry
- Retina Scan
- Iris Scan
- Facial recognition

Advantages of "Someone You Are" type authentication

- Easy to use
- Does not often require remembering secrets
- Difficult to crack

Disadvantages of "Someone You Are" type authentication

- Difficult to implement correct.
- Chances of errors are high (recall you trying fingerprints several times at Aadhar enrolment centre?).
- Difficult to change (requires physical presence for re-provisioning).
- Cannot be used by non-person entities (hardware devices, applications, OS, programs, processes, etc.).

4. Something You Do

Definition : *The authentication methods based on "Something You Do" rely upon your way of performing a given task.*

This authentication method also collects biometric patterns but while performing a given task. Unlike static biometric patterns, these are dynamic biometric patterns which are used for authentication purpose.

Definition : *Dynamic Biometrics are job performing characteristics that can be used for authentication while an individual is performing a given task.*

Some of the examples of authentication methods based on "Something You Do" are,

- Voice print (or pattern).
- Keystroke Dynamics (how hard you press the keys and how fast).
- Handwriting characteristics (remember old movies where handwriting matches were done?).

Scanned with CamScanner

Advantages of "Something You Do" type authentication

- Easy to use.
- Does not often require remembering secrets.

Disadvantages of "Someone You Are" type authentication

- Difficult to implement correct.
- Not too difficult to crack (consider replaying a recorded voice).
- Chances of errors are high (have you tried voice to text yet?).
- Difficult to change (requires physical presence for re-provisioning).
- Cannot be used by non-person entities (hardware devices, applications, OS, programs, processes, etc.).

5. Somewhere You Are

Definition : *The authentication methods based on "Somewhere You Are" rely upon your physical location.*

Increasingly, the devices and systems have location awareness. Your mobile phone or laptops (via network connectivity) precisely know where you are located at a particular moment. "Somewhere You Are" uses the location information for authentication.

For example, Google Smart Lock for Android allows you to set Trusted Locations. Say, if you are at home, your phone may not require you to unlock it (via PIN, pattern, password or fingerprint) before use.

https://support.google.com/accounts/answer/6160273?hl=en

Set your Android device to automatically unlock

You can keep your Android phone or tablet unlocked in some situations, like when your phone is in your pocket or you're near home. When you use Smart Lock, you won't need to unlock with your PIN, pattern, or password. The features you can use depend on your device.

If you want to change your screen lock, learn more about screen lock settings.

Note: Some of these steps work only on Android 9 and up. Learn how to check your Android version.

Turn on automatic unlock

1. Make sure you have a screen lock. Learn how to set a screen lock.
2. Open your device's Settings app.
3. Tap **Security & location** > **Smart Lock**.
4. Enter your PIN, pattern, or password.
5. Pick an option and follow the on-screen steps.

After setup, when you turn on your screen, you'll see a pulsing circle at the bottom around the Lock.

Important: When you don't use your device for 4 hours, and after it restarts, you'll need to unlock it.

Turn off automatic unlock

1. Open your device's Settings app.
2. Tap **Security & location** > **Smart Lock**.
3. Enter your PIN, pattern, or password.
4. Turn off **On-body detection** and remove all trusted devices, trusted places, trusted faces, and Voice Match voices.
5. Optional: If you want to turn off your screen lock, learn how to change your screen lock.

Scanned with CamScanner

"Somewhere You Are" is also quite widely used in corporate IT. In many environments, if you are on an office network (LAN or Wi-Fi), you can login using only a password, but if you are out of the office you must use VPN or an additional mechanism for authentication.

Advantages of "Somewhere You Are" type authentication

- Easy to use.
- Does not often require remembering secrets.
- Can be used for authenticating non-person entities (mobile, laptops or other location aware devices).

Disadvantages of "Somewhere You Are" type authentication

- Not too difficult to crack (consider theft and someone carrying the device to a trusted location).
- Cannot be used for authenticating individuals.
- Requires network connectivity for location awareness.

4.1.2 Comparison between the Authentication Types

Table 4.1.1 : Comparison between the Type of Authentication

Type of authentication	Ease of use	Ease of Change	Ease of implementation	Error rate	Support non-person entities
Something You Know	High	High	High	Low	Yes
Something You Have	High	Low	Medium	Low	Yes
Someone You Are	High	Low	Low	High	No
Something You Do	Medium	Low	Low	High	No
Somewhere You are	High	High	Medium	Medium	Yes

4.2 Factors of Authentication

Each type of authentication that you learnt in this section is called a factor of authentication. Based on the number of factors you choose for effectively carrying out and completing the authentication process, you have three types of authentication scenarios.

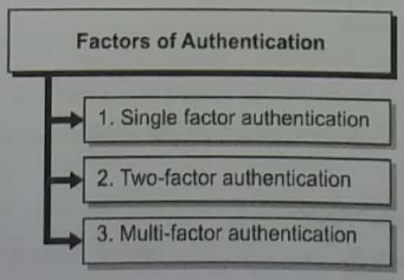


Fig. 4.2.1 : Factors of Authentication

1. Single Factor Authentication

Definition : *Single factor authentication requires only ONE of the types of authentication for successfully carrying out the authentication process.*

Scanned with CamScanner

For example, you could just use password or token. This is most widely used. It is often treated as a weak form of authentication.

2. Two-factor Authentication

✍ ***Definition :*** *Two-factor authentication requires you to use any TWO types of authentication, one after another, for successfully carrying out the authentication process.*

It is considered as a strong form of authentication. For example, for your online transactions, you are first required to give the account password and then you receive an OTP. You are required to put the correct OTP for successfully authenticating your account details and carrying out and completing your transaction. Another example could be ATM. You must possess you Debit Card (Something You Have) and put in the right PIN (Something You Know) for withdrawing money.

3. Multi-factor Authentication

✍ ***Definition :*** *Multi-factor authentication requires you to use MORE than two types of authentication.*

It is often used in a high security environment. For example, you may be first required to give your fingerprint, after which you can access an application where you are required to provide username and password. For carrying out a transaction on that application, you might require an OTP.

Note :	If two authentication techniques from the same type of authentication are used, it is not considered two-factor authentication. It would be considered a single factor authentication. For example, you cannot consider consecutive requirement of two passwords or a password and a PIN to be two-factor authentication. Two-factor authentication essentially requires two different ways to authenticate.

4.3 One-way and Mutual Authentication Schemes

4.3.1 Needham Schroeder Authentication Protocol (Challenge Response Based Authentication)

Needham Schroeder proposed two authentication protocols – one using symmetric keys and one using asymmetric keys. Let's learn about both of them.

4.3.1(A) The Needham–Schroeder Symmetric Key Based Authentication Protocol

In symmetric key based authentication protocol, there are 3 entities :

- 2 users – let's call them Alice (A) and Bob (B)
- 1 Server (S)

✍ ***Definition :*** *The goal of this protocol to generate and share a key that can be used for securing communication between the two users - A and B.*

Note here that the Needham–Schroeder Symmetric Key Based Authentication Protocol forms a basis for Kerberos based authentication. It solves the key distribution problem.

Assume the following primitives :

- A and B are identities of Alice and Bob respectively,
- K_{AS} is a symmetric key known only to A and S,

Scanned with CamScanner

- K_{BS} is a symmetric key known only to B and S,
- N_A and N_B are nonce (random number used once) generated by A and B respectively,
- K_{AB} is the symmetric key that needs to be generated and shared between A and B for secure communication.

Protocol Operation

Following is the sequence of activities that the protocol follows.

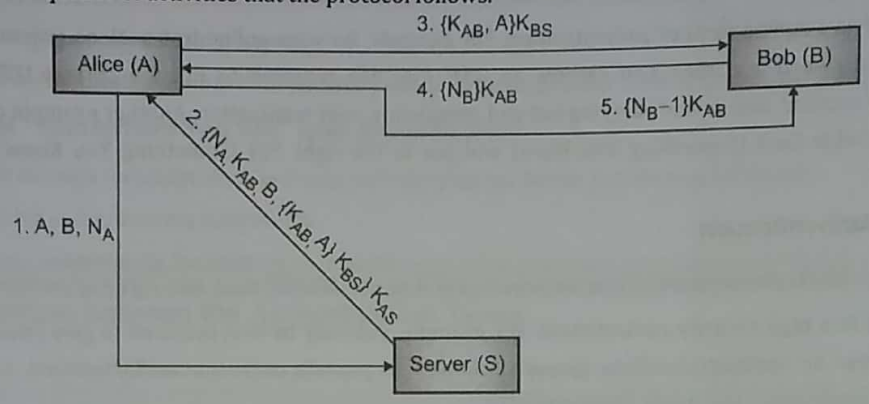


Fig. 4.3.1 : Protocol Workflow

1. Alice sends a message to the server identifying herself and telling the server that she wants to communicate with Bob.
2. The server generates K_{AB} and sends it to Alice encrypting the entire response with K_{AS}.
   (a) A copy of K_{AB} is encrypted using K_{BS} for Alice to forward to Bob and also identify herself and,
   (b) A copy for Alice herself.
3. Alice forwards the key K_{AB} to Bob. Bob decrypts it with the key K_{BS}.
4. Bob sends Alice a nonce N_B encrypted using K_{AB} to show that he has the key.
5. Alice performs a simple operation on the nonce N_B, re-encrypts it and sends it back verifying that she is still alive and that she holds the key as well.

Attack on the protocol

This protocol is vulnerable to replay attack. The attacker can grab the older and compromised value for K_{AB}. He can then replay the message $\{K_{AB}, A\}K_{BS}$ to Bob, who will accept it. Kerberos solves this problem by adding timestamp to avoid replaying older communication.

4.3.1(B) The Needham–Schroeder Asymmetric Key Based Authentication Protocol

In asymmetric key based authentication protocol, there are 3 entities as well

- 2 users – let's call them Alice (A) and Bob (B)
- 1 Server (S)

Definition : *The goal of this protocol is to share the respective public keys between the two users - A and B.*

Assume the following primitives:

- 3 key pairs : P stands for Public Key, Q stands for Private Key,
- K_{PA} and K_{QA} : Public and Private Keys of A respectively,
- K_{PB} and K_{QB} : Public and Private Keys of B respectively,

Scanned with CamScanner

- K_{PS} and K_{QS}: Public and Private Keys of S respectively,
- K_{PS} is known to both A and B and is trusted.

Protocol Operation

Following is the sequence of activities that the protocol follows.

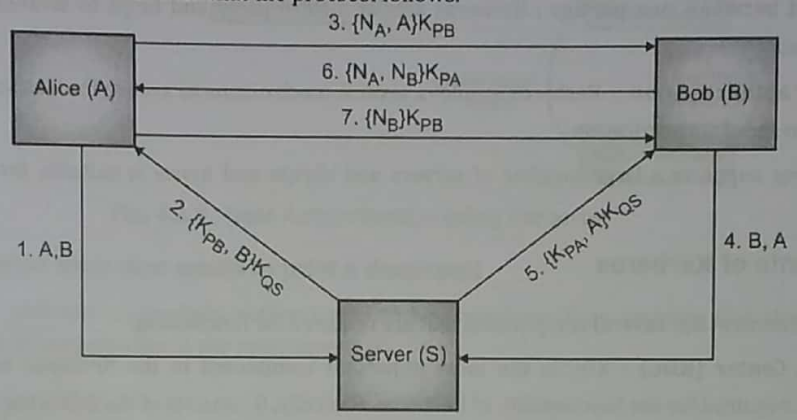


Fig. 4.3.2 : Protocol Workflow

1. A requests B's public keys from S.
2. S responds with B's public key K_{PB} alongside B's identity, encrypted by the server's private key K_{QS}.
3. A chooses a random nonce N_A and sends it to B by encrypting it using B's public Key K_{PB} received in step 2.
4. B now knows that A wants to communicate. So, B requests A's public keys from S.
5. S responds with A's public key K_{PA} alongside A's identity, encrypted by the server's private key K_{QS}.
6. B chooses a random nonce N_B and sends it to A along with N_A to prove his ability to decrypt with K_{QB}.
7. A confirms N_B to B, to prove her ability to decrypt with K_{QA}.

Attack on the protocol

This protocol is vulnerable to Man-in-the-Middle attack. The attacker can make A and B believe that they are communicating. This could be fixed by updating the step 6 by passing along the identity - $\{N_A, N_B, B\}K_{PA}$. So, A would know who she is actually communicating with instead of being attacked by the middle-man.

4.3.2 Kerberos Authentication Protocol

Q.	Explain working of Kerberos.	MU - May 16, 10 Marks
Q.	Explain Kerberos protocol that supports authentication in distributed system.	MU - May 18, 10 Marks
Q.	Write short notes on Kerberos.	MU - May 19, 4 Marks

Definition : *Kerberos is a network-based authentication protocol.*

It was developed around mid-1980s at MIT. It works on the client/server model and uses symmetric key cryptography. Today, Kerberos is extensively used for authentication in Microsoft Windows, Unix, Linux and Apple OS.

Scanned with CamScanner

4.3.2(A) Problems Addressed by Kerberos

1. **Sharing passwords over the network :** With Kerberos, you need not share passwords over the network for authentication.
2. **Establishing trust between two parties :** Kerberos acts as a third party and helps to establish trust between two non-trusting parties.
3. **Difficult to spoof authentication :** Kerberos employs several mechanisms to secure the authentication process and makes it difficult to spoof authentication.
4. **Scalable :** Kerberos supports a large number of servers and clients and hence is suitable for distributed network architectures.

4.3.2(B) Components of Kerberos

The Kerberos environment has several components that are required for functioning.

1. **Key Distribution Center (KDC) :** KDC is the most important component in the Kerberos environment. It holds all the information required for the functionality of Kerberos. Basically, it consists of the following sub-components.
   - (a) **Authentication Service (AS) :** The Authentication Service (AS) issues TicketGranting Tickets (TGTs) that is used to connect to the Ticket Granting Service (TGS). It also verifies principals that require authentication.
   - (b) **Ticket Granting Service (TGS) :** The TGS issues the tickets to the clients using which the clients can connect to the desired server.
   - (c) **Principals :** In the Kerberos terminology, Principals can be users, clients, servers, applications or network services that require authentication. The KDC has the information about each principal account and its secret key. For example, a user could be a principal requiring access to a print server that could be another principal.
   - (d) **KDC Database :** All the principal related information is stored in the KDC database.
   - (e) **Realm :** In Kerberos terminology, a realm is the set of principals who can authenticate to each other. It is a logical grouping of principals. One KDC can have one realm or several realms.
2. **Client :** Typically, a client is the principal that requires to authenticate to another principal (server).
3. **Server :** Typically, a server is the principal that holds resources that the client is interested in and provides the resources that can be consumed after successful authentication.

4.3.2(C) How does it work?

Kerberos heavily uses the concept of tickets that works very much like your train ticket. A ticket is just a temporary proof. Let's understand the working in detail.

Scenario 1 - User authenticating to a computer

1. The user enters the username and password on the computer.
2. The Kerberos software running on the computer sends the username to the Authentication Service (AS).
3. The Authentication Service checks if the username is present. If yes, it sends back a Ticket Granting Ticket (TGT) which is encrypted with the user's pre-shared secret key (password).
4. If the user entered the correct password, she can decrypt the TGT and then is granted access to the computer.

Scanned with CamScanner

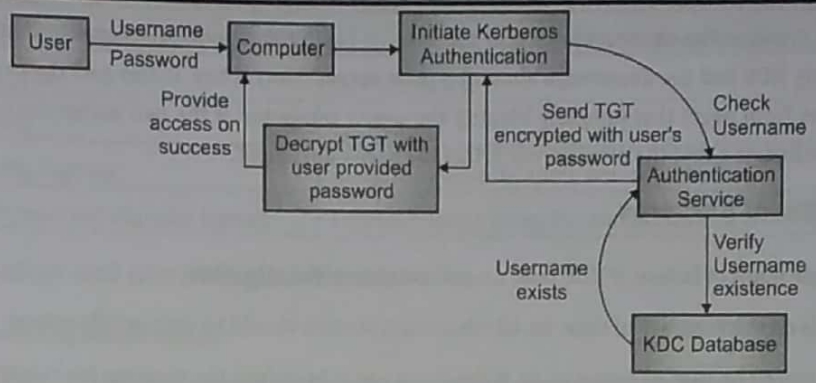


Fig. 4.3.3 : User Authentication using Kerberos

Scenario 2 - Authenticated User now wants to print a document

From Scenario 1, the user has successfully authenticated to her computer. Now, suppose that she wants to print a document and hence needs to authenticate to the print server.

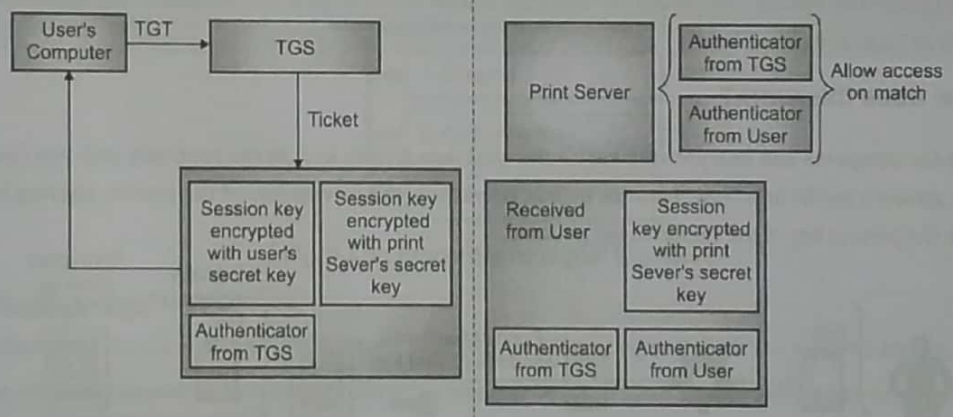


Fig. 4.3.4 : User Authenticating to a Service using Kerberos

The following steps are taken :

1. The computer sends the TGT it received earlier (when the user wanted to authenticate) to the TGS. The TGT is a proof for the TGS that the user has already been successfully authenticated (as in scenario 1).
2. TGS creates a new ticket and puts two copies of the same session key (temporary secret key) in it. It encrypts the first copy of the session key with the user's secret key and the second copy with print server's secret key. This ticket also contains an authenticator information that holds the value of the user's computer's IP Address, sequence number and timestamp from where the TGT came. It then sends the ticket to the user's computer.
3. The user decrypts the ticket created by the TGS with her secret key and obtains the session key. It adds another set of authenticator information (computer's IP Address, sequence number and timestamp) to it and sends the ticket to the print server.

Scanned with CamScanner

4. The print server receives the ticket and extracts the session key by decrypting it. It knows that the KDC created the ticket because only KDC had the knowledge about the print server's secret key. It also gets the two authenticators (one from TGS and one from user) that uniquely identify the user's computer. If the two authenticators match, the user is successfully authenticated, and the print server prints the user's document.

4.3.2(D) Limitations of Kerberos

1. KDC can be a single point of failure. If KDC fails, no authentication can take place.
2. Kerberos depends on the accuracy of time. So, all clients and servers should be time synchronized.
3. Session keys are stored on user's computer. So, if the computer is breached, the sessions key might be stolen.

4.4 Digital Signature

Q. What is a digital signature? **MU - May 16, Dec. 16, 4 Marks**

Definition : *A digital signature is a hash value that has been encrypted with the sender's private key. The act of signing means encrypting the message's hash value with a private key (since no one else knows the sender's private key).*

4.4.1 How does this work?

So, the sender computes and encrypts the hash value with her private key. At the receiving end, you decrypt the hash value with the sender's public key. Now, because no one else knows the private key of the sender, altering hash value and re-signing with the private key of the sender is not possible.

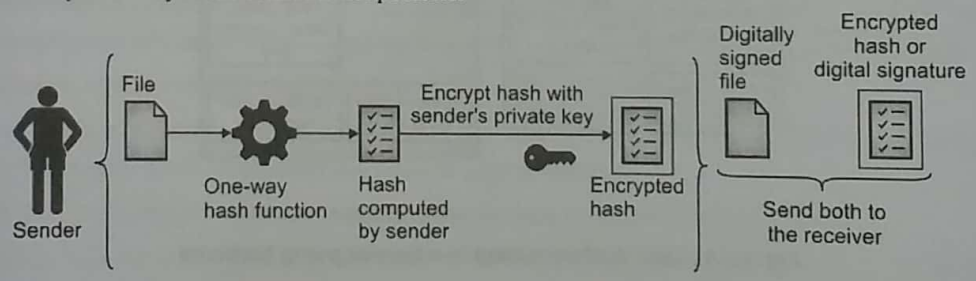


Fig. 4.4.1 : Digital Signature Generation at the Sender's end

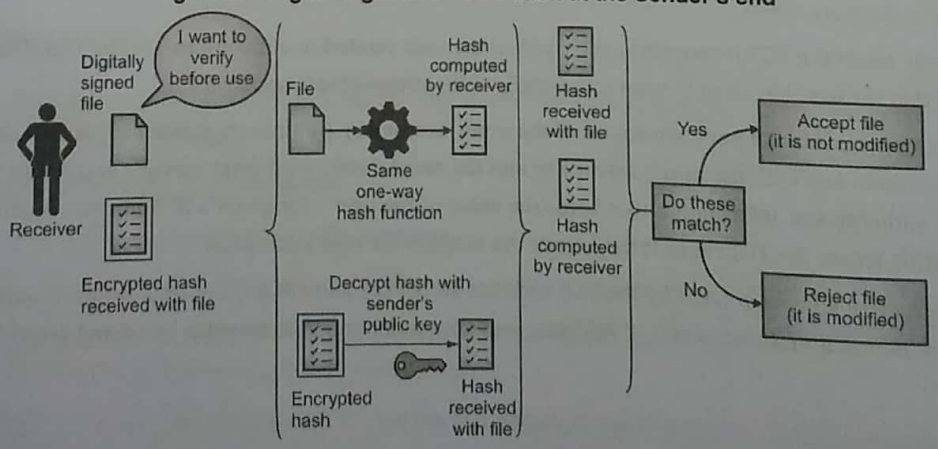


Fig. 4.4.2 : Digital Signature verification at the Receiver's end

Scanned with CamScanner

Processing applied on a message	Security Property achieved
Encryption	Confidentiality
Hashing	Integrity
Digitally Signing	Integrity, authentication, non-repudiation
Encryption and digitally signing	Confidentiality, Integrity, authentication, non-repudiation

4.4.2 Application and use of Digital Signature

1. Sending and receiving secure emails.
2. Signing documents. For example, you can sign income tax returns using digital signature.
3. Sending and receiving important files. For example, insurance policy documents, Aadhar card e-letter, etc.

4.4.3 Properties of Digital Signature

Digital signature provides three security properties.

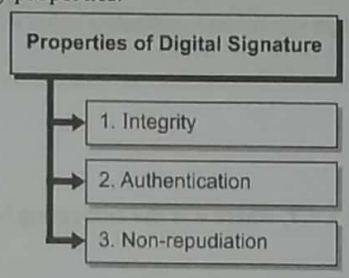


Fig. 4.4.3 : Properties of Digital Signature

1. **Integrity :** Via hash value calculation.
2. **Authentication :** Via the ability to prove sender's identity by decrypting hash with the sender's known public key.
3. **Non-repudiation :** Sender cannot deny sending the message because she used her private key to encrypt the hash.

4.5 Attacks on Digital Signature

Attacks on digital signature are a combination of attacks on cryptosystems and hashing. Please refer to those sections for a refresher.

4.6 RSA Digital Signature Scheme

Q. Explain any digital signature algorithm in detail. **MU - May 16, Dec. 16, 6 Marks**

✍ ***Definition :*** *RSA signatures are based on public key cryptography.*

RSA uses public key cryptography for creating and verifying digital signatures.

Key Generation

RSA digital signatures work on public and private key pairs. They can be generated by the regular key pair generating method by a Certificate Authority (CA) or on the user's system by herself. Recall from your reading on RSA algorithm that the keys are as following :

- The public key = (n, e)
- The private key = (n, d)

Scanned with CamScanner

Message Signing

To sign a message, M

- Calculate the hash value of the message M at sender's end
  h = hash(M)
- Encrypt *h* using RSA private key
  Signature S = $(h)^d$ mod n

Signature Verification

- Decrypt Signature *S* using public key
  h' = $(S)^e$ mod n
- Calculate the hash value of the message M at receiver's end
  h = hash(M)
- If *h* = *h'*, the signature is valid else the signature is invalid

Ex. 4.6.1 : Alice chooses public key as (7, 33) and B chooses public key as (13, 221). Calculate their private keys. A wishes to send message m = 5 to B. Show the message signing and verification using RSA digital signature.

MU - May 19, 10 Marks

Soln. :

For Alice

Public key is (7, 33). Hence, n = 33 and e = 7, where e is the encrypting key. Let's calculate Alice's private key which would give us d, decrypting key.

Assume that 3 and 11 were the chosen prime factors for n = 33.

n = p * q = 3 * 11 = 33

r = (p – 1) * (q –1) = (3 –1) * (11– 1)

r = 2 * 10 = 20

d = e^{-1} mod r

ed ≡ 1 mod 20

7d ≡ 1 mod 20

Let's calculate modulo inverse using extended Euclidean algorithm (swapping 7 and 20)

Index i	quotient q for i	Remainder r for i	s for i	t for i
0		20	1	0
1		7	0	1
2	20 / 7 = 2	20 – 7 *2 = 6	1 – 0*2 = 1	0 – 1*2 = –2
3	7 / 6 = 1	7 – 6 * 1 = 1	0 – 1 * 1 = –1	1 – (–2)*1 = 3
4	6 / 1 = 6	6 – 1 * 6 = 0	Do not calculate	Do not calculate

- Let's re-swap the values.
  x = 3
  y = –1
- Putting the values in the extended Euclidean algorithm, you get
  ax + by = 1

Scanned with CamScanner

$3*7 + 20*(-1) = 1$

- Since, you have to find multiplicative inverse in mod 20, divide both sides by mod 20.

  $3*(7) \bmod 20 + 20*(-1) \bmod 20 = 1 \bmod 20$

  $3*(7) \bmod 20 + 0 = 1 \bmod 20$

  $3*(7) \bmod 20 = 1 \bmod 20$

- Hence, multiplicative inverse is 3.
- Therefore, the value of decrypting key, $d = 3$.
- Hence, the Private Key of user Alice is (3, 33).

For User B

Public key is (13, 221). Hence, n = 221 and e = 13, where e is the encrypting key. Let's calculate B's private key which would give us *d*, decrypting key.

Assume that 13 and 17 were the chosen prime factors for n = 221.

$n = p * q = 13 * 17 = 221$

$r = (p - 1) * (q - 1) = (13 - 1) * (17 - 1)$

$r = 12 * 16 = 192$

$d = e^{-1} \bmod r$

$ed \equiv 1 \bmod 192$

$13d \equiv 1 \bmod 192$

Let's calculate modulo inverse using extended Euclidean algorithm (swapping 13 and 192)

Index i	quotient q for i	Remainder r for i	s for i	t for i
0		192	1	0
1		13	0	1
2	192 / 13 = 14	192 – 13 * 14 = 10	1 – 0*14 = 1	0 – 1*14 = –14
3	13 / 10 = 1	13 – 10 * 1 = 3	0 – 1 *1 = –1	1 – (–14)*1 = 15
4	10 / 3 = 3	10 – 3 * 3 = 1	1 – (–1) * 3 = 4	–14 – 15 * 3 = –59
5	3 / 1 = 3	3 – 1* 3 = 0	Do not calculate	Do not calculate

Let's re-swap the values.

$x = -59$

$y = 4$

- Putting the values in the extended Euclidean algorithm, you get

  $ax + by = 1$

  $13*(-59) + 192*(4) = 1$

- Since, you have to find multiplicative inverse in mod 192, divide both sides by mod 192.

  $13*(-59) \bmod 192 + 192*(4) \bmod 192 = 1 \bmod 192$

  $13*(-59) \bmod 192 + 0 = 1 \bmod 192$

Scanned with CamScanner

13*(−59) mod 192 = 1 mod 192

Recall our discussion on calculating mod for negative numbers. You need to keep adding mod until the number turns positive and then calculate mod on the positive number you got.

−59 + 192 = 136

136 mod 192 = 136

- Hence, multiplicative inverse is 136.
- Therefore, the value of decrypting key, $d = 136$.
- Hence, the Private Key of user B is (136, 221).

Now, the question isn't asking you to encrypt and decrypt the plaintext message 5. It is asking you to show the message signing and verification using RSA digital signature. Hence, do not encrypt the plaintext message 5. It is not required.

- As per RSA digital signature,
- Message Signing
- To sign a message, M
- Calculate the hash value of the message M at sender's end
- $h = hash(M)$
- Encrypt h using RSA private key
- Signature $S = (h)^d \bmod n$
- Signature Verification
- Decrypt Signature S using public key
- $h' = (S)^e \bmod n$
- Calculate the hash value of the message M at receiver's end
- $h = hash(M)$
- If $h = h'$, the signature is valid else the signature is invalid

Since, hash algorithm or the hash value of the message to be digitally signed is not given, let's assume that the hash value h of the plaintext message 5 is 12. Let's create and verify the digital signature using the RSA digital signature scheme.

To sign a message, M

- Encrypt h using RSA private key
- Signature $S = (h)^d \bmod n$

User A wishes to send the message to B. Hence, you would use A's private key (3, 33) for encrypting the hash value of the message. The encryption key is 3 from the private key. We have assumed that 12 is the hash value of the plaintext message 5.

$S = 12^3 \bmod 33$

$S = 12$

User A would send the message and its signature S = 12 to user B.

Now, user B can verify the signature as

- Decrypt Signature S using public key
- $h' = (S)^e \bmod n$
- The public key of the user A is (7, 33).

$h' = 12^7 \bmod 33$

$h' = 12$

- You find that $h = h' = 12$. Hence, the digital signature is verified.

Scanned with CamScanner

Review Questions

Here are a few review questions to help you gauge your understanding of this chapter. Try to attempt these questions and ensure that you can recall the points mentioned in the chapter.

[A] Computer Security Concepts

Q. 1 Describe the various categories of authentication methods with examples. **(8 Marks)**

Q. 2 Describe "Something You Know" as an authentication method category and list its advantages and disadvantages. **(8 Marks)**

Q. 3 Describe "Something You Have" as an authentication method category and list its advantages and disadvantages. **(8 Marks)**

Q. 4 Describe "Someone You Are" as an authentication method category and list its advantages and disadvantages. **(8 Marks)**

Q. 5 Describe "Something You Do" as an authentication method category and list its advantages and disadvantages. **(8 Marks)**

Q. 6 Describe "Somewhere You Are" as an authentication method category and list its advantages and disadvantages. **(8 Marks)**

Q. 7 Provide a comparison between the various authentication method types. **(8 Marks)**

Q. 8 Describe the term "Factors of Authentication" and give examples. **(6 Marks)**

[B] Needham Schroeder Authentication Protocol

Q. 9 With a block diagram, describe the Needham–Schroeder Symmetric Key Based Authentication Protocol. **(8 Marks)**

Q. 10 With a block diagram, describe the Needham–Schroeder Asymmetric Key Based Authentication Protocol. **(8 Marks)**

[C] Kerberos Authentication Protocol

Q. 11 What is Kerberos? List the problems it addresses. **(8 Marks)**

Q. 12 Describe the various components of Kerberos and explain how it works. **(8 Marks)**

Q. 13 With a block diagram, explain how a user can authenticate to a computer using Kerberos. **(8 Marks)**

(Copyright No. - L82548/2019)

Scanned with CamScanner

Q. 14 With a block diagram, explain how an authenticated user can authenticate to a print server using Kerberos for printing a document. **(8 Marks)**

Q. 15 What is Kerberos? List its limitation. **(4 Marks)**

[D] Digital Signature

Q. 16 What is a digital signature? How does it work? **(8 Marks)**

Q. 17 What is a digital signature? What are its applications? **(4 Marks)**

Q. 18 What is a digital signature? List its properties. **(4 Marks)**

❑❑❑

Scanned with CamScanner

5 Network Security and Applications

Syllabus

At the end of this unit, you should be able to understand and comprehend the following syllabus topics :

- Network security basics
- TCP/IP vulnerabilities (Layer wise)
  - Packet Sniffing
  - ARP spoofing
  - Port scanning
  - IP spoofing
  - Denial of Service (DoS)
    - Classic DoS attacks
      - ICMP flood
      - SYN flood
      - UDP flood
    - Distributed Denial of Service (DDoS)
    - Defences against Denial of Service Attacks
- Internet Security Protocols
  - SSL
  - IPSEC
- Secure Email
  - PGP
  - S/MIME
- Firewalls
- Intrusion Detection Systems (IDS)
  - Host based
  - Network based

Scanned with CamScanner

5.1 Network Security Basics

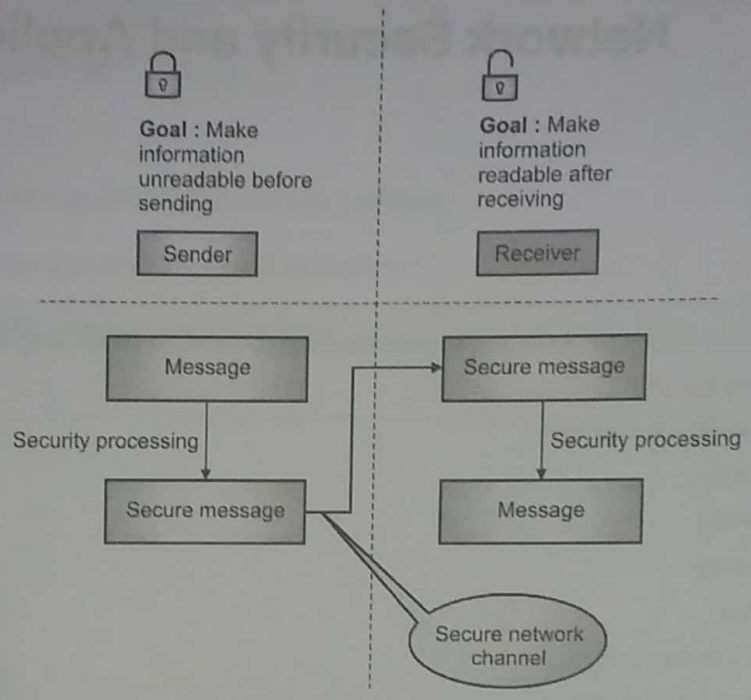


Fig. 5.1.1 : Transmitting a Message Securely over a Network

Network channel and protocols play a crucial role in information security. The information is not very useful when just stored locally on a computer. It must move to different devices in its lifetime to be useful. The way information is transferred from one device to another depends on several networking parameters. **Look at the simplistic diagram here.**

The network channel is the core mechanism to ensure that the message is transferred securely. If there are network vulnerabilities, as you will learn in this chapter, the information cannot be reliably sent.

You might have a question – if you are already securing the message before sending, why do you need a secure channel as well? The answer is that if the message can be intercepted (picked up) from the channel, the secure message can be attacked such that to make it insecure. Consider an example. You seal your letter in an envelope. But, if the person carrying the envelope is not reliable enough, would you still send the envelope with her? Not really, right? Same way, the network medium (or channel) should be secure enough to transfer the message to the destination reliably.

5.2 TCP/IP Vulnerabilities (Layer Wise)

Recall from our previous discussion on OSI model that you can conceptually view the networking as seven layers that inter-operate. Each of these layers have their respective vulnerabilities and requires appropriate protection mechanism. Some of these attacks and vulnerabilities are highlighted in the Table 5.2.1.

Scanned with CamScanner

Table 5.2.1 : TCP/IP Vulnerabilities (Layer Wise)

Layer Number	Layer Name	Attacks / Vulnerabilities
7	Application	Layer 7 DoS attacks (HTTP flood) SQL Injection Cross-site scripting DNS Spoofing
6	Presentation	Malicious SSL requests Inspecting SSL encryption packets
5	Session	Session Hijacking SSH downgrade Session sniffing
4	Transport	SYN Flood UDP Flood Port Scanning Other DoS attacks
3	Network	IP Spoofing Source Address Spoofing ICMP Flood Packet Sniffing Teardrop attack Other DoS attacks
2	Data Link	ARP Flooding ARP Spoofing MAC flooding DHCP Spoofing
1	Physical	Wire-cuts Disrupting the signal Any other media transmission disturbance

Let's learn a few of these in detail.

5.2.1 Packet Sniffing

Definition : *Packet Sniffing is the act of intercepting (capturing) of network traffic and logging it for further analysis.*

Scanned with CamScanner

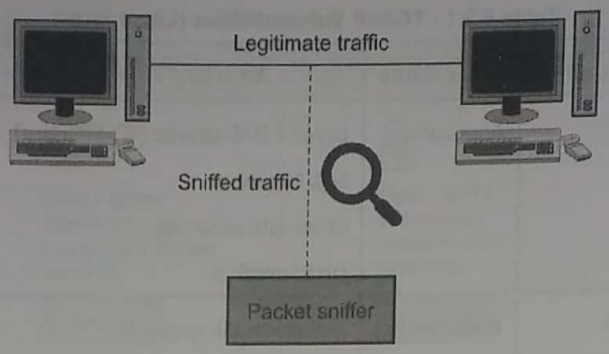


Fig. 5.2.1 : Packet Sniffing

Packet sniffing is carried out using either software programs or hardware devices. These are called packet analysers or just packet sniffers. Wireshark is one of the most widely used Packet Sniffing program.

Common Attacks

Some of the common attacks carried out using packet sniffing technique are as following:

- Capturing sensitive information.
- Analyse communication patterns.
- Learn about the network infrastructure.
- Exploit network vulnerabilities.

Protection

You cannot, to a great extent, avoid network sniffing attempts since it is external to the network and mostly beyond your control. For example, how do you protect someone with Wi-Fi analysing device to capture your Wi-Fi packets? However, some of the common protection mechanisms to reduce the impact from sniffing are as follows:

- **Encryption :** Use TLS, VPN tunnels or application level encryption wherever possible to avoid sending and receiving data in plaintext.
- **Use secure protocols :** Discourage the use of protocols such as HTTP, FTP, TELNET, RPC and prefer using secure protocols such as HTTPS, FTPS, SSH, etc.
- **Isolate and Segment Networks :** Design your network in such a way that the sensitive systems are adequately segmented from rest of the network.

5.2.2 ARP Spoofing

✍ ***Definition :*** *ARP spoofing is a technique by which the attacker associates her MAC address with the IP address of a legitimate device.*

It is also called ARP cache poisoning or ARP poison routing.

During network communication, a device, when wishes to interact with another target device, requires mapping the IP address of the target to the MAC address of the target. In ARP spoofing, the attacker maliciously provides MAC address of her device so that the wrong mapping of target IP address to target MAC is created.

Scanned with CamScanner

ARP Table (Expected)	
Target IP Address	Target MAC Address
10.12.92.56	AA:00:FE:12:E4:56
ARP Table (Spoofed)	
Target IP Address	Target MAC Address
10.12.92.56	BB:11:FE:12:F7:84

Once the attacker's MAC address is linked to the target IP address, the attacker will begin receiving any data that is intended for that IP address. ARP spoofing can enable malicious parties to intercept, modify or even stop data in-transit. ARP spoofing attacks can only occur on local area networks that utilize the Address Resolution Protocol (ARP).

Common Attacks

Some of the common attacks carried out using ARP spoofing technique are as following:

- **Denial-of-service attacks :** DoS attacks often use ARP spoofing to link multiple IP addresses with a single target's MAC address. As a result, traffic that is intended for many different IP addresses will be redirected to the target's MAC address, overloading the target with traffic.
- **Session hijacking :** Session hijacking attacks can use ARP spoofing to steal session IDs and thus granting attackers access to private or sensitive data.
- **Man-in-the-middle attacks :** MITM attacks can use ARP spoofing to intercept and modify traffic between victims and carry out further attacks.

Protection

Some of the protection mechanisms for ARP spoofing are as follows:

- **Static ARP entries :** You could create static ARP entries so that any MAC entries are approved for use.
- **ARP spoofing detection software :** You could use software to detect ARP spoofing attacks and prevent spoofed MAC entries.

5.2.3 Port Scanning

Q. What is Authentication Header (AH)? How does it protect against replay attacks? **MU - May 19, 5 Marks**

In computer networking terminology,

Definition : *A port is an endpoint of communication that serves the service requests.*

Each service on the OS requires a port to communicate with any other service or device on the network. Hence, indirectly, if you can find out which ports are open for connection on a particular system, it is as good as knowing which services are running on the system. Once you know which services are running on the system, you can further craft attacks on the system exploiting vulnerabilities in those services.

Hence,

Definition : *Port scanning is a technique using which you can identify the state of ports and indirectly know about the running services that could be exploited.*

Scanned with CamScanner

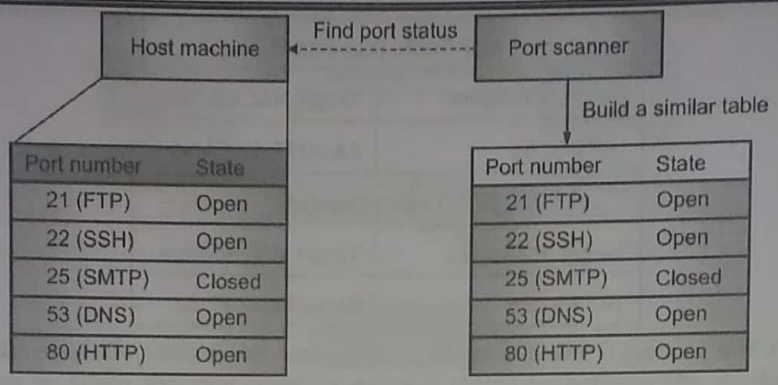


Fig. 5.2.2 : Port Scanning

5.2.3(A) Port Scanning Techniques

There are several port scanning techniques. The following techniques are commonly used.

1. Connect Scan
2. SYN Scan
3. FIN Scan
4. NULL Scan

Let's learn about them.

1. **Connect Scan :** This is the simplest of all the port scanning techniques. The port scanner utilizes the OS network functions to identify any open ports. It tries to establish the connection to all the ports serially one after the another. The ports that are open would complete the 3-way TCP handshake and ports which are not open would not.
2. **SYN Scan :** In this technique, the port scanner sends the SYN packet to the ports it desires to test openness for. Ports that are open respond with SYN-ACK packet and ports that are not open respond with RST packet.
3. **FIN Scan :** In this technique, the port scanner sends the FIN packet to the ports it desires to test openness for. Ports that are open ignore the packet and ports that are not open respond with RST packet.
4. **NULL Scan :** In this technique, the port scanner sends a TCP packet with no TCP flags set. Ports that are open ignore the packet and ports that are not open respond with RST packet.

Protection

1. **Close unnecessary ports :** Close unnecessary ports on the system. Any service that you are not using and is installed and running could be vulnerable to attacks.
2. **Firewalls :** Firewalls can stop scanning or connection establishment from any non-approved devices. They can filter various ports and avoid providing the state of the ports to any unauthorized port scanners.
3. **Intrusion Detection Systems :** Intrusion Detection Systems can identify port scanning attempts and can raise notifications accordingly.

Scanned with CamScanner

5.2.4 IP Spoofing

Q. Write in brief about - IP spoofing. **MU - Dec. 15, 5 Marks**

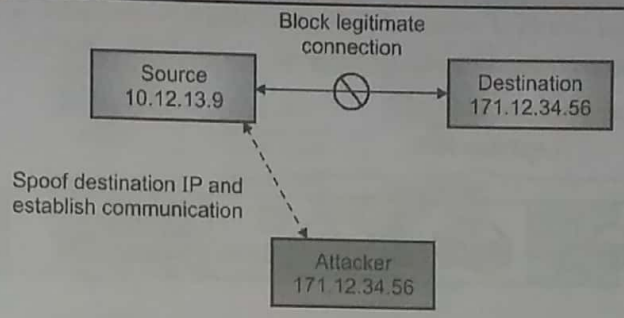


Fig. 5.2.3 : IP Spoofing

IP addresses are, in a way, identities of the systems that are communicating over a network.

✍ ***Definition :*** *IP Spoofing is a technique whereby an attacker impersonates (tries to steal the identity of) another machine by manipulating IP packets.*

The goal of the attacker is to get unauthorized access to communication by manipulating her IP address as the actual target IP address.

Common Attacks

Some of the common attacks carried out using IP spoofing technique are as following:

- Steal sensitive information by impersonating as the legitimate IP address.
- Bypass authentication in networks where machines are trusted by their IP addresses.
- Carry out of Denial of Service (DoS) attacks by manipulating that the traffic is coming from a legitimate source.

Protection

Some of the protection mechanisms for IP spoofing are as follows:

- Do not trust IP addresses for authentication. Use stronger form of authentication such as certificates.
- Use packet filtering to reject packets with private (internal) IP addresses.
- Monitor network using controls such as network monitoring tools, IDS, etc.

5.2.5 Denial of Service (DoS) and Distributed Denial of Service (DDoS)

Q. What is a Denial of service attack? What are the different ways in which an attacker can mount a DOS attack on a system? **MU - Dec. 15, May 17, 10 Marks**

Q. Write in brief about Denial of service attacks. **MU - May 16, 5 Marks**

Q. Explain briefly with example, How the following Denial of service attacks occurs. **MU - Dec. 16, Dec. 17, 5 Marks**

Q. Explain different types of Denial of Service attacks. **MU - May 19, 10 Marks**

Remember our discussion on availability – one of the CIA triads? DoS and DDoS are attacks on the availability triad.

Scanned with CamScanner

Definition : *Denial of Service (DoS) is an attack from a single source such that the resources are exhausted on the target beyond its serving capacity.*

Definition : *Distributed Denial of Service (DDoS) is an attack from multiple sources such that the resources are exhausted on the target beyond its serving capacity.*

Let's understand these using quick examples.

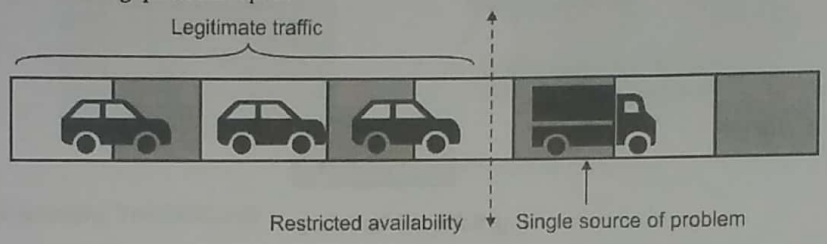


Fig. 5.2.4 : Analogy for DoS

In this example, a single source (truck – the attack) is blocking the road (resources) for all other vehicles (legitimate traffic) behind it.

The vehicles behind the truck have limited mobility and the road (resources) is exhausted (full capacity being consumed by the truck). This is what is DoS in the typical sense. A single source of attack exhausts all target resources such that the target is unavailable to serve the legitimate requests.

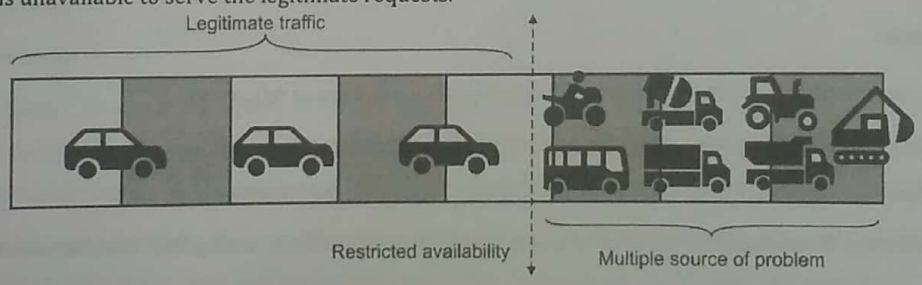


Fig. 5.2.5 : Analogy for DDoS

In the second example, there are multiple sources of jam (attacks) that have completely blocked the road (resources) despite that the road is wider this time when compared with the previous example. This is what is typically Distributed Denial of Service (DDoS).

Note : These days computer and other equipment have large capacities. So, it is not very effective to carry out DoS attack (the resources are sufficient to not cause any availability problem). Instead, attackers choose DDoS to attack the target such that to exhaust its capacity (such that even a larger capacity does not help to protect availability to a great extent). Hence, in our discussion, we can use the terms DoS and DDoS to actually mean just DDoS. Any DDoS attack is actually a DoS attack which is amplified (multiplied) using various sources of attack.

5.2.5(A) Botnet

Understanding botnet (robot networks) is crucial to understating DDoS attacks.

Definition : *A botnet refers to a group of computers which have been infected by an attacker and is under her complete control.*

Scanned with CamScanner

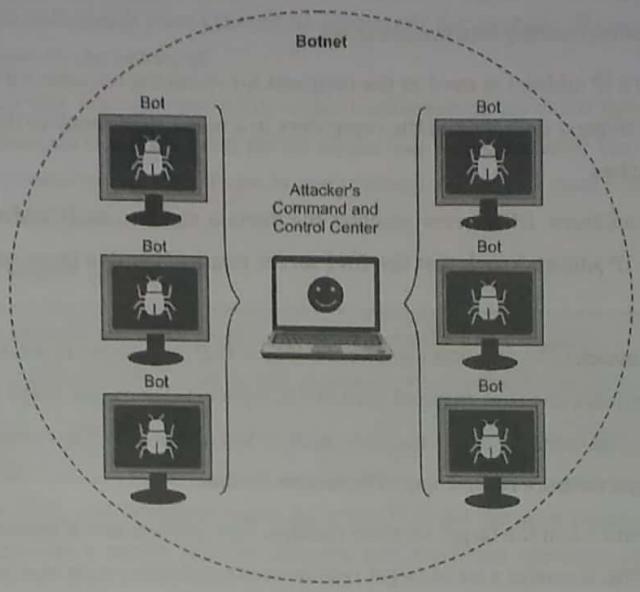


Fig. 5.2.6 : Botnet

These are the computers that the attacker uses to carry out DDoS attacks. All the computers in the group work together under the command of the attacker to hit a particular target and bring down its availability.

5.2.5(B) Types of DDoS Attacks

There are various types of DDoS attacks. Some of them are outlined here.

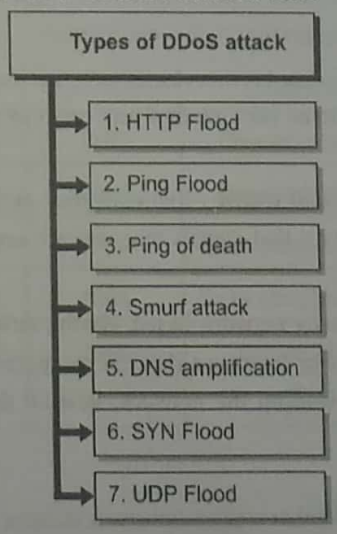


Fig. 5.2.7 : Types of DDoS Attacks

1. **HTTP flood :** HTTP flood is similar to you hitting refresh on your browser several times. It is just that it is done at a much large scale to crash the webserver and restrict the legitimate users from reaching the webserver.
2. **Ping (ICMP) Flood :** In this attack, the target machine is sent so many ping requests that it is overwhelmed and fails to respond.

Scanned with CamScanner

3. **Ping of Death :** In this attack, the target machine is sent malformed packets such that the system is unable to understand and process them resulting into system crash.
4. **Smurf attack :** The victim's IP address is used as the recipient for receiving responses from broadcast communication. The attacker then crafts a request such that all the computers in a network respond to the victim's IP address such that it is overwhelmed and crashes.
5. **DNS Amplification :** As you know, DNS server resolves the domain name to an IP address. The attacker crafts a DNS request (with the target's IP address) such that the DNS server responds with a large amount of data and crashes the target.
6. **SYN Flood :** In SYN flood attacks, the attacker exploits the way a TCP connection is established. After sending the SYN packet to the target, the attacker does not respond with the ACK packet. The target keeps on waiting for the ACK until it runs out of the resources. The attacker sends multiple such SYN packets until the resources on the target are totally consumed and the target can no more receive any SYN packets further.
7. **UDP flood :** UDP flood occurs when the target receives multiple UDP packets and it needs to check if there are any UDP ports listening for UDP traffic. It wastes a lot of target resources in conducting such searches for port numbers and thus the target becomes too busy to serve any legitimate traffic thus impacting its availability.

5.2.5(C) Preventing DDoS Attacks

Here are some ways to potentially prevent DDoS attacks.

1. **Reduce attack surface area :** By reducing the attack surface area we mean reducing the number of points exposed for attacks. These could be network ports, number of services running, open networks, unrestricted administrative access, unpatched OS or applications or anything else that could be exploited.
2. **Plan for scale :** DDoS attacks target the limited resources. If you have resources spread over a large scale (or say cloud computing), the DDoS attack impact could be very minimal and could be absorbed by the large resource pool without impacting availability.
3. **Know what is a normal and an abnormal traffic :** Knowing your network traffic patterns would let you plan for action if you see any change in the pattern that signals that you are under attack. You can then appropriately take actions to remedy any impact from DDoS.
4. **Deploy firewalls :** DDoS attacks are mostly network based. Having stringent (strong) firewall rules, such that only appropriate traffic is allowed, is a great way to ensure that the resources are not wasted in serving the DDoS traffic. Only the traffic that is legitimate is allowed on the network. Rest all the traffic is dropped without impacting the resources.

5.3 Internet Security Protocols

World Wide Web or just web is a collection of web servers that run several websites that hold the desired information. The Internet as a whole is a collection of such servers and various communication devices and protocols. You mostly use browsers (such as Chrome, Firefox, Internet Explorer, Safari, etc.) or applications (for example, Mobile Apps) to browse the web and fetch the desired information or just complete a desired interaction such as making a purchase.

Scanned with CamScanner

Let me pause you here and ask a simple question. Don't you feel that your interaction with the Internet (which generally is an insecure and unsafe place) should be protected? For example, if you type your Facebook password, should it be available to everyone on the network?

To make a purchase when you provide your bank account information, isn't that information very confidential and requires secure handling as you pass it through your device all the way to the website? Yes, I am sure you understand that your interaction with the web requires security. There is one protocol that we all need – SSL (obsolete now) followed by TLS (currently used). Let's learn about it.

5.3.1 Secure Socket Layer (SSL)

Q. List the functions of the different protocols of SSL. Explain the handshake protocol. **MU - Dec. 15, 5 Marks**

Q. What are the different protocols in SSL? How do the client and server establish an SSL connection ? **MU - Dec. 17, 5 Marks**

Q. What is the need of SSL? **MU - May 19, 4 Marks**

The Secure Sockets Layer (SSL) and Transport Layer Security (TLS) are the most widely use web security protocol. It is essentially a protocol that provides a secure channel between two machines operating over the Internet or an internal network.

✍ ***Definition :*** *A Secure Socket Layer (SSL) is a cryptographic protocol designed to protect communication between two entities.*

SSL underwent several revisions and is now followed by a more secure protocol called TLS. The Table 5.3.1 is a quick version history of SSL/TLS.

Table 5.3.1 : History of SSL/TLS

Protocol	Published	Status
SSL 1.0	Unpublished	Unpublished
SSL 2.0	1995	Obsoleted in 2011
SSL 3.0	1996	Obsoleted in 2015
TLS 1.0	1999	To be obsolete in 2020
TLS 1.1	2006	To be obsolete in 2020
TLS 1.2	2008	Currently good
TLS 1.3	2018	Currently good

Goals of SSL

1. **Cryptographic Security :** Establish and provide a secure connection between two parties.
2. **Interoperability :** Two unrelated applications should be able to establish SSL connection.
3. **Extensibility :** Provides a framework for using various algorithms and methods without changing the protocol.
4. **Efficiency :** Performance enhancement mechanism to avoid overloading the system when protocol is in use.

Scanned with CamScanner

5.3.1(A) Overview of SSL Protocol

SSL protocol works in layers. At each layer, messages may include fields for length, description, and content. SSL takes messages to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies a MAC, encrypts, and transmits the result. On the other side, received data is decrypted, verified, decompressed, and reassembled and then delivered to higher level clients.

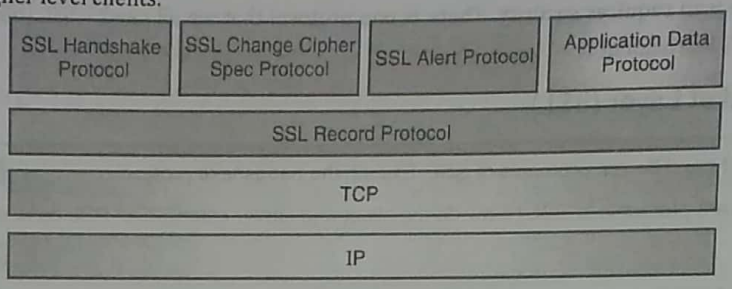


Fig. 5.3.1 : Overview of SSL protocol

Let's learn about each one of them in detail.

Session and Connection States

An SSL session is stateful which means that parameters negotiated during the session establishment persist (stay the same) until the session is terminated. The SSL handshake protocol coordinates the states of the client and server. It is thus important to preserve Session and Connection States.

The Table 5.3.2 summarizes a Session State.

Table 5.3.2 : Session State

Fields	Purpose
Session Identifier	A session ID chosen by the server to identify an active or resumable session state.
Peer Certificate	X.509 Certificate of the other party in the communication.
Compression method	The algorithm used to compress data prior to encryption.
Cipher Specification	Chosen encryption algorithm such as AES and a hash algorithm such as SHA.
Master Secret	48-byte secret shared between the client and server.
Is resumable	A Boolean flag indicating whether a session ID can be used to initiate new connections.

The Table 5.3.3 summarizes a Connection State.

Table 5.3.3 : Connection State

Fields	Purpose
Server and client random	Byte sequences for establishing connection.
Server write MAC secret	The secret used in MAC operations on data written by the server.
Client write MAC secret	The secret used in MAC operations on data written by the client.
Server write key	The bulk cipher key for data encrypted by the server and decrypted by the client.
Client write key	The bulk cipher key for data encrypted by the client and decrypted by the server.
Initialization vectors	Random number to initialize encryption operation.
Sequence numbers	Sequence numbers for transmitted and received messages for each connection.

(Copyright No. - L82548/2019)

Scanned with CamScanner

5.3.1(B) SSL Record Layer Protocol

Definition : *The SSL Record Layer is the last protocol that receives the raw data from the higher application layers and other SSL protocols such as handshake.*

Its core function is to facilitate (perform) data transfer. The basic unit of data in SSL is a record. Each record consists of a five-byte record header, followed by data.

There are four types of records in SSL.

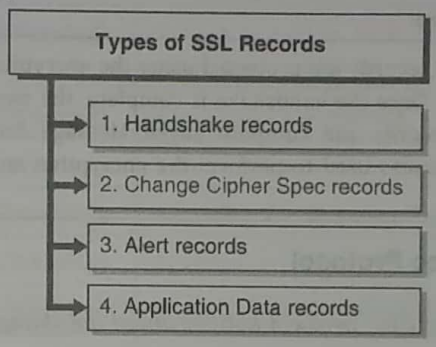


Fig. 5.3.2 : Types of SSL records

The five-byte format of an SSL Record Header shown in Fig. 5.3.3.

SSL record type (1-byte)	SSL Major Version (1-byte)	SSL Minor Version (1-byte)	Length of data in the record (2-bytes)

Fig. 5.3.3 : Format of an SSL record header

A simplistic block diagram of steps involved in the SSL Record Protocol shows in Fig. 5.3.4.

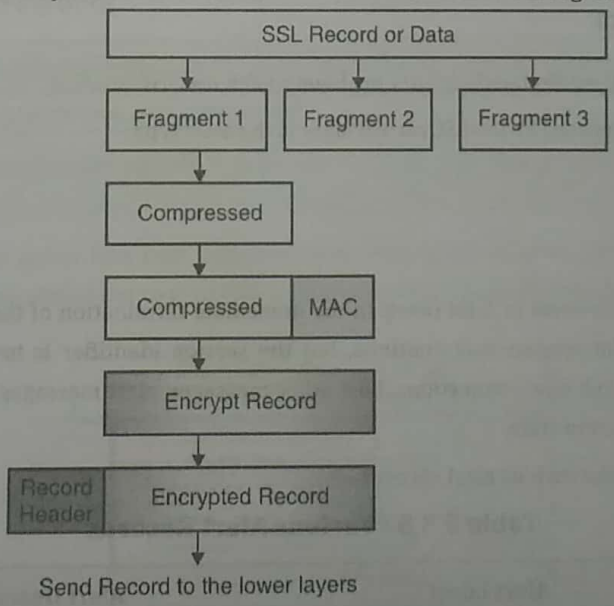


Fig. 5.3.4 : Block diagram of SSL record

At a high level the SSL Record Protocol performs three operations as shown in the Table 5.3.4.

Scanned with CamScanner

Table 5.3.4 : Operations of SSL Record Protocol

Operation Performed	Purpose
Fragmentation	Break original data into SSL Plaintext records of 2^{14} bytes or less.
Compression and Decompression	All SSL Plaintext records are compressed using the compression algorithm defined in the current session state. The compression algorithm translates an SSL Plaintext structure into an SSL Compressed structure.
Payload Protection	All SSL Compressed records are protected using the encryption and MAC algorithms defined in the current CipherSpec. Once the handshake is complete, the two parties have shared secrets that are used to encrypt records and compute keyed Message Authentication Codes (MACs) on their contents. The techniques used to perform the encryption and MAC operations are defined by the CipherSpec.

5.3.1(C) SSL Change Cipher Spec Protocol

✍ ***Definition :*** *The Change Cipher Spec protocol notifies about the changes in cipher parameters.*

The protocol consists of a single message, which is encrypted and compressed. The Change Cipher Spec Protocol notifies the communicating parties about any change in the previously negotiated Cipher Specifications or Keys.

The keys or the algorithms need to be changed at times for reasons such as renewing the session or resuming the session. The change cipher spec message is sent by both the client and server to notify the receiving party that subsequent records will be protected under the just-negotiated CipherSpec and keys.

5.3.1(D) SSL Alert Protocol

✍ ***Definition :*** *The SSL Alert Protocol signals problems with an SSL session.*

One of the content types supported by the SSL record layer is the alert type.

Alert messages notify the;

- Severity of the alert and,
- A description of the alert

Alert messages with a severity level of *fatal* result in the immediate termination of the connection. In this case, other connections corresponding to the session may continue, but the session identifier is invalidated, preventing the failed session from being used to establish new connections. Like other messages, alert messages are encrypted and compressed, as specified by the current connection state.

The Table 5.3.5 summarizes the various alert records.

Table 5.3.5 : Various Alert Records

Alert Code	Alert Message	Alert Level	Alert Description
0	close_notify	1 (Warning)	Notifies the recipient that the sender will not send any more messages on this connection.
10	unexpected_message	2 (Fatal)	An inappropriate message was received.

Scanned with CamScanner

Alert Code	Alert Message	Alert Level	Alert Description
20	bad_record_mac	2 (Fatal)	A record is received with an incorrect MAC.
30	decompression_failure	2 (Fatal)	The decompression function received improper input.
40	handshake_failure	2 (Fatal)	The sender was unable to negotiate an acceptable set of security parameters given the options available.
41	no_certificate	1 (Warning)	Sent in response to a certification request if no appropriate certificate is available.
42	bad_certificate	1 (Warning)	A certificate was corrupt.
43	unsupported_certificate	1 (Warning)	A certificate was of an unsupported type.
44	certificate_revoked	1 (Warning)	A certificate was revoked by its signer.
45	certificate_expired	1 (Warning)	A certificate has expired or is not currently valid.
46	certificate_unknown	1 (Warning)	Some other (unspecified) issues.
47	illegal_parameter	2 (Fatal)	A field in the handshake was out of range or inconsistent with other fields.

The alert record consists of 2 bytes of information from the Table 5.3.5.

Alert Level (1-byte)	Alert Code (1-byte)

Fig. 5.3.5 : Alert record

5.3.1(E) SSL Handshake Protocol

Q. Write in brief about SSL handshake protocol. **MU - May 16, 5 Marks**

Q. Explain all phases of SSL Handshake protocol in detail. **MU - May 19, 6 Marks**

✍ ***Definition :*** *The cryptographic parameters of the session state are produced by the SSL handshake protocol.*

When an SSL client and the server first start communicating, they need to agree upon certain parameters. There are also several steps that need to be carried out to establish a secure session. At a high level, Fig. 5.3.6 shows the four steps are carried out.

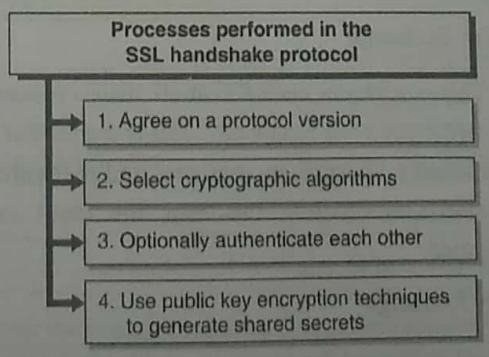


Fig. 5.3.6 : Processes performed in the SSL Handshake Protocol

Scanned with CamScanner

The steps can be detailed shown in the Fig. 5.3.7 simplistic handshake process.

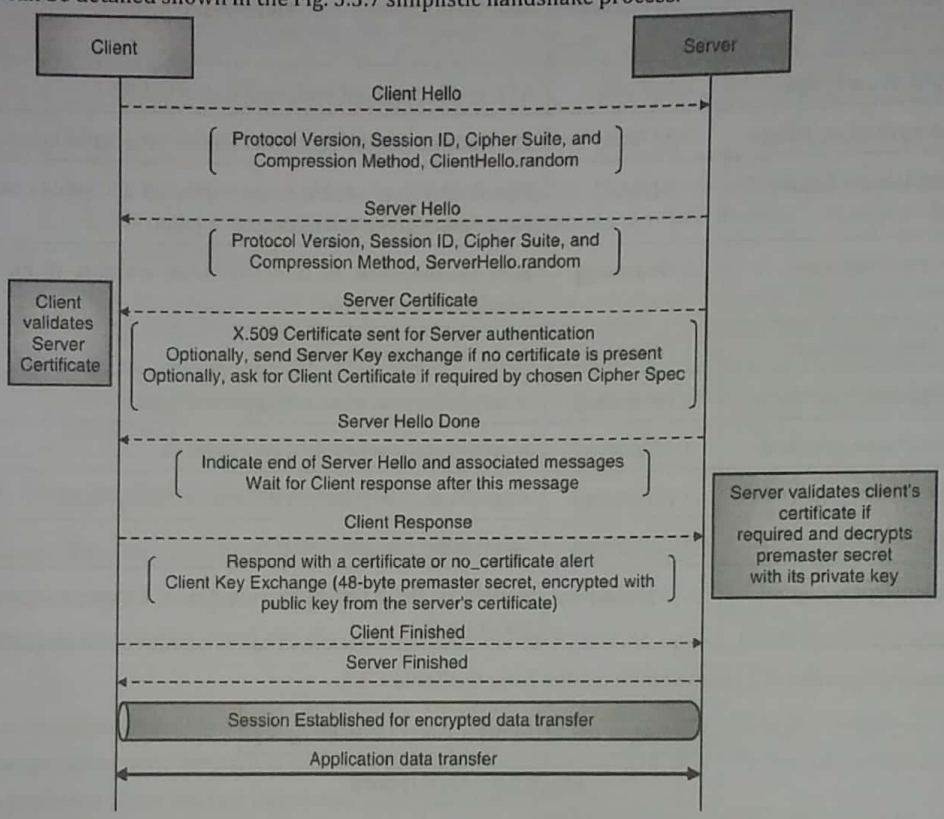


Fig. 5.3.7 : Handshake Process Diagram

Step 1 : Hello messages (Establish security capabilities)

The hello phase messages are used to exchange security enhancement capabilities between the client and server.

1. Client Hello : When a client first connects to a server it is required to send the client hello as its first message. The client can also send a client hello in response to a hello request or on its own initiative in order to renegotiate the security parameters in an existing connection. The list of parameters sent are shown in Fig. 5.3.7.
2. Server Hello : The server processes the client hello message and responds with server hello message. The list of parameters sent are in the diagram.

Step 2 : Server Authentication and Key Exchange

This is the most important step. This is why you need SSL at all. Before proceeding to interact with the server you should find out "Is this really the server you want to talk to?". This is crucial. For example, if you want to do a banking transaction, before providing your account information, username and password, you MUST validate that the website you are on (server behind the website) is legitimate. In this step, the client validates the server certificate. Any certificate related errors are highlighted.

Step 3 : Client authentication and Key Exchange

If the certificate is found valid, client exchanges the keying material that would be subsequently used to encrypt the messages. The client generates a 48-byte premaster secret, encrypts it using the public key from the server's certificate and sends the result in an encrypted pre-master secret message.

Scanned with CamScanner

Step 4 : Connection establishment and data transfer

Once all the connection parameters are negotiated and exchanged, a connection between the server and the client is established. Once the connection is established, the data transfer begins between the server and the client. The data is encrypted based on the negotiated terms.

5.3.2 Transport Layer Security (TLS)

As you learnt earlier, SSL is obsolete. TLS replaced SSL in 1999. The underlying working of TLS is very similar to SSL.

TLS is more efficient and secure than SSL. It provides stronger message authentication, key-material generation and supports pre-shared keys, secure remote passwords, elliptical-curve keys and Kerberos. TLS and SSL are not interoperable, but TLS provides backward compatibility for devices using SSL.

HTTPS

Now that you learnt how SSL works, let's learn about one of its implementations – HTTPS application protocol.

✍ ***Definition :*** *HTTPS establishes a secure SSL / TLS tunnel before beginning data transfer.*

The Hypertext Transfer Protocol (HTTP) is an application protocol used to transfer data on distributed and connected systems. HTTPS is the secure version of HTTP. The 'S' at the end of HTTPS stands for 'Secure'. It means that all the communications between your client (browser, mobile apps) and the server (website, web application) is encrypted. HTTPS is often used to protect confidential online interactions such as online banking.

Conceptually, HTTPS is very simple. Simply use HTTP over TLS (previously SSL) instead of HTTP. The use of TLS (previously SSL) ensures that the adequate protection mechanisms such as encryption, server authentication, hashing, and optionally client authentication are effectively applied, and the communication is adequately protected.

Comparison between HTTP and HTTPS

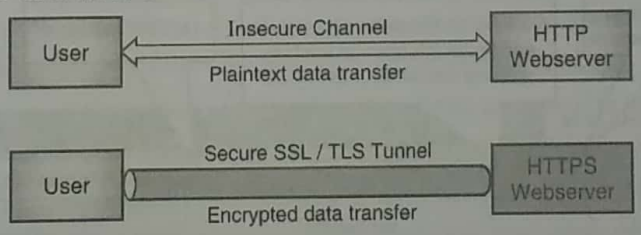


Fig. 5.3.8 : Comparison between HTTP and HTTPS

Table 5.3.6 : Comparison between HTTP and HTTPS

HTTP	HTTPS
Data transfer in plaintext	Data transfer in ciphertext
Default port is 80	Default port is 443
Does not require SSL/TLS or Certificates	Requires SSL/TLS implementation with Certificates
URL has http://	URL has https://
Should be avoided	Should be preferred
Search engines do not favour the insecure websites	Improved reputation of the website in search engine
Users worried about their data	Users confident about the security of their data

Scanned with CamScanner

Motivation / Benefits of using HTTPS

1. **Increasing sensitivity of data :** With the proliferation (widespread use) of internet, a lot of sensitive communication such as online banking, ticketing, shopping, etc. is taking place over the Internet. There is an ever increasing need to ensure that the communication is secure (confidentiality and integrity of the information is enforced). Information such as your password or credit card number is not transferred in plaintext that can potentially be captured and then misused.
2. **Authentication :** One of the critical use cases that HTTPS serves is that using it you can potentially authenticate a website or a business. HTTPS connection is established using X.509 certificates and certificate authorities do proper business or website validation before issuing certificates. Certificates help you prove that a website is indeed legitimate, and you are indeed interacting with the right website. This avoids several online frauds where a similar looking banking or e-commerce site can capture your confidential details.
3. **Privacy requirements :** Often times, the nature of communication is private even if it is not confidential.

   For example, your health reports, your chats, your location details, etc. require that they are adequately protected when transferred over the network. Use of HTTPS ensures that encryption is applied to all data seamlessly and the private information is adequately protected during transfer.

Format, Port Number and Representation

Typical format of HTTPS is https://www.example.com. It works over port 443 by default. You would have seen various websites with HTTPS enabled. These days browsers show green colour in the URL for HTTPS protected websites and warning for non-HTTPS websites.

An example of a HTTPS protected website is shown in Fig. 5.3.9.

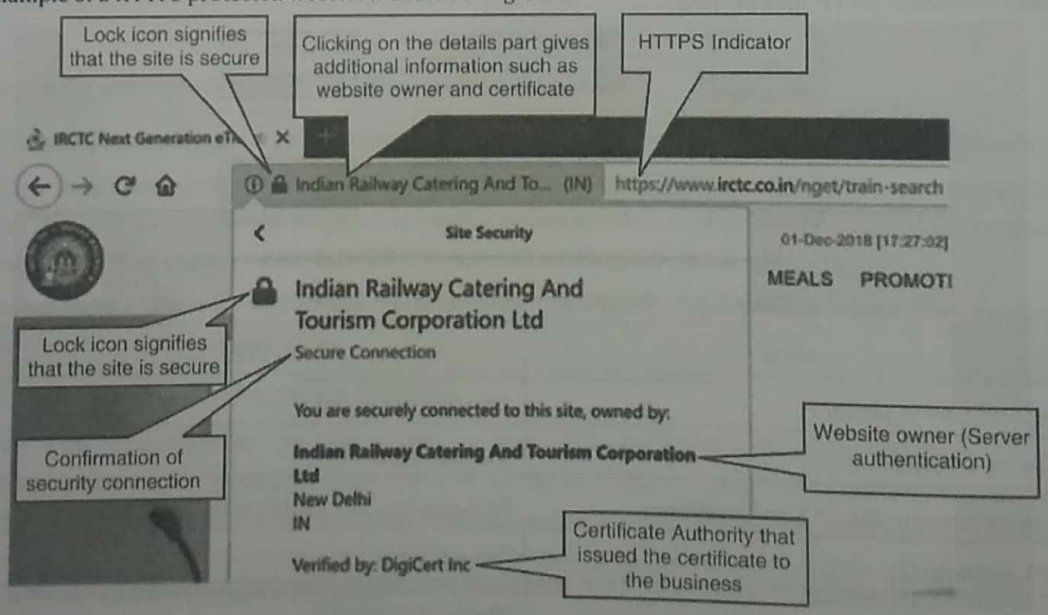


Fig. 5.3.9 : HTTPS protected website

Scanned with CamScanner

An example of a non-HTTPS website in shown in Fig. 5.3.10.

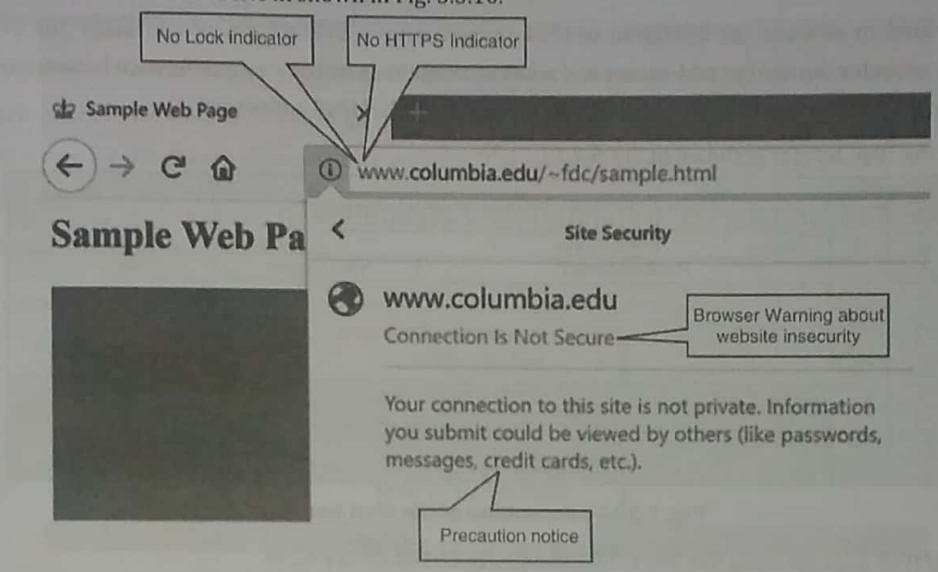


Fig. 5.3.10 : Non-HTTPS website

5.3.3 Internet Protocol Security (IPSec)

Q. Write in brief about IPSec protocols for security. **MU - May 16, 5 Marks**

IP stands for Internet Protocol. IP defines a set of protocols that can be used for communication between any two devices on the network. A network protocol is a standard set of rules that determines how systems will communicate across networks. Two different systems that use the same protocol can communicate and understand each other very similar to how two people can communicate and understand each other by using the same language.

IP provides addressing and routing mechanisms for each packet of data that needs to move across the network. Each device on the network must have a unique IP address to communicate with any other device on the network.

Note : It is assumed that you have a general understanding of computer networking. While this section does not dive deeper into computer networks, it focuses on specific security topics around networking.

IPv4 : IPv4 is IP version 4. This is the most common IP addressing scheme used today despite certain challenges. It is 32-bit long and thus has an address space of 2^{32} = 4,294,967,296. This means you can maximally have 4,294,967,296 (approximately 4.3 billion) IPv4 addresses. There are many more devices than the number 4,294,967,296. An outline of the IPv4 header is shown in Fig. 5.3.11.

Offsets Octet	Octet	0								1								2								3							
	Bit	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	26	27	28	29	30	31
0	0	Version				IHL				DSCP						ECN		Total Length															
4	32	Identification																Flags				Fragment Offset											
8	64	Time To Live								Protocol								Header Checksum															
12	96	Source IP Address																															
16	128	Destination IP Address																															
20	160	Options (if IHL > 5)																															
24	192																																
28	224																																
32	256																																

Fig. 5.3.11 : An outline of the IPv4 header

An example of IPv4 address looks like 121.56.78.214.

Scanned with CamScanner

IPv6

IPv6 was created to address the limitation of IPv4 to have only 4,294,967,296 IP addresses due to 32-bit length. IPv6 more or less provides the similar addressing and routing capabilities but one core difference between IPv4 and IPv6 is the address space. IPv6 address is 128-bit long and thus you can have 2^{128} IPv6 addresses!

An outline of the IPv6 header is shown in Fig. 5.3.12.

Offsets Octet	Octet	0								1								2								3							
	Bit	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	26	27	28	29	30	31
0	0	Version				Traffic Class									Flow Label																		
4	32	Payload Length																Next Header								Hop Limit							
8	64	Source IP Address																															
12	96																																
16	128																																
20	160																																
24	192	Destination IP Address																															
28	224																																
32	256																																
36	288																																

Fig. 5.3.12 : An outline of the IPv6 header

An example of IPv6 address looks like 2001:0:9d38:6abd:2c37:10da:8554:4234.

