Network security and Cryptography

UNIT 03

Asymmetric key cryptography / Public key cryptography:

Asymmetric encryption, also known as **public-key cryptography**, is a type of encryption that uses a pair of keys to encrypt and decrypt data. The pair of keys includes a public key, which can be shared with anyone, and a private key, which is kept secret by the owner.

In asymmetric encryption, the sender uses the recipient's public key to encrypt the data. The recipient then uses their private key to decrypt the data. This approach allows for secure communication between two parties without the need for both parties to have the same secret key. Asymmetric encryption has several advantages over symmetric encryption, which uses the same key for both encryption and decryption. One of the main advantages is that it eliminates the need to exchange secret keys, which can be a challenging process, especially when communicating with multiple parties.

Additionally, asymmetric encryption allows for the creation of digital signatures, which can be used to verify the authenticity of data. Asymmetric encryption is commonly used in various applications, including secure online communication, digital signatures, and secure data transfer. Examples of asymmetric encryption algorithms include RSA, Diffie-Hellman, and Elliptic Curve Cryptography (ECC). In asymmetric key cryptography, the private key is a separate key from the public key that is kept private by the owner of the public key while the public key is made available to everyone. Anyone can encrypt a message using the public key, but only the holder of the private key can unlock it. With no chance of the communication being intercepted and read by a third party, anyone can send a secure message to the public key's owner.

Features of Asymmetric key cryptography:

The Main Features of Asymmetric Encryption (also known as public-key cryptography) are:

Dual keys: Asymmetric encryption uses a pair of keys, including a public key and a private key. The public key can be freely shared with anyone, while the private key is kept secret and known only to the key owner.

Encryption and decryption: Asymmetric encryption uses the public key to encrypt data and the private key to decrypt data. This allows secure communication between two parties without the need to exchange secret keys.

Digital signatures: Asymmetric encryption enables the creation of digital signatures, which can be used to verify the authenticity of data. A digital signature is created by encrypting a hash of the data with the sender's private key.

Security: Asymmetric encryption is considered more secure than symmetric encryption because it eliminates the need to exchange secret keys, which can be a security risk. Additionally, the private key is kept secret, which makes it harder for attackers to intercept or tamper with the data.

Slow processing: Asymmetric encryption is slower than symmetric encryption because it involves more complex mathematical operations. This can make it less suitable for applications that require fast data processing.

Advantages of Asymmetric key cryptography:

Enhanced Security: Asymmetric encryption provides a higher level of security compared to symmetric encryption where only one key is used for both encryption and decryption with asymmetric encryption a different key is used for each process and the private key used for decryption is kept secret by the receiver making, it harder for an attacker to intercept and decrypt the data.

Authentication: Asymmetric encryption can be used for authentication purposes which means that the receiver can verify the sender s identity. This is achieved by the sender encrypting a message with their private key which can only be decrypted with their public key if the receiver can successfully decrypt the message, it proves that it was sent by the sender who has the corresponding private key.

Non-repudiation: Asymmetric encryption also provides non-repudiation which means that the sender cannot deny sending a message or altering its contents this is because the message is encrypted with the sender s private key and only their public key can decrypt it. Therefore, the receiver can be sure that the message was sent by the sender and has not been tampered with.

Key distribution: Asymmetric encryption eliminates the need for a secure key distribution system that is required in symmetric encryption with symmetric encryption, the same key is used for both encryption and decryption and the key needs to be securely shared between the sender and the receiver asymmetric encryption, on the other hand, allows the public key to be shared openly and the private key is kept secret by the receiver.

Versatility: Asymmetric encryption can be used for a wide range of applications including secure email communication online banking transactions and e-commerce it is also used to secure SSL/TSL connections which are commonly used to secure internet traffic.

Disadvantages of asymmetric key cryptography:

Slower Performance: Asymmetric algorithms, such as RSA, are computationally intensive and much slower compared to symmetric key cryptography. This can lead to inefficiencies when handling large amounts of data.

Larger Key Sizes: Asymmetric cryptography requires much larger key sizes to achieve the same level of security as symmetric algorithms. For example, a 2048-bit RSA key offers roughly the same security as a 128-bit symmetric key.

Key Management Complexity: Asymmetric cryptography involves two keys (public and private). Ensuring secure distribution and management of these keys can be complex, especially in large-scale systems. Public keys must be distributed securely, and private keys must be stored safely to avoid compromise.

Vulnerable to Man-in-the-Middle Attacks: If the authenticity of a public key is not verified, an attacker could provide a forged public key, leading to potential man-in-the-middle attacks. Public key infrastructure (PKI) is needed to validate keys, adding complexity.

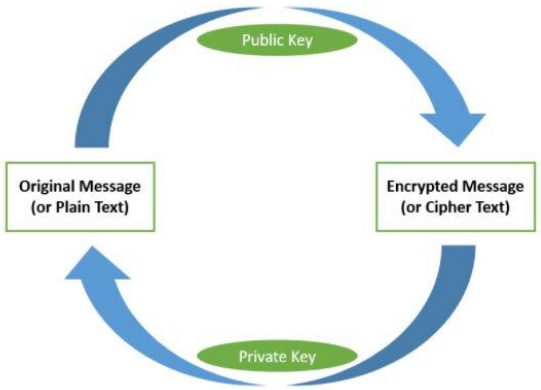
More Resource-Intensive: Asymmetric encryption algorithms require more computational resources (CPU power and memory), making them unsuitable for environments with limited processing capabilities, like embedded systems or mobile devices.

For example: In asymmetric encryption, Alice wants to send a secure message to Bob. She uses Bob's public key to encrypt the message, and only Bob can decrypt it using his private key. This ensures that even if someone intercepts the message, they can't read it without Bob's private key.

The RSA algorithm:

RSA algorithm is an asymmetric cryptography algorithm, named after its inventors **Rivest, Shamir, and Adelman (RSA).**

You may wonder what i mean by saying the RSA algorithm is asymmetric, it 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.



How does the RSA algorithm work?

Below is the entire process how RSA works, starting from creating the key pair, to encrypting and decrypting the information.

Key Generation

Step 1: Choose two large prime numbers **p** and **q**, and then calculating their product **N**, as shown:

$$N=p*q$$

Here, 'n' is the length of the key.

Step 2: Compute Euler's Totient function Z, which represents the number of integers less than n that are coprime with n:

$$f(n)=(p-1)\times(q-1)$$

This step is crucial for calculating the private key.

Step 3: Choose a public exponent e, which is typically a small odd number that is coprime with f(n). Ensure e satisfies:

$$1<e<f(n)$$

Step 4: Calculate $e\times d = 1 \bmod (p-1)(q-1)$ or $e\times d = 1 \bmod ((f)n)$

- You can bundle private key pair as (n,d)
- You can bundle public key pair as (n,e)

2. **Encryption:**

Consider a sender who sends the plain text message to someone whose public key is **(n,e).** To encrypt the plain text message in the given scenario, use the following syntax –

If the plaintext is P then ciphertext C is:

C = Pe mod n

This step turns the original message into an unreadable ciphertext.

3. **Decryption**

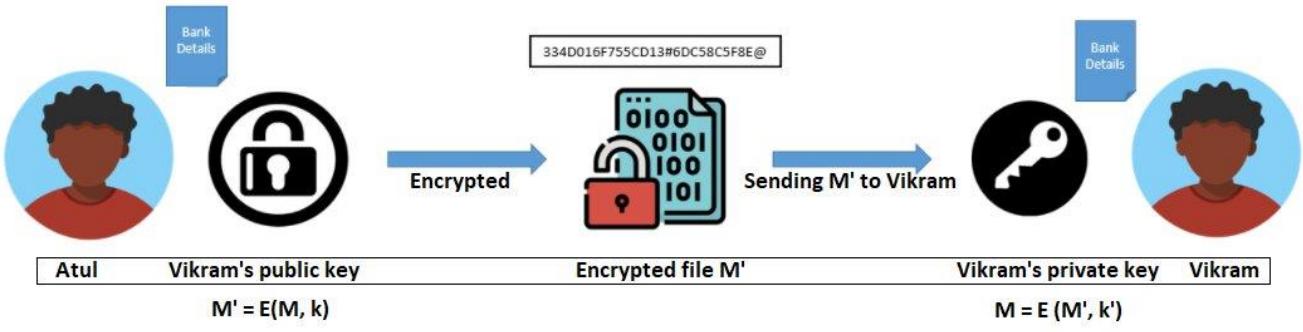
Considering receiver **C** has the private key **d**, the result modulus will be calculated as –

Plaintext = Cd mod n

Example of RSA algorithm

Suppose that Atul wants to send his bank details to Vikram. So, is there any way to send the bank details on the same public unencrypted network ensuring only Vikram can read it?

For Vikram to receive Atul's bank details, Atul needs Vikram's Public key. Using this, he can encrypt his message and send it. Vikram, who carries his private key, can decrypt the message sent by Atul. It's a one-way process of sending messages to the intended recipient.



Here,

- **M** denotes an original message
- **E** denotes encryption function
- **k** denotes Vikram's public key
- **M'** denotes an encrypted message
- **k'** denotes Vikram's private key
- **D** denotes the decryption function

Advantages of RSA algorithm:

- **Security**: RSA is considered very secure and reliable for protecting sensitive data.
- **Public-key cryptography**: Uses a pair of keys – public for encryption and private for decryption, enhancing security without sharing secret keys.
- **Key exchange**: Enables secure exchange of secret keys between parties without transmitting the key directly.
- **Digital signatures**: Allows verification of message authenticity by signing with a private key and verifying with a public key.
- **Widely used**: Popular in secure communications, online banking, and e-commerce due to its robustness.

Disadvantages of RSA algorithm:

- **Slow processing**: RSA is slower compared to other algorithms (e.g., symmetric encryption) when handling large volumes of data.
- **Large key size**: Requires longer key lengths (2048+ bits) for stronger security, consuming more computational power and storage.
- **Vulnerable to side-channel attacks**: Leaks via power consumption, timing, or electromagnetic radiation can expose the private key.
- **Complex key management**: Ensuring the secure handling of private keys can be challenging and critical.
- **Not ideal for large data**: Its slower speed makes it unsuitable for applications requiring frequent encryption/decryption of large datasets.

ElGamal cryptography:

The ElGamal cryptographic algorithm is an asymmetric key encryption algorithm based on the **Diffie-Hellman key exchange**. It was invented by **Taher ElGamal** in **1985**. The algorithm is widely used for secure data transmission and has applications in digital signatures and encryption. The security of the ElGamal algorithm relies on the difficulty of solving the **Discrete Logarithm Problem** in a finite field, which is computationally hard to reverse.

It has two primary uses:

1. **Encryption**: ElGamal is used for encrypting messages where public key cryptography is required.
2. **Digital Signatures**: A variant of ElGamal is used for creating digital signatures, ensuring message authenticity and integrity.

How ElGamal Works:

1. Key Generation:

Select a large prime number **p** and a generator **g**, where g is a primitive root modulo p.
Private Key: Select a private key **x** randomly such that $\mathbf{1 \leq x \leq p-2}$.
Public Key: Compute the public key **y** as:

$$\mathbf{y=g^x \bmod p}$$

The public key is (p,g,y) and the private key is x.

2. Encryption:

- **Step 1**: Bob (the sender) wants to send a message m to Alice. He first converts m into a number m such that $0 \leq m \leq p-1$
- **Step 2**: Bob randomly selects an ephemeral key k, where $1 \leq k \leq p-2$, and computes: $c_1 = g^k \bmod p$
- **Step 3**: Bob computes the shared secret using Alice's public key y: $s = y^k \bmod p$
- **Step 4**: Bob encrypts the message by calculating: $c_2 = (m \cdot s) \bmod p$
- **Step 5**: The ciphertext sent to Alice is (c_1, c_2).

3. **Decryption:**

- **Step 1**: Alice receives the ciphertext (c_1, c_2).
- **Step 2**: She computes the shared secret using her private key x: $s = c_1^x \bmod p$
- **Step 3**: Alice recovers the original message by calculating: $m = (c_2 \cdot s^{-1}) \bmod$.

  Here, s^{-1} is the modular inverse of s mod p, which can be computed using the extended Euclidean algorithm.

What is a digital signature?

A digital signature is a mathematical technique used to validate the authenticity and integrity of a digital document, message or software. It's the digital equivalent of a handwritten signature or stamped seal, but it offers far more inherent security. A digital signature is intended to solve the problem of tampering and impersonation in digital communications.

Digital signatures can provide evidence of origin, identity and status of electronic documents, transactions and digital messages. Signers can also use them to acknowledge informed consent.

Digital signature attacks:

Possible attacks on digital signatures include the following:

- **Chosen-message attack.** The attacker either obtains the victim's public key or tricks the victim into digitally signing a document they don't intend to sign.
- **Known-message attack.** The attacker obtains messages the victim sent and a key that enables the attacker to forge the victim's signature on documents.
- **Key-only attack.** The attacker has access to the victim's public key and re-creates the victim's signature to digitally sign documents or messages that the victim doesn't intend to sign.

- **Malware Attack**: Malware captures the private key or tampers with the signing process.
- **Insider Attack**: A trusted insider misuses access to create fraudulent signatures.

How digital signatures work:

Digital signatures are created and verified by using public key cryptography, also known as asymmetric cryptography. By the use of a public key algorithm, such as RSA, one can generate two keys that are mathematically linked- one is a private key, and another is a public key.

The user who is creating the digital signature uses their own private key to encrypt the signature-related document. The only way to decrypt that document is with the use of signer's public key.

This technology requires all the parties to trust that the individual who creates the signature has been able to keep their private key secret. If someone has access the signer's private key, there is a possibility that they could create fraudulent signatures in the name of the private key holder.

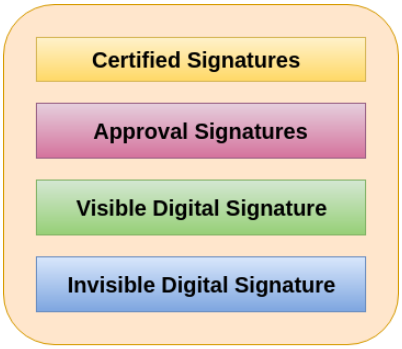
The steps which are followed in creating a digital signature are:

1. **Key Generation**: Each user generates a pair of keys — a **private key** (kept secret) and a **public key** (shared with others).
2. **Signing**:
    - The document is passed through a **hash function**, creating a unique hash (or message digest) representing the document.
    - The hash is then encrypted using the sender's **private key**, creating the **digital signature**.
3. **Verification**:
    - The recipient uses the sender’s **public key** to decrypt the digital signature, retrieving the original hash.
    - The recipient re-hashes the original document using the same hash function.

- If the decrypted hash matches the newly computed one, the signature is valid, proving the document is authentic, unaltered, and signed by the owner of the private key.

Types of Digital Signature:

They are described below:



Types of Digital Signature

Certified Signatures

The certified digital signature documents display a unique blue ribbon across the top of the document. The certified signature contains the name of the document signer and the certificate issuer which indicate the authorship and authenticity of the document.

Approval Signatures

The approval digital signatures on a document can be used in the organization's business workflow. They help to optimize the organization's approval procedure. The procedure involves capturing approvals made by us and other individuals and embedding them within the PDF document. The approval signatures to include details such as an image of our physical signature, location, date, and official seal.

Visible Digital Signature

The visible digital signature allows a user to sign a single document digitally. This signature appears on a document in the same way as signatures are signed on a physical document.

Invisible Digital Signature

The invisible digital signatures carry a visual indication of a blue ribbon within a document in the taskbar. We can use invisible digital signatures when we do not have or do not want to display our signature but need to provide the authenticity of the document, its integrity, and its origin.

Application of Digital Signature:

The important reason to implement digital signature to communication is:

- Authentication
- Non-repudiation
- Integrity

Authentication

Authentication is a process which verifies the identity of a user who wants to access the system. In the digital signature, authentication helps to authenticate the sources of messages.

Non-repudiation

Non-repudiation means assurance of something that cannot be denied. It ensures that someone to a contract or communication cannot later deny the authenticity of their signature on a document or in a file or the sending of a message that they originated.

Integrity

Integrity ensures that the message is real, accurate and safeguards from unauthorized user modification during the transmission.

Advantages of Digital Signatures:

1. **Authentication**: Digital signatures verify the identity of the sender, ensuring the message comes from the correct source.
2. **Data Integrity**: They ensure the data hasn't been altered in transit, as even minor changes to the document will result in a mismatched hash during verification.
3. **Non-repudiation**: The signer cannot deny having signed the document since the signature is uniquely tied to the sender's private key.
4. **Security**: Digital signatures use strong encryption techniques, making it extremely difficult for unauthorized parties to forge a signature or tamper with the document.
5. **Efficiency**: They reduce the need for physical paperwork, enabling fast and secure document signing, saving time and resources in workflows like legal contracts, financial transactions, and more.
6. **Environmentally Friendly**: By eliminating the need for paper-based signatures, they contribute to reduced paper waste.

Disadvantages of Digital Signatures:

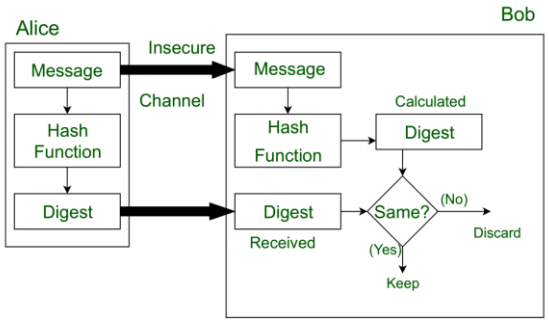
1. **Cost**: Implementing digital signature technology can be expensive due to the need for specialized software, hardware, and certificate authorities (CAs) to issue and manage keys.
2. **Complexity**: The underlying technology can be complex for non-technical users, leading to possible user error or misunderstanding of the system.
3. **Private Key Security**: If a user's private key is compromised (stolen or lost), the security of the digital signature is breached. Maintaining the security of private keys is crucial.
4. **Trust in Certification Authorities (CAs)**: Digital signatures rely on trusted third parties (CAs) to issue digital certificates. If a CA is compromised or untrustworthy, the digital signatures it verifies could be invalid.
5. **Technological Dependence**: Digital signatures rely on technology and internet access. In regions with limited infrastructure, implementation and verification can be problematic.

Message Digest:

A message digest or hash value is a numeric string generated using the cryptographic hash function. The message is passed to the hash function. This Hash function computes a unique hash value for the provided message and this hash value acts as digital fingerprint of the message.

Message Digest is used to ensure the integrity of a message transmitted over an insecure channel (where the content of the message can be changed).

Let's assume, Alice sent a message and digest pair to Bob. To check the integrity of the message Bob runs the cryptographic hash function on the received message and gets a new digest. Now, Bob will compare the new digest and the digest sent by Alice. If, both are same then Bob is sure that the original message is not changed.



Properties of a Message Digest:

- **Hash function:** The message digest is always a unique numeric hash value. It cannot be the same for two or more messages. Once the message digest value is generated for a message, it's only associated with that message.

- **Fixed Size**: Regardless of the size of the input data, the digest will always be of a fixed length. For any length of message a fixed length hash value is generated and that depends on the implementation of the hash function.

- **Unique Output**: If you use the hash function multiple times for the same function again and again, then you only get the same digest value each time. Ideally, even a small change in the input will produce a significantly different digest. This property is known as the **avalanche effect.** For example, if you send "hii" to the hash function the generated message

digest value will be the same each time when you pass “hii” to the hash function. It’s not going to change.

- **Non-Reversible**: We cannot get the original message back from the message digest because it is generated using the one-way function.

Uses of message digest:

1. **Data Integrity**: Message digests are commonly used to verify the integrity of data. By comparing the digest of the received data with the digest of the original data, you can detect any changes or corruption.
2. **Digital Signatures**: In digital signatures, a hash of the message is signed with a private key. The recipient can use the corresponding public key to verify the signature, ensuring both the authenticity and integrity of the message.
3. **Password Storage**: Instead of storing plain-text passwords, systems often store the hash of the password. During login, the system hashes the entered password and compares it with the stored hash.
4. **Data Deduplication**: Hashes can be used to identify duplicate data by comparing the digests of different files or data blocks.
5. **Efficient Data Retrieval**: In data structures like hash tables, message digests are used to quickly locate and retrieve data.

Message Authentication Code (MAC):

Message Authentication Code (MAC), also referred to as a tag, is a technique used to verify the integrity and authenticity of a message. It ensures that the message has not been altered and that it indeed comes from a legitimate source. This is typically achieved using a cryptographic algorithm that generates a unique code or tag for the message, known as a Message Authentication Code (MAC).

MAC is used to authenticate the origin and nature of a message. In other words, MAC ensures that the message is coming from the correct sender, has not been changed, and that the data transferred over a network or stored in or outside a

system is legitimate and does not contain harmful code. MACs can be stored on a hardware security module, a device used to manage sensitive digital keys.

How it works:

Steps in MAC Operation:

1. **Key Sharing:**
    - The sender and receiver must share a **secret key**, which is agreed upon beforehand. This key is known only to both parties and is used to generate and verify the MAC.
2. **Message Input:**
    - The sender wants to send a message, which can be of arbitrary length (e.g., a file, email, or data packet).
3. **MAC Generation:**
    - The sender uses a **MAC generation algorithm** that takes the message and the secret key as input.
    - The algorithm outputs a fixed-size MAC, which is a cryptographic hash or code that is unique to the message and the:

      MAC=MAC_function(message,key)

    - Even a small change in the message would result in a completely different MAC being generated.
4. **Message Transmission:**
    - The sender transmits both the original message and the generated MAC to the receiver.
5. **MAC Verification:**
    - Upon receiving the message and MAC, the receiver uses the same MAC generation algorithm, with the **same secret key**, to compute a new MAC based on the received message.
    - The receiver then compares this newly generated MAC with the received MAC.
    - If the two MACs match, the message is considered authentic (i.e., it was not altered in transit and was sent by the correct source). If the MACs don't match, it means the message might have been altered or tampered with.

HMAC:

HMAC (Hash-based Message Authentication Code) is a type of message authentication code (MAC) that is acquired by executing a cryptographic hash function on the data that is to be authenticated and a secret shared key. Like any of the MACs, it is used for both data integrity and authentication.

Digital signatures are nearly similar to HMACs i.e. they both employ a hash function and a shared key. The difference lies in the keys i.e. HMAC uses a symmetric key (same copy) while Signatures uses an asymmetric (two different keys).

How HMAC works:

HMAC provides a valid and reliable way for transacting parties to ensure that their messages have not been tampered by an unauthorized or malicious party, also referred to as a **threat actor**.

The HMAC code or key consists of two parts:

1. A shared set of cryptographic keys for the sender (client) and recipient (server). The sender and recipient use the same key to generate and verify the HMAC.
2. A generic cryptographic hash function, like SHA-1.

The formula for HMAC is represented as the following:

HMAC = hashFunc(secret key + message)

In a messaging transaction between a client and a server involving HMAC, the client creates a unique HMAC or hash by hashing the request data with the private keys and sending it as part of a request. The server receives the request and regenerates its own unique HMAC. It then compares the two HMACs. If they are equal, the client is trusted and considered legitimate, and the request is executed. This process is often called a **secret handshake**.

Advantages of HMAC

- Strong security due to the combination of a secret key and hash function, protecting against message tampering and forgery.
- HMACs are ideal for high-performance systems like routers due to the use of hash functions which are calculated and verified quickly unlike the public key systems.
- Digital signatures are larger than HMACs, yet the HMACs provide comparably higher security.
- HMACs are used in administrations where public key systems are prohibited.

Disadvantages of HMAC

- The security of HMAC relies on the strength of the chosen hash function (e.g., SHA-256). If the hash function is compromised, HMAC is also affected.
- HMACs uses shared key which may lead to non-repudiation. If either sender or receiver's key is compromised then it will be easy for attackers to create unauthorized messages.
- Securely managing and distributing secret keys can be challenging.
