NAME – CHETAN SHARMA

REG NO – 20BCE0744

ASSESSMENT 3

SYMMETRIC KEY ALGORITHM

AES (ADVANCE ENCRYPTION STANDARD)

ALGORITHM AND WORKING

A popular symmetric encryption technique is called the Advanced Encryption Standard (AES). It employs a key with a length of 128, 192, or 256 bits and operates on blocks of data that are typically 128 bits in size. AES uses several encryption and decryption rounds, each requiring a specific set of procedures. Here is a high-level explanation of how AES works:

1. Key Expansion: The original key is expanded to generate a key schedule. This key schedule contains a series of round keys, which will be used in each encryption and decryption round.
2. Initial Round: The input plaintext block is XORed with the first round key.
3. Rounds: AES consists of multiple rounds (10, 12, or 14 rounds depending on the key size). Each round consists of four main operations:

   a. SubBytes: Each byte of the state is replaced with a corresponding byte from the S-box substitution table. This step provides confusion and non-linearity.

   b. ShiftRows: Bytes in each row of the state are shifted cyclically to the left. This step provides diffusion.

   c. MixColumns: Each column of the state is transformed using a mathematical operation. This step provides further diffusion and mixing.

   d. AddRoundKey: The state is XORed with the round key.

4. Final Round: The final round is similar to the normal rounds, but it excludes the MixColumns operation.

5. Output: After all the rounds are completed, the resulting state represents the encrypted ciphertext block.

For decryption, the process is similar but involves using the round keys in reverse order and applying inverse operations (InverseSubBytes, InverseShiftRows, InverseMixColumns) instead.

STRENGTH

The AES algorithm has several strengths and advantages that contribute to its widespread adoption and recognition as a secure encryption algorithm. Here are some key strengths and advantages of AES:

1. Security: When used properly, AES is thought to be extremely secure and resistant to a variety of cryptographic assaults. It has passed strict inspection and undergone in-depth research by cryptography specialists throughout the world. The complete AES algorithm has not been the target of any real-world assaults.

2. AES has three key length options: 128 bits, 192 bits, and 256 bits. The encryption is more secure the larger the key size. Based on the desired level of security and performance requirements, AES provides flexibility in selecting a suitable key length.

3. AES is meant to be computationally efficient and can be implemented in high-performance hardware and software. It can swiftly encrypt and decode data and has been optimised for use with contemporary computer architectures, making it appropriate for a wide range of applications.

4. AES is a standard encryption algorithm that has received widespread adoption from organisations, businesses, and governments around the world. International cryptographic standards, including the U.S. National Institute of

Standards and Technology (NIST), have endorsed and approved it. The widespread use of AES ensures compatibility and interoperability across various platforms and systems.

5. Attack Resistance: AES has been carefully tested against a number of attacks, including related-key attacks, linear cryptanalysis, and differential cryptanalysis. It exhibits considerable resilience to these attacks, which adds to the certainty of its security.

6. Design that is Transparent and Clear: AES has a design that is transparent and clear, making it simple to comprehend and use. The algorithm's simplicity lowers the possibility of implementation mistakes and weaknesses.

7. The AES standard is openly accessible, allowing for in-depth examination, peer review, and inspection by the cryptographic community. This openness promotes confidence in the algorithm by allowing for the identification of any potential flaws or vulnerabilities.

WEAKNESS

Although AES is regarded as a robust encryption algorithm, it's vital to remember that no algorithm is impervious to all potential threats. Here are some possible AES drawbacks or issues to be aware of:

1. AES implementations may be subject to side-channel assaults such timing attacks, power analyses, or analyses of electromagnetic radiation. In order to obtain the secret key, these attacks take advantage of data that has leaked through side channels (such as timing fluctuations or power consumption). These vulnerabilities can be reduced by defences such hardware protections and constant-time implementations.

2. AES may become vulnerable to attacks utilising quantum computers, although currently being impervious to assaults utilising traditional cryptography. The fundamental mathematical issues that underlie AES, like as factoring big numbers or resolving the discrete logarithm problem, may be cracked by quantum computers. To solve this problem, post-quantum algorithms are being developed.