✍ ***Definition :*** *IPSec is a suite of protocols that protects IP traffic.*

IP does not have any integrated security mechanisms by itself and hence IPSec (short form for IP Security) is additionally used to provide security for IP traffic. The IPSec suite consists of following security protocols.

Table 5.3.7 : Security protocols/ Services Provided by IPSec

Protocol Name	Functionality Provided
Authentication Header (AH)	• Data Integrity • Data Origin Authentication • Protection from replay attacks.
Encapsulating Security Payload (ESP)	• Confidentiality • Data Origin Authentication • Data Integrity.
Internet Security Association and Key Management Protocol (ISAKMP)	Framework for Authentication and Key Exchange.
Internet Key Exchange (IKE)	Authenticated keying material for use with ISAKMP.

Note here that IPSec is a framework. It does not mandate which hashing and encryption algorithms should be used or how keys should be exchanged between the communicating devices. Key management can be handled manually or automated by a key management protocol such as ISAKMP.

5.3.3(A) Security Association

Q. What are security associations ? **MU - Dec. 17, 4 Marks**

Security Association (SA) is a fundamental concept with respect to IPSec.

Scanned with CamScanner

Definition : *Security Association (SA) holds several information that determines how security services would be consumed by the communicating devices.*

IPSec provides many options for performing security services such as encryption, integrity and authentication.

The network devices that wish to establish an IPSec connection, must negotiate and arrive at exactly which algorithms and parameters to use for the chosen IPSec security services. The security association is a mechanism to hold all the agreed terms (algorithms, parameters, etc.) for a given IPSec communication session.

5.3.3(B) Modes of Operation

IPSec can work in two modes.

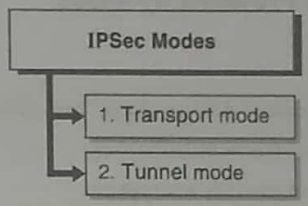


Fig. 5.3.13 : IPSec Modes

1. **Transport Mode :** In this mode, only the payload (data) part of the information is protected. The addressing and routing information is not protected. It is like a sealed envelope with address on it. The message inside the envelop is protected whereas the source and destination addresses are not.

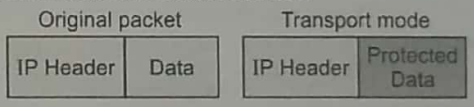


Fig. 5.3.14 : Transport Mode

2. **Tunnel Mode :** In this mode, both the payload (data) as well as the addressing information is protected. In this mode, the entire packet is protected, and a new IP header is added by IPSec. The original IP header information along with the payload information is protected. Tunnel mode provides more security than the transport mode.

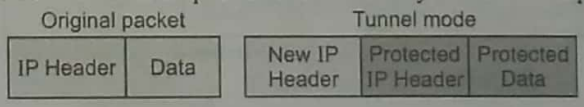


Fig. 5.3.15 : Tunnel Mode

5.3.3(C) Applications / Benefits / Usage of IPSec

1. **Establish Virtual Private Network (VPN) :** IPSec is predominantly used to establish VPN connection. VPN connections are generally used to access private networks over the internet. For example, you can access your college or your organization's network from home over the internet.
2. **Connecting two or more branch networks :** IPSec can be used to extend or connect branch networks. For example, if you have two branches of office each using its own network, the branches can be connected using IPSec. The network traffic then can securely move between the branches.
3. **General Security Benefits :** IPSec adds general security benefits to the core IP protocol. It provides benefits such as data confidentiality, data integrity, data origin authentication and protection from several attacks on the core IP protocol.

Scanned with CamScanner

5.3.3(D) How does IPSec work?

Overall, communication over IPSec has 5 broad steps.

1. **Initiate IPSec process :** IPSec communication begins with the identification of traffic that requires IPSec security.
2. **IKE Phase 1 :** In this phase, the IKE SAs are negotiated and agreed.
3. **IKE Phase 2 :** In this phase, next set of SAs for actual data transfer are negotiated and agreed.
4. **Data Transfer :** Data is transferred between the communicating entities.
5. **Termination :** The IPSec connection is terminated once the data transfer is complete.

5.3.3(E) Security Protocols/ Services Provided by IPSec

1. Authentication Header (AH)

✍ ***Definition :*** *The Authentication Header (AH) protocol provides data integrity and data original (source address) authentication over the network communication.*

It also provides replay protection. It does not provide encryption.

AH calculates the Integrity Check Value (ICV) over non-changing fields of the IP header:

- Next Header
- Payload Len
- Reserved
- Security Parameter Index (SPI)
- Sequence number
- Padding bytes

ICV is a hash value which is often computed using SHA-1 or other hashing algorithms.

(i) Transport mode for AH, Fig. 5.3.16 is show before and after applying AH.

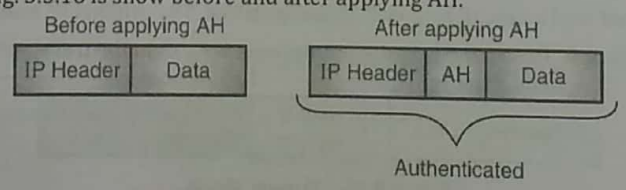


Fig. 5.3.16 : Transport mode for AH

(ii) Tunnel mode for AH, Fig. 5.3.17 is show before and after applying AH.

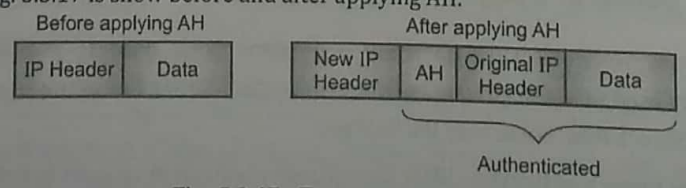


Fig. 5.3.17 : Tunnel mode for AH

If you recall our previous discussion on hash values, hash values provide integrity. AH uses hashing algorithms to find out hash value of the IP header and attaches it with the IP header. This way it not only provides data integrity (since hash is calculated on payload as well) but also data origin integrity or authentication (since source address is part of the IP header as well).

Scanned with CamScanner

2. Encapsulating Security Payload (ESP)

✍ ***Definition :*** *The Encapsulating Security Payload (ESP) protocol is designed to provide confidentiality (through encryption), data integrity and data origin authentication over network communication.*

ESP can be applied with AH or without AH. Note here that ESP can itself provide integrity. It does not need AH for integrity. You have an option to additionally calculate integrity using AH. ESP in transport mode encrypts the actual payload (data) so that it cannot be read by an unauthorized entity. In tunnel mode, the IP header information is encrypted as well.

If you choose integrity service, the Integrity Check Value (ICV) is calculated on the following fields in the IP header.

- Security Parameter Index (SPI)
- Sequence Number
- Payload Data
- ESP trailer

If you choose confidentiality service, the ciphertext consists of the following fields in the IP header.

- Payload (Data)
- ESP trailer

(i) Transport mode for ESP, Fig. 5.3.18 shows before and after applying ESP.

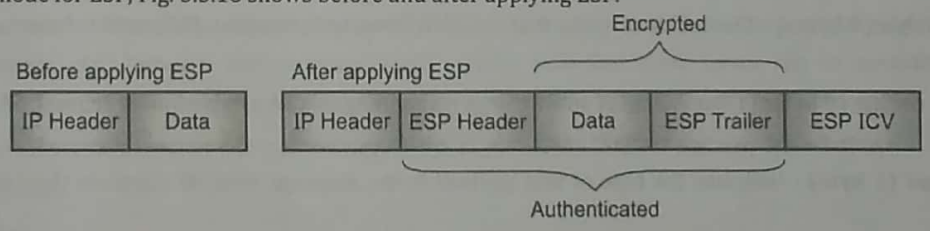


Fig. 5.3.18 : Transport mode for ESP

(ii) Tunnel mode for ESP, Fig. 5.3.19 shows before and after applying ESP.

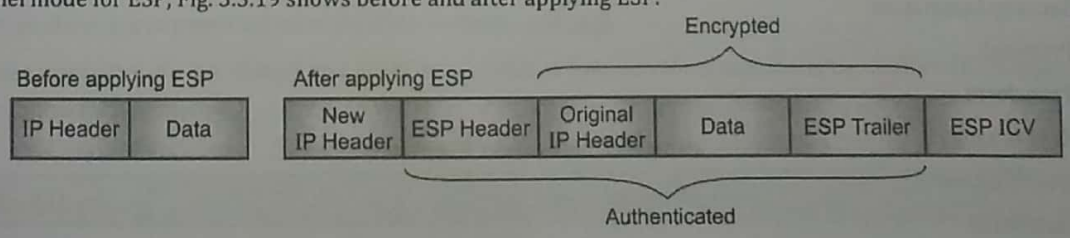


Fig. 5.3.19 : Tunnel mode for ESP

3. Internet Security Association and Key Management Protocol (ISAKMP)

✍ ***Definition :*** *Internet Security Association and Key Management Protocol (ISAKMP) provides a framework for authentication and key exchange.*

ISAKMP does not define the exact algorithms to be used. It is just a framework within which various exchange protocols can work.

Scanned with CamScanner

ISAKMP defines the procedures for;

- Authenticating communication devices
- Creation and management of Security Associations (SA)
- Key generation techniques
- Threat mitigation

ISAKMP is distinct from key exchange protocols in order to cleanly separate the details of security association management (and key management) from the details of key exchange. There may be many different key exchange protocols, each with different security properties. However, a common framework is required for agreeing to the format of SA attributes, and for negotiating, modifying, and deleting SAs. ISAKMP serves as this common framework. ISAKMP header is shown in Fig. 5.3.20.

<table>
<tr><td colspan="8">0 1 2 3 4 5 6 7</td><td colspan="8">8 9 10 11 12 13 14 15</td><td colspan="8">16 17 18 19 20 21 22 23</td><td colspan="8">24 25 26 27 28 29 30 31</td></tr>
<tr><td colspan="32">Initiator cookie</td></tr>
<tr><td colspan="32">Responder cookie</td></tr>
<tr><td colspan="8">Next payload</td><td colspan="8">Major version</td><td colspan="8">Minor version</td><td colspan="8">Exchange type</td><td colspan="8">Flags</td></tr>
<tr><td colspan="32">Message ID</td></tr>
<tr><td colspan="32">Length</td></tr>
</table>

Fig. 5.3.20 : ISAKMP header

- **Initiator cookie (8 bytes) :** The cookie of entity that initiated Security Association (SA) establishment, SA notification, or SA deletion.
- **Responder cookie (8 bytes) :** The cookie of entity that is responding to a SA establishment request, SA notification, or SA deletion.
- **Next Payload (1 byte) :** Indicates the type of first payload in the message. ISAKMP supports the following payload types :
  - None
  - SecurityAssociation
  - Proposal
  - Transform
  - KeyExchange
  - Identification
  - Certificate
  - CertificateRequest
  - Hash
  - Signature
  - Nonce
  - Notification
  - Delete
  - VendorID
  - NAT Discovery Payload

Scanned with CamScanner

- NAT Original Address Payload
- Reserved
- PrivateUse

- **Major version (4-bits) :** Major version of the ISAKMP protocol in use.
- **Minor version (4-bits) :** Minor version of the ISAKMP protocol in use.
- **Exchange Type (1 byte) :** The type of exchange in a given ISAKMP session. The primary difference between exchange types is the ordering of the messages and the payload ordering within each message.
- **Message ID (4 bytes) :** The unique message identifier.
- **Length (4 bytes) :** The length, in bytes, of the total message (header + payloads).

ISAKMP offers two phases of negotiation :

- **Phase 1 :** In the first phase, two entities agree on how to protect further negotiation traffic between themselves, establishing an ISAKMP SA.
- **Phase 2 :** The second phase of negotiation is used to establish security associations for other security protocols. This second phase can be used to establish many security associations. The security associations established by ISAKMP during this phase can be used by a security protocol to protect many message/data exchanges.

4. Internet Key Exchange (IKE)

Definition : *Internet Key Exchange (IKE) is the protocol used to set up a Security Association (SA) in the IPsec protocol suite.*

Recall from our earlier discussion on security association – A security association is a set of negotiated terms (algorithms, parameters, etc.) between two communicating entities such that these terms can be used in successive communication.

The following attributes are used by IKE and are negotiated as part of the ISAKMP Security Association :

- Encryption algorithm
- Hashing algorithm
- Authentication method
- Information about a group over which to do Diffie-Hellman exchange.

All of these attributes are mandatory and MUST be negotiated between the communicating entities. IKE supports the following attributes for negotiation.

Table 5.3.8 : IKE negotiated attributes

Attribute For	Supported Attributes
Encryption algorithm	DES, IDEA, Blowfish, 3DES, CAST, RC5, AES
Hashing algorithm	MD5, SHA, TIGER
Authentication method	Pre-shared key, DSS Signature, RSA Signature, Encryption with RSA, Revised encryption with RSA
Group information	MODP (modular exponentiation group), ECP (elliptic curve group over GF[P]), EC2N (elliptic curve group over GF[2^N])

IKE works in two phases :

In Phase 1, following functions are carried out :

Scanned with CamScanner

- Mutual authentication of the communicating entities.
- Negotiating cryptographic parameters.
- Creating session keys.

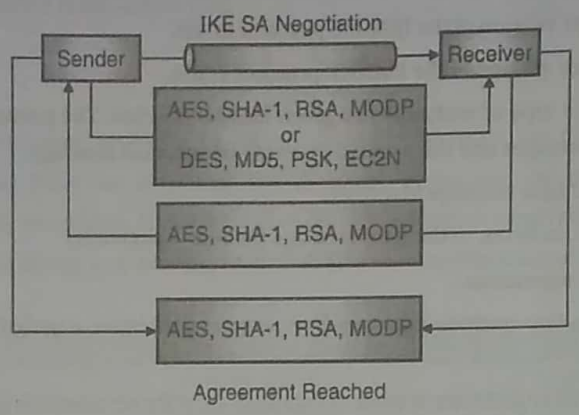


Fig. 5.3.21 : IKE Agreement Process

In Phase 2, an IPSec tunnel is negotiated by creating keying material for the IPSec tunnel to use (either by using the IKE phase one keys as a base or by performing a new key exchange).

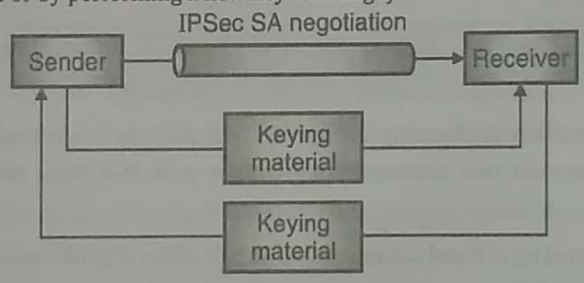


Fig. 5.3.22 : IPSec Tunnel Negotiation

5.4 Secure Email

Q. Write in brief about Email security. **MU - May 16, 5 Marks**

Around billions of emails are sent across the globe every day. Emails have become the primary source of official communication. With such a wide use of emails, attackers are inclined and motivated to intercept emails and get the message and at times modifying the messages before the recipient gets it. It is important that you secure the email communication as any other form of communication. In this section, you will learn about a couple of email security standards that you could use.

5.4.1 Pretty Good Privacy (PGP)

Q. How does PGP achieve confidentiality and authentication in emails? **MU - Dec. 15, 5 Marks**

Definition : *Pretty Good Privacy (PGP) is an email security program that was developed in 1991. It is based on public key cryptography.*

Scanned with CamScanner

1. Web of Trust

Fig. 5.4.1 : Trust using Public Key Cryptography

In Public Key Cryptography system that depends upon a third-party Certificate Authorities (CAs) to establish trust, there is no mutual trust amongst the users.

Each user trusts a reputed CA and thus CA plays a predominant role in establishing the trust so that communication can happen between users. If there is no CA, the trust relationship is not established and thus the communication may not happen.

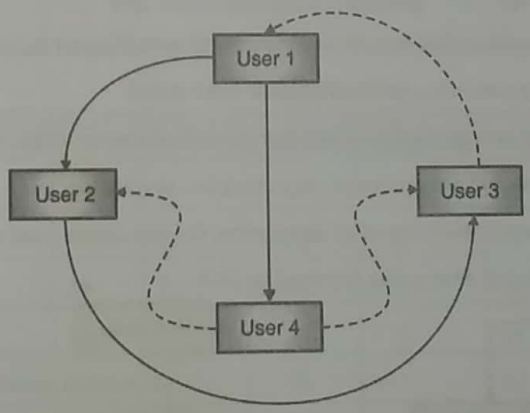


Fig. 5.4.2 : Web of Trust

Unlike the traditional Public Key Cryptography system that depends upon a CA to establish trust amongst the users, the earlier implementations of PGP did not use the regular CAs for issuing certificates. It used "Web of Trust" where each user generates and distributes his or her public key, and users sign each other's public keys, which creates a community of users who trust each other. This is different from the CA approach, where no one trusts each other; they only trust the CA.

So, basically, PGP is a system of "I don't know you, but my friend Alice says that you can be trusted, so I will trust you on her words". In the Fig. 5.4.2, as you understand, there is a trust relationship (User 1, User 2) and (User 2, User 3). Now, when User3 needs to communicate with User 1, it establishes a trust inherited from its prior trust on User 2. There is no third-party involved in this scenario. Each user keeps in a file, referred to as a key ring, a collection of public keys he has received from other users. Each key in that ring has a parameter that indicates the level of trust assigned to that user and the validity of that particular key.

Scanned with CamScanner

5.4.1(A) PGP Services

PGP provides the following services. You can use one or more services at a time. For example, if you intend to use only encryption service, you can do so. If you intend to use only the digital signature service, you can do so. Let's learn a brief about these services.

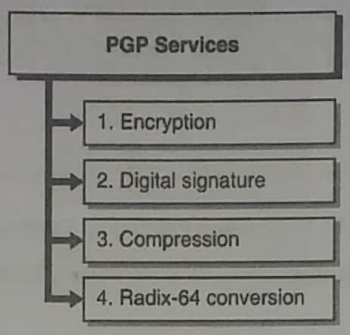


Fig. 5.4.3 : PGP Services

1. **Encryption**

   (a) The sender creates a message.

   (b) PGP generates a random number that is used as symmetric key to encrypt it.

   (c) The symmetric key is encrypted using receiver's public key.

   (d) Encrypted message and the encrypted symmetric key are sent to the receiver.

   (e) The receiver decrypts the encrypted symmetric key using her private key.

   (f) Once the receiver gets the symmetric key after decryption, the key can be used to decrypt the message.

   Fig. 5.4.4 illustrates steps involved in encryption followed by PGP.

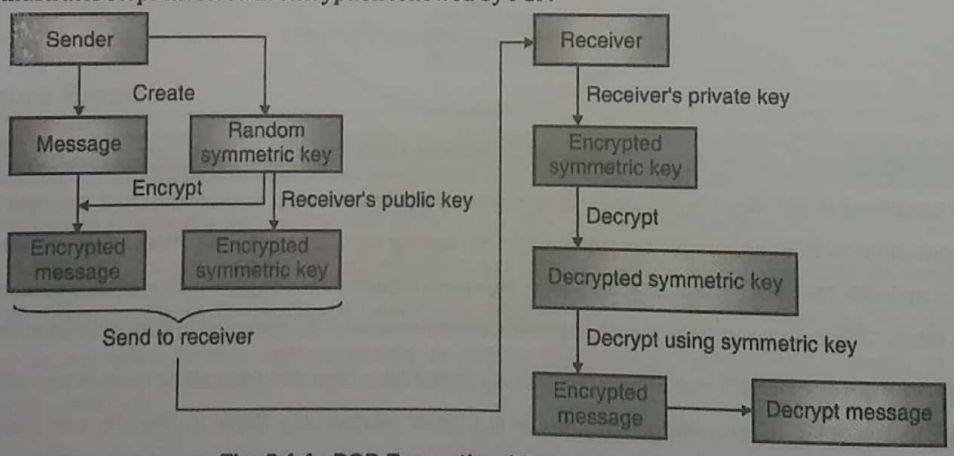


Fig. 5.4.4 : PGP Encryption / Decryption Process

2. **Digital Signature :** The digital signature uses a hash code or message digest algorithm, and a public-key signature algorithm. You have already learnt digital signature in detail in Chapter 2. Refer it for a quick refresher.

3. **Compression :** PGP compresses the message after applying the signature but before encryption. Compression has the added side effect that some types of attacks can be avoided by the fact that even the slightly altered, compressed data does not decompress without errors. This side security benefit is operationally useful.

Scanned with CamScanner

4. **Radix-64 conversion :** R64 conversion is useful for compatibility of emails across varied systems. PGP's underlying native representation for encrypted messages, signature certificates, and keys is a stream of arbitrary octets. Some systems only permit the use of blocks consisting of seven-bit, printable text.

So, for transporting PGP's native raw binary octets through channels that are not safe to raw binary data, a printable encoding of these binary octets is needed. PGP provides the service of converting the raw 8-bit binary octet stream to a stream of printable ASCII characters, called Radix-64 encoding.

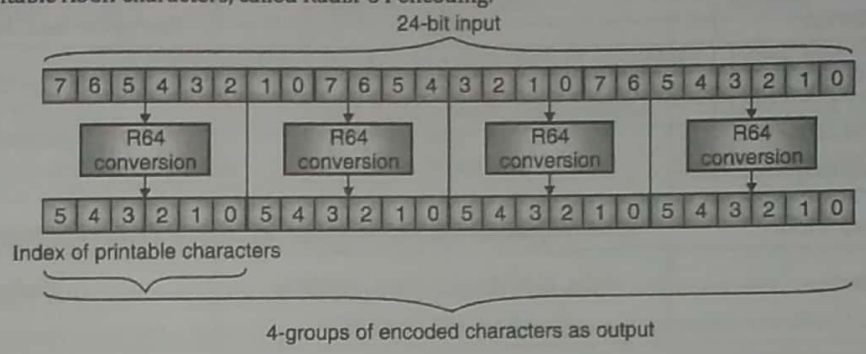


Fig. 5.4.5 : Radix-64 encoding

Each 6-bit group is used as an index into an array of 64 printable characters as shown in Table 5.4.1.

Table 5.4.1 : Encoding Map

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v	(pad)	=
14	O	31	f	48	w		
15	P	32	g	49	x		
16	Q	33	h	50	y		

Scanned with CamScanner

5.4.1(B) PGP Algorithms

The Table 5.4.2 is a summary of various PGP services and algorithms they support.

Table 5.4.2 : Summary of various PGP services and Algorithms

PGP Service	Supported Algorithm	Purpose
Public Key	RSA	Encrypt or Sign (Symmetric Key)
Public Key	Elgamal	Encrypt (Symmetric Key)
DSA (Digital Signature Algorithm)	DSS	Sign (Message)
Symmetric Key	IDEA, TripleDES, CAST5, Blowfish, AES	Bulk encryption (message)
Hash	MD5, RIPEMD160, SHA1, SHA256, SHA384, SHA224, SHA512	Hashing
Compression	ZIP, ZLIB, BZip2	Compress messages

5.4.2 S/MIME

✍ ***Definition :*** *S/MIME (Secure/Multipurpose Internet Mail Extensions) provides a consistent way to send and receive secure MIME data (emails).*

It is based on certificates (Public Key Cryptography) and works as you have learnt in previous sections and units.

5.4.2(A) S/MIME Services

S/MIME provides the following cryptographic security services for electronic messaging applications.

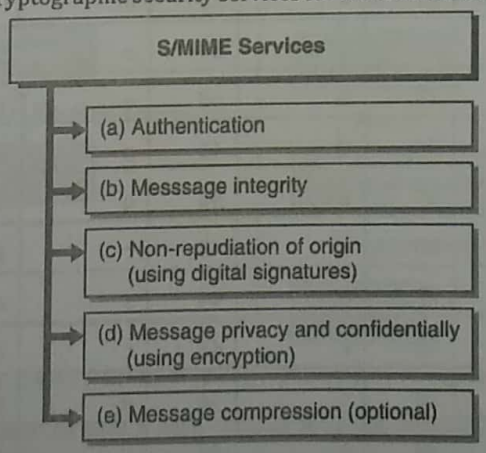


Fig. 5.4.6 : S/MIME Services

These services are provided using same techniques as you have learnt in the previous sections and units.

Scanned with CamScanner

5.4.2(B) S/MIME Algorithms

Summary of various S/MIME services and algorithms they support are shown in The Table 5.4.3.

Table 5.4.3 : S/MIME Supported services and algorithms

Sr. No.	S/MIME Service	Supported Algorithms	Purpose
1.	Message Integrity	SHA-256, SHA-1, MD5	Hashing
2.	Non-repudiation, Authentication	RSA and DSA with Hashing algorithms	Digital Signature
3.	Key Encryption	RSA, RSAES-OAEP, Diffie-Hellman	Encrypting Symmetric key
4.	Privacy and Confidentiality	AES, DES, Triple DES	Message Encryption

5.4.2(C) S/MIME Cryptographic Message Syntax (CMS)

S/MIME standard describes a protocol for adding cryptographic signature and encryption services to MIME (email) data. The MIME standard provides a general structure for the content of Internet messages and allows extensions for new content-type-based applications. The S/MIME specification defines how to create a MIME (email) body part that has been cryptographically enhanced according to the Cryptographic Message Syntax (CMS).

There are 4 types of CMS used in S/MIME :

1. **Data Content Type :** This is the original plaintext form of the email message. It has an identifier that is referred whenever it is compressed, encrypted or digitally signed.
2. **SignedData Content Type :** SignedData content type is used when a sender needs to apply a digital signature to a message. Applying a signature to a message provides authentication, message integrity, and non-repudiation of origin.
3. **EnvelopedData Content Type :** This content type is used to apply data confidentiality (via encryption) to a message. A sender needs to have access to a public key (for encrypting the symmetric key used for actual encryption of the message) for each intended message recipient to use this service. At the receiver's end, the receiver uses her private key to decrypt the symmetric key used for encrypting the original message. Once the symmetric key is available, it can be used to decrypt the encrypted message and thus read the email message.
4. **CompressedData Content Type :** This content type is used to apply data compression to a message. This content type does not provide authentication, message integrity, non-repudiation, or data confidentiality. It is only used to reduce the size of the message.

5.4.3 Firewalls

Q. What is a firewall ? What are the firewall design principles ? **MU - May 16, 5 Marks**

Q. What are firewals ? **MU - Dec. 16, 3 Marks**

Your computer is connected to the internet. How do you protect it from someone trying to access it over the internet? How do you prevent some rogue programs on your computer to send information to the attacker? Firewalls could be a mechanism.

✍ ***Definition :*** *Firewalls are network security systems that protect the computing resources on a trusted network from unauthorized access.*

(Copyright No. - L82548/2019)

Scanned with CamScanner

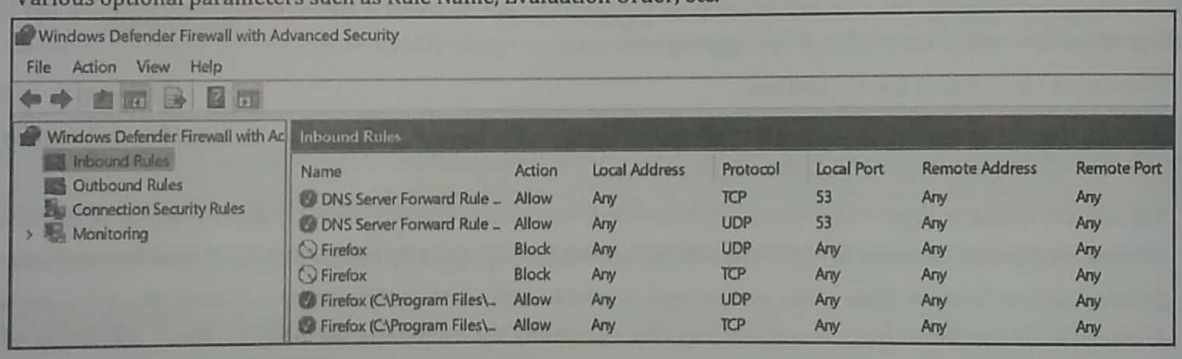
For example, you can access google.com website but not it's webserver's operating system. If you try connecting to the webserver except over the HTTP or HTTPS, the connection would be denied. That's what a firewall does at a high level.

You need to define various rules, as per your security requirements, in the firewall and the firewall evaluates those rules before granting or denying access to the requested resource.

5.4.3(A) Components of a Firewall Rule

Typically, a firewall rule consists of the following parameters :

1. Source IP address or hostname.
2. Destination IP address or hostname.
3. Source Port number.
4. Destination Port number.
5. Direction of communication [inbound or outbound].
6. Protocol name [TCP, UDP, ICMP or various others].
7. Action [allow, deny, log, etc.].
8. Various optional parameters such as Rule Name, Evaluation Order, etc.



Name	Action	Local Address	Protocol	Local Port	Remote Address	Remote Port
DNS Server Forward Rule ...	Allow	Any	TCP	53	Any	Any
DNS Server Forward Rule ...	Allow	Any	UDP	53	Any	Any
Firefox	Block	Any	UDP	Any	Any	Any
Firefox	Block	Any	TCP	Any	Any	Any
Firefox (C:\Program Files\...	Allow	Any	UDP	Any	Any	Any
Firefox (C:\Program Files\...	Allow	Any	TCP	Any	Any	Any

This is a snapshot of Microsoft® Windows® Firewall.

5.4.3(B) Classification of Firewalls

Q. Explain the different types of firewalls and mention the layer in which they operate. **MU- Dec. 16, 7 Marks**

Q. What are the types of firewalls ? **MU- May 17, 5 Marks**

Firewalls can be classified based on various attributes. Let's learn about their types.

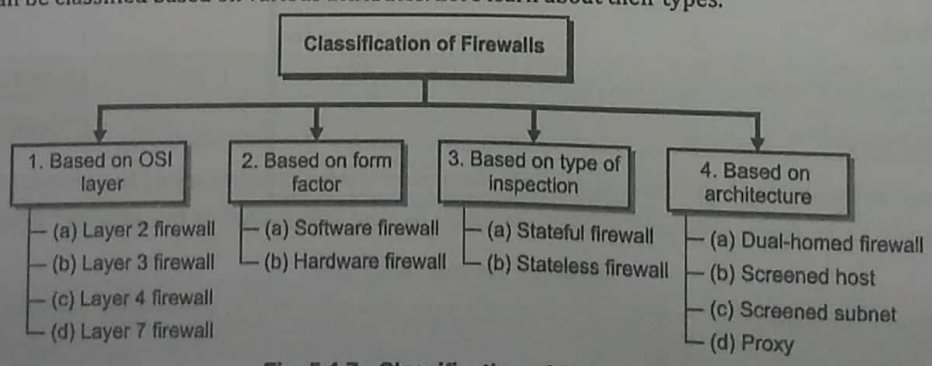


Fig. 5.4.7 : Classification of Firewalls

Scanned with CamScanner

1. **Based on the OSI Layer :** As you understand, OSI is a conceptual networking model. Based on the various layers, firewalls can be classified as following :

   (a) **Layer 2 Firewall :** These firewalls work at the "Data Link" layer of the OSI model. These firewalls require MAC, VLAN or device hardware level information to operate. One of the greatest advantage of these types of firewalls is that they are not IP dependent.

   (b) **Layer 3 Firewall :** These firewalls work at the "Network" layer of the OSI model. These filter traffic based on source/destination IP, port, and protocol. These are one of the most prevalent types of firewalls in use today. These are also called as Stateless firewalls. These are also called first-generation firewalls.

   (c) **Layer 4 Firewall :** These firewalls work at the "Transport" layer of the OSI model. These firewalls do everything that a Layer 3 firewall does and additionally track the active network connections and allow/deny traffic based on the state of those connections. These can effectively stop DoS attacks such as the ones based on TCP SYN/ACK as these are aware of the state of connection. These are also called as Stateful firewalls. These are also called second-generation firewalls.

   (d) **Layer 7 Firewall :** These firewalls are called Layer 7 but can work at three layers – Session, Presentation and Application. For simplicity, these are just called Layer 7 firewalls. Layer 7 firewalls do everything that a Layer 4 firewall does and additionally include the ability to intelligently inspect the contents of the network packets passing through them.

   For example, a Layer 7 firewall could deny all the HTTP requests from Korean IP addresses. They have the actual packet content level visibility and are the most advanced types of firewall in use today. These are also called third-generation firewalls.

2. **Based on the form factor :** Form factor or the footprint is the way the firewall is actually packaged and deployed. They can be classified as,

   (a) **Software Firewalls :** These firewalls work as a software program and require an operating system to run them. All the implementation logic is coded in software and they are installed, patched, upgraded and maintained like a regular computer software. These firewalls could work at any of the OSI layers as discussed before.

   (b) **Hardware Firewalls :** Firewalls can also be deployed as a hardware device. Hardware firewall may have better performance and they come packaged in a ready to use hardware device. Like any other firewall, you need to configure it as per your security requirements.

3. **Based on the type of inspection :** Firewalls can keep track of connections or just work based on the configured rules. Based on their inspection types, these can be classified as,

   (a) **Stateful Firewalls :** These firewalls keep track of the state of connections apart from the defined firewall rules. These precisely understand various handshake protocols and can effectively stop attacks that try to manipulate connection establishment or maintenance process.

   (b) **Stateless Firewalls :** Stateless firewalls typically work at the Layer 3 and take decisions based on the defined rule parameters such as IP, Port and Protocol. These do no track connection states and cannot effectively protect against attacks that manipulate connection processes.

4. **Based on architecture :** Firewalls can be deployed in many ways. They have special properties that make them suitable for one deployment type over the another. Based on the deployment possibilities, firewalls can be classified as,

   (a) **Dual-homed Firewalls :** A Dual-Homed Firewall has two interfaces - one facing the external network and the other facing the internal network. It receives the external packets on one of its interfaces, evaluates firewall rules, and passes on the traffic to the designated internal resources via the second interface. The two interfaces are kept separate to isolate the external traffic with the internal traffic physically.

(Copyright No. - L82548/2019)

Scanned with CamScanner

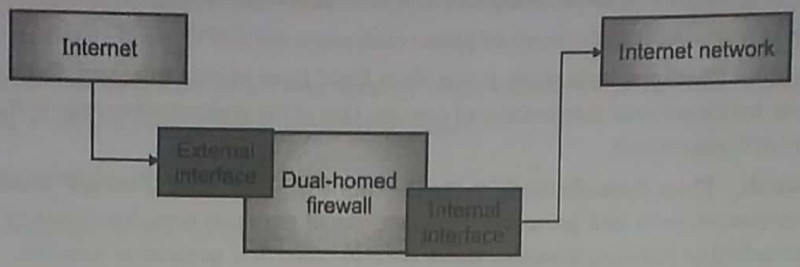


Fig. 5.4.8 : Dual-homed Firewalls

(b) Screened Host : In a screened host firewall, all internet (and other regulated) traffic goes through the firewall, no matter what. The internet router device first screens (filters) all the packets that are relevant to the network and then passes it to the Screened Host firewall for further inspection and applying rules.

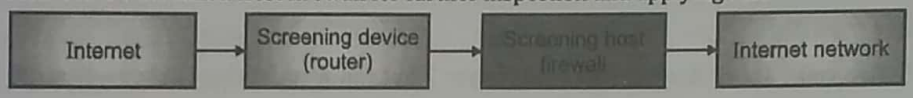


Fig. 5.4.9 : Screened Host

(c) Screened Subnet : In screened subnet architecture, two firewalls are used. One just after the external network and the one just before the internal network. Any network that lies between the two firewalls is called a demilitarized zone (DMZ). You place your public facing servers such as webservers, email servers etc. in DMZ. An attacker would have to bypass both the firewalls before she can hit the internal network. This kind of architecture is commonly used in the industry today.

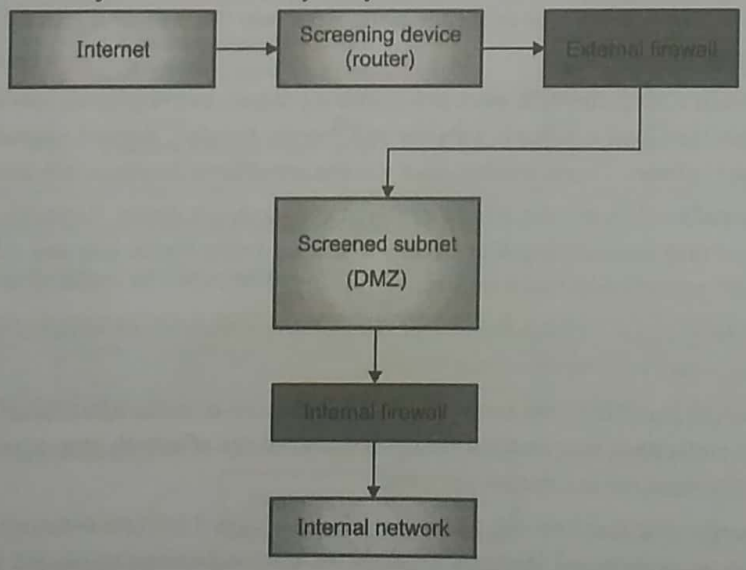


Fig. 5.4.10 : Screened Subnet

(d) Proxy firewall : A proxy firewall stands between the trusted and the untrusted network and takes allow or deny decisions after careful inspection of what is being passed along. Like a regular proxy, the proxy firewall breaks the connection between the source and the destination. After examining the traffic, it self-establishes a connection with the destination and passes the intended traffic to the destination as if the packets were originating from it.

Scanned with CamScanner

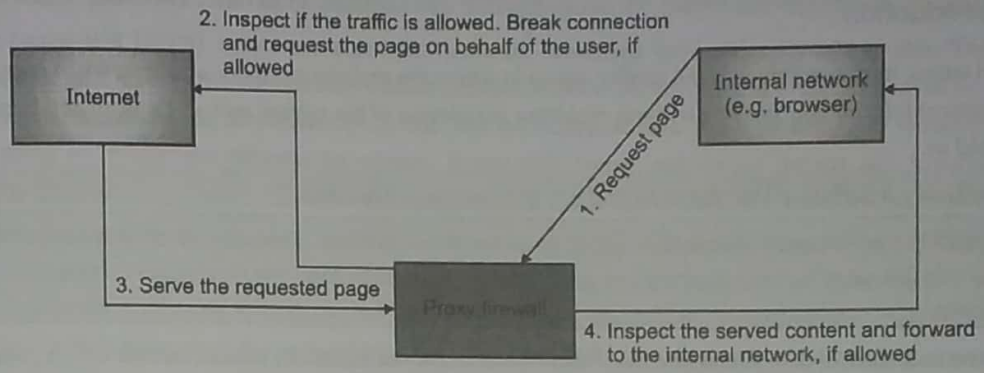


Fig. 5.4.11 : Proxy Firewell

5.4.3(C) Challenges in Managing and Deploying Firewalls

1. **Performance :** Since the traffic (as well as the content) needs to pass through the firewalls, there is a little performance degradation of the network. The adequate traffic examination may add up to a few milliseconds of latency on each packet.
2. **Business agility :** Firewall rules are usually manually added, edited or deleted. The pace of business might be too high to require several changes to the firewall rules frequently. Keeping up with these changes without making errors is difficult.
3. **Costs :** Modern (or advanced) firewalls that provide content and protocol level inspection may be cost-prohibitive for small or medium sized organizations.
4. **Insider attacks :** Firewalls are usually designed and deployed to protect a trusted network from an untrusted network. But, if there were other vulnerabilities (such as a missing OS security patch) that were exploited such that an attacker is already on the trusted network, firewalls might not be able to protect or limit damages to the other resources on the trusted network.
5. **Managing firewalls themselves :** Like your OS, printers or other software or hardware devices, firewalls need to be installed, patched, updated, etc. to remain operational. This adds a management overhead. Additionally, firewalls could have known vulnerabilities that need to be patched else a firewall that itself is lacking protection may not be very useful in providing you the required level of protection.

Note : Irrespective of what challenges or limitations firewalls may have, they are heavily used throughout the industry. You cannot just imagine any network without several firewalls in place to monitor, inspect and manage legitimate traffic and separate it from the illegitimate traffic. So, just know what some of the management or technical challenges are and do not worry too much about them.

5.4.4 Intrusion Detection Systems (IDS)

Q. Explain the significance of an Intrusion Detection System for securing a network. **MU - May 16, 6 Marks**

Q. Write short notes on IDS. **MU - May 19, 4 Marks**

Intrusion means to encroach (or to capture) a place. For example, suppose there is a vacant site and you manage to build a small hut there without seeking permission of the site owner, that is precisely what intrusion is. You, as an individual, are trying to intrude on someone's property.

Scanned with CamScanner

5.4.4(A) Introduction

In digital terms, intrusion refers to the similar situation where the malicious code or attackers try to encroach (forcibly enter and capture) information systems without requiring permission of the system owner. An Intrusion Detection System (IDS) is defined as,

Definition : *A software that helps to find out if a system is breached.*

Note : Breach is a word used in information security domain to describe any form of attack or unauthorized actions. You can use this word to mean anything that refers to the undesired actions and outcomes with respect to information security.

So, in a nutshell, IDS can help you to find out if there were undesired actions or attacks carried out on your information systems. IDS works using various techniques as we will see later in this section. Note here that IDS does not help to prevent the attacks unlike anti-virus. It is only a system that can gather system information and find out if everything looks alright or not.

5.4.4(B) Need for IDS

IDS is one of the software-based security mechanisms that help to protect information system. At a high level, it is needed for the following reasons,

1. **Defense in Depth :** As you saw in the security architecture section, security is about minimizing the damage that can be possibly done. Defense in depth (or the layered approach) of security designing ensures that even if one of the controls is to fail, the overall security of the system would still be possibly healthy. IDS fulfils this need to bring an added layer of protection where any breaches or their possibilities can be identified quickly.
2. **Automate intrusion detection :** Imagine that you have a large set of machines, say 1,000 and more. How would you inspect each and every machine and find out if there were attacks or attempts to attack it? IDS helps you to automate this need and alert you when it detects any threat or likely a breach.
3. **Corrective actions :** Learning from threats or breaches that the IDS identifies, you can take corrective actions on your infrastructure design and could possibly strengthen its security. You might have some unprotected areas in your infrastructure that can be highlighted with the use of IDS.

5.4.4(C) Types of IDS

Broadly speaking, IDS can be classified based on **what** it monitors and **how** it monitors.

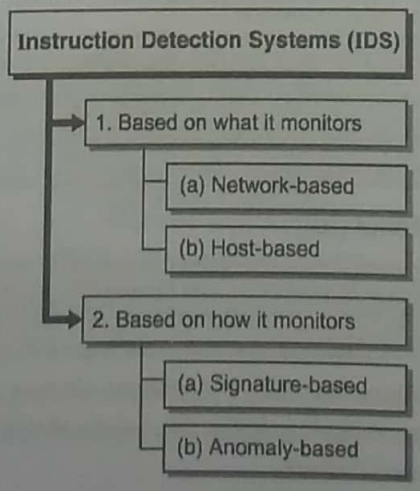


Fig. 5.4.12 : Types of IDS

Scanned with CamScanner

1. **Based on what it monitors :** IDS can be classified into Network-based IDS (NIDS) and Host-based IDS (HIDS).

(a) **Network-based IDS (NIDS) :** Network-based IDS evaluate intrusions from the networking side. They watch all network traffic as it reaches the various information systems. If there are any alerting situations based on the network traffic analysis, it notifies the administrator to take the corrective actions. NIDS do not have visibility into what's actually going on within the information system. It can only watch and detect threats and breaches from the networking viewpoint.

(b) **Host-based IDS (HIDS) :** Host-based IDS are typically installed on the individual information systems and then they watch for suspicious activities occurring on the system. System entities such as system services and processes, system files, privileged user actions, downloads, etc. are closely monitored to detect any undesired activities. HIDS do not have visibility into what's going on at the networking side of the system. It can only watch and detect activities with respect to individual machines only.

2. **Based on how it monitors :** IDS can be classified into Signature-based and Anomaly-based.

(a) **Signature-based :** Like banks and other organizations use human signature to validate requests and transactions, similarly Signature-based IDS has a pre-loaded database of various attack signatures (patterns of a possible attack). When it watches the activities, it constantly compares the activities' patterns with that in the database. If a match is found, it raises an alert. If you notice, there are 3 things to understand here:

(i) Signature based IDS can only detect attacks if it already and historically knows about an attack pattern.

(ii) For new types of attack, signature-based IDS would not raise alerts.

(iii) It is important for you to update the signature definitions time to time (like how you do in anti-virus system).

(b) **Anomaly-based :** Anomaly typically means "deviation from routine". For example, if you wake up at 7 AM every day and one day you wake up at 4 AM, that is an anomaly situation. If I were to plot your wake-up time graph, 4 AM would show up away from your regular wake-up time. That 4 AM point on the graph is called outlier (or away from other samples).

Similarly, the Anomaly-based IDS first establishes the baseline (common routine) of activities. It might take up to 2-3 weeks to "learn" what's right for a system. Once the learning phase is over, it would watch out for any activities that are not part of that baseline and raise alerts. If you notice, there are 3 things to understand here as well :

(i) It does not require signature and hence can possibly detect new attacks.

(ii) It requires a learning period during which the system should have undergone all possible activities.

(iii) If you plan to use the system for other purposes, you need to retrain the IDS.

5.4.4(D) Limitations and Challenges of IDS

1. **Does not prevent attacks :** As you understand, IDS can only detect and raise alerts when it finds a likelihood of a breach. It cannot prevent or block the breach from happening.

2. **High rate of false alerts (noise) :** IDS might generate a lot of false alerts. It could happen so for example, when there is a new traffic from a source that IDS has not seen before. You need to spend your resources to take a note of each alert and appropriately deal with it – either fix it or ignore it.

3. **Complex systems :** IDS systems are typically complex in nature and require regular administrative actions and tuning for adequate operations.

4. **Bypassing IDS :** Advanced attackers know what actions and activities a version and brand of IDS can detect and what not. They tune their activities to bypass such detection mechanisms and go undetected.

(Copyright No. - L82548/2019)

Scanned with CamScanner

Review Questions

Here are a few review questions to help you gauge your understanding of this chapter. Try to attempt these questions and ensure that you can recall the points mentioned in the chapter.

[A] Network Security Basics

Q. 1 Write a short note on network security. **(4 Marks)**

[B] TCP/IP Vulnerabilities (Layer wise)

Q. 2 Describe packet sniffing and protection against it. **(8 Marks)**

Q. 3 Describe ARP spoofing and protection against it. **(8 Marks)**

Q. 4 Describe port scanning, its techniques and protection against it. **(8 Marks)**

Q. 5 Describe IP spoofing and protection against it. **(8 Marks)**

Q. 6 Describe DNS spoofing and protection against it. **(8 Marks)**

Q. 7 Describe DDoS attack and protection against it. **(8 Marks)**

Q. 8 Describe the types of DDoS attacks. **(8 Marks)**

Q. 9 Write a short note on botnet. **(4 Marks)**

[C] Internet Security Protocols

Q. 10 What is SSL? List its goals. **(6 Marks)**

Q. 11 With a block diagram, give a brief Overview of SSL protocol. **(8 Marks)**

Q. 12 Describe SSL Record Layer Protocol. **(8 Marks)**

Q. 13 Describe SSL Alert Protocol. **(8 Marks)**

Q. 14 Describe SSL Handshake Protocol. **(8 Marks)**

Q. 15 What is HTTPS? Why do we need it? **(4 Marks)**

Q. 16 Compare HTTP and HTTPS. **(8 Marks)**

Q. 17 List the Motivation / Benefits of using HTTPS. **(6 Marks)**

Q. 18 Write a short note on Security Association. **(4 Marks)**

Q. 19 Describe IPSec modes of operation. **(8 Marks)**

Q. 20 List theapplicationsof IPSec. **(4 Marks)**

Q. 21 List the benefits of IPSec. **(4 Marks)**

Q. 22 List the usage of IPSec. **(4 Marks)**

Q. 23 List the steps involved in the working of IPSec. **(4 Marks)**

Q. 24 Describe IPSec Authentication Header (AH). **(8 Marks)**

Q. 25 Describe IPSec Encapsulating Security Payload (ESP). **(8 Marks)**

(Copyright No. - L82548/2019)

Scanned with CamScanner

Q. 26 Describe Internet Security Association and Key Management Protocol (ISAKMP). **(8 Marks)**

Q. 27 Describe Internet Key Exchange (IKE). **(8 Marks)**

[D] Secure Email

Q. 28 What is Pretty Good Privacy (PGP)? Explain the concept of "Web of Trust". **(8 Marks)**

Q. 29 Describe PGP Services. **(8 Marks)**

Q. 30 Describe the Radix-64 conversion process in PGP. **(8 Marks)**

Q. 31 List the various algorithms used in PGP and their purpose. **(6 Marks)**

Q. 32 What is S/MIME? What services does it provide? **(6 Marks)**

Q. 33 List the various algorithms used in S/MIME and their purpose. **(4 Marks)**

Q. 34 Write a short note on S/MIME Cryptographic Message Syntax (CMS). **(4 Marks)**

[E] Firewalls

Q. 35 Write a firewall rule of your choice and describe its components. **(8 Marks)**

Q. 36 Classify firewalls based on the OSI Layer. **(4 Marks)**

Q. 37 Classify firewalls based on Form Factor. **(4 Marks)**

Q. 38 Classify firewalls based on the type of Inspection. **(4 Marks)**

Q. 39 Classify firewalls based on architecture. **(8 Marks)**

Q. 40 What are some of the major challenges in managing and deploying firewalls? **(6 Marks)**

[F] Intrusion Detection Systems (IDS)

Q. 41 What is Intrusion Detection Systems (IDS)? Why do you need it? **(6 Marks)**

Q. 42 Describe the types of Intrusion Detection Systems (IDS). **(8 Marks)**

Q. 43 Explain the limitations and challenges of IDS. **(4 Marks)**

❑❑❑

Scanned with CamScanner

6 System Security

Syllabus

At the end of this unit, you should be able to understand and comprehend the following syllabus topics :

- Software Vulnerabilities
  - Buffer Overflow
    - Format string
  - SQL injection
- Malware
  - Viruses
  - Worms

6.1 Software Vulnerabilities

Q. List various Software Vulnerabilities. How vulnerabilities are exploited to launch an attack? **MU - May 19, 5 Marks**

Today it is hard to imagine any software without any vulnerabilities. Some vulnerabilities are found during software testing (for example penetration testing, static code analysis, dynamic code analysis, etc.) and others are found in the field (when the software is put to mass use). Software vulnerabilities that are found after the software has been released are mitigated (fixed) by giving out software patches or updates. You would have seen your browser, operating system and apps updating multiple times in its course of lifetime. These updates include both feature updates (new functionalities in the software) and also security updates (fixing new vulnerabilities found in the software).

Software developing companies have security vulnerability update programs and teams that regularly investigate all reported security vulnerabilities form the field. As you learnt in the previous chapter, these companies also run bug bounty programs that help them find security vulnerabilities in their products and mitigate them before these vulnerabilities could be mass exploited.

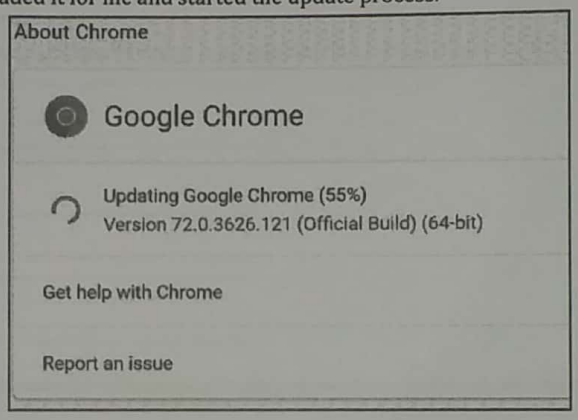
Let's see an example, Depending on the type of vulnerability, it can be exploited either manually or using scripts or tools. Attacker has deep understanding of how the vulnerability works and she carries out the required steps in the right sequence to exploit the vulnerability. How a particular vulnerability is exploited really depends on the type of vulnerability. There is no fix mechanism. It depends on the complexity of the vulnerability and also on the skills of the attacker.

Scanned with CamScanner

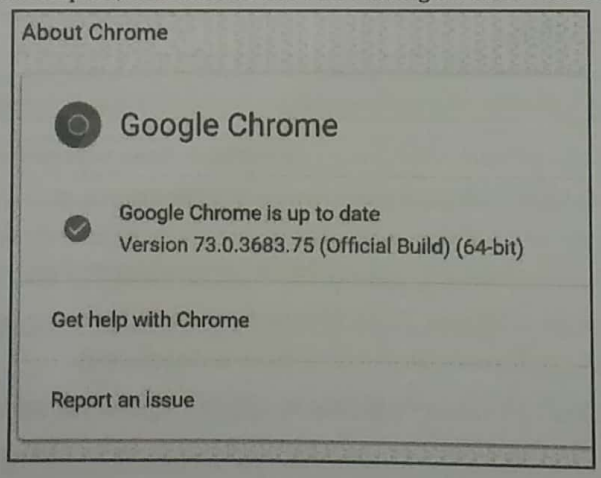
[An interesting note – I was writing this section on 17-Mar-2019. I opened the Google Chrome browser on my laptop to give you an example of security updates in the Google Chrome browser and you know what? When I opened the browser, I found that an update was waiting to be installed. It gave me a smile.

It would make a perfect example for you! It showed how security vulnerabilities are updated regularly in the products to avoid mass exploits. I stopped writing for a while and updated the browser that contained several security fixes. Writing can wait, security fixes cannot. I also grabbed a few snapshots for you meanwhile.]

As soon as I went to the version section of the Google Chrome browser, the browser automatically found that a new update was waiting and downloaded it for me and started the update process.



- Once the update process was complete, it showed me that I am running the latest version of Google Chrome.



Now, I wanted to find out what security updates were installed as part of the version 73.0.3683.75. The Chrome release website had the complete listing of the security vulnerabilities that were fixed in the version 73.0.3683.75.

https://chromereleases.googleblog.com/2019/03/stable-channel-update-for-desktop_12.html

Scanned with CamScanner

Chrome Releases: Stable Channe

https://chromereleases.g...

Chrome 73.0.3683.75 contains a number of fixes and improvements -- a list of changes is available in the log. Watch out for upcoming Chrome and Chromium blog posts about new features and big efforts delivered in 73.

Security Fixes and Rewards

Note: Access to bug details and links may be kept restricted until a majority of users are updated with a fix. We will also retain restrictions if the bug exists in a third party library that other projects similarly depend on, but haven't yet fixed.

This update includes 60 security fixes. Below, we highlight fixes that were contributed by external researchers. Please see the Chrome Security Page for more information.

[$TBD][913964] High CVE-2019-5787: Use after free in Canvas. Reported by Zhe Jin（金哲），Luyao Liu(刘路遥) from Chengdu Security Response Center of Qihoo 360 Technology Co. Ltd on 2018-12-11

[$N/A][925864] High CVE-2019-5788: Use after free in FileAPI. Reported by Mark Brand of Google Project Zero on 2019-01-28

[$N/A][921581] High CVE-2019-5789: Use after free in WebMIDI. Reported by Mark Brand of Google Project Zero on 2019-01-14

[$7500][914736] High CVE-2019-5790: Heap buffer overflow in V8. Reported by Dimitri Fourny (Blue Frost Security) on 2018-12-13

[$1000][926651] High CVE-2019-5791: Type confusion in V8. Reported by Choongwoo Han of Naver Corporation on 2019-01-30

[$500][914983] High CVE-2019-5792: Integer overflow in PDFium. Reported by pdknsk on 2018-12-13

[$TBD][937487] Medium CVE-2019-5793: Excessive permissions for private API in Extensions. Reported by Jun Kokatsu, Microsoft Browser Vulnerability Research on 2019-03-01

[$TBD][935175] Medium CVE-2019-5794: Security UI spoofing. Reported by Juno Im of Theori on 2019-02-24

[$N/A][919643] Medium CVE-2019-5795: Integer overflow in PDFium. Reported by

- Do you see that 60 security fixes were done in this version? Huge, isn't it?
- Do you see the $ values against some of the security vulnerabilities? Those are the bug bounty program rewards that would be given to the security researchers who reported the vulnerabilities to the Google Chrome security team.
- By the way, do you also see a vulnerability by name "buffer overflow"? Smile, you will read about it in this chapter.

Many software vulnerabilities are well-known, and the applications are frequently examined to check if those vulnerabilities exist. You will read about a few such vulnerabilities in this section. Gear up!

6.2 Buffer Overflow

Q. Write in brief about - Buffer overflow attack. MU - Dec. 15, 5 Marks

Q. Write short notes on Buffer overflow. MU - May 19, 4 Marks

Scanned with CamScanner

Buffer overflow is also called buffer overrun or buffer overwrite. But before you proceed with learning what a buffer overflow is let's understand what a buffer is.

Definition : *Buffer is a section of computer memory for temporarily storing information.*

When you run programs or enter data into programs, they are stored in the buffer until they are finally saved on the hard disk. All the programs that you run are loaded into memory before they can be executed. Any data that the program requires for its operation is stored in the memory as well. This memory space is called virtual address space. The virtual address space is divided into various segments and each segment is designated a specific purpose.

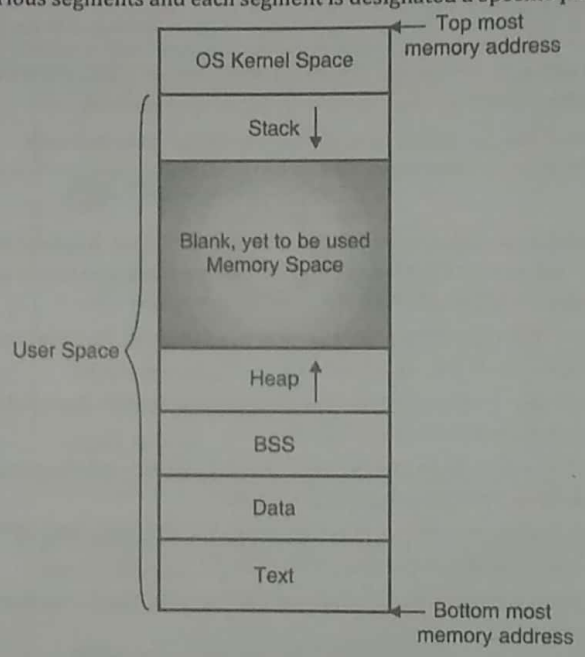


Fig. 6.2.1 : Memory Layout

- **OS Kernel Space :** The topmost memory space is occupied and reserved for the OS Kernel. User programs or data do not have access to this memory area.
- **Stack :** Stack is where the user executable programs are loaded before execution. The Stack grows downwards (towards lower memory addresses) to occupy the unused memory space. The stack pointer references to the top of the stack (the last occupied stack address) all the time.
- **Heap :** Heap is where the values for various dynamic variables, as required by the executing programs, are stored. Heap space is dynamically allotted as per the program needs and the exact heap space is not pre-determined. Heap grows upwards (towards higher memory addresses) to occupy the unused memory space.
- **BSS :** The BSS (Block Started by Symbol) segment, also known as uninitialized data, contains all the global variables and static variables that are initialized to zero or do not have explicit initialization in the program. For example, if you define 'static int a', the value of *a* would be set to 0 and would be placed in the BSS.
- **Data :** The Data segment contains any global or static variables which have a pre-defined value. For example, if you define 'int a = 10', the value of *a* would be set to *10* and would be placed in the Data segment.

Scanned with CamScanner

- **Text :** Text segment, also known as the Code segment, contains executable instructions of the loaded program. It is read-only to avoid any changes during runtime. Any attempt to write in this segment will cause the OS kernel to kill the process with a segmentation fault error.

Now that you understand the virtual address space and how programs are laid out in the memory, let's define what a buffer overflow is.

Definition : *Buffer overflow is a condition in which the input information crosses the boundary limits or capacity of any segment in the virtual address space thus overwriting information in other segments.*

Adversaries (attackers) exploit such a condition to crash a system or to insert specially crafted code that allows them to gain control of the system. So, for example, an attacker can fill the stack or the heap area so much so that it exhausts all free memory on the system and crash it.

A real-life example could be a water glass that has say a limited capacity of 250 ml. if you keep pouring water in it, the water begins to spill over after its capacity is full. The spilled water could dirty the surface on which the glass is kept or may even damage the surface.

Let's take an example to understand it better.

Assume the following code snippet written in C language.

```
char A[8];
int B=10;
strcpy(A,"EVILINPUT");
```

A is a character array that can hold up to 8 characters. It is followed by an integer *B* that is initialized with the value of 10. Assume that both are adjacent (side by side) in the memory. So, a typical memory layout would be as following.

A								B
								10

Now, the variable *A* is assigned the value of "EVILINPUT", which is 9 characters, without checking the length of the input. What happens?

A								B
E	V	I	L	I	N	P	U	T

This assignment overwrites the value of *B*. The variable *A* should have been restricted to the limit or the boundary that is assigned to it. But that does not happen (especially in the low-level languages like C and C++). The attacker can take advantage of such erroneous coding practices and can potentially exploit the system.

As per National Vulnerability Database (NVD), following is the number of reported vulnerabilities against buffer overflow.

https://nvd.nist.gov/vuln/search/statistics?form_type=Basic&results_type=statistics&query=buffer+overflow&queryType=phrase&search_type=all

Scanned with CamScanner

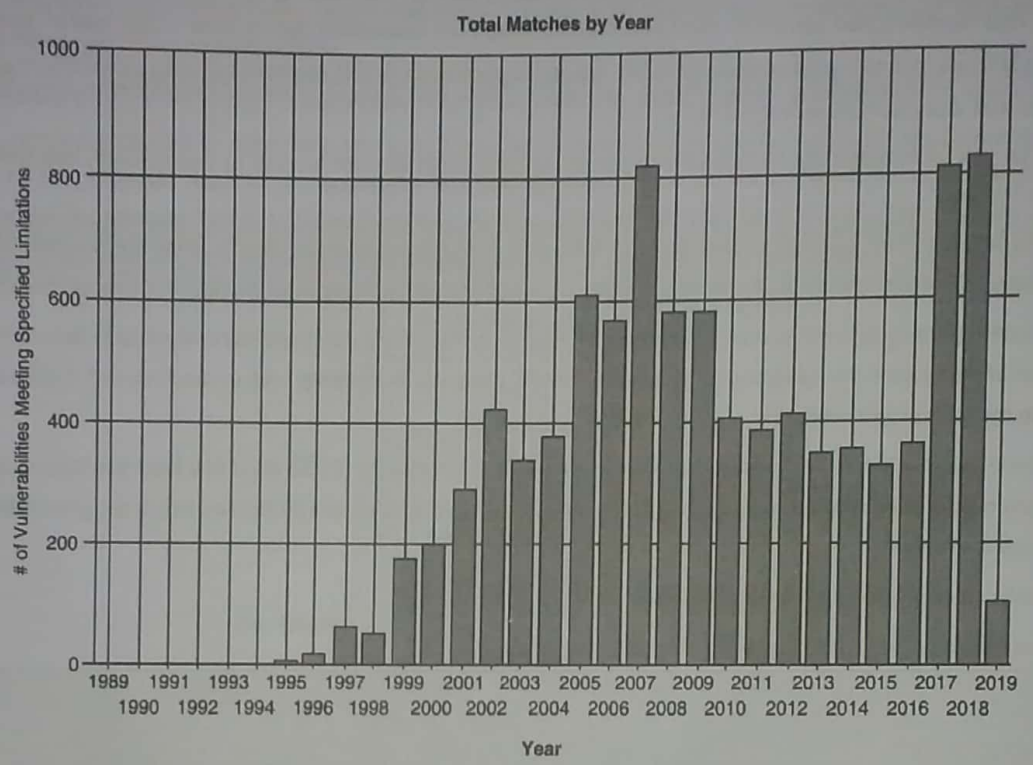


Fig. 6.2.2 : The number of reported vulnerabilities against buffer overflow

Note :	Don't get confused by the year 2019. I am taking this snapshot in Mar 2019. The rest of the year is still to pass-by. I am sure that there would be more buffer overflow vulnerabilities discovered in the later months of the year.

Buffer overflow is one of the earliest types of vulnerabilities (first reported in 1988 as part of the Morris Worm) found in software and is still found very frequently even in the latest applications such as Google Chrome as you learnt earlier.

6.2.1 Causes of Buffer Overflow and Suggested Protection Mechanism

There can be several causes that can lead to buffer overflows. Let's understand them.

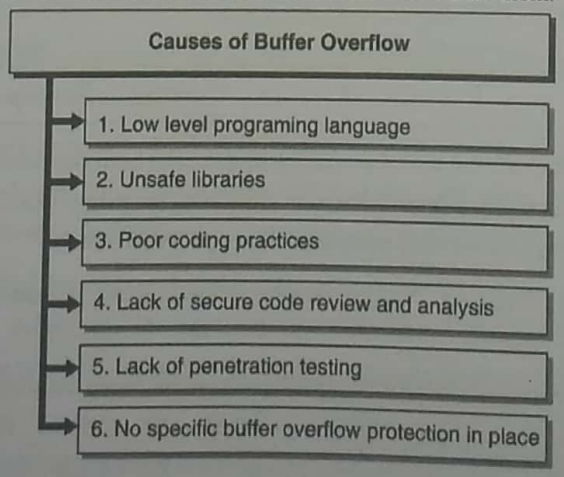


Fig. 6.2.3 : Causes of Buffer Overflow

Scanned with CamScanner

6.2.1(A) Low-level Programming Languages

There is an old saying that with great power comes great responsibility. Low-level languages such as C and C++ give a lot of power to the developers to handle the executional aspects of their programs such as memory allocation and garbage collection. Also, these languages were developed at times when the programming community was just using assembly language, and these were considered to be very close to assembly language in terms of performance. Also, those times software attacks weren't considered such prevalent (widespread) enough.

Unlike the modern programming languages, C and C++ do not check for overflows themselves. If your program does not check for the overflow conditions, the CPU will raise no errors or warnings and would allow overflow to happen. Also, in C and C++, arrays are compile-time entities only. At execution time, there are only pointers. Hence, there is no method to check any out of bound access at the program runtime.

Hence, until and unless, you are writing system level applications, it is recommended to choose modern programming languages.

Language/Environment	Direct Memory Access	Safety from buffer overflow
Java	Not allowed	Safe
.NET	Not allowed	Safe
Perl	Not allowed	Safe
Python	Not allowed	Safe
Ruby	Not allowed	Safe
C	Allowed	Unsafe
C++	Allowed	Unsafe
Assembly	Allowed	Unsafe

These languages do not allow buffer overflows. If attempts are made to exceed the defined boundaries, these raise exceptions (runtime errors) and terminate the program. So, an attacker might not be able to proceed further and carry out attacks.

6.2.1(B) Unsafe Libraries

In programming terms, a library is a collection of pre-written programs, routines, system calls, functions, classes, metadata, etc. that you can use when you are writing your programs. You don't have to write code yourself to carryout primitive (basic) tasks such as taking an input from the user or printing output on screen.

As you understand, these libraries are compiled from various other programmers like yourself. These libraries might not have sufficient checks for buffer overflow protection. Use of such libraries can make your program vulnerable to buffer overflow as well. Wherever possible, you should look for alternative libraries that have built-in buffer overflow protection.

Following is the sample list of C functions not having buffer overflow protection and their respective safer alternatives. You should be extremely cautious when using the unsafe functions and prefer using safer alternatives.

Scanned with CamScanner

Unsafe function	Safer alternative
gets()	fgets()
scanf()	scanf_s()
strcpy()	strlcpy()
strcat()	strlcat()
sprintf()	snprintf()
vsprintf()	vsnprintf()

Your applications written in C should use the secure methods provided by the Safe C Runtime library, also known as Safe CRT. Also, C++ applications should use the Standard Template Library classes, often called STL, which automatically detect overflow conditions and throw an exception when such a condition occurs.

6.2.1(C) Poor Coding Practices

If you are programming in a language like C, that gives you direct interaction with the system memory, you need to be more careful. Memory allocation, memory deallocation, pointer address, return address, buffer size, buffer operations, etc. are something that you need to do yourself without getting much assistance from the underlying programming language or the system. Let's take an example.

Assume the following code snippet written in C language.

```
char A[8];
int B=10;
strcpy(A,"EVILINPUT");
```

What is a better alternative?

```
char A[8];
int B=10;
strlcpy(A,"EVILINPUT", sizeof(A));
```

You replaced *strcpy()* with *strlcpy()* that allows you to control precisely that irrespective of what size of input is given to the variable *A*, only the input up to the capacity of the variable *A* would be considered. This is much better when compared to the previous code when it comes to protecting from buffer overflow vulnerability. You must do all sorts of input validation:

- Length checks
- Boundary checks
- Data type checks

As a programmer, you need to exercise discipline when writing code specially with languages that give you a lot of power and system level direct interaction capabilities.

6.2.1(D) Lack of Secure Code Review and Analysis

You review your responses to examination questions and rectify any oversight issues or mistakes, isn't it? Similarly, as an industrial practice, code is subject to review and analysis. Your peer (colleague) can review what you wrote and can

Scanned with CamScanner

point out any issues that could be potentially dangerous and should be fixed before putting code into practice. Also, there are a lot of tools that can automate the process of testing your code and identify potential vulnerabilities.

Some of them are,

Software	Language(s)
.NET Security Guard	.NET, C#, VB.net
Agnitio	ASP, ASP.NET, C#, Java, JavaScript, Perl, PHP, Python, Ruby, VB.NET, XML
Brakeman	Ruby, Rails
Checkmarx Static Code Analysis	Multiple
CodeSec	C, C++, C#, Java, JavaScript, PHP, Kotlin, Lua, Scala, TypeScript, Android
CodeSonar	C, C++, Java
Coverity	Multiple
DevBug	PHP
Find Security Bugs	Java, Scala, Groovy
FindBugs	Java
FlawFinder	C, C++
Fortify	Multiple
Google CodeSearch Diggity	Multiple
Klocwork	C, C++, C#, Java
Microsoft FxCop	.NET
Microsoft PREFast	C, C++
ParaSoft	C, C++, Java, .NET
phpcs-security-audit	PHP
PMD	Multiple
Polyspace Static Analysis	C, C++, Ada
Puma Scan	.NET, C#
Puma Scan Professional	.NET, C#
PVS-Studio	C, C++, C#
RIPS	PHP
RIPS NextGen	PHP
SonarCloud	Multiple
Splint	C
Veracode	Multiple
VisualCodeGrepper	C/C++, C#, VB, PHP, Java, PL/SQL

Scanned with CamScanner

These tools help you to find potential vulnerabilities in your code quickly without you having to go through your code line by line. Usually these tools are used in combination with the manual code review. So, if you are writing some code and putting it directly into practice, it is not recommended. Ensure that it has been reviewed manually as well as scanned by a tool of your choice before you actually put it to use for everyone else. You should cover the following in your code review and analysis.

- Arrays (For example, int a[10])
- Format strings (For example, printf(), fprintf(), %x, %s, %n, %d, %u, %c, %f)
- Functions that are known to be vulnerable to buffer overflow (For example, gets())

6.2.1(E) Lack of Penetration Testing

Recall from the last chapter that the goal of penetration testing is to identify and fix the vulnerabilities before any threat can exploit those vulnerabilities. It involves thinking like an attacker and conducting tests to find out vulnerabilities in the system. Quite a few programs are never penetration tested. Consequently, the attacker might discover such vulnerabilities and can potentially exploit the system. So, before you put your code into practice, it is important to carry out tests to discover any buffer overflow vulnerabilities and fix them. There is a wide variety of tools and methods that can conduct penetration testing to discover buffer overflow vulnerability.

6.2.1(F) No Specific Buffer Overflow Protection in Place

As you know, buffer overflow is quite an old and known vulnerability. There have been several protection mechanisms developed since then to protect the system and applications from exploits. It is highly recommended to have these protection mechanisms in place. Let's learn about them.

6.2.2 Specific Protection Mechanisms for Buffer Overflow

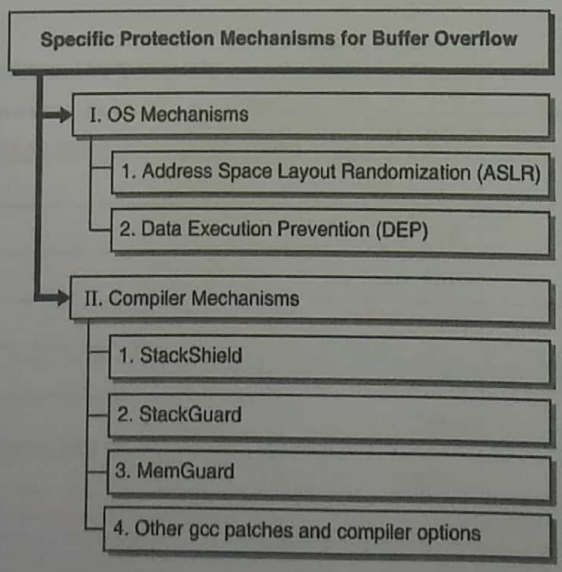


Fig. 6.2.4 : Specific Protection Mechanisms for Buffer Overflow

Scanned with CamScanner

6.2.2(A) Operating System Based Buffer Overflow Protection Mechanisms

There are certain protection mechanisms that can be applied at the operating system level to avoid buffer overflow attacks. These mechanisms help to secure memory access and provide protection. These mechanisms are independent from the program that you are executing and are applied for all programs globally. Let's take a look at them.

1. Address Space Layout Randomization (ASLR)

To exploit the buffer overflow vulnerability, the attacker must guess the memory address at which the program and its parts are located.

> ✍ ***Definition :*** *Address Space Layout Randomization (ASLR) is a memory protection technique that tries to prevent an attacker from guessing the memory addresses by randomizing the memory address layout of various program components each time the program runs.*

The program is loaded at a different base address in memory each time it is executed. Also, the addresses for memory segments like the heap and the stack are randomized. This makes it harder for attackers to guess the correct address and carry out exploits on buffer overflow.

For example, carefully look at the following snapshot. The program WinRAR is loaded at different base address on each invocation. In the first instance, the base address starts with 00DE whereas in the second instance, the base address starts with 0137. Such a randomization makes it hard for the attacker to guess the memory locations of programs and the related objects.

First invocation

00DE0000	00001000	WinRAR		PE header
00DE1000	0010E000	WinRAR	.text	code
00EEF000	0001A000	WinRAR	.rdata	imports
00F09000	000A1000	WinRAR	.data	data
00FAA000	00001000	WinRAR	.gfids	
00FAB000	00001000	WinRAR	.tls	
00FAC000	00037000	WinRAR	.rsrc	resources
00FE3000	0000D000	WinRAR	.reloc	relocations

Second invocation

01370000	00001000	WinRAR		PE header
01371000	0010E000	WinRAR	.text	code
0147F000	0001A000	WinRAR	.rdata	imports
01499000	000A1000	WinRAR	.data	data
0153A000	00001000	WinRAR	.gfids	
0153B000	00001000	WinRAR	.tls	
0153C000	00037000	WinRAR	.rsrc	resources
01573000	0000D000	WinRAR	.reloc	relocations

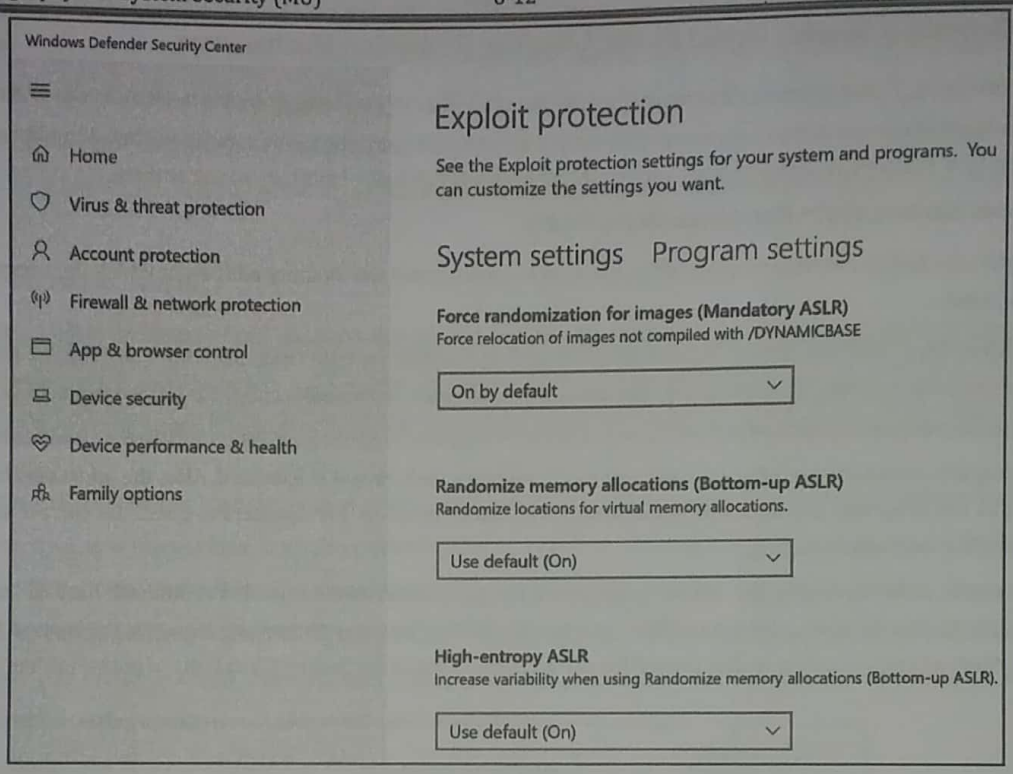
ASLR is set at the operating system level and the programs supporting it can randomize their memory addresses on each execution.

On Linux, you can enable ASLR from command line as following.

```
# sysctl -w kernel.randomize_va_space=2→ Enable ASLR
# sysctl -w kernel.randomize_va_space=0→ Disable ASLR
```

On Windows®, you can enable ASLR from Windows® Security Settings.

Scanned with CamScanner



2. Data Execution Prevention (DEP)

Definition : *Data Execution Prevention (DEP) is a set of hardware and software technologies that perform additional checks on memory to help prevent malicious code from running on a system.*

Note here that DEP is a general feature that prevents the misuse of memory pages to run unauthorised code. It is not specific to buffer overflow but can also be used for protection from buffer overflow and hence listed here.

When using DEP, any program is allowed to run only from the certain authorised memory locations. If an attacker attempts to execute the program from an unauthorised memory location, such an attempt is denied.

For example, if the attacker tries to execute code from stack or heap memory segments, such an attempt is denied. Stack and heap are not meant for code execution but to temporarily store data.

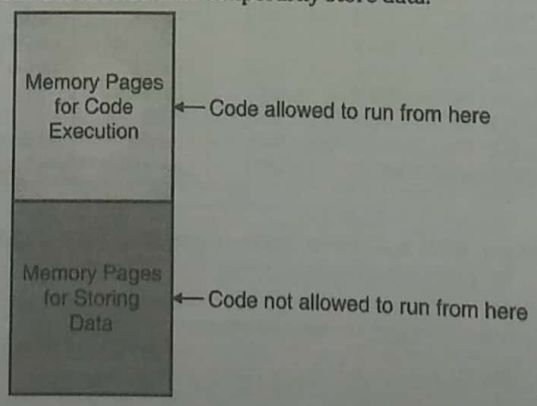


Fig. 6.2.5 : Memory pages marked as Data only

Scanned with CamScanner

DEP can be implemented using,

- The processor technology (hardware)
  - The no-execute page-protection (NX) processor feature as defined by AMD processors.
  - The Execute Disable Bit (XD) feature as defined by Intel processors.
- Software technology

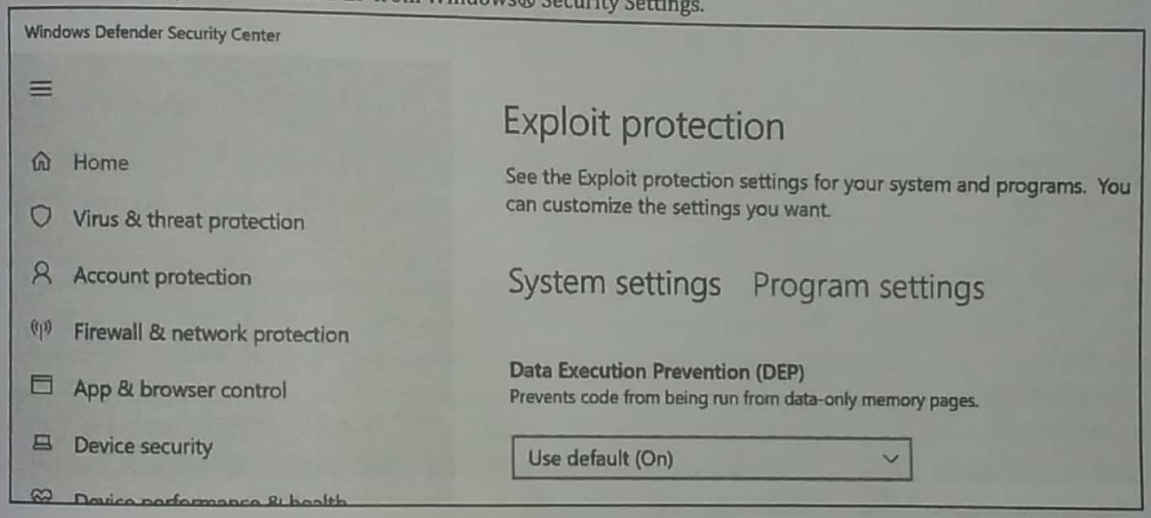
Irrespective of what technology is used, the memory pages are mapped and marked for code execution or data only.

DEP is set at the operating system level.

On Linux, you can enable DEP from command line as following.

```
# sysctl -w kernel.exec-shield=1  →Enable DEP
# sysctl -w kernel.exec-shield=0 →Disable DEP
```

On Windows®, you can enable DEP from Windows® Security Settings.



Advantages of OS-based buffer overflow protection mechanisms

1. The OS-based mechanisms do not require any program level changes.
2. OS takes the complete control over how memory is protected seamlessly.

Disadvantages of OS-based buffer overflow protection mechanisms

1. These mechanisms are architecture and platform specific and sometimes may not work equally across different operating systems.
2. It requires compatible processors that can enforce protection mechanisms as required by the operating system.
3. There could be system-wide performance impact.

6.2.2(B) Compiler Based Buffer Overflow Protection Mechanisms

The GNU Compiler Collection (GCC) is the compiler for C, C++ and several other languages. Overtime it has been updated with several compile time options (arguments and flags) using which you can adequately protect against buffer overflow. Your program needs to be compiled with the right compiler options to make use of such compile time buffer overflow protection mechanism.

(Copyright No. - L82548/2019)

Scanned with CamScanner

Functions are a fundamental entity in programming languages. The assembler implementation of them has specific processor instructions. The two major instructions with respect to functions are the CALL (used for invoking the function) and the RET (used for going to the return address so that the rest of the program can continue to execute after the function completes its tasks).

A GCC compiled program, in order to call a function, pushes all the function parameters in the stack and issues the CALL instruction followed by the function address. When the function has to return to its caller it issues the RET instruction.

Before you learn about how various compiler-based protection mechanisms work, let's understand the memory layout of the stack with respect to the program execution process.

Consider a sample program.

```
char *func(char *msg)
{
        int var1;
        char buf[80];
        int var2;
        strcpy(buf,msg);
        return msg;
}
int main(int argv, char **argc)
{
        char *p;
        p = func(argc[1]);
        exit(0);
}
```

This program when executed is laid out on the stack as following :

func()	var2	4 bytes
func()	buf	80 bytes
func()	var1	4 bytes
func()	saved frame pointer	4 bytes
func()	return address	4 bytes
func()	func()'s arguments	4 bytes
main()	P	4 bytes
main()	saved frame pointer	4 bytes
main()	return address	4 bytes
main()	main()'s arguments	12 bytes

In the buffer overflow attack, the attacker tries to manipulate this stack layout such that to overwrite the return address of the function to her desired location so that her manipulated code can run and exploit the system.

(Copyright No. - L82548/2019)

Scanned with CamScanner

func()	var2	4 bytes
func()	Overwrite to the desired address of the manipulated code	80 bytes
func()		4 bytes
func()		4 bytes
func()		4 bytes
func()	func()'s arguments	4 bytes
main()	P	4 bytes
main()	saved frame pointer	4 bytes
main()	return address	4 bytes
main()	main()'s arguments	12 bytes

Now that you understand how buffer overflow vulnerability is typically exploited, let's learn about the compiler-based protection mechanisms.

1. StackShield

Definition : *StackShield is a compiler-based buffer overflow exploit protection technique.*

It works by copying the return address of the function to an area that cannot be overflown (for example, data segment).

When the function is invoked, its return address is copied to a safe region in the memory. When function is about to return, its return address from the stack is compared with the previously cloned return address (from safe memory region) when the function was invoked. If the two addresses match, it means that the return address on the stack was not modified (by buffer overflow) and the function successfully returns to the return address. If the two addresses don't match, the program terminates and does not let the attacker exploit the buffer overflow she created. It does not let the function return to the manipulated return address thus avoiding the exploit from buffer overflow.

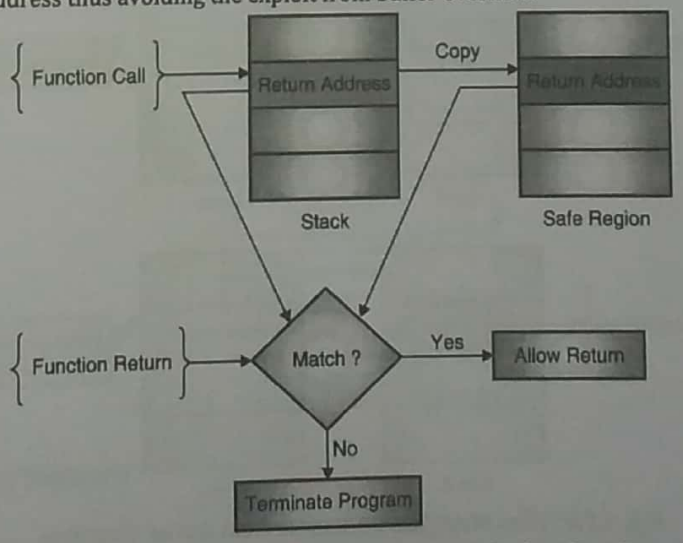


Fig. 6.2.6 : How StackShield protects from Buffer Overflow

2. StackGuard

Definition : *StackGuard is a compiler-based buffer overflow exploit protection technique.*

Scanned with CamScanner

It works by inserting a canary (random value) next to the return address. If the buffer overflow exploit is carried out, it would need to overwrite the canary value. Before jumping to the return address, the canary value written during the function call is checked. If it is not modified, the return is allowed else the program is terminated.

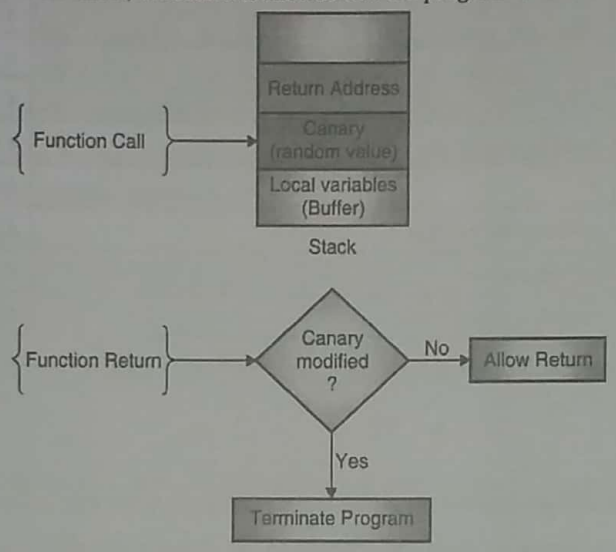


Fig. 6.2.7 : How StackGuard protects from Buffer Overflow

3. MemGuard

Definition : *MemGuard is a compiler-based buffer overflow exploit protection technique.*

It works by marking the return address as non-writable during the lifetime of the function. Any write attempt to this protected region during the function's active lifetime results in exception. The program is terminated, and the buffer overflow exploit is avoided.

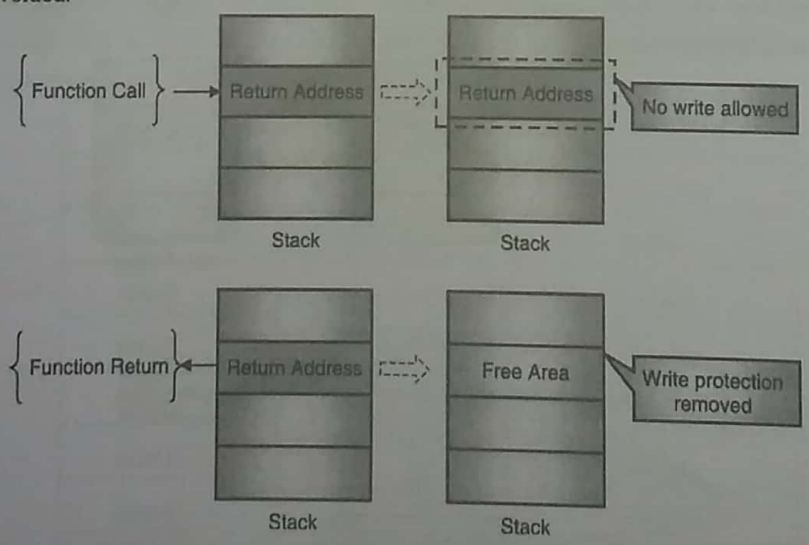


Fig. 6.2.8 : How MemGuard protects from Buffer Overflow

4. Other gcc patches and compiler options

Overtime there have been several additions to the gcc compiler options (flags and arguments to compile with) for safeguarding against the buffer overflow technique. They are listed here for your reference.

Scanned with CamScanner

- -fstack-protector-strong argument
- /GS switch for Microsoft Visual Studio
- -qstackprotect – IBM compiler

Advantages of Compiler-based buffer overflow protection mechanisms

1. Since the protection mechanism is directly applied at the program level, there is no system-wide performance impact.
2. It does not depend upon the architecture or processor capabilities.
3. It does not require program level changes. Just compiling it with the right compiler options provides protection.

Disadvantages of Compiler-based buffer overflow protection mechanisms

1. You require source code to compile it with the right compiler options. If you do not have access to the source code, protection cannot be provided.
2. Each program and library need to be compiled with the right compiler options to provide overall protection from buffer overflow.

6.3 Types of Buffer Overflow

There could be several types of buffer overflow attacks that could be potentially carried out. They are classified as the following based on the area of memory they exploit or based on the type of exploit.

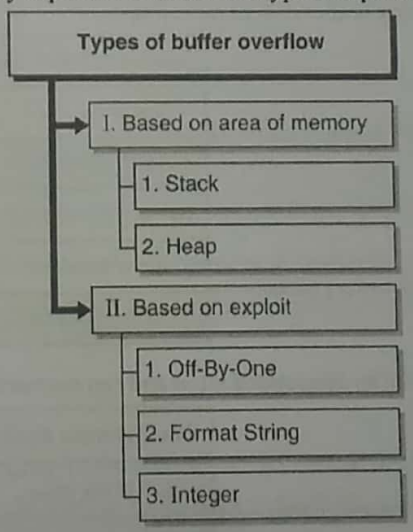


Fig. 6.3.1 : Types of Buffer Overflow

6.3.1 Based on Area of Memory

6.3.1(A) Stack Overflow

Stack overflow is the most common type of buffer overflow technique. You have learnt about it in the previous sections on how the stack can be manipulated by overwriting the return address and returning the functional pointer to an address from where the attacker can execute the malicious code. This is also called stack smashing.

Scanned with CamScanner

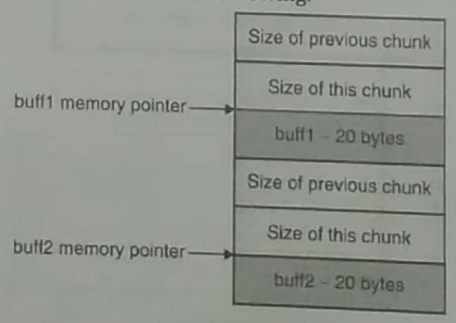
6.3.1(B) Heap Overflow

A heap overflow or heap overrun exploits the heap area. As you know, the memory on the heap area is dynamically allocated by your program at runtime. For example, when you use *malloc* or the *C++ new* operator, the memory is assigned from the heap area. It typically saves the program data and its size is not known previously.

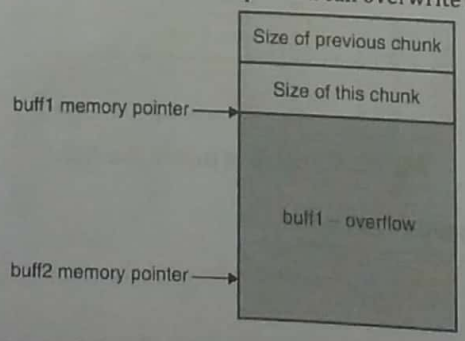
Heap overflow is carried out by corrupting the data in such a way so as to cause overwrites in the other areas of the heap. The program could crash, or the data could be generally corrupted. Let's see an example.

```
int main(void)
{
  char *buff1, *buff2;
  buff1 = malloc(20);
  buff2 = malloc(20);
  gets(buff1);
  free(buff1);
  exit(0);
}
```

For the given program, the heap memory is allocated as following.



In the example, the *gets(buff1)* can take an unbounded input and can overwrite *buff2* space in the heap as well.



Then, the function *free(buff1)* frees not only *buff1* space but also *buff2* space.

Scanned with CamScanner

Size of previous chunk
Size of this chunk
Free Space
Free Space
Free Space
Free Space

Now, if you try calling *buff2*, it might crash the program or make it work in an unexpected way.

6.3.2 Based on Exploit

6.3.2(A) Off-By-One

This type of overflow is usually caused due to the programmer's oversight. The array element definition usually starts from 0 and while addressing the elements in loops, you might refer to an array element which is outside the definition.

Let's take an example.

```
void func(char *str)
{
  char buff[128];
  int i;

  for (i=0;i<=128;i++)
    buff[i] = str[i];
}
int main(int argc, char **argv)
{
func(argv[1]);
}
```

If you look at the example code carefully, you see <= operator in the *for* loop. The array definition is for 128 elements but the *for loop* starting from element 0 and going until 128 would make it 129 elements. This would cause overflow by one byte.

6.3.2(B) Format String

Languages such as C provide formatting parameters that can be effectively used to process variables as per the required formatting. Some of these parameters are as mentioned in the Table 6.3.1.

Table 6.3.1 : Format String Parameters

Parameters	Output	Passed as
%p	External representation of a pointer to void	Reference
%x	Read data from stack (hexadecimal)	Value
%s	Read character strings	Reference
%n	Writes the number of characters into a pointer	Reference

Scanned with CamScanner

In the Format String exploit, the attacker passes the specialized formatted input values such as to read the stack, write to the stack or cause a system crash. Let's see an example.

```
#include <stdio.h>
void main(void)
{
  char a[10];
  gets(a);
printf(a);
}
```

1. Crashing the program

The attacker may provide input as "%s%s%s%s%s".

- For each %s, *printf()* will fetch a number from the stack, treat this number as an address, and print out the memory contents pointed by this address as a string, until a NULL character is encountered.
- If the number fetched by *printf()* is not an address, the memory pointed by this number might not exist and the program will crash.

```
%s%s%s%s%s
Segmentation fault
```

2. Viewing the stack

The attacker may provide input as "%x %x %x %x %x"

For each %x, *printf()* will display the stack content in the hexadecimal format.

```
%x %x %x %x %x
bcf81011 bcd5d9f0 fbad2288 bcf81012 0
```

3. Writing to the stack

Let's take a different example.

```
#include <stdio.h>
void main(void)
{
  int test=5;
  char buff[10];
  gets(buff);
printf("Original value of test:%d\n",test);
printf(buff,&test);
printf("\nNew value of test:%d\n",test);
}
```

In this example, the statement *printf(buff,&test)* can overwrite the value of *test*. Let's see a malicious input.

Assume the attacker enters "iamahacker%n" as input for *buff*. The *%n* format parameter counts the number of characters before it and writes the count of characters to the memory location of *test*. The value of *test* is thus modified, and the attacker has successfully written to the stack exploiting the *%n* format parameter.

Scanned with CamScanner

```
iamahacker%n
Original value of test:5
Iamahacker
New value of test:10
```

6.3.2(C) Integer

The integer overflow can occur when you take integer inputs but their processing results into larger values than the variable can hold.

Let's see an example.

```
#include <stdio.h>

void main(void)

{

  unsigned short i = 60000;

  unsigned short j = 50000;

  unsigned short result = i * j;

printf ("%d",result);

}
```

Unsigned short data type can hold 2 bytes up to the numbers 0 to 65,535. In the given example, both the variables are under the limit of what unsigned short can hold but the result of their multiplication is too large to be accommodated in an unsigned short variable. This results in memory fault and incorrect result.

```
24064

...Program finished with exit code 5
```

6.4 SQL Injection (SQLi)

Q. Explain briefly with example, how the SQL injection attack. **MU - Dec. 17, 3 Marks**

✍ ***Definition :*** *SQL Injection (SQLi) is a vulnerability exploiting which an attacker can inject malicious SQL commands and cause the database server to execute them.*

Using SQLi, the attacker can,

- Read sensitive and unauthorised data from the database,
- Modify data,
- Execute administrative commands on the database,
- And can also Issue commands to the operating system running the database,

Note here that the presence of SQLi attack does not mean that the SQL databases are inherently vulnerable or exploitable directly. This exploit only works if the application developer has programmed the application in such a way that the attack is possible. SQL databases just behave as they should. So, even if the attack is called SQLi, the fix for this exploit is not in the SQL databases. The fix is required in the application developer's code as we will see later on.

(Copyright No. - L82548/2019)

Scanned with CamScanner

SQLi is proven to be one of the most dangerous and impactful exploits on database-based applications. It has maintained the first position in the OWASP's list of Top Ten Most Critical Application Security Risks from several years.

https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

OWASP Top 10 - 2013	→	OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	∪	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	∪	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

6.4.1 How does SQLi Work?

Q. With the help of examples explain non malicious programming errors. **MU - Dec. 15, 5 Marks**

There could be several non-malicious programming errors such as buffer overflow, SQL Injection, Cross-site scripting etc. that can be exploited by attacker. Let's take an example of SQL Injection (SQLi).

SQLi can potentially be exploited where the user input is directly used to form SQL database queries. Instead of providing the expected input, the attacker can provide malicious input such that the original and desired SQL query is changed. The attacker can control the behaviour of the SQL queries and can manipulate what SQL queries are run.

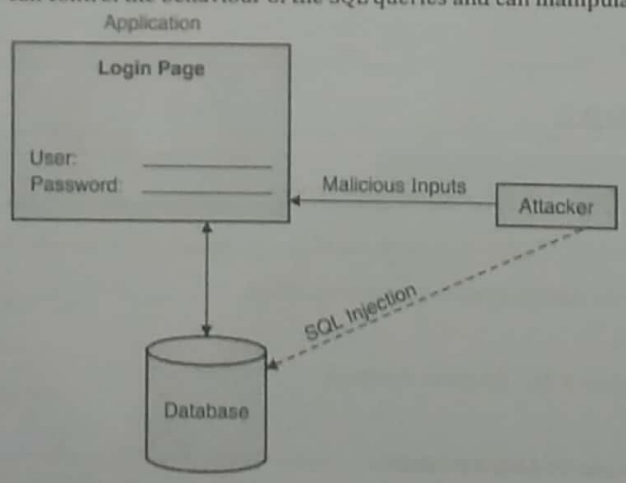


Fig. 6.4.1 : SQL Injection

Let's see an example.

```
query = "SELECT * FROM users WHERE name = ' + userName + ';"
```


Scanned with CamScanner

This query takes *userName* from the user. If the user passes a valid input such as Joe as the *userName*, the query statement is correctly formed as following.

```
query = "SELECT * FROM users WHERE name = ' + Joe + ';"
query = "SELECT * FROM users WHERE name = 'Joe';"
```

But what if the user is malicious and passes *'OR '1'='1* as input?

```
query = "SELECT * FROM users WHERE name = ' + ' OR '1'='1 + ';"
query = "SELECT * FROM users WHERE name = '' OR '1'='1';"
```

So, what happened here?

The *name* variable is searched for blank (' ') and then there is an *OR* statement which will always evaluate to True. ('1'='1'). So, overall, the SQL query formed will always run and print the entire *users* table!

So, the intention of the developer was to get the user name from the user and populate that value to form the dynamic query which can then show that particular user's details. But the malicious input provided by the user resulted in forming a query that would print the entire table and disclose the details of all the users contained in the database.

Let's consider some other malicious inputs.

```
' OR '1'='1 --
' OR '1'='1 {
' OR '1'='1 /*
```

In SQL, you can provide comments using -- (double hyphen), { } (braces) or /*...*/ (C-style multi-line comments). An attacker can pass injection statements followed by comments. Then the rest of the query is not executed, and the attacker can only execute the part of the query she wishes to execute.

Let's take the following query as an example.

```
query = "SELECT * FROM users WHERE
            username = ' + username + '
            AND password = ' + password + ';"
```

In this example, the system is trying to authenticate the user by taking the user name and the corresponding password. So, if the user name is valid and the password entered is also valid, the query will return the result else it will not return any results.

Now, an attacker can pass the input as *' OR '1'='1';"* -- for *username* and any random value for *password*.

```
query = "SELECT * FROM users WHERE
            username = '' OR '1'='1';" -- '
            AND password = ' + password + ';"
```

Such a malicious input comments out the password check in the SQL query and returns True. The user is successfully authenticated bypassing providing any user name or password.

The attacker can also add more SQL queries as malicious input. For example, consider the following query.

```
query = "SELECT * FROM users WHERE
            username = '' OR '1'='1'; DROP table users;" -- '
            AND password = ' + password + ';"
```


Scanned with CamScanner

The malicious input of *' OR '1'='1'; DROP table users;"* -- can now not only authenticate the user but also can delete the table containing the user information.

6.4.2 Types of SQLi

There are several variants of SQLi.

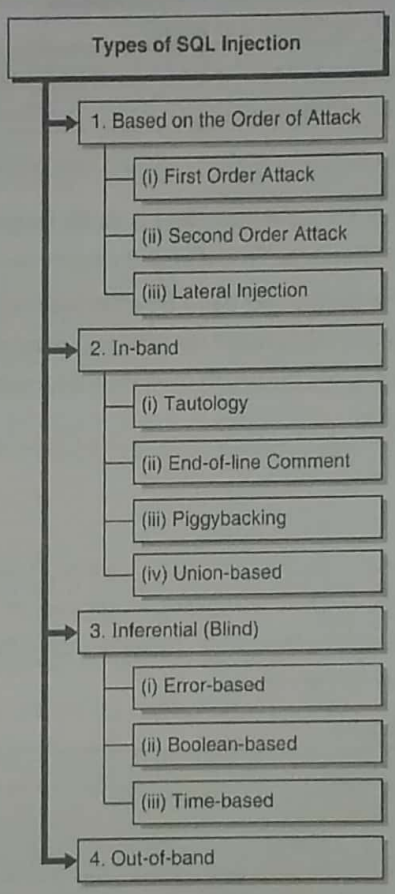


Fig. 6.4.2 : Types of SQL injection

6.4.2(A) Based on the Order of Attack

1. First order attack

✍ ***Definition :*** *In the first order attack, the attacker enters a malicious input and causes the SQLi attack immediately.*

The injected malicious SQL query is run immediately, and the attackers gets the results. The SQLi examples that you have learnt in the previous section are all first order attacks. A single query is manipulated, and the results are immediately obtained.

2. Second order attack

✍ ***Definition :*** *In the second order attack, the attacker first injects the malicious query (or data) into the database and then gets that malicious query (or use the malicious data) subsequently executed by another activity directly from the database.*

Scanned with CamScanner

The result of second order SQLi attack is not obtainable immediately. There might be some time elapsed between the time at which the malicious query (or the malicious data) is injected and the time when the malicious query is actually executed (or malicious data is actually used), and the result of the attack is seen.

Let's see an example.

```
INSERT INTO users(username, password) VALUES("admin--", "random");
UPDATE users SET password = '1' WHERE username = admin--
```

Usually there are two SQLi attacks done.

First, the attacker manages to insert a new user into the database (using the first order attack that you learnt earlier or by just following the process of creating a new account on the system). Note here that the user name added is specifically having -- at the end of it, but since it is passed inside " ", the SQL query will accept that as the value and insert it into the database without complaining.

In the second statement, the attacker wishes to reset her password for *admin--* account she created. She passes *admin--* as the user name whose password needs to be reset. The database executes the query but due to -- at the end of *admin--*, it actually resets the password of the original *admin user*. Now, since the password of the original admin user is reset, the attacker can login as the admin on the system.

So, as you learnt in the example, there were two parts to the SQLi. First a value was stored in the database and then that stored value was exploited to carry out the SQLi.

3. Lateral Injection

Definition : *In Lateral Injection, the attacker modifies the system functions to inject malicious query.*

Using Lateral SQL Injection, an attacker can exploit stored procedure (SQL functions). For example, the function *TO_CHAR()* can be manipulated by changing the environment variables NLS_Date_Formator NLS_Numeric_Characters. You can include any text in the format model and can exploit SQLi.

6.4.2(B) In-band

1. Tautology

Tautology means a statement that is always true. For example, "a beginner who has just started" is a tautology.

Definition : *Tautology is a form of SQLi where conditional statements are injected that always evaluate to true.*

Following is a simple example of a tautology.

```
SELECT * FROM users WHERE username="or 1=1 --
```

The use of 1 = 1 -- makes the entire *WHERE* condition as a tautology (always evaluating to true).

2. End-of-line Comment

Definition : *End-of-line comment is a form of SQLi where comments are used to terminate the SQL query prematurely (inappropriately) to run only a part of the query.*

Following is a simple example.

```
SELECT * FROM users WHERE username='admin'-- AND password = adminpasswd
```

The use of -- terminates the query prematurely and the query thus returns the entire details of the *admin user* without requiring the *admin* password.

Scanned with CamScanner

3. Piggybacking

Piggybacking means to ride on someone's shoulder.

Definition : *Piggybacking is a form of SQLi where additional queries are injected to run with the original query.*

Following is a simple example.

```
SELECT * FROM users WHERE username = 'admin' OR '1'='1'; DROP table users;
```

The use of ; after the first query allows to add the second query. When the first query completes, the database server runs the second query followed by the ; (*DROP table users*). The attacker can add as many queries as she wishes separated by ; .

4. Union-based

The UNION operator is used to combine the result-set of two or more SELECT statements.

Definition : *Union-based is a form of SQLi where the UNION operator is used to inject the second query to the original query.*

Following is a simple example.

```
SELECT username FROM users where username = ' ' OR '1'='1'
UNION SELECT empname FROM employees
```

There are two tables *users* and *employees*, and both have a field containing names. The attacker can use the *UNION* operator to reveal the employee names from the *employees* table.

6.4.2(C) Inferential (blind)

1. Error-based

Definition : *Error-based is a form of SQLi where the attacker causes errors to learn about the environment.*

For gathering information about the environment such as database type, version, OS, table names, etc. the attacker might cause system errors by injecting random queries. If those errors are not sufficiently handled in the application, then those errors would be visible in the application. The attacker can use the gathered information to launch more sophisticated and known attacks.

Following is a simple example.

```
SELECT * FROM users WHERE username= ' ' AND pin=convert(int,(select top 1 name from sysobjects where xtype='u'))
```

Here the attacker purposely creates an error by trying to convert name into integer. *sysobjects* is a system table that stores all the table names present on the database server. The attacker tries to check the table names starting from first entry *(top 1)* in the *sysobjects* table. The attacker is only interested in the tables created by the user *(xtype='u')*.

The query when executed gives the following error.

```
Microsoft OLE DB Provider for SQL Server (0x80040E07)
Error converting nvarchar value 'CreditCards' to a column of data type int
```

The attacker gets to know two things from this error :

- The database server running behind the application is Microsoft® SQL Server®
- There is a table by name CreditCards present on the database server.

The attacker can then use this information to launch further attacks.

Scanned with CamScanner

2. Boolean-based

Definition : *Boolean-based is a form of SQLi where the attacker injects true/false statements to learn about the environment.*

Following is a simple example.

Suppose the attacker injects the condition such that it always evaluates to true.

```
SELECT * FROM users WHERE username='' or 1=1 -- AND password = mypasswd
```

Suppose the response from the application is *"Invalid Password"*.

The attacker can infer (guess) two things from it:

- Either her malicious input was detected and rejected.
- Or the check for password is a separate query (because -- would have commented out that password check).

Now, suppose the attacker changes the condition such that it always evaluates to false.

```
SELECT * FROM users WHERE username='' or 1=0 -- AND password = mypasswd
```

Suppose the response from the application is *"Invalid Username and Password"*.

The attacker learns that the response is different from the previous one when she inserted a true condition. This would conclude that the field *username* is injectable and could be exploited.

3. Time-based

Definition : *Time-based is a form of SQLi where the attacker injects random queries and evaluates the time it takes to run the query and get the result to learn about the environment.*

Time-based SQLi injects the SQL query to the database such that it forces the database to wait for a specified amount of time (in seconds) before responding. The response time indicates to the attacker whether the result of the query is true or false.

Following is a simple example.

```
SELECT accounts FROM users WHERE login='abc' and
ASCII(SUBSTRING((select top 1 name from sysobjects),1,1)) > X WAITFOR 5 -- ' AND
pass='' AND pin=0
```

In this example, the attacker waits for 5 seconds if the ASCII code of the first character from the first table name in *sysobjects* is greater than the ASCII code of the character X. If the ASCII code is greater, then there would be a 5 second delay and if the ASCII code is not greater, then there would not be a 5 second delay.

6.4.2(D) Out-of-band

Definition : *Out-of-band is a form of SQLi where the attacker injects the SQL query via one medium and receives the results via another medium.*

It is not very common because it depends on the features enabled on the database server such as the ability to send emails or notifications. Out-of-band techniques offer the attacker alternatives for evaluating attack results.

Scanned with CamScanner

6.4.3 Preventing SQLi

Some of the ways in which SQLi can be prevented are as following.

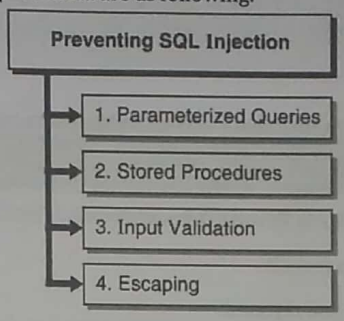


Fig. 6.4.3 : Preventing SQL Injection

1. Parameterized Queries

The primary cause of SQLi exploit is that the user input is directly used to form the SQL query.

✍ ***Definition :*** *In parameterized queries, the user input is separated from the SQL query such that the input is taken literally and not as part of the SQL query.*

Let's see an example.

Assume a SQL query that is vulnerable to SQLi.

```
SELECT * FROM users WHERE username= ''' + name + '''
```

Here the variable *name* is part of the query. This allows the attacker to provide malicious input for the *name* variable and that malicious input becomes the part of the query and the SQLi gets exploited.

Instead of writing the query this way, parameterized queries separate the input variable from the actual query.

```
paraName = getRequestString(name);
paraQuery = "SELECT * FROM users WHERE username=@0";
db.Execute(paraQuery, paraName);
```

The database server then takes the variable input literally instead of part of the SQL query. So, even if the attacker provides a malicious input, the database server would not execute it. The *@0* provides the argument number that should be taken as input for executing the query in the *db.Execute* function call.

Note : The example given previously is as per ASP.NET language. Each language has its own way of constructing parameterized queries. For example, following is an example of creating parameterized query in Java.

```
String custname = request.getParameter("customerName");
String query = "SELECT account_balance FROM user_data WHERE user_name = ? ";
PreparedStatement pstmt = connection.prepareStatement( query );
pstmt.setString( 1, custname);
ResultSet results = pstmt.executeQuery( );
```

2. Stored Procedures

✍ ***Definition :*** *A stored procedure is like a function or a subroutine that you can save in your database and call and execute multiple times.*

Scanned with CamScanner

For example, following is a stored procedure that does not require a parameter.

```
CREATE PROCEDURE ShowAllStudents
AS
SELECT * FROM students
GO;

EXEC ShowAllStudents;
```

Following is a stored procedure that takes one parameter.

```
CREATE PROCEDURE ShowAllStudents @Class nvarchar(30)
AS
SELECT * FROM students WHERE Class = @Class
GO;

EXEC ShowAllStudents Class = "BE";
```

Following is a stored procedure that takes multiple parameters.

```
CREATE PROCEDURE ShowAllStudents @Class nvarchar(30), @Dept nvarchar(30)
AS
SELECT * FROM students WHERE Class = @Class AND Dept = @Dept
GO;
EXEC ShowAllStudents Class = "BE", Dept = "Computer";
```

The idea behind using stored procedures is to separate out the query from the input data. It works very similar to the parameterized queries and can be effectively used to prevent SQLi.

3. Input Validation

SQLi exploit is possible because the user input is not adequately validated for correctness. As a developer, you should evaluate the user input before accepting or processing it. Lookout for things like,

- Length of the input
- Format of the input
- Any special characters that you don't believe could be a legitimate value

4. Escaping

As you learnt in XSS, you can escape (replace) user input with the respective codes. In this technique, all the user input is escaped before putting it in a query. It varies from database to database in its implementation. When you escape user input, it is treated literally, and the database server does not treat it as part of SQL query even if the attacker desires so.

For example, following are some escape sequences for MySQL database server.

Escape Sequence	Character Represented by Sequence
\0	An ASCII NUL (X'00') character
\'	A single quote (') character
\"	A double quote (") character
\b	A backspace character
\n	A newline (linefeed) character

Scanned with CamScanner

Escape Sequence	Character Represented by Sequence
\r	A carriage return character
\t	A tab character
\Z	ASCII 26 (Control+Z); see note following the table
\\	A backslash (\) character
\%	A % character; see note following the table
_	A _ character; see note following the table

This is an old technique and should not be used quite often. It can be complex and error-prone.

6.5 Malware (or Malicious Code)

Malware (or Malicious code) refers to,

✍ ***Definition :*** *The category of programs that infect the information systems and are motivated to gain unauthorised access to the system, information contained therein or to fulfil any other malicious objectives.*

These are usually spread through emails, social media, malicious websites, documents and other attachments. You should be very careful while clicking on a link or opening an email from a stranger or downloading an interesting attachment. These all could be mechanisms to infect your system.

6.5.1 Concept Building-Activities related to Malware

Before you understand various types of malware, let's understand various activities associated with malware. A malware may exhibit some of these activities or all of them.

1. **Installation :** How the malware reaches the system, e.g. from attachment.
2. **Detection and removal :** How the malware's presence can be detected, e.g. anti-virus.
3. **Payload :** Actual function that the malware performs, e.g. delete system files.
4. **Trigger :** Event that invokes (or activates) the malware, e.g. clicking a file.
5. **Replication :** Capability of the malware to further copy itself and infect other systems.
6. **Eradication :** The malware might remove itself after delivering the payload.

6.5.2 Types of Malware

Let's learn about some of the common forms of *malware (or malicious code).*

Scanned with CamScanner

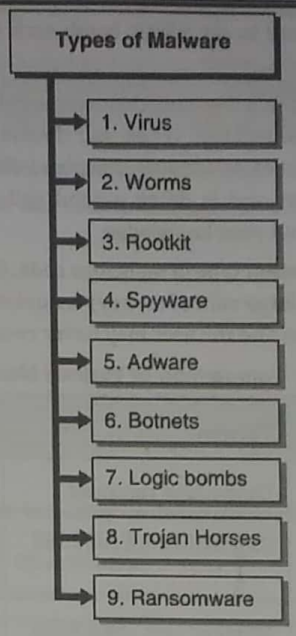


Fig. 6.5.1 : Types of Malware

1. **Virus :** Virus is a category of malicious code that cannot self-replicate itself to deliver the payload. It requires a trigger (user action) to activate and then deliver the payload (harm the system in a particular way). It could deliver various payloads such as deleting system files, displaying specific messages or stealing information. ILOVEYOU and Melissa are two well-known viruses.
2. **Worms :** Worms are smaller malicious programs that can self-replicate and trigger without needing a user action. They are targeted towards infecting the systems and delivering the required payload of their own. One of the most common examples of worm is Stuxnet which was used to damage Iran's uranium infrastructure.
3. **Rootkit :** Once the attacker has access to the system, she can upload a set of tools that can remain on the system for her future access and further exploiting the system by becoming administrator (elevation of privilege attack). Such set of tools are called rootkits. Rootkits allow access without the attacker having to redo the attack to gain access to the system.
4. **Spyware :** Spyware is a category of malicious programs which are secretly downloaded (say when you are browsing a website) and installed on the system for the purpose of gaining access to the sensitive information such as passwords. These can be further used to take control over your system and install other malicious programs.
5. **Adware :** Adware is a special category of malicious programs that are intended to serve advertisements through banners, pop-ups or screen messages. They do not perform a malicious activity *per se but* could be very annoying. The purpose of adware is to drive sales through continuous ads.
6. **Botnets :** Bot is a short word for 'robot'. Botnets are a huge network of malicious programs (bots) that do various tasks for its master. These are typically used in the web-based environments where the master uses these automatic programs to carry out malicious actions. For example, botnets can send thousands of spam emails to millions of people every day.
7. **Logic bombs :** Logic bombs are malicious programs that are automatically triggered when the desired logic is met. For example, an attacker can set the program to delete all files when it becomes 12.12.12.12.12.12 (12 Dec. 2012 at 12th hour, 12th Minute and 12th Second). Logic bombs can also be used like a time bomb.

Scanned with CamScanner

For example, the administrator may choose to set a logic bomb such that if she does not deactivate it before a time, automatically the system is infected.

8. **Trojan Horses :** These are malicious programs that hide themselves in legitimate (right) looking programs.
For example, a trojan might be called notepad.exe. When you double click it, it might call the original notepad.exe program but also do several other things such as sending sensitive information to the attacker's machine. You believe that the program you clicked is legitimate and is doing a good and desired action, but it is actually carrying out malicious activities behind the scene without your knowledge.

9. **Ransomware :** Ransomware is the most recent type of malicious code. In this, the attacker takes over the user's system and locks it out and demands a ransom (a large sum of money) for unlocking it. If the user does not pay the money, the attacker usually erases the complete system and the user may never recover the system and data that was on it.

Table 6.5.1 : Comparison of Various Malicious Code

Type of malware	Purpose	Trigger	Replication	Countermeasure
Virus	Deliver payload	User action required	User action required	Anti-virus
Worms	Deliver payload	User action not required	User action not required	Anti-virus
Rootkits	Future access	Attacker uses these tools	Not applicable	Anti-virus
Spyware	Gather sensitive information	User action not required	Not applicable	Anti-malware
Adware	Serve advertisements	User action not required	Not applicable	Anti-malware
Botnets	Deliver payload	By botnet master	In thousands to form a network	Anti-malware
Logic bombs	Deliver payload	User action not required	Not applicable	Separation of duties
Trojan Horse	Deliver payload	User action not required	Not applicable	Anti-virus
Ransomware	Extort money	By the attacker	Not applicable	Backup of data

Now that you have an overall idea of malware, let's dive deeper into some of them.

6.5.2(A) Virus and Worms

Q. Write a brief on - Viruses and their types. **MU - Dec. 15, 5 Marks**

Q. What are the different types of viruses and worms? How do they propagate? **MU - Dec. 16, 10 Marks**

What is a virus?

Definition : *Virus is a category of malicious code that cannot self-replicate itself to deliver the payload.*

It requires a trigger (user action) to activate and then deliver the payload (harm the system in a particular way). It could deliver various payloads such as deleting system files, displaying specific messages or stealing information. ILOVEYOU and Melissa are two well-known viruses.

Scanned with CamScanner

What is a worm?

Definition : *Worms are smaller malicious programs that can self-replicate and trigger without needing a user action.*

They are targeted towards infecting the systems and delivering the required payload of their own. One of the most common examples of worm is Stuxnet which was used to damage Iran's uranium infrastructure.

Types of virus

Viruses are usually classified based on :

1. How they live in memory
2. How they spread
3. How they hide
4. What they deliver

Let us understand the classification first. We will then detail them out.

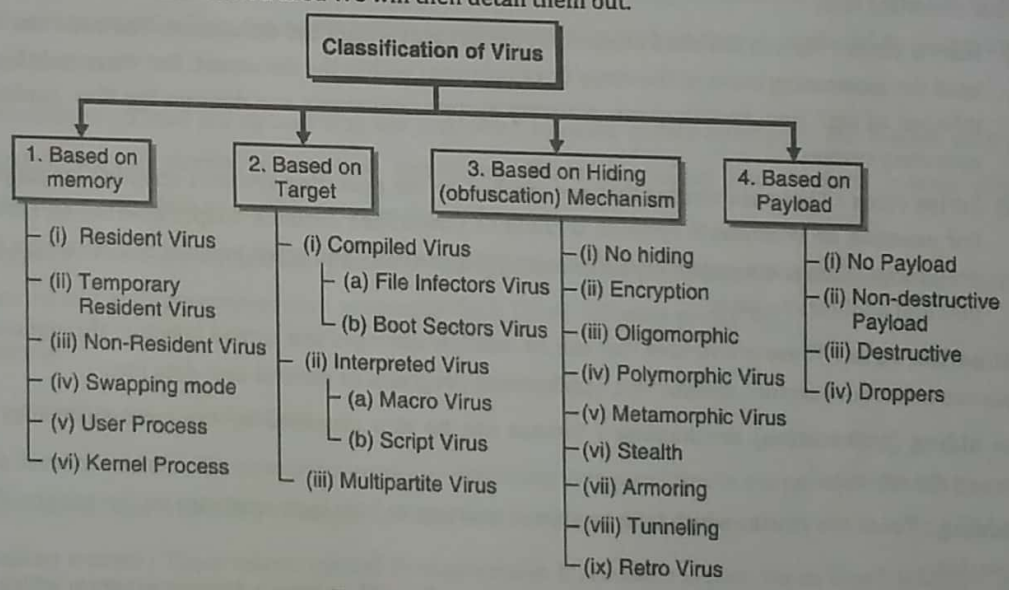


Fig. 6.5.2 : Classification of Virus

1. **Based on memory :** This classification depends upon how the virus stays within a system.

   (a) **Resident virus :** These types of viruses stay in memory and infect any file or program that is opened.

   (b) **Temporary resident virus :** These viruses stay in memory for some time and then vacate the memory based on time or after some event (say infecting 5 files).

   (c) **Non-resident virus :** These viruses do not stay in memory. These infect the files by copying their payloads to the files directly. The relevant files are searched and then the payload is copied to them.

   (d) **Swapping mode :** These viruses use swap space for residing. These switch from swap space to memory, infect a few files and return back to the swap space.

   (e) **User process :** These viruses run with user level access and privileges. These can mostly infect only the user files without touching the system files (since for infecting system files, administrative level access is needed).

Scanned with CamScanner

(f) **Kernel process :** These viruses are very dangerous as they get installed at the system level (say via device drivers or through system applications). These viruses run with the full administrative access and privileges and can infect any file, program or application on the system.

2. **Based on target :** This classification depends upon what the virus choses to infect (the target of infection).

(a) **Compiled virus :** These are viruses that run as machine executable. These are directly run by the operating system. The virus' source code is compiled to form an executable file. These can be further classified into,

(i) **File Infector virus :** File infectors corrupt the specific types of files. These often manipulate the file headers such that when the file is opened, it actually places the file pointer to the malware code execution point.

(ii) **Boot sector virus :** These viruses infect the boot sectors on the hard disk. Boot sectors are key partitions that help to power on and bring up the system for operation.

(b) **Interpreted virus :** The source code of the interpreted viruses is run at the execution timeline by line. Unlike compiled viruses, the interpreted viruses are run by the legitimate application programs when invoked. These are further classified into,

(i) **Macro virus :** Macros are code blocks that are present within the documents. The code blocks are usually used for automating some of the steps or calculations within the document. But these code blocks could be infected as well such that they run arbitrary system commands and damage the files, applications or the operating system.

(ii) **Script virus :** These are viruses that are invoked by the operating system's scripting language interpreter. For example, on Windows® systems, it could be PowerShell or Batch scripts whereas on Linux it could be shell script. Scripts are code blocks that exist independently and when invoked could damage the system or files and applications on it.

(c) **Multipartite virus :** These are viruses that can do multiple damages and spread infection throughout the system. These can infect boot sectors, system files, applications, programs or general user data files.

3. **Based on hiding (obfuscation) mechanism :** Viruses can be also classified by the mechanism they use to hide themselves on the system.

(a) **No hiding :** These are viruses which take no special measure to hide their existence on the system. Viruses rarely follow this.

(b) **Encryption :** These viruses use encryption to hide themselves. They have a decrypt program which exposes the virus when it's time to spread the infection on the system. Once the infection task is carried out, the virus again encrypts itself to hide.

(c) **Oligomorphic :** These are also called semi-polymorphic. Basically, these viruses use several decryption routines to hide itself as much as possible. With each infection, it changes the decryption routines to avoid getting detected by the anti-virus tools.

(d) **Polymorphic :** These viruses have some sort of mutation (changing) engine that changes the appearance of the viruses with each infection to make their detection very difficult. These make several changes to the encryption settings and also changes the decryption code accordingly.

(e) **Metamorphic :** These viruses change the virus body instead of appearance. It is done by using unneeded functions or commands or changing the order of virus code.

(f) **Stealth :** These viruses try to hide the infection by restoring the older properties of the file (time of modification, file size, etc.) after infection.

Scanned with CamScanner

(g) **Armoring :** An armoring virus is a virus that makes analysis very difficult. These viruses use various techniques to hide themselves from detection and removal.

(h) **Tunnelling :** These viruses attach themselves to the system interrupts. Whenever there is an interrupt call, first the virus executes itself and then passes the control to the original interrupt.

(i) **Retro virus :** These viruses are specifically designed to bypass any security tools such as firewall, anti-virus or intrusion detection systems. The virus developers understand to a great extent how such security programs work and make it increasingly hard for these programs to detect viruses.

4. **Based on payload :** Based upon what the viruses deliver or do to the target system, they can be classified as under.

   (a) **No payload :** These viruses do not affect systems as such or infect files. These are generally productivity killers that irritate by slowing down the system or creating minor nuisances.

   (b) **Non-destructive payload :** These viruses as well are non-harming in nature except that they pop up repeated messages or graphic. They can control some hardware devices such as mouse, keyboard or cdrom and make them to malfunction.

   (c) **Destructives :** These are viruses that are motivated to cause severe damage to the system such that they are damaged beyond normal operational conditions. These can affect boot sectors, system files, programs, applications, user data or anything else on the system.

   (d) **Droppers :** These are viruses that just sit on the system and let the attacker use the system's resources to launch other attacks and cybercrimes such as identity theft, DDoS, software piracy, etc.

Types of worms

There are majorly two types of worms.

1. **Network Service worms :** These worms spread by exploiting vulnerabilities in the network service associated with an OS or application. Once it infects a system, it spreads to other networked systems.
2. **Mass mailing worms :** These worms spread through emails. It reaches a system via an email, infects it and then looks for other email addresses in the system to send a copy of itself to other users.

Table 6.5.2 : Comparison between Virus and Worm

Comparison Parameter	Virus	Worms
Trigger	User action required	User action not required
Replication	User action required	User action not required
Spread	Mostly remains on the infected system	Spreads throughout the network
Countermeasures	Antivirus	Antivirus, Firewall
Detection Complexity	High	Medium
Damage Caused	Usually high	Usually medium

Scanned with CamScanner

6.5.3 General Guidelines for Preventing Malware

Some of the most common and general guidelines for preventing any form of malware are as following.

1. Regularly scan your systems using an adequate anti-malware program
2. Do not open emails and attachments from unknown sources
3. Control the urge to click on every URL that you see
4. Restrict visiting sites that support software piracy, promise free videos, music, books, etc.
5. Do not install programs from unknown or non-reputed sources
6. Regularly update your OS and installed applications to fix any known vulnerabilities
7. Do not plug external devices such as a pen drive from unknown sources
8. Limit physical access to the system

Review Questions

Here are a few review questions to help you gauge your understanding of this chapter. Try to attempt these questions and ensure that you can recall the points mentioned in the chapter.

[A] Software Vulnerabilities

Q. 1 With a simple code example, explain buffer overflow. **(6 Marks)**

Q. 2 You are reviewing code written by a colleague. She has used gets() in her code. What do you suggest and why? **(4 Marks)**

Q. 3 What can cause buffer overflow? **(7 Marks)**

Q. 4 Write a short note on ASLR. **(4 Marks)**

Q. 5 How Data Execution Prevention (DEP) can be used to provide protection from buffer overflow? **(5 Marks)**

Q. 6 Classify buffer overflow protection techniques. **(6 Marks)**

Q. 7 Julius is interested in buffer overflow protection. He comes to you asking about the advantages and disadvantages of OS-based and compiler-based protection mechanisms. What do you tell him? **(6 Marks)**

Q. 8 What is a canary? How and where it is used? **(6 Marks)**

Q. 9 Draw a block diagram of StackShield and explain. **(6 Marks)**

Q. 10 Explain any three types of buffer overflow. **(6 Marks)**

Q. 11 Write a short note on Format String vulnerability. **(4 Marks)**

Q. 12 Compare the types of XSS. **(6 Marks)**

Q. 13 Persisted XSS can exploit any visitor. Comment. **(6 Marks)**

Scanned with CamScanner

Q. 14 Alex is worried about XSS. You tell him to relax and follow some guidelines on preventing XSS. He asks you which guidelines you would recommend. What do you tell him? **(6 Marks)**

Q. 15 How SQLi works? **(6 Marks)**

Q. 16 Classify SQLi. **(8 Marks)**

Q. 17 Write a short note on tautology. **(4 Marks)**

Q. 18 Database programmers are joining your company today. You are sent to explain company guidelines on preventing SQLi. What do you explain? **(7 Marks)**

[B] Malware

Q. 19 Compare various types of malware on the basis of Purpose, Trigger, Replication and suitable Countermeasure. **(6 Marks)**

Q. 20 Draw the virus classification diagram (no explanation required). **(6 Marks)**

Q. 21 Provide a comparison between a virus and a worm. **(8 Marks)**

Q. 22 Write a short note on Trojans (Trojan Horses). **(4 Marks)**

Q. 23 Write a short note on Logic Bombs. **(4 Marks)**

Q. 24 What are the components of bots? **(6 Marks)**

Q. 25 Describe various botnet architectures. **(8 Marks)**

Q. 26 Explain the attacks carried out by botnets. **(8 Marks)**

Q. 27 What is a rootkit? Explain its types. **(8 Marks)**

Q. 28 What are the ways of detecting rootkits? **(6 Marks)**

Q. 29 Ron wants to ensure that everyone in the company is aware of malware protection guidelines. You are required to make a presentation covering the malware protection guidelines. Which guidelines would you include in your presentation? **(4 Marks)**

❑❑❑

Scanned with CamScanner