BHARATI VIDYAPEETH DEEMED UNIVERSITY
COLLEGE OF ENGINEERING, PUNE - 43

DEPARTMENT OF COMPUTER ENGINEERING

Lab Manual

Computer & Information Security

B.Tech. Computer Sem V

(2021 course)

DEPARTMENT OF COMPUTER ENGINEERING

VISION OF THE INSTITUTE

"To be World Class Institute for Social Transformation through Dynamic Education"

MISSION OF THE INSTITUTE

- To provide quality technical education with advanced equipment, qualified faculty members, infrastructure to meet needs of profession and society.
- To provide an environment conductive to innovation, creativity, research and entrepreneurial leadership.
- To practice and promote professional ethics, transparency and accountability for social community, economic and environmental conditions.

VISION OF THE DEPARTMENT

"To pursue and excel in the endeavour for creating globally recognized computer engineers through quality education".

MISSION OF THE DEPARTMENT

1. To impart engineering knowledge and skills confirming to a dynamic curriculum.

2.To develop professional, entrepreneurial & research competencies encompassing continuous intellectual growth.

3. To produce qualified graduates exhibiting societaland ethical responsibilities in working environment.

PROGRAM EDUCATIONAL OBJECTIVES

Graduates upon completion of B. Tech Computer Engineering Programme will able to**:**

1.Demonstrate technical and professional competencies by applying engineering fundamentals, computing principles and technologies.

2.Learn, Practice, and grow as skilled professionals/ entrepreneur/researchers adapting to the evolving computinglandscape.

3.Demonstrate professional attitude, ethics, understanding of social context and interpersonal skills leading to a successful career.

PROGRAM SPECIFIC OUTCOMES:

PSO 1: To design, develop and implement computer programs onhardware towards solving problems.

PSO 2: To employ expertise and ethical practise through continuing intellectual growth and adapting to the working environment.

PROGRAMME OUTCOMES

1. Apply the knowledge of mathematics, science, engineering fundamentals, and computing for the solution of complex engineering problems.
2. Identify, formulate, research literature, and analyse complex engineering problems reaching substantiated conclusions using computer engineering foundations, principles, and technologies.
3. Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for public health and safety, and cultural, societal, and environmental considerations.
4. Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues, and the consequent responsibilities relevant to the professional engineering practice.
7. Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and the need for sustainable development.
8. Apply ethical principles while committed to professional responsibilities and norms of the engineering practice.
9. Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings

10. Communicate effectively on complex engineering activities with the engineering community and with the society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. Apply the engineering and management principles to one's work, as a member and leader in a team.
12. Recognise the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

GENRAL INSTUCTIONS:

- Equipment in the lab is meant for the use of students. Students need to maintain a proper decorum in the computer lab. Students must use the equipment with care.
- Students are required to carry their reference materials, files and records with completed assignment while entering the lab.
- Students are supposed to occupy the systems allotted to them and are not supposed to talk or make noise in the lab.
- All the students should perform the given assignment individually.
- Lab can be used in free time/lunch hours by the students who need to use the systems should take prior permission from the lab in-charge.
- All the Students are instructed to carry their identity cards when entering the lab.
- Lab files need to be submitted on or before date of submission.
- Students are not supposed to use pen drives, compact drives or any other storage devices in the lab.
- For Laboratory related updates and assignments students should refer to the notice board in the Lab.

Course Name:- *Computer & Information Security*

HoursPerWeek:2 Hrs.

Course Outcomes: On completion of the course, students will have the ability to:

1. Explain the basics of network security.
2. Compare different techniques of cryptography
3. Discuss details of key and certificate management
4. Discuss details about system security
5. Recite Network and Transport Layer security
6. Apply knowledge of network security and cryptography in real life

HOW OUTCOMES ARE ASSESSED?

Course Outcome	Assignment Number	Level	Proficiency evaluated by
Explain the basics of network security	1,2,3	3,3,3	Study and anlaysis
Compare different techniques of cryptography	3,4,5,6,7	3,3,3,3,3	Study and anlaysis
Discuss details of key and certificate management	3,4,7	3,3,3	Performing experiments and reporting results
Discuss details about system security	7,8,9,10	3,3,3,3	Performing experiments and reporting results
Recite Network and Transport Layer security	8	3	Performing experiments and reporting results
Apply knowledge of network security and cryptography in real life	8,9,10	2,3,3	Performing experiments and reporting results

CO-PO mapping

CO Statements	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2
CO1														
CO2														
CO3														
CO4														
CO5														
CO6														

Guidelines for Student's Lab Journal

- The laboratory assignments are to be submitted by student in the form of journal. The Journal consists of prologue, Certificate, table of contents, and handwritten write-up of each assignment(Title,Objectives,ProblemStatement,Outcomes,software&Hardware requirements, Date of Completion, Assessment grade/marks and assessor's sign, Theory-Concept, algorithms, printouts of the code written using coding standards, sample test cases etc.
- Oral Examination will be based on the term work submitted by the student in the form of journal.
- Candidate is expected to know the theory involved in the experiment.
- The practical examination should be conducted if the journal of the candidate is completed in all respects and certified by concerned faculty and head of the department
- All the assignment mentioned in the syllabus must be conducted.

List of Assignments

No	Detail of Assignment
1.	Enlist and explain goals of security, types of Security attacks. Study and analysis of security vulnerabilities for E-commerce services(any case study)
2.	Introduction to Cryptography based Security Tools.
3.	Write a Program in C/Java to implement symmetric encryption.
4.	Write a Program in C/Java to implement asymmetric encryption.
5.	Introduction to GnuPG encryption system.
6.	Implementation of Decryption techniques using secret key in GnuPG.
7.	Study and implementation of Digital certificates using Java, Study of DNS certificates.
8.	Case study on Secure Socket Layer (SSL) and Transport Layer Security (TLS), Secure Electronic Transaction (SET).
9.	Case Study on use and implementation of Firewall.(any one)
10.	Case Study on Recent trends in IOT security, IDS, Cloud Security.

Course Co-ordinator **Head**

EXAMINATION SCHEME

Oral Exam: 25 Marks
Term Work: 25 Marks
Total: 50 Marks
Minimum Marks required: 10 Marks each head

PROCEDURE OF EVALUATION

Each practical/assignment shall be assessed continuously on the scale of 25 marks. The distribution of marks as follows.

Sr. No	Evaluation Criteria	Marks for each Criteria	Rubrics
1	**Timely Submission**	**07**	➢ Punctuality reflects the work ethics. Students should reflect that work ethics by completing the lab assignments and reports in a timely manner without being reminded or warned.
2	**Presentation**	**06**	➢ Student are expected to write the technical document (lab report) in their own words. The presentation of the contents in the lab report should be complete, unambiguous, clear, understandable. The report should document approach/algorithm/design and code with proper explanation.
3	**Understanding**	**12**	➢ Correctness and Robustness of the code is expected. The Learners should have an in-depth knowledge of the practical assignment performed.The learner should be able to explain methodology used for designing and developing the program/solution. He/she should clearly understand the purpose of the assignment and its outcome.

LABORATORY USAGE

Students use Linux/Unix on computers for executing the lab experiments, document the results and to prepare the technical documents for making the lab reports.

OBJECTIVES:-

i)To Explain basics of cryptography, how it has evolved, and some key encryption techniques.

ii)To learn security policies such as authentication, integrity and confidentiality.

PRACTICAL PRE-REQUISITE:-

1. Basic knowledge of computer network..

Practical Experiment 1:

AIM: Enlist and explain goals of security, types of Security attacks. Study and analysis of security vulnerabilities for E-commerce services(any case study)

The aim of this practical experiment is to understand the goals of security, explore various types of security attacks, and conduct a study and analysis of security vulnerabilities in E-commerce services using a relevant case study.

Goals of Security:

Security in the context of information technology and E-commerce services typically aims to achieve the following goals:

- Confidentiality: Ensuring that sensitive information is accessible only to authorized individuals or entities and remains protected from unauthorized access or disclosure.
- Integrity: Maintaining the accuracy and consistency of data, systems, and processes, preventing unauthorized modifications or alterations.
- Availability: Ensuring that systems, services, and data are consistently available and operational for legitimate users, minimizing downtime due to technical failures or attacks.
- Authentication: Verifying the identity of users, systems, or entities to ensure that only authorized individuals or processes gain access.
- Authorization: Granting appropriate access privileges to authorized users while preventing unauthorized access to resources.
- Non-repudiation: Ensuring that actions or transactions cannot be denied by the parties involved, preventing disputes by providing evidence of actions.

Types of Security Attacks:

A cyber-attack is an exploitation of computer systems and networks. It uses malicious code to alter computer code, logic or data and lead to cybercrimes, such as information and identity theft.

We are living in a digital era. Now a day, most of the people use computer and internet. Due to the dependency on digital things, the illegal computer activity is growing and changing like any type of crime.

Web-based attacks

These are the attacks which occur on a website or web applications. Some of the important web-based attacks are as follows-

1. Injection attacks

It is the attack in which some data will be injected into a web application to manipulate the application and fetch the required information.

Example- SQL Injection, code Injection, log Injection, XML Injection etc.

2. DNS Spoofing

DNS Spoofing is a type of computer security hacking. Whereby a data is introduced into a DNS resolver's cache causing the name server to return an incorrect IP address, diverting traffic to the attacker?s computer or any other computer. The DNS spoofing attacks can go on for a long period of time without being detected and can cause serious security issues.