3. Key management: Proper key management is essential for maintaining the security of any encryption system. Weak key distribution, storage, or creation procedures can weaken the security of AES. To maintain the security of AES, strong, random keys must be used, and secure key management procedures must be followed.

4. Considerations for Key Length: AES allows keys with lengths of 128, 192, and 256 bits. Although a bigger key size offers better security, it also necessitates more processing power. The intended security level and the system's performance needs should be taken into account when choosing the key size. In some circumstances, longer key sizes might not offer any further security advantages if key management is not done properly.

5. Cryptoanalysis Developments: New attacks against AES could always be found in the future as new cryptographic methods and improvements in computing power are developed. Updating cryptographic systems in accordance with the most recent security advice and research is crucial.

REAL WORLD EXAMPLES

AES is a cryptographic algorithm that is used in secure communication protocols to safeguard data sent over networks. For instance, it is used to encrypt email messages and attachments in secure email protocols like S/MIME (Secure/Multipurpose Internet Mail Extensions).

- AES is frequently used in virtual private networks (VPNs), which enable private and secure communication over public networks. It guarantees the confidentiality and security of any data sent between a user and a VPN server.

- Wireless Networks: To secure data transmissions between devices and access points in wireless networks, such as Wi-Fi networks, AES is utilised. It is a component of the widely used WPA2 (Wi-Fi Protected Access II) protocol for secure wireless communication.

- Disc encryption: AES is used in full-disk encryption programmes to safeguard a disc drive's whole contents. In the event that a device is lost or stolen, this helps protect the data on laptops, desktop computers, and other storage devices by prohibiting unauthorised access.

- Secure Messaging Apps: AES is frequently used to encrypt communications and ensure end-to-end encryption in messaging apps like WhatsApp and Signal. This indicates that only the messages' intended receivers will be able to decrypt and read them.

CODE

Java imports

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.GCMParameterSpec;
import java.util.Base64;
```

aes class

```
public class AES_ENCRYPTION {
    private SecretKey key;
    private final int KEY_SIZE = 128;
```

```
    private final int DATA_LENGTH = 128;
    private Cipher encryptionCipher;

    public void init() throws Exception {
        KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");
        keyGenerator.init(KEY_SIZE);
        key = keyGenerator.generateKey();
    }
```

Encryption

```
    public String encrypt(String data) throws Exception {
        byte[] dataInBytes = data.getBytes();
        encryptionCipher = Cipher.getInstance("AES/GCM/NoPadding");
        encryptionCipher.init(Cipher.ENCRYPT_MODE, key);
        byte[] encryptedBytes = encryptionCipher.doFinal(dataInBytes);
        return encode(encryptedBytes);
    }
```

Decryption

```
    public String decrypt(String encryptedData) throws Exception {
        byte[] dataInBytes = decode(encryptedData);
        Cipher decryptionCipher = Cipher.getInstance("AES/GCM/NoPadding");
        GCMParameterSpec spec = new GCMParameterSpec(DATA_LENGTH, encryptionCipher.getIV());
        decryptionCipher.init(Cipher.DECRYPT_MODE, key, spec);
        byte[] decryptedBytes = decryptionCipher.doFinal(dataInBytes);
        return new String(decryptedBytes);
    }
```

Encoding and decoding

```
    private String encode(byte[] data) {
        return Base64.getEncoder().encodeToString(data);
    }

    private byte[] decode(String data) {
        return Base64.getDecoder().decode(data);
    }
```

Main function

```
    public static void main(String[] args) {
        try {
            AES_ENCRYPTION aes_encryption = new AES_ENCRYPTION();
            aes_encryption.init();
```

```
            String encryptedData = aes_encryption.encrypt("Hello, welcome to the ethical
hacking");
            String decryptedData = aes_encryption.decrypt(encryptedData);

            System.out.println("Encrypted Data : " + encryptedData);
            System.out.println("Decrypted Data : " + decryptedData);
        } catch (Exception ignored) {
        }
    }
```

OUTPUT

```
Encrypted Data : 5FXTzE62fBD1jE3YKp3Pi55Z6pch5c0TwKBeA++WWaDQja+GqmJU17EYGte8XQdfPsEbIZg=
Decrypted Data : Hello, welcome to the ethical hacking
```

SECURITY ANALYSIS

The cryptographic community has studied the security of the AES algorithm in great detail since its creation. Here are some significant security assessments and analyses done on AES:

1. Cryptanalysis: To find potential flaws and vulnerabilities, cryptanalysts analysed AES in great detail. AES has been subjected to a number of cryptanalytic approaches, including differential cryptanalysis, linear cryptanalysis, and algebraic assaults. Numerous attack versions have been proposed by researchers, but the whole AES algorithm has not yet been the target of any workable attacks.

2. Before being chosen as the Advanced Encryption Standard, AES underwent a thorough examination procedure by the National Institute of Standards and Technology (NIST). The cryptography community was encouraged by NIST to submit algorithms, and a five-year competition assessed the proposals' performance and security. AES won as a result of its superior security features.

3. Academic Research: Experts from all around the world are still delving into the security of AES. Numerous scholarly articles have been written on a variety of AES-related topics, such as key schedule analysis, algebraic properties, and resilience to known attacks. These investigations aid in locating any potential weaknesses and further guarantee the security of AES.

4. International Standards: The International Organisation for Standardisation (ISO) and the Internet Engineering Task Force (IETF) have both embraced AES, which is acknowledged as an international standard. AES is widely used, and its presence in security protocols and standards is evidence of the confidence people have in its security.

5. Analysis that is ongoing: AES security analysis is a continuous activity. Researchers regularly look for potential weaknesses in the method as computing power increases and new cryptographic approaches are developed. This involves investigating quantum assaults, side-channel attacks, and other sophisticated attacks to make sure that AES is protected against developing security risks.

ASYMMETRIC KEY ALGORITHMS

RSA

ALGORITHM AND WORKING

This is how the RSA algorithm works:

1. Key Generation:
    - Choose two distinct prime numbers, p and q.
    - Calculate the modulus, n, by multiplying p and q: n = p * q.
    - Compute Euler's totient function, ϕ(n), where ϕ(n) = (p - 1) * (q - 1).
    - Choose an integer, e, such that 1 < e < ϕ(n) and e is coprime (relatively prime) to ϕ(n). In other words, e and ϕ(n) should not have any common factors other than 1.
    - Find the modular multiplicative inverse of e modulo ϕ(n), denoted as d. This means finding d such that (e * d) % ϕ(n) = 1.
    - The public key consists of the pair (e, n), and the private key consists of the pair (d, n).
2. Encryption:
    - Convert the plaintext message into a numerical representation, often using a predefined mapping.
    - To encrypt a message m, calculate the ciphertext c using the formula: c = (m^e) % n. This involves raising the plaintext message to the power of e and taking the modulus n.
3. Decryption:
    - To decrypt the ciphertext c, compute the original plaintext message m using the formula: m = (c^d) % n. This involves raising the ciphertext to the power of d and taking the modulus n.

ADVANTAGES

1. Security Strength: The inability to factor huge composite numbers into their prime factors is the foundation for RSA's security. Factoring is thought to be a computationally challenging problem, and this complexity is where RSA's strength resides. RSA encryption is regarded as secure against assaults by

contemporary classical computers and the majority of known quantum methods as long as the key size is properly chosen.

2. Public-Key Cryptography: The RSA algorithm requires two different keys for encryption and decryption, making it a public-key encryption algorithm. Everyone has access to the public key, which enables secure communication with numerous parties without the need to first exchange a private key. When secure communication between numerous entities is necessary, this feature makes key management easier.

3. Digital signatures, which offer data authenticity, integrity, and non-repudiation, can be created using RSA. The sender can establish their identity and guarantee that the communication hasn't been tampered with during transmission by signing a message with their private key. By utilising the associated public key to validate the signature, the sender's legitimacy is established.

4. Key Exchange: Two parties can exchange keys securely by using RSA. RSA can be used to implement the modular exponentiation-based Diffie-Hellman key exchange protocol. In the absence of any past shared secrets, this enables two entities to create a shared secret key across an unsecured channel.

5. Standardisation and Wide Use: RSA has undergone significant research, testing, and standardisation, which has helped to ensure its wide adoption and compatibility with a variety of platforms and systems. It is easily implementable in many applications because it is supported by a variety of cryptographic libraries and frameworks.

WEAKNESS

1. Key Size and Performance: Longer key sizes are needed to withstand rising computational power and improvements in factoring algorithms because the security of RSA depends on the challenge of factoring huge integers. Larger key sizes make encryption and decryption processes take longer and use more computer power. This could be a drawback for applications that need quick processing or environments with limited resources.

2. RSA is susceptible to attacks from strong quantum computers. Vulnerability to Quantum Computers. Shor's algorithm, a quantum algorithm, has the potential to defeat RSA encryption because it is effective at factoring big numbers. As quantum computer technology develops, there is a greater chance that RSA will be vulnerable to quantum assaults. Therefore, it is crucial to create and implement post-quantum cryptography algorithms in order to ensure long-term security.

3. Lack of perfect forward secrecy (PFS): PFS is a property that guarantees the confidentiality of earlier communications even in the event that the private key is compromised in the future. PFS is not integrated by RSA. All previously encrypted messages can be decoded using a compromised RSA private key. Additional procedures, such as routine key changes or the use of hybrid encryption techniques, are required to achieve PFS.

4. Key Management and Trust: To verify the validity of public keys, RSA uses either a public key infrastructure (PKI) or a trusted third party. The reliability and integrity of the key management system play a significant role in RSA's security. Vulnerabilities in encryption or authentication might result from compromised or poorly validated public keys.

5. Padding Vulnerabilities: To produce secure and randomly generated ciphertexts, RSA encryption requires the employment of padding techniques. Attacks like the infamous padding oracle exploit might result from improper

or insecure padding techniques. To keep RSA secure, specified padding techniques must be implemented correctly and followed strictly.

6. Potential Side-Channel Attacks: RSA implementations are susceptible to side-channel attacks, in which a hacker exploits unexpected information leaks like timing, power consumption, or electromagnetic emissions to learn more about the encryption or decryption process. Implementing countermeasures carefully is necessary to defend against side-channel assaults.

REAL WORLD EXAMPLES

1. Secure Web Communication (HTTPS): The SSL/TLS protocol suite, which enables secure communication over the internet, is fundamentally built around RSA. RSA is often used during the initial handshake to create a secure connection between your browser and the web server when you visit a website that uses HTTPS (secure HTTP). In order to agree on a shared symmetric key for encrypting the actual data transmission, the client and server employ RSA for key exchange.

2. Digital Signatures: The RSA algorithm is frequently used to create and validate digital signatures. Data integrity, authentication, and non-repudiation are all provided via digital signatures. When signing digital documents, software, or transactions, both businesses and individuals can use RSA to verify the identity of the signer and guarantee that the content hasn't been changed since the signature was made.

3. Secure Email Communication: Many email applications and servers offer secure communication using RSA encryption. Email messages can be encrypted with RSA to prevent eavesdropping and unauthorised access, and to exchange keys.

4. Virtual Private Networks (VPNs): To create secure connections between remote users and business networks, VPNs use RSA. For key exchange and authentication purposes in VPN protocols (like OpenVPN), it is employed. This protects the confidentiality and integrity of the data being transferred.

5. For secure remote login and file transfer between systems, Secure Shell (SSH) frequently uses RSA. It is used to establish a secure connection and authenticate the client and server during the SSH handshake.

6. Secure File Storage: To encrypt the symmetric encryption keys used to protect files, several file encryption programmes and storage providers use RSA. The symmetric key is encrypted using the recipient's RSA public key, making sure that only the intended recipient and the owner of the associated private key can decode and access the data.

CODE :

```
import java.io.*;
import java.math.*;
import java.util.*;

public class rsa {
    public static double gcd(double a, double h)
    {
        /*
        * This function returns the gcd or greatest common
        * divisor
        */
        double temp;
        while (true) {
            temp = a % h;
            if (temp == 0)
                return h;
            a = h;
            h = temp;
        }
    }
    public static void main(String[] args)
    {
```

```
        double p = 3;
        double q = 7;

        // Stores the first part of public key:
        double n = p * q;

        // Finding the other part of public key.
        // double e stands for encrypt
        double e = 2;
        double phi = (p - 1) * (q - 1);
        while (e < phi) {
            /*
            * e must be co-prime to phi and
            * smaller than phi.
            */
            if (gcd(e, phi) == 1)
                break;
            else
                e++;
        }
        int k = 2; // A constant value
        double d = (1 + (k * phi)) / e;

        // Message to be encrypted
        double msg = 12;

        System.out.println("Message data = " + msg);

        // Encryption c = (msg ^ e) % n
        double c = Math.pow(msg, e);
        c = c % n;
        System.out.println("Encrypted data = " + c);

        // Decryption m = (c ^ d) % n
        double m = Math.pow(c, d);
        m = m % n;
        System.out.println("Original Message Sent = " + m);
    }
}
```

OUTPUT:

```
Message data = 12.0
Encrypted data = 3.0
Original Message Sent = 12.0
```

SECURITY ANALYSIS

1. Key Size: The size of the used keys has a significant impact on the security of RSA. The encryption is more secure the larger the key size. However, the real-world security of decreasing key sizes has been compromised by improvements in computing power and factoring techniques. Therefore, suggested key sizes have grown over time. Key sizes of 2048 bits or more are now regarded as secure for the majority of applications, whereas key sizes of 3072 bits or more are advised for long-term security.

2. The difficulty of factoring large integers into their prime factors affects the security of RSA algorithms. Although there are no effective traditional algorithms for factoring huge numbers, the study of factoring has advanced significantly. Large numbers can be factored effectively by specialised factoring algorithms like General Number Field Sieve (GNFS), but their computational complexity is still quite significant.

3. RSA is susceptible to attacks from quantum systems that are capable of performing efficient integer factorization. Large numbers can be factored exponentially more quickly using Shor's method, a quantum algorithm. The danger to RSA encryption increases with the development of quantum computing technology. In order to maintain secure communication in the era of quantum computing, post-quantum cryptography is currently being actively investigated and developed.

4. Side-Channel Attacks: Attackers may take advantage of information leakage via unintended channels like timing, power usage, or electromagnetic emissions to compromise RSA systems. These assaults can make the secret key or other private information visible. To reduce side-channel vulnerabilities, countermeasures like secure implementation strategies, constant-time algorithms, and hardware safeguards are needed.

5. In order to thwart assaults like the padding oracle attack developed by Bleichenbacher, RSA encryption relies on secure padding techniques. Implementations of padding that are inadequate or unsafe might result in flaws and decryption attacks. To guarantee the security of RSA encryption, proper padding schemes and implementation techniques, such as OAEP (Optimal Asymmetric Encryption Padding), are advised.

6. Key administration and Trust: The safe generation, storage, and administration of the private keys are essential to the security of RSA encryption. Data encryption can be broken through the compromising of a private key. To verify the legitimacy and integrity of public keys used in RSA encryption, trust in the public key infrastructure (PKI) is crucial.

HASH FUNCTIONS

SHA256

ALGORITHM AND WORKING

The SHA-2 family of cryptographic hash functions includes the SHA-256 (Secure Hash Algorithm 256-bit) function. The output is a fixed-size 256-bit hash value, which is a string of 64 hexadecimal characters. It typically accepts an input, usually a message or some data. The hash value is specific to the input data, therefore even a small change will result in a noticeably different hash.

1. Padding: The input data is padded to ensure it meets the required length for processing. The padding ensures the input size is a multiple of the block size used by the algorithm.
2. Initialization: SHA-256 uses an initial set of constant values known as the "initial hash values" (H) and a set of predefined "round constants" (K).
3. Message Processing: The input data is processed in fixed-size blocks (512 bits each) and divided into 16 32-bit words for further processing.
4. Compression: Each block goes through a series of hash computations called "rounds" (64 rounds for SHA-256). The algorithm operates on a "message schedule" (W) that is derived from the input block and previously processed blocks.
5. Round Operations: Each round involves several operations, including logical functions (bitwise operations such as AND, OR, XOR), bitwise rotations, and addition modulo 2^32. These operations modify the hash values from the previous round and update them with new values.
6. Final Hash: After all the rounds are completed for the last block, the resulting hash values are combined to produce the final hash. This final hash is the output of the SHA-256 algorithm.

STRENGTH

1. Collision Resistance: One of SHA-256's main advantages is that it can withstand collision assaults. When two distinct inputs result in the same hash output, a collision happens. It is theoretically impossible to locate a collision because doing so would demand an enormous amount of computing power.

2. Deterministic and Reliable: SHA-256 guarantees determinism and reliability by consistently producing the same output hash for the same input data. For activities like password storage, where user input can be verified by comparing hash values, this attribute is essential.

3. No matter the size of the input, SHA-256 always produces a fixed-size 256-bit hash output. Given that they always have a set length, hash values can be effectively stored and transmitted thanks to this constant size.

4. Widely Supported and Used: SHA-256 is widely used in a variety of cryptographic frameworks, tools, and applications. It is compatible with various platforms and systems thanks to its widespread use and standardisation.

5. Efficiency and Speed: The SHA-256 algorithm is built to strike a balance between security and computing speed. It is suitable for real-time applications and systems with high data throughput requirements because it can quickly handle massive amounts of data.

6. Security through cryptography: SHA-256 is thought to be cryptographically safe and offers a high level of protection against unauthorised data manipulation or change. Pre-image attacks cannot be used against it, making it computationally impossible to reconstruct the original input from the hash result.

7. Versatility: SHA-256 is a versatile hash function that may be used for a variety of cryptographic tasks, such as data integrity checks, digital signatures, password storage (in hashed form), and blockchain technologies like Bitcoin, where it serves as the underlying hash function.

WEAKNESS

1. Length Extension Attacks: Length extension attacks can be used against SHA-256. In such attacks, a malicious party that is aware of a message's hash can add data to it without being aware of the original message, producing a legitimate hash for the expanded message. Utilising HMAC constructs or other secure protocols with extra security measures can help to mitigate this flaw.

2. Although it is not specific to SHA-256, the development of potent quantum computers in the future may jeopardise the security of many cryptographic algorithms, including SHA-256. It may be possible to break cryptographic techniques based on the standard computing paradigm with quantum computers because they may be able to solve some mathematical problems more quickly than conventional computers. Post-quantum cryptography techniques are being created to overcome this issue.

3. Limited Input Size: The SHA-256 algorithm only processes data in 512-bit fixed-size blocks. The input data must be broken into numerous blocks for processing if it exceeds this block size. But if not handled correctly, this division process could result in new complications, weaknesses, or attack routes.

4. Deterministic Output: While SHA-256's deterministic nature is often a benefit, it can occasionally be a drawback. An attacker with access to precomputed hash tables (rainbow tables), for instance, can readily compare stored hash values to a precomputed set of hashes when using SHA-256 to store passwords, possibly jeopardising the security of poorly hashed passwords.

5. Dependence on Hash Functions: The security of SHA-256 is dependent on the characteristics and safety of the underlying hash functions. The security of SHA-256 can be weakened if any flaws or vulnerabilities are found in these supporting operations.

REAL WORLD EXAMPLES

1. Length Extension Attacks: SHA-256 is susceptible to length extension attacks. Such attacks allow a malevolent person who is aware of a message's hash to add information to it without being aware of the original message, resulting in a valid hash for the extended message. This issue can be reduced by using HMAC constructions or other safe protocols with additional security protections.

2. The creation of strong quantum computers may compromise the security of several cryptographic algorithms, including SHA-256, while this threat is not specific to SHA-256. Quantum computers may be able to answer some mathematical problems more quickly than conventional computers, making it possible to break encryption approaches based on the traditional computing paradigm. To solve this problem, post-quantum cryptography methods are being developed.

3. SHA-256 only analyses data in 512-bit fixed-size blocks due to its limited input size. If the input data is larger than this block size, it must be divided into many blocks before processing. But if not done properly, this division process may produce new difficulties, flaws, or attack points.

4. Deterministic Output: Although the deterministic nature of SHA-256 is frequently advantageous, it can occasionally be a disadvantage. When employing SHA-256 to store passwords, an attacker having access to precomputed hash tables (rainbow tables), for example, might quickly compare stored hash values to a precomputed set of hashes, potentially jeopardising the security of badly hashed passwords.

5. Dependence on Hash Functions: The properties and security of the underlying hash functions have an impact on the security of SHA-256. If any weaknesses or vulnerabilities are discovered in these supporting operations, the security of SHA-256 may be compromised.

CODE:

```
import java.math.BigInteger;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

// Java program to calculate SHA hash value

class sha256 {
    public static byte[] getSHA(String input) throws NoSuchAlgorithmException
    {
        MessageDigest md = MessageDigest.getInstance("SHA-256");

        // digest() method called
        // to calculate message digest of an input
        // and return array of byte
        return md.digest(input.getBytes(StandardCharsets.UTF_8));
    }

    public static String toHexString(byte[] hash)
    {
        // Convert byte array into signum representation
        BigInteger number = new BigInteger(1, hash);

        // Convert message digest into hex value
        StringBuilder hexString = new StringBuilder(number.toString(16));

        // Pad with leading zeros
        while (hexString.length() < 64)
        {
            hexString.insert(0, '0');
        }

        return hexString.toString();
    }

    // Driver code
    public static void main(String args[])
    {
```

```
        try
        {
            System.out.println("HashCode Generated by SHA-256 for:");

            String s2 = "welcome to ehical hacking cyber security";
            System.out.println("\n" + s2 + " : " + toHexString(getSHA(s2)));
        }
        // For specifying wrong message digest algorithms
        catch (NoSuchAlgorithmException e) {
            System.out.println("Exception thrown for incorrect algorithm: " + e);
        }
    }
}
```

OUTPUT:

```
welcome to ehical hacking cyber security : c5cbf10481dd296de92ac9c5d39a7c6fe9d28e3d1fa06dec2320e5bc53a36c69
```

SECURITY ANALYSIS

1. Collision Resistance: A hash function's main security feature is collision resistance, which states that it should be computationally impossible to identify two separate inputs that yield the same hash output. According to the security analysis of SHA-256, it has a high level of collision resistance. The theoretical security strength of SHA-256 is predicated on the birthday paradox, which states that establishing a collision would require an astronomically large number of computations, despite the fact that no practical collisions have been discovered.

2. Pre-image Resistance: This characteristic, which states that it should be computationally impossible to infer the original input data from its hash value, is another crucial one. There have been numerous analyses of SHA-256, but no real-world attacks have been discovered that can undo the hash function and recover the original input from the hash value.

3. Resistance to Differential and Linear Cryptanalysis: SHA-256 has been developed to withstand the powerful cryptanalytic methods used to assess

the security of cryptographic algorithms, differential and linear cryptanalysis. There are no obvious holes in SHA-256's defence against these kinds of attacks, according to study.

4. Bit Independence: The hash algorithm SHA-256 possesses the property of bit independence, which states that altering a single bit in the input should result in an output that appears unrelated to the original hash value. Small changes in the input are certain to cause big changes in the output thanks to this property.

5. Length Extension Attacks: SHA-256 is susceptible to length extension attacks, as was already mentioned. These attacks use the algorithm's deterministic nature to add more data to a given hash value. However, by including extra precautions like HMAC structures, this flaw can be reduced.

6. Cryptanalysis and Security Research: The cryptography community has conducted in-depth cryptanalysis and security research on SHA-256. Despite the fact that no real-world assaults have been identified that seriously jeopardise SHA-256's security, continued research is necessary to identify any potential flaws or fresh threats.