3. Session Hijacking

It is a security attack on a user session over a protected network. Web applications create cookies to store the state and user sessions. By stealing the cookies, an attacker can have access to all of the user data.

4. Phishing

Phishing is a type of attack which attempts to steal sensitive information like user login credentials and credit card number. It occurs when an attacker is masquerading as a trustworthy entity in electronic communication.

5. Brute force

It is a type of attack which uses a trial and error method. This attack generates a large number of guesses and validates them to obtain actual data like user password and personal identification number. This attack may be used by criminals to crack encrypted data, or by security, analysts to test an organization's network security.

6. Denial of Service

It is an attack which meant to make a server or network resource unavailable to the users. It accomplishes this by flooding the target with traffic or sending it information that triggers a crash. It uses the single system and single internet connection to attack a server. It can be classified into the following-

Volume-based attacks- Its goal is to saturate the bandwidth of the attacked site, and is measured in bit per second.

Protocol attacks- It consumes actual server resources, and is measured in a packet.

Application layer attacks- Its goal is to crash the web server and is measured in request per second.

7. Dictionary attacks

This type of attack stored the list of a commonly used password and validated them to get original password.D

8. URL Interpretation

It is a type of attack where we can change the certain parts of a URL, and one can make a web server to deliver web pages for which he is not authorized to browse.

9. File Inclusion attacks

It is a type of attack that allows an attacker to access unauthorized or essential files which is available on the web server or to execute malicious files on the web server by making use of the include functionality.

10. Man in the middle attacks

It is a type of attack that allows an attacker to intercepts the connection between client and server and acts as a bridge between them. Due to this, an attacker will be able to read, insert and modify the data in the intercepted connection.

System-based attacks

These are the attacks which are intended to compromise a computer or a computer network. Some of the important system-based attacks are as follows-

1. Virus

It is a type of malicious software program that spread throughout the computer files without the knowledge of a user. It is a self-replicating malicious computer program that replicates by inserting copies of itself into other computer programs when executed. It can also execute instructions that cause harm to the system.

2. Worm

It is a type of malware whose primary function is to replicate itself to spread to uninfected computers. It works same as the computer virus. Worms often originate from email attachments that appear to be from trusted senders.

3. Trojan horse

It is a malicious program that occurs unexpected changes to computer setting and unusual activity, even when the computer should be idle. It misleads the user of its true intent. It appears to be a normal application but when opened/executed some malicious code will run in the background.

4. Backdoors

It is a method that bypasses the normal authentication process. A developer may create a backdoor so that an application or operating system can be accessed for troubleshooting or other purposes.

5. Bots

A bot (short for "robot") is an automated process that interacts with other network services. Some bots program run automatically, while others only execute commands when they receive specific input. Common examples of bots program are the crawler, chatroom bots, and malicious bots.

Case Study: Security Vulnerabilities in an E-commerce Website

For the case study, consider an E-commerce website that sells electronic gadgets. The following security vulnerabilities could be analyzed:

1. Financial Frauds or Payment Frauds

This type is one of the most typical for eCommerce and dates back to the very first attempts of the businesses going online. Often, scammers used to make unauthorized transactions and immediately wipe out the trails. Or else, they can use the fake emails, accounts, and names, and even IP addresses to look like the real customer.

After they have requested a refund with, for instance, a fake screenshot, most eCommerce platforms basically give them money for nothing, especially if they’re not aware of this financial trick.

With being reported in over 70% of all attacks, payment frauds are still one of the top reasons why companies experience huge cost losses.

How to Solve the Problem

Make sure your eCommerce platform cooperates only with verified and authoritative payment systems. Additionally, some companies make it possible to conduct a transaction only after logging in to the individual account before any purchase, which minimizes the risks of financial fraud and prevents common security vulnerabilities as well.

2. Spam Attacks

Though emails are considered to be the most powerful marketing channel for eCommerce, they

are also the typical web security vulnerabilities hackers can easily take advantage of.

The random comments left on the product pages, under your blog posts, or the contact forms can not only harm the customers' trust but also slow down your platform as well.

Needless to say, that one infected link left by a spammer is more than enough to affect your site's speed, provide access to personal customer information and other sensitive data.

Additionally, the spamming activity can become a serious threat to the customers' security as well, which can easily undermine your site's credibility.

How to Solve the Problem

Use anti-spamming software for security vulnerabilities detection and its successful removal. Such software can easily spot the infected URLs and safely remove them from your site so that no one can see them.

Typically, such software type uses various algorithms to filter the comments and detect the computer-generated links which can be potentially dangerous for your site's security, and even provide you with the details about the email of the actual sender if it is possible.

3. Triangulation Fraud

One of the most recent and large-scale security vulnerabilities detected nowadays is triangulation fraud. This stands for creating a fake site with an identical interface and products at a cheaper price.

After the customers complete the transaction, they basically donate the money to the criminals, as the products they wanted to purchase simply don't exist and never be shipped to them.

The reason why this type of fraud is harmful to your eCommerce platform is that you can lose your new clients, loyal customers, and their trust as well: no one wants to go back to the site (even with the slight differences in a brand's name or interface) after being cheated there at once.

How to Solve the Problem

Basically, no one can stop scammers from creating a platform that looks just like your online store. However, it's possible to prevent your customers from being fooled by simply informing them about this issue and pointing out the real domain of your eCommerce platform.

Even a simple information letter can in fact prevent your customers from money loss and also strengthen your store's authority as well.

4. Web Application Security Vulnerabilities

At present, the level of competition in different business areas makes companies do their best to meet all the customers' needs. For online stores, web applications are simply a must to attract more clients to their platform.

For instance, it's essential for eCommerce clients to create the wish lists of the products they want to buy next, look for the featured products, check the special offers and get the personalized list of products they are probably interested in.

The use of smartphones has only enhanced the demand for web app creation. However, having created one is still not that easy as to maintain and update it regularly.

Developers should not only focus on the customer experience and comfortable interface but also on the software security vulnerabilities as well. Having omitted this stage of web app development, you can easily give a green light to the criminals for the attack. They can easily play

with fake transactions and refunds, gift cards, and even the critical data of your eCommerce platform.

How to Solve the Problem

Find a dedicated team of developers you can trust, and ask them to conduct the app checking for the bugs, error codes, and other software issues that need to be fixed.

Additionally, professional app developers can try various scenarios of hacking, such as cookie poisoning, remote command execution, cross-site scripting, etc to check the specific software security vulnerability of your app and easily resolve it.

5. Bot Attack

Some criminals also attack eCommerce sites with bots, that basically act like real users and can hardly be detected by the security system.

This is why bot attack is considered to be one of the common security vulnerabilities you should always keep in mind. Usually, you can check the bot traffic in the site's analytics and get the records about the exact time and details of their behavior.

However, bots are not just fake users that can boost your traffic to slow down the site's speed. Instead, they can also steal the personal information of your customers, record their log-in credentials and bank information, manipulate the products' prices and randomly block them, thus making your eCommerce platform less secure and user-friendly.

How to Solve the Problem

To make sure your site is secure enough and won't go down during any of the hacking attempts, always introduce a CAPTCHA test for critical actions such as logging in or products' purchase.

In addition, track the traffic and block the one generated from the suspicious sources, analyze the failed log-in attempts and protect your mobile apps.

Large companies also consider employing bot migration software - the perfect solution for minimizing IT security vulnerabilities.

6. Brute Force Attacks

Brute-force attacks refer to the hacking method of guessing the system passwords. So far that's one of the most dangerous security vulnerability types that can attack your online store's panel and attempt to get full access to it.

During this attack, the various programs and complex algorithms are used to generate any possible combination to crack your site's password. After that, any scenario is possible: criminals can ask for the reward or steal the client's personal data, send spam offers, etc — all they planned to do since the site owner has lost access to the admin panel.

How to Solve the Problem

This attack can't be predicted but can be prevented instead. For minimizing the site's security vulnerability, developers recommend using strong, complex passwords and do not store them on your digital files, computer documents, browsers, etc.

In addition to that, you can protect the site by changing the password regularly (for instance, on a monthly basis or once a quarter).

Conclusion:

This practical experiment provides a hands-on opportunity to explore the goals of security, understand various types of security attacks, and apply this knowledge to analyze security vulnerabilities in a real-world E-commerce context. By conducting this analysis and proposing mitigation strategies, participants can develop a deeper understanding of information security in E-commerce services.

Practical Experiment 2:

AIM- Introduction to Cryptography based Security Tools.

Theory:-

The aim of this practical experiment is to introduce participants to the fundamental concepts of cryptography and explore various cryptography-based security tools commonly used to enhance information security.

Introduction to Cryptography:

Cryptography involves techniques for secure communication in the presence of adversaries. It employs mathematical algorithms to transform plaintext data into ciphertext, making it unreadable without the appropriate decryption key. Cryptography serves as a cornerstone of modern information security, providing confidentiality, integrity, authentication, and non-repudiation.

1. OpenSSL:

- OpenSSL is a widely used open-source library that provides cryptographic functions and protocols, including SSL/TLS for secure communication. It offers a range of cryptographic operations such as encryption, decryption, digital signatures, and hash functions.

2. GnuPG (GNU Privacy Guard):

- GnuPG is a free and open-source software implementing the OpenPGP (Pretty Good Privacy) standard. It is used for encrypting and decrypting data, as well as creating and verifying digital signatures.

3. TrueCrypt and VeraCrypt:

- TrueCrypt (now discontinued) and VeraCrypt are tools for creating encrypted virtual disks or containers. They provide on-the-fly encryption, allowing users to create secure storage areas that can only be accessed with the correct password or key.

4. Hashing Tools (e.g., md5sum, sha256sum):

- These command-line tools calculate and display hash values (checksums) of files using different hash algorithms. They are used to verify the integrity of files and detect changes or corruption.

5. Hashcat:

- Hashcat is a popular password cracking tool that utilizes the power of GPUs to perform brute-force and dictionary attacks on hashed passwords. It helps assess the strength of passwords and the security of hash functions.

6. Wireshark:

- Wireshark is a network protocol analyzer that can capture and display the data traveling back

and forth on a network. While not a cryptographic tool itself, it can be used to analyze encrypted traffic and identify potential security issues.

7. KeePass:

- KeePass is a password manager that securely stores passwords and other sensitive information in an encrypted database. It requires a master password or key file to unlock the database.

8. Tor (The Onion Router):

- Tor is a network of volunteer-operated servers that enhances privacy and security by routing internet traffic through a series of encrypted relays. It helps users anonymize their online activities.

9. OpenSSH:

- OpenSSH provides encrypted communication over a network, enabling secure remote login and file transfer. It uses public-key cryptography for authentication and secure communication.

10. Cryptography Libraries in Programming Languages:

- Many programming languages offer built-in or external cryptography libraries. For example, Python has the cryptography library, Java has the javax.crypto package, and C# has the System.Security.Cryptography namespace.

These tools cater to a wide range of cryptographic needs, from secure communication and data encryption to password management and vulnerability assessment. It's important to choose the right tool based on your specific requirements and the level of security you need to achieve.

Conclusion:

This practical experiment offers participants a comprehensive introduction to cryptography and its practical applications through various security tools. By gaining hands-on experience with encryption, digital signatures, hash functions, and SSL/TLS, participants will develop a solid foundation in cryptography-based information security, enabling them to appreciate its significance in modern digital communication and data protection.

Practical Experiment 3:

AIM:-Write a program in C/JAVA to implement Symmetric encryption.

THEORY:-

Symmetric encryption is a type of encryption where the same key is used for both the encryption and decryption processes. In other words, the sender and the recipient of the encrypted data share a common secret key that is used to both scramble (encrypt) and unscramble (decrypt) the information.

Here's a simplified overview of how symmetric encryption works:

Key Generation: A secret key is generated by the sender or a designated authority. This key must be kept confidential between the sender and the recipient.

Encryption: The sender uses the secret key to encrypt the plaintext (original) data into ciphertext (encrypted) data. This process involves applying a mathematical algorithm to the plaintext using the secret key, which transforms the data into an unreadable format.

Transmission: The encrypted data is transmitted over a communication channel to the recipient.

Decryption: The recipient uses the same secret key to decrypt the received ciphertext back into the original plaintext. This process involves applying the reverse of the encryption algorithm using the same secret key.

Key characteristics of symmetric encryption:

Efficiency: Symmetric encryption is generally faster and requires fewer computational resources compared to asymmetric encryption (public-key encryption).

Confidentiality: As long as the secret key remains secure, symmetric encryption provides strong confidentiality for the data being transmitted.

Key Management: The main challenge in symmetric encryption is the secure distribution and management of the secret key. If an unauthorized party gains access to the key, they can decrypt the data.

Scalability: Symmetric encryption becomes challenging to manage when there are multiple parties that need to communicate securely, as each pair of participants would require a unique secret key.

Examples of symmetric encryption algorithms include Advanced Encryption Standard (AES), Data Encryption Standard (DES), and Triple DES (3DES).

It's important to note that while symmetric encryption is efficient for securing data transmission, it does not provide a solution for key exchange or digital signatures, which are areas where asymmetric (public-key) encryption is commonly used in conjunction with symmetric encryption.

CODE IN C:-

```
#include <stdio.h>    // header file
#include <stdlib.h>   // header file
#include <string.h>   // header file
#define SIZE 30
```

```
// Conversion of characters
// of the string to lowercase
voidtoLowerCase(char plain[], intps)
{
inti;
for (i = 0; i<ps; i++) {
if (plain[i] > 64 && plain[i] < 91)
plain[i] += 32;
    }
}


// spaces is removed from the string
intremoveSpaces(char* plain, intps)
{
inti, count = 0;
for (i = 0; i<ps; i++)
if (plain[i] != ' ')
plain[count++] = plain[i];
plain[count] = '\0';
return count;
}

// here, 5x5 key square is generated
voidgenerateKeyTable(char key[], intks, char keyT[5][5])
{
inti, j, k, flag = 0, *dicty;

    // hashmap of 26 character that willl store the alphabet count
dicty = (int*)calloc(26, sizeof(int));

for (i = 0; i<ks; i++) {
if (key[i] != 'j')
dicty[key[i] - 97] = 2;
    }
```

```
dicty['j' - 97] = 1;

i = 0;
   j = 0;
for (k = 0; k <ks; k++) {
if (dicty[key[k] - 97] == 2) {
dicty[key[k] - 97] -= 1;
keyT[i][j] = key[k];
j++;
if (j == 5) {
i++;
              j = 0;
          }
       }
   }
for (k = 0; k < 26; k++) {
if (dicty[k] == 0) {
keyT[i][j] = (char)(k + 97);
j++;
if (j == 5) {
i++;
              j = 0;
          }
       }
   }
}

// characters and position of letters od digraph is searched from the key
void search(char keyT[5][5], char a, char b, intarr[])
{
inti, j;

if (a == 'j')
       a = 'i';
else if (b == 'j')
```

```
        b = 'i';

for (i = 0; i< 5; i++) {
for (j = 0; j < 5; j++) {
if (keyT[i][j] == a) {
arr[0] = i;
arr[1] = j;
            }
else if (keyT[i][j] == b) {
arr[2] = i;
arr[3] = j;
            }
        }
    }
}

// this function will find the modulus with 5
int mod5(int a)
{
if (a < 0)
        a += 5;
return (a % 5);
}

// this function calls the decrypt function
void decrypt(char str[], char keyT[5][5], intps)
{
inti, a[4];
for (i = 0; i<ps; i += 2) {
search(keyT, str[i], str[i + 1], a);
if (a[0] == a[2]) {
str[i] = keyT[a[0]][mod5(a[1] - 1)];
str[i + 1] = keyT[a[0]][mod5(a[3] - 1)];
        }
else if (a[1] == a[3]) {
```

```
str[i] = keyT[mod5(a[0] - 1)][a[1]];
str[i + 1] = keyT[mod5(a[2] - 1)][a[1]];
        }
else {
str[i] = keyT[a[0]][a[3]];
str[i + 1] = keyT[a[2]][a[1]];
        }
    }
}

// this function wil call the decrypt function
voiddecryptByPlayfairCipher(char str[], char key[])
{
charps, ks, keyT[5][5];

    // Key text
ks = strlen(key);
ks = removeSpaces(key, ks);
toLowerCase(key, ks);

    // ciphertext
ps = strlen(str);
toLowerCase(str, ps);
ps = removeSpaces(str, ps);

generateKeyTable(key, ks, keyT);

decrypt(str, keyT, ps);
}

// main code
int main()
{
charstr[SIZE], key[SIZE];
```

```
   // Key text that needs to be encrypted
strcpy(key, "Algorithm");
printf("Key text: %s\n", key);

   // Ciphertext that needs to be decrypted
strcpy(str, "ulroaliocvrx");
printf("Plain text: %s\n", str);

   // encryption is done using Playfair Cipher algorithm
decryptByPlayfairCipher(str, key);

printf("Deciphered text: %s\n", str);

return 0;
}
```

CONCLUSION:-

The above code will decrypt the cipher text to plain text using the Playfair cipher decryption technique. First, we call decryptByPlayfairCipher() which removes the space from cipher text and converts it to lowercase then call generateKeyTable() to generate a square grid. Then decrypt() will use that square grid to convert the cipher text to plain text.

Practical Experiment 4:

AIM:- WRITE A PROGRAM IN C/JAVA TO IMPLEMENT ASYMMETRIC ENCRYPTION.

THEORY:-

RSA algorithm is an asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys i.e. Public Key and Private Key. As the name describes that the Public Key is given to everyone and the Private key is kept private.

An example of asymmetric cryptography:

1. A client (for example browser) sends its public key to the server and requests some data.
2. The server encrypts the data using the client's public key and sends the encrypted data.
3. The client receives this data and decrypts it.

Since this is asymmetric, nobody else except the browser can decrypt the data even if a third party has the public key of the browser.

The idea of RSA is based on the fact that it is difficult to factorize a large integer. The public key consists of two numbers where one number is a multiplication of two large prime numbers. And private key is also derived from the same two prime numbers. So if somebody can factorize the large number, the private key is compromised. Therefore encryption strength totally lies on the key size and if we double or triple the key size, the strength of encryption increases exponentially. RSA keys can be typically 1024 or 2048 bits long, but experts believe that 1024-bit keys could be broken in the near future. But till now it seems to be an infeasible task.

The RSA algorithm is a very fast algorithm for encryption and decryption. It is used in many applications like encryption and decryption of messages. The algorithm is based on the idea that if we know the public and private keys, then we can encrypt and decrypt messages. An RSA user creates two large prime numbers, p and q, and computes $n = pq$. Then they computes the totient function, $\lambda(n) = (p - 1)(q - 1)$. Choosing e as a relatively prime number to $\lambda(n)$ and $1 < e < \lambda(n)$ they release e as the public key.

Then they computes the private key, d, as follows:

$d * e = 1 \bmod \lambda(n)$

Encrytion is done as follows:

$c = m^e \bmod n$

over all the characters.

Decryption is done as follows:

$m = c^d \bmod n$

over all the characters.

RSA Algorithm:

1. Ask the user to enter two prime numbers and validate them.
2. Store the prime numbers in variables.

3. Compute n = pq.
4. Compute λ(n) = (p – 1)(q – 1).
5. Choose a random number e as a relatively prime number to λ(n) and 1 < e < λ(n).
6. Compute d = e^{-1} mod λ(n).
7. Print the public and private keys.
8. Ask the user to enter a message and store it in a variable.
9. Encrypt the message using the public key.
10. Decrypt the message using the private key.
11. Print the encrypted and decrypted message.

CODE IN C:-

```
/*
 * C program to Implement the RSA Algorithm
 */

#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>
#include<string.h>
longint p, q, n, t, flag, e[100], d[100], temp[100], j, m[100], en[100], i;
charmsg[100];
int prime(long int);
voidce();
longint cd(long int);
void encrypt();
void decrypt();
void main()
{
printf("\nENTER FIRST PRIME NUMBER\n");
scanf("%d", &p);
flag = prime(p);
if (flag == 0)
   {
printf("\nWRONG INPUT\n");
```

```
getch();
exit(1);
   }
printf("\nENTER ANOTHER PRIME NUMBER\n");
scanf("%d", &q);
flag = prime(q);
if (flag == 0 || p == q)
   {
printf("\nWRONG INPUT\n");
getch();
exit(1);
   }
printf("\nENTER MESSAGE\n");
fflush(stdin);
scanf("%s", msg);
for (i = 0; msg[i] != NULL; i++)
m[i] = msg[i];
   n = p * q;
   t = (p - 1) * (q - 1);
ce();
printf("\nPOSSIBLE VALUES OF e AND d ARE\n");
for (i = 0; i< j - 1; i++)
printf("\n%ld\t%ld", e[i], d[i]);
encrypt();
decrypt();
}
int prime(long intpr)
{
inti;
   j = sqrt(pr);
for (i = 2; i<= j; i++)
   {
if (pr % i == 0)
return 0;
   }
```

```
return 1;
}
voidce()
{
int k;
   k = 0;
for (i = 2; i< t; i++)
   {
if (t % i == 0)
continue;
flag = prime(i);
if (flag == 1 &&i != p &&i != q)
      {
e[k] = i;
flag = cd(e[k]);
if (flag > 0)
         {
d[k] = flag;
k++;
         }
if (k == 99)
break;
      }
   }
}
longint cd(long int x)
{
longint k = 1;
while (1)
   {
      k = k + t;
if (k % x == 0)
return (k / x);
   }
}
```

```
void encrypt()
{
longintpt, ct, key = e[0], k, len;
i = 0;
len = strlen(msg);
while (i != len)
   {
pt = m[i];
pt = pt - 96;
      k = 1;
for (j = 0; j < key; j++)
      {
         k = k * pt;
         k = k % n;
      }
temp[i] = k;
ct = k + 96;
en[i] = ct;
i++;
   }
en[i] = -1;
printf("\nTHE ENCRYPTED MESSAGE IS\n");
for (i = 0; en[i] != -1; i++)
printf("%c", en[i]);
}
void decrypt()
{
longintpt, ct, key = d[0], k;
i = 0;
while (en[i] != -1)
   {
ct = temp[i];
      k = 1;
for (j = 0; j < key; j++)
      {
```

```
        k = k * ct;
        k = k % n;
      }
pt = k + 96;
m[i] = pt;
i++;
   }
m[i] = -1;
printf("\nTHE DECRYPTED MESSAGE IS\n");
for (i = 0; m[i] != -1; i++)
printf("%c", m[i]);
}
```

CONCLUSION:-

RSA algorithm is an asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys i.e. Public Key and Private Key. As the name describes that the Public Key is given to everyone and the Private key is kept private.

Practical Experiment 5:

AIM:-Introduction to GnuPG encryption system.

THEORY:-

PGP (Pretty Good Privacy) is an encryption program that provides cryptographic privacy and authentication for data communication (Refer to: https://en.wikipedia.org/wiki/Pretty_Good_Privacy). And OpenPGP is a standard that PGP software should follow. One of the software is GnuPG.

GnuPG helps people to protect their privacy by encrypting or signing their mails and guarantees that only the recipient could decrypt them or others could validate the content. It's not under any government regulation, which in a sense means 'absolute' freedom.

In many distrosGnuPG has been built in. If not, find it in official repos.

$ gpg--generate-key*# generate a new pair of keys*

It's quite simple to get your own private key and public key. The former one is for decryption, and latter one for encryption (though this statement is not so rigorous).

Then you can send your public key (including subkeys) to a key server like hkp://keys.gnupg.net (hkp is a protocol specialized for PGP).

$ gpg--keyserver keys.gnupg.net --send-key[last 8 chars of your key fingerprint]

Now you're able to encrypt your e-mails with GnuPG in E-mail management software like Evolution. If you would like send others an e-mail, use his public key. But do remember that it's better to check the fingerprint of the key via telephone, messaging software or ask him in person.

Here be careful that the master public key is usually used for certify your subkeys. It's recommended to use subkeys for encryption or signing, since it'll be much easier to revoke a subkey rather than a master key.

INSTALLATION:-

GnuPG can be downloaded from the GnuPG homepage and you should follow the instructions relating to your own system. This page assumes you are using a Mac OS X or Linux based enviroment and Windows users would have to adjust accordingly (in fact this page would be more appropriate reading).

For installation on a Mac the easiest method is to install Fink and then issue the following command in your terminal:

yoda:~ ian$ sudo apt-get install gnupg

If you get a message about not being in the sudoers file make sure to add your username into /etc/sudoers using the root account. Many programs can be installed in this way with Fink and I highly recommend it for anyone who enjoys the UNIX aspect of OS X.

Verify your installation by typing the following command and checking that a path is returned as follows:

```
yoda:~ ian$ which gpg
/sw/bin/gpg
```

ENCRYPTING A FILE:-

Now that we have generated our keys we can go ahead and encrypt a file. For this example we will make a gpgtest directory in our home directory for our test file:

yoda:~ ian$ mkdir ~/gpgtest

yoda:~ ian$ cd ~/gpgtest

yoda:~/gpgtestian$ echo 'this is a secret!!!' >secret.txt

yoda:~/gpgtestian$ cat secret.txt

this is a secret!!!

Here we first create our directory and then change to it, we then create a simple text file containing line 'this is a secret!!!' and then test the contents of our file using cat.

For now you can see that the secret file is the only one we have in our directory:

yoda:~/gpgtestian$ ls

secret.txt

We will now go ahead and encrypt this file, specifying the recipient as ourselves (remember we can do this because we have our own address in our public keyring).

We will use the -e (encrypt) flag of gpg, along with the -r (recipient) flag and our recipient and finally the name of the file we wish to encrypt:

yoda:~/gpgtestian$ gpg -e -r ian@atkinson.co.uk secret.txt

gpg: checking the trustdb

gpg: checking at depth 0 signed=0 ot(-/q/n/m/f/u)=0/0/0/0/0/1

yoda:~/gpgtestian$ ls

secret.txt

secret.txt.gpg

As you can see, using the -e option gpg has named the encypted file by simply appended .gpg to our clear text file. In many cases this will be suitable as it's easier to keep track of a file and it's encrypted equivalent if they have the same root file name. Of course, depending on the level of security you want to obtain, changing the file name and removing the .gpg extension is likely to draw less attention.

If we wanted we could specify the name of the output file by using the -o (output) option, for example we could save our encrypted file as encrypted.txt rather than secret.txt.gpg as follows:

yoda:~/gpgtestian$ gpg -e -r ian@atkinson.co.uk -o encrypted.txt secret.txt

This file is now encrypted, if you try using the cat command again you should just see a lot of random characters (this may make your terminal go weird though so either take my word for it or be prepared to launch a new one!).

It is worth noting that whilst we have used a plain text file for our example, we can encrypt other types of data of necessary, such as an image.

The encryption process will also hide the file type from the operating system since the headers become encrypted, further helping the privacy of our document. Notice in the following example

how, when encypted, the PNG image file is no longer recognised as an image by the system and it simply ‘data’:

yoda:~/gpgtestian$ file picture.png

picture.png: PNG image data, 1319 x 968, 8-bit/color RGB, non-interlaced

yoda:~/gpgtestian$ gpg -e -r ian@atkinson.co.uk -o encrypted_picture.png picture.png

yoda:~/gpgtestian$ file encrypted_picture.png

encrypted_picture.png: data

DECRYPTING A FILE:-

Now that we have encrypted our file, we can also decrypt it. You may wish to decrypt a file for editing or of couse just to view the contents. We decrypt using the -d flag as follows:

yoda:~/gpgtestian$ gpg -d secret.txt.gpg

You need a passphrase to unlock the secret key for

user: "Ian Atkinson (home) <ian@atkinson.co.uk>"

1024-bit ELG-E key, ID 71445CC9, created 2006-04-06 (main key ID 0EB6F689)

Enter passphrase: my passphrase

gpg: encrypted with 1024-bit ELG-E key, ID 71445CC9, created 2006-04-06

"Ian Atkinson (home) <ian@atkinson.co.uk>"

this is a secret!!!

Gpg will ask us for our passphrase, which you chose when generating your keys. If you type the phrase in successfully then the content of the file will be printed to stdout (the screen).

This isn’t usually very useful, and certainly isn’t for non text files, so what is more common is to output to a file in the same way as we did when encrypting using the -o flag. We can save the decrypted file as notasecret.txt as follows:

yoda:~/gpgtestian$ gpg -d -o notasecret.txt secret.txt.gpg

This is all the information you need in order to encrypt and decrypt files for yourself. You can encrypt any of your own sensetive data in this way and keep it safe without learning any further commands.

However, if you wish to encrypt files for other people, or have people encrypt files for you, then you must learn about importing and exporting public keys.

CONCLUSION:-

GnuPG is an implementation of PGP (Pretty Good Privacy), which is a form of public key/private key encryption. The strength of the encryption comes from the fact that a file can be encrypted for a given recipient using only the public key, yet both keys are needed in order to decrypt the file.

So the idea is to give your public key to your friends and collegues, but keep the private key closely guarded.

With GnuPG the keys are associated with an ID consisting of a name, comment and e-mail address. When specifying recipients you may use either the name or the e-mail adress but due to the complexity of dealing with the spaces in the names we will be using the e-mail address.

Practical Experiment 6:

AIM:- Implementation of Decryption techniques using secret key in GnuPG.

THEORY:-

GnuPG (GNU Privacy Guard) is a widely used open-source software tool that provides encryption and decryption of data, primarily for securing communication and files. It uses a combination of asymmetric (public-key) and symmetric (secret-key) cryptography for encryption and decryption. In this response, I'll provide you with a basic overview of how to implement decryption using a secret key in GnuPG.

Here's a step-by-step guide to decrypting a file using a secret key in GnuPG:

1. Import the Secret Key:

If you haven't already imported the secret key into your keyring, you need to do so. The secret key is required to decrypt the data encrypted with the corresponding public key.

gpg --import path/to/secret-key.asc

2. Decrypt the File:

Once you have the secret key imported, you can use the gpg command to decrypt the file. Replace encrypted-file.gpg with the actual filename of the encrypted file you want to decrypt.

gpg --decrypt encrypted-file.gpg

GnuPG will prompt you for the passphrase associated with the secret key. Enter the passphrase, and GnuPG will decrypt the file and output the result to the terminal.

3. Output to a File:

If you want to save the decrypted content to a file, you can use the --output option.

gpg --decrypt encrypted-file.gpg --output decrypted-file.txt

This command will decrypt the content of encrypted-file.gpg and save it to decrypted-file.txt.

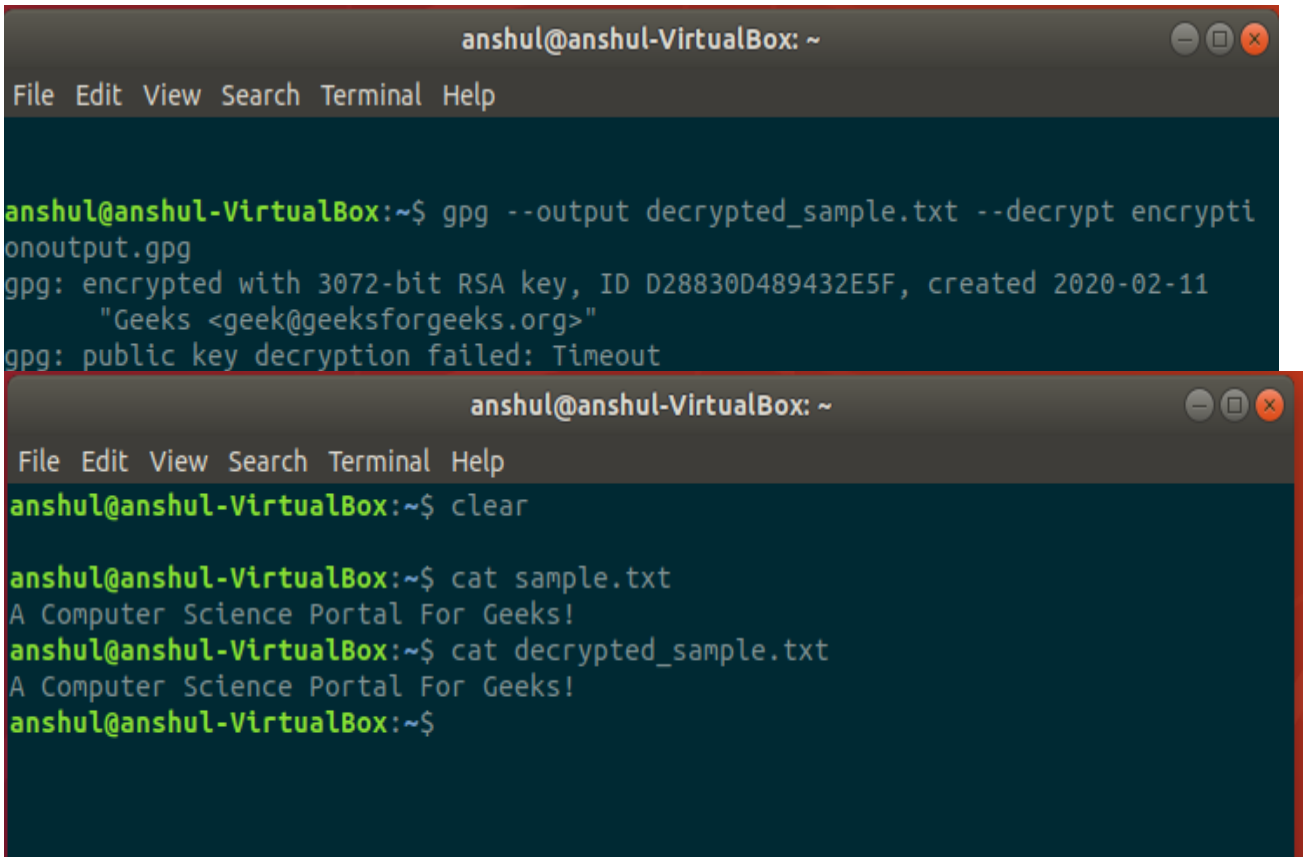
Remember that the above steps assume that you have the necessary secret key and passphrase to decrypt the file. If the file was encrypted using your public key, you need the corresponding secret key. If the file was encrypted using someone else's public key, they will need to provide you with the corresponding secret key or decrypt the file themselves.

Also, ensure that you're following best practices for securely managing your secret keys and passphrases, as they are critical for the security of your encrypted data.

Please note that GnuPG is a powerful tool with many options and features. The above steps provide a basic overview of decrypting a file using a secret key. You can refer to the GnuPG documentation for more advanced usage and options.

GnuPG is an encryption module that uses OpenPGP at its core. PGP stands for Pretty Good Privacy. It is an encryption program that provides authentication and cryptographic privacy for data communication. In the age where *data is the new oil*, the modern age thieves won't intrude in through doors, windows or roofs, but instead, via wires and electrical signals in form of a few lines of code and commands. Nothing in this world is secure, which leads to an obvious conclusion that these thieves are inevitable.

But instead of trying to figure out which door these thieves will intrude through, we might just need to focus on what is it they need … *Data*. Data is the holy grail of wine which contains the



DECRYPTING A FILE

Now that we have encrypted our file, we can also decrypt it. You may wish to decrypt a file for editing or of couse just to view the contents. We decrypt using the -d flag as follows:

yoda:~/gpgtestian$ gpg -d secret.txt.gpg

You need a passphrase to unlock the secret key for

user: "Ian Atkinson (home) <ian@atkinson.co.uk>"

1024-bit ELG-E key, ID 71445CC9, created 2006-04-06 (main key ID 0EB6F689)

Enter passphrase: my passphrase

gpg: encrypted with 1024-bit ELG-E key, ID 71445CC9, created 2006-04-06

"Ian Atkinson (home) <ian@atkinson.co.uk>"

this is a secret!!!

Gpg will ask us for our passphrase, which you chose when generating your keys. If you type the phrase in successfully then the content of the file will be printed to stdout (the screen).

This isn't usually very useful, and certainly isn't for non text files, so what is more common is to

output to a file in the same way as we did when encrypting using the -o flag. We can save the decrypted file as notasecret.txt as follows:

yoda:~/gpgtestian$ gpg -d -o notasecret.txt secret.txt.gpg

This is all the information you need in order to encrypt and decrypt files for yourself. You can encrypt any of your own sensetive data in this way and keep it safe without learning any further commands.

However, if you wish to encrypt files for other people, or have people encrypt files for you, then you must learn about importing and exporting public keys.

CONCLUSION:-

The decryption process described above assumes that you have the necessary secret key and passphrase to decrypt the data. It's essential to manage secret keys and passphrases securely, as they are critical for the security of your encrypted data.

GnuPG provides additional features and options for more advanced usage, such as specifying different keyrings, using multiple recipients, or signing and verifying data.

Practical Experiment 7:

AIM:- Study and Implementation of Digital Certificates using Java, Study of DNS certificates.

Theory:-

Digital Signatures are an Asymmetrically encrypted hash of a digital message(data). It is a value that can provide a guarantee of authenticity, non-repudiation, and integrity. In other terms, it means you can verify the sender, date & time and message content have not been revealed or compromised.

DNS (Domain Name System):

1. Introduction to DNS: DNS is a hierarchical decentralized naming system that associates domain names with IP addresses. It plays a crucial role in the functioning of the internet by translating user-friendly domain names into IP addresses that machines understand.
2. Components of DNS: Key components include:

- Domain Name Registrar: A company that manages domain name registrations.
- Name Servers (DNS Servers): Servers that store DNS records and respond to DNS queries.
- DNS Records: Information stored in DNS servers, including A (IPv4 address), AAAA (IPv6 address), MX (mail exchange), CNAME (canonical name), and more.

3. DNS Resolution Process: When a user enters a domain name, their computer follows a series of steps to resolve the domain to an IP address. These steps involve querying DNS servers and caching.
4. DNSSEC (Domain Name System Security Extensions): DNSSEC is a suite of extensions that adds security to the DNS protocol by digitally signing DNS data. It helps prevent DNS spoofing and other attacks.

SSL/TLS Certificates:

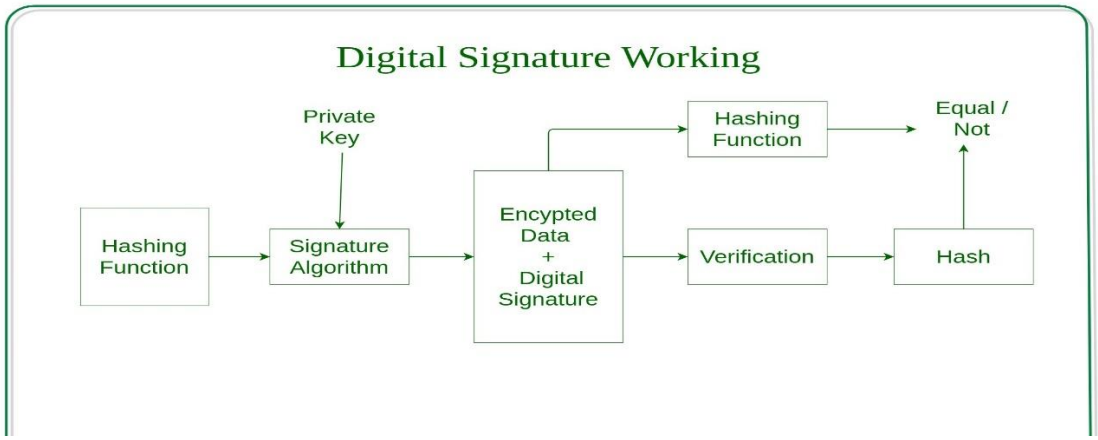
1. Introduction to SSL/TLS: SSL (Secure Sockets Layer) and its successor TLS (Transport Layer Security) are cryptographic protocols used to secure communication over a network. They ensure data confidentiality and integrity between a client and a server.
2. SSL/TLS Handshake: When a client connects to a secure website (using HTTPS), an SSL/TLS handshake occurs. It involves a series of steps to establish a secure connection, including key exchange, cipher negotiation, and verification of the server's SSL/TLS certificate.
3. SSL/TLS Certificates: SSL/TLS certificates are digital certificates issued by Certificate Authorities (CAs). They contain information about the certificate holder, the public key, and the digital signature of the CA. SSL/TLS certificates are used to verify the authenticity of a website and enable encrypted communication.

4. Types of SSL/TLS Certificates: There are different types of SSL/TLS certificates, including:

- Domain Validated (DV) Certificates: Basic certificates that verify domain ownership.

- Organization Validated (OV) Certificates: Include additional organization verification.
- Extended Validation (EV) Certificates: Provide the highest level of validation and display the company name in the browser's address bar.
- Wildcard Certificates: Secure multiple subdomains.
- Multi-Domain (SAN) Certificates: Secure multiple domains with a single certificate.

5. Certificate Chain: SSL/TLS certificates are issued in a chain of trust. The root CA signs intermediate CAs, which in turn sign end-entity certificates (website certificates).
6. Certificate Revocation: Certificates can be revoked if compromised or no longer valid. Online Certificate Status Protocol (OCSP) and Certificate Revocation Lists (CRLs) are used to check the status of certificates.



ity and esolved in IoT ıe input vill then receiver ıe.

Digital Signature Flow:

Let “A” and “B” be the fictional actors in the cryptography system for better understanding.

“A” is the sender and calculates the hash of the message and attaches signature which he wants to send using his private key.

The other side “B” hashes the message and then decrypts the signature with A’s public key and compares the two hashes

If “B” finds the hashes matching then the message has not been altered or compromised.

Implementing Digital Signatures:

Let us implement the digital signature using algorithms SHA and RSA and also verify if the hash matches with a public key.

Approach:

Create a method named Create_Digital_Signature() to implement Digital Signature by passing two parameters input message and the private key. In this method we will get an instance of the signature object passing the signing algorithm and assign it with a private key and finally pass the input this will return byte array.

public static byte[] Create_Digital_Signature(byte[] input, PrivateKeyprivateKey);

signature.initSign(privateKey);

signature.update(input);

The next step is to generate asymmetric key pair using RSA algorithm and SecureRandom class functions.

SecureRandomsecureRandom =new SecureRandom();

KeyPairGeneratorkeyPairGenerator = KeyPairGenerator.getInstance(ALGORITHM);

Finally verifying the signature using public key. Verify_Digital_Signature() method is used to check whether the signature matches by passing it the input, signature, and public key.

Signature signature = Signature.getInstance(SIGNING_ALGORITHM);

signature.initVerify(publickey);

signature.update(input);

Example:

Input:msg = "GEEKSFORGEEEKS IS A COMPUTER SCIENCE PORTAL"

Output:

Signature Value:

80429D3FA203437B4098CAF774D96C827B6CC2489F437A82926DA2EFCE64EF68FB33235B9F6BA8E3B033235B9F6BA8

Verification: true

Below is the implementation:

```
// Java implementation for Generating
// and verifying the digital signature

packagejava_cryptography;

// Imports
importjava.security.KeyPair;
importjava.security.KeyPairGenerator;
importjava.security.PrivateKey;
importjava.security.PublicKey;
importjava.security.SecureRandom;
importjava.security.Signature;
importjava.util.Scanner;
```

```
importjavax.xml.bind.DatatypeConverter;

public class Digital_Signature_GeeksforGeeks {

   // Signing Algorithm
private static final String
      SIGNING_ALGORITHM
      = "SHA256withRSA";
private static final String RSA = "RSA";
private static Scanner sc;

   // Function to implement Digital signature
   // using SHA256 and RSA algorithm
   // by passing private key.
public static byte[] Create_Digital_Signature(
byte[] input,
PrivateKey Key)
throws Exception
   {
      Signature signature
         = Signature.getInstance(
            SIGNING_ALGORITHM);
signature.initSign(Key);
signature.update(input);
returnsignature.sign();
   }

   // Generating the asymmetric key pair
   // using SecureRandom class
   // functions and RSA algorithm.
public static KeyPairGenerate_RSA_KeyPair()
throws Exception
   {
SecureRandomsecureRandom
         = new SecureRandom();
```

```
KeyPairGeneratorkeyPairGenerator
        = KeyPairGenerator
            .getInstance(RSA);
keyPairGenerator
        .initialize(
            2048, secureRandom);
returnkeyPairGenerator
        .generateKeyPair();
   }

   // Function for Verification of the
   // digital signature by using the public key
public static boolean
Verify_Digital_Signature(
byte[] input,
byte[] signatureToVerify,
PublicKey key)
throws Exception
   {
      Signature signature
         = Signature.getInstance(
            SIGNING_ALGORITHM);
signature.initVerify(key);
signature.update(input);
return signature
        .verify(signatureToVerify);
   }

   // Driver Code
public static void main(String args[])
throws Exception
   {

      String input
         = "GEEKSFORGEEKS IS A"
```

```
            + " COMPUTER SCIENCE PORTAL";
KeyPairkeyPair
        = Generate_RSA_KeyPair();

      // Function Call
byte[] signature
        = Create_Digital_Signature(
input.getBytes(),
keyPair.getPrivate());

System.out.println(
        "Signature Value:\n "
        + DatatypeConverter
            .printHexBinary(signature));

System.out.println(
        "Verification: "
        + Verify_Digital_Signature(
input.getBytes(),
            signature, keyPair.getPublic()));
   }
}
```

CONCLUSION:-

It seems there might be a confusion between DNS (Domain Name System) and SSL/TLS certificates. DNS is a system used to translate human-friendly domain names (like www.example.com) into IP addresses that computers use to identify each other on a network. SSL/TLS certificates, on the other hand, are used to secure the communication between a client (like a web browser) and a server.

The digital signature is a mechanism that verifies the authority of digital messages as well as documents. It is very popular because it provides more security than other signatures. In Java, JDK Security API is used to create and implement digital signatures. In this section, we will discuss the digital signature mechanism and also implement the digital signature mechanism in a Java program.

Practical Experiment 8:

AIM:- Case Study on Secure Socket Layer(SSL) and Transport Layer Security(TLS), Secure Electronic Transaction (SET).

THEORY:-

Let's consider a case study involving the implementation of Secure Sockets Layer (SSL) to secure online communication for a fictional e-commerce website called "SecureMart."

Case Study: SecureMart - Implementing SSL for Secure Online Shopping

Background:

SecureMart is a growing online marketplace that allows customers to browse and purchase a wide range of products. As the business expands, the need for secure online transactions becomes crucial to protect customer data and build trust.

Challenges:

1. Data Security: Customer personal and financial information, such as credit card details and addresses, must be securely transmitted over the internet to prevent interception and unauthorized access.
2. Customer Trust: Establishing customer trust is essential for the success of SecureMart. Customers need assurance that their information is safe and that they are interacting with a legitimate website.
3. Regulatory Compliance: Compliance with data protection regulations, such as GDPR or CCPA, is vital to avoid legal and financial penalties.

Solution:

SecureMart decides to implement SSL (Secure Sockets Layer) to secure the communication between customers and its website.

Steps Taken:

1. SSL Certificate Selection:

SecureMart selects an Extended Validation (EV) SSL certificate from a reputable Certificate Authority (CA). An EV certificate provides the highest level of validation and displays the company name in the browser's address bar, enhancing customer trust.

2. Certificate Acquisition:

SecureMart purchases the EV SSL certificate from the CA. The CA verifies the organization's identity and issues the certificate along with the corresponding private key.

3. Website Configuration:

The IT team at SecureMart configures the web server to enable SSL/TLS. They install the SSL certificate and private key on the server. The website is now accessible using the "https://" protocol.

4. Secure Data Transmission:

When customers visit SecureMart's website, their browsers establish a secure connection using SSL/TLS. Data exchanged between the customer's browser and the server, such as login credentials and payment information, is encrypted using strong encryption algorithms.

5. Browser Indicators:

With the EV SSL certificate, modern browsers display the company's name in the address bar, indicating a secure and verified connection. This visual cue enhances customer trust.

6. Regular Certificate Renewal:

SecureMart sets up reminders to renew the SSL certificate before it expires to ensure uninterrupted secure communication.

Benefits and Results:

1. Enhanced Security: SSL encryption ensures that customer data is protected during transmission, reducing the risk of interception and data breaches.
2. Customer Trust: The EV SSL certificate and browser indicators reassure customers that they are interacting with a legitimate and secure website, increasing their confidence in making online purchases.
3. Compliance: Implementing SSL helps SecureMart comply with data protection regulations, as it safeguards sensitive customer information.
4. Business Growth: With improved security and customer trust, SecureMart attracts more customers and experiences increased sales, contributing to business growth.
5. Positive Reputation: SecureMart gains a reputation for prioritizing customer security and privacy, leading to positive word-of-mouth and repeat business.

In summary, the case study of SecureMart demonstrates the importance of implementing SSL to secure online communication, enhance customer trust, and achieve regulatory compliance. By choosing an EV SSL certificate, configuring the web server correctly, and ensuring regular certificate renewal, SecureMart successfully addresses security challenges and creates a secure and trustworthy online shopping experience for its customers.

Let's explore a case study involving the implementation of Transport Layer Security (TLS) to secure communications for a fictional healthcare technology company called "HealthTech Secure."

Case Study: HealthTech Secure - Implementing TLS for Secure Patient Data Transmission

Background:

HealthTech Secure is a technology company that specializes in providing secure communication solutions for healthcare providers. Their platform enables the exchange of sensitive patient information between hospitals, clinics, and medical practitioners.

Challenges:

1. Patient Data Privacy: HealthTech Secure handles sensitive patient data, including medical records, diagnoses, and treatment plans. Ensuring the privacy and confidentiality of this data is critical.
2. Legal and Regulatory Compliance: HealthTech Secure must comply with healthcare data protection regulations, such as HIPAA (Health Insurance Portability and Accountability Act) in the United States.
3. Secure Communication: Healthcare providers need a reliable and secure way to transmit patient data electronically while preventing eavesdropping and unauthorized access.

Solution:

HealthTech Secure decides to implement Transport Layer Security (TLS) to secure the

communication between healthcare providers using their platform.

Steps Taken:

1. TLS Certificate Selection:

HealthTech Secure acquires an Organization Validated (OV) TLS certificate from a trusted Certificate Authority (CA). The OV certificate verifies the organization's identity and authenticity.

2. Certificate Acquisition:

HealthTech Secure obtains the OV TLS certificate from the CA. The certificate includes a public key and is signed by the CA.

3. Platform Configuration:

HealthTech Secure integrates the TLS certificate into their communication platform. They configure the platform to support TLS and use strong encryption algorithms for secure data transmission.

4. Data Encryption and Decryption:

When healthcare providers exchange patient data using HealthTechSecure's platform, the data is encrypted using the public key from the TLS certificate before transmission. The recipient decrypts the data using the corresponding private key.

5. Client Authentication:

HealthTech Secure implements mutual TLS authentication to ensure that both the client (healthcare provider) and the server (HealthTech Secure platform) verify each other's authenticity using their respective certificates.

6. Regular Certificate Renewal:

HealthTech Secure sets up automated reminders for certificate renewal before it expires to ensure continued secure communication.

Benefits and Results:

1. Patient Data Security: Implementing TLS encryption ensures that patient data is protected during transmission, reducing the risk of unauthorized access and data breaches.
2. Regulatory Compliance: HealthTechSecure's use of TLS encryption helps them comply with healthcare data protection regulations, such as HIPAA, by ensuring the confidentiality and integrity of patient data.
3. Provider Trust: Healthcare providers trust HealthTechSecure's platform as a secure means of transmitting sensitive patient data, contributing to stronger relationships and partnerships.
4. Business Growth: The reputation for prioritizing patient data security leads to increased adoption of HealthTechSecure's platform, driving business growth.
5. Industry Leadership: HealthTech Secure becomes recognized as a leader in providing secure communication solutions for the healthcare industry, further enhancing its brand reputation.

In summary, the case study of HealthTech Secure illustrates the importance of implementing TLS for secure communication in the healthcare sector. By obtaining an OV TLS certificate, configuring their platform to support TLS encryption, and enabling mutual authentication, HealthTech Secure successfully addresses data security and compliance challenges, leading to enhanced trust and growth in the healthcare technology market.

Let's explore a case study involving the implementation of secure electronic transactions for a fictional online banking and financial services company called "SecureBank."

Case Study: SecureBank - Implementing Secure Electronic Transactions for Online Banking

Background:

SecureBank is a leading online banking and financial services company that provides a range of services, including account management, fund transfers, bill payments, and investment opportunities. As the digital banking landscape evolves, ensuring secure electronic transactions becomes paramount.

Challenges:

1. Data Security: SecureBank deals with sensitive financial and personal information of customers. Ensuring the confidentiality and integrity of this data during electronic transactions is crucial.
2. Customer Trust: Building and maintaining trust is essential for online banking. Customers need confidence that their financial transactions are secure and protected.
3. Regulatory Compliance: SecureBank must comply with financial regulations and standards, such as PCI DSS (Payment Card Industry Data Security Standard).

Solution:

SecureBank implements a comprehensive approach to secure electronic transactions, incorporating encryption, authentication, and compliance measures.

Steps Taken:

1. Secure Online Platform:

SecureBank develops a secure online banking platform with robust encryption protocols to protect data during transmission. They use HTTPS (HTTP Secure) to encrypt the communication between customers' browsers and their servers.

2. Two-Factor Authentication (2FA):

SecureBank implements two-factor authentication for customer accounts. Customers are required to provide a combination of their password and a unique, time-sensitive code from a mobile app or a hardware token.

3. SSL/TLS Certificates:

SecureBank acquires Extended Validation (EV) SSL/TLS certificates from a reputable Certificate Authority. EV certificates enhance trust by displaying the company name in the browser's address bar.

4. Secure Mobile App:

SecureBank develops a secure mobile banking app that utilizes strong encryption for data at rest and in transit. The app enforces biometric authentication (e.g., fingerprint or face recognition) for added security.

5. Regular Security Audits:

SecureBank conducts regular security audits and vulnerability assessments of its online platform

and mobile app to identify and address potential weaknesses.

6. Compliance with Standards:

SecureBank ensures compliance with relevant industry standards such as PCI DSS. They implement controls to protect cardholder data during payment transactions.

Benefits and Results:

1. Data Protection: The implementation of strong encryption and two-factor authentication ensures that customer data is secure during electronic transactions, reducing the risk of unauthorized access.
2. Customer Trust: The use of EV SSL/TLS certificates and two-factor authentication builds customer trust in the security of SecureBank's online banking services.
3. Regulatory Compliance: SecureBank's adherence to PCI DSS and other standards demonstrates their commitment to protecting sensitive financial information and ensures compliance with industry regulations.
4. User Adoption: Customers feel confident using SecureBank's online banking and mobile app due to the robust security measures in place, leading to increased adoption of digital banking services.
5. Brand Reputation: SecureBank gains a reputation for prioritizing customer security, contributing to positive brand perception and customer loyalty.

In summary, the case study of SecureBank illustrates the importance of implementing secure electronic transactions in the online banking industry. By employing encryption, two-factor authentication, compliance measures, and secure mobile apps, SecureBank successfully addresses data security and compliance challenges, leading to enhanced customer trust and growth in the digital banking market.

Practical Experiment 9:

AIM:- Case Study on use and implementation of Firewall.

THEORY:-

Let's explore a case study involving the use and implementation of a firewall for a fictional company called "TechCoSolutions."

Case Study: TechCo Solutions - Implementing a Firewall for Network Security

Background:

TechCo Solutions is a medium-sized technology company that offers IT consulting, software development, and cloud services to various clients. The company manages sensitive client data, proprietary software code, and critical business information, making network security a top priority.

Challenges:

1. Network Security: With an increasing number of cyber threats, TechCo Solutions needs a robust solution to protect its internal network and systems from unauthorized access, malware, and other cyberattacks.
2. Remote Workforce: Due to the pandemic, a significant portion of TechCo's workforce is now working remotely, requiring secure remote access to the company's resources.
3. Compliance: TechCo must adhere to industry regulations and standards, such as HIPAA (Health Insurance Portability and Accountability Act) and GDPR (General Data Protection Regulation), to ensure data security and privacy.

Solution:

TechCo Solutions implements a multi-layered firewall strategy to safeguard its network and data.

Steps Taken:

1. Firewall Selection:

TechCo selects a next-generation firewall (NGFW) that offers advanced features beyond traditional firewall capabilities, including intrusion prevention, application control, and deep packet inspection.

2. Firewall Deployment:

TechCo deploys firewalls at various entry points in its network architecture, including the perimeter, data center, and remote access points. This ensures comprehensive protection against both external and internal threats.

3. Rule Configuration:

The IT team configures firewall rules to control inbound and outbound traffic. They create rules based on application type, user role, IP addresses, and other criteria to enforce a least-privilege access model.

4. Intrusion Prevention:

TechCo enables intrusion prevention features on the firewall to detect and block suspicious activities, such as port scanning and unauthorized access attempts.

5. Application Control:

The NGFW includes application control capabilities, allowing TechCo to regulate the use of specific applications to prevent security vulnerabilities and maintain productivity.

6. Remote Access VPN:

To facilitate secure remote work, TechCo sets up a virtual private network (VPN) on the firewall. Employees use VPN connections to access the company's resources securely from remote locations.

7. Logging and Monitoring:

TechCo implements real-time logging and monitoring of firewall activities. Security analysts regularly review logs to identify and respond to potential security incidents.

Benefits and Results:

1. Enhanced Network Security: The multi-layered firewall approach ensures a strong defense against unauthorized access, malware, and other threats, protecting sensitive data and company resources.
2. Remote Work Enablement: The VPN functionality allows remote employees to securely access the company's network and systems, maintaining productivity while ensuring data security.
3. Regulatory Compliance: The firewall's features and configuration help TechCo meet compliance requirements, such as HIPAA and GDPR, by enforcing data protection measures.
4. Reduced Attack Surface: By carefully configuring firewall rules, TechCo minimizes the attack surface, reducing the risk of successful cyberattacks.
5. Incident Response: The real-time logging and monitoring capabilities aid in the timely detection and response to security incidents, minimizing potential damage.
6. Customer Trust: TechCo's commitment to robust network security enhances its reputation and builds trust with clients who rely on the company's services.

In summary, the case study of TechCo Solutions highlights the importance of implementing a multi-layered firewall strategy to address network security challenges. By deploying next-generation firewalls, configuring rules, enabling intrusion prevention, and facilitating secure remote access, TechCo successfully strengthens its network security posture, ensures compliance, and maintains its reputation as a trusted technology partner.

Practical Experiment 10:

AIM:- Case Study on Recent trends in IOT security, IDS, Cloud Security.

THEORY:-

RECENT TRENDS IN IOT SECURITY:

1. Edge Computing Securit y: With the increasing adoption of edge computing in IoT deployments, security measures are being designed to address vulnerabilities at the edge. Edge security includes measures like secure boot, hardware-based security, and edge-specific threat detection.

2. Zero Trust Architecture: IoT security is adopting the principles of Zero Trust, which means not assuming trust based on location or network boundaries. Devices and users are continuously authenticated and authorized regardless of their location.

3. AI and Machine Learning for Threat Detection: AI and machine learning are being used to analyze and detect abnormal patterns in IoT device behavior. These technologies help in identifying potential security threats and anomalies in real-time.

4. Blockchain for IoT Security: Blockchain technology is being explored to enhance the security of IoT devices, particularly in areas where device identity, data integrity, and trust are crucial.

5. Device Identity Management: Establishing and managing the identity of IoT devices is gaining importance. Solutions like unique device certificates and identity management platforms are being used to prevent unauthorized access.

6. Hardware-Based Security: Hardware-based security solutions, including trusted platform modules (TPMs) and secure enclaves, are being integrated into IoT devices to provide a higher level of protection for sensitive data and cryptographic operations.

7. Regulations and Standards: Governments and industry bodies are introducing regulations and standards for IoT security. These regulations aim to ensure a baseline level of security across IoT devices and systems.

8. Supply Chain Security: Ensuring the security of the entire IoT device supply chain is becoming critical. Manufacturers are being urged to implement secure development practices and supply chain integrity checks.

9. Firmware and Software Updates: Secure mechanisms for delivering and verifying firmware and software updates are being developed to ensure that devices remain protected against emerging threats.

10. Consumer Awareness and Education: As IoT devices become more common in households, there is an increasing need to educate consumers about the security risks associated with these devices and how to protect themselves.

Please note that the field of IoT security is rapidly evolving, and new trends and developments may have emerged since my last update. To get the most recent and accurate information about IoT security trends, I recommend checking reputable sources in the field of cybersecurity and IoT.

RECENT TRENDS IN IDS:

1. Behavioral Analysis and Anomaly Detection: Traditional signature-based IDS are being complemented with behavioral analysis and anomaly detection techniques. Machine learning and AI algorithms are employed to detect deviations from normal patterns of network and user behavior, helping to identify previously unknown threats.

2. Network Traffic Analysis: IDS solutions are focusing on deep packet inspection and network traffic analysis to detect advanced threats and attacks that may not have recognizable signatures. This approach enables the detection of more sophisticated attacks, including zero-day exploits.

3. Cloud and Hybrid Environments: As organizations move towards cloud and hybrid infrastructures, IDS solutions are adapting to monitor and analyze network traffic and activities across diverse environments, including on-premises, cloud, and containerized systems.

4. IoT and OT Security Integration: IDS solutions are expanding to cover Internet of Things (IoT) and Operational Technology (OT) environments, providing monitoring and protection for a wider range of connected devices and industrial systems.

5. Threat Intelligence Integration: IDS platforms are integrating threat intelligence feeds to enhance their ability to identify and respond to emerging threats. This integration helps in enriching the context of detected activities.

6. User and Entity Behavior Analytics (UEBA): IDS is incorporating UEBA capabilities to analyze user behaviors and detect insider threats or compromised accounts based on deviations from normal activities.

7. Automated Threat Response: IDS solutions are integrating with Security Orchestration, Automation, and Response (SOAR) platforms to enable automated responses to detected threats. This can include actions like isolating affected devices or blocking suspicious network traffic.

8. Enhanced Visualization and Reporting: Improved visualization tools provide security

analysts with more intuitive insights into network activities, making it easier to identify and investigate potential threats.

9. Container and Microservices Security: As containerization and microservices architectures become prevalent, IDS solutions are adapting to monitor and protect these environments by analyzing traffic between containers and microservices.

10. Zero Trust Security: IDS is aligning with the principles of Zero Trust by continuously monitoring and assessing the security posture of devices and users, regardless of their location within the network.

11. Decoy and Deception Technologies: Some IDS solutions are incorporating deception technologies, setting up decoy systems and fake assets to lure attackers and divert their attention.

It's important to note that the field of IDS is dynamic, and new trends and technologies may have emerged since my last update. To stay current with the latest developments in IDS, I recommend consulting reputable sources in the cybersecurity industry and staying connected with relevant conferences and research publications.

RECENT TRENDS IN Cloud Security:

1. Zero Trust Architecture (ZTA): Zero Trust principles are being applied to cloud security, emphasizing the need to authenticate and authorize every user and device accessing cloud resources, regardless of their location.

2. Cloud-Native Security: With the rise of serverless computing, containers, and microservices, security solutions are adapting to protect cloud-native applications and environments.

3. Identity and Access Management (IAM): IAM solutions are becoming more granular and adaptive, supporting dynamic access controls based on user behavior and context.

4. Security as Code: Security is being integrated into the DevOps process through practices like DevSecOps, where security controls are automated and treated as code.

5. Multi-Cloud Security: As organizations adopt multi-cloud strategies, security tools are being developed to provide consistent visibility and control across multiple cloud platforms.

6. Container Security: Container security solutions are focusing on runtime protection, vulnerability scanning, and image signing to secure containerized applications.

7. Serverless Security: Security tools are emerging to protect serverless functions, focusing on

runtime monitoring, vulnerability management, and access controls.

8. Cloud Workload Protection: Solutions are being developed to secure virtual machines and workloads in the cloud, including runtime protection, intrusion detection, and network segmentation.

9. Data Protection and Privacy: With increasing data regulations, cloud security is emphasizing encryption, data masking, and data loss prevention to ensure compliance and protect sensitive data.

10. Cloud Security Posture Management (CSPM): CSPM tools are being used to continuously assess and manage the security configuration of cloud resources, helping organizations identify and remediate misconfigurations.

11. Threat Detection and Response: Cloud-based threat detection and response solutions are being used to monitor cloud environments for suspicious activities and automate incident response.

12. API Security: As APIs play a crucial role in cloud applications, API security solutions are focusing on authentication, authorization, and monitoring of API traffic.

13. Serverless Function Security: Security measures are being developed to protect the code and data associated with serverless functions, addressing risks like code injection and data exposure.

14. Compliance Automation: Cloud security solutions are helping organizations automate compliance monitoring and reporting to meet regulatory requirements.

15. Cloud Security Training and Awareness: Organizations are investing in cloud security training for their teams to raise awareness about best practices and emerging threats.

Please note that the field of cloud security is rapidly evolving, and new trends and developments may have emerged since my last update. To get the most recent and accurate information about cloud security trends, I recommend checking reputable sources in the cybersecurity industry and staying connected with relevant conferences and research publications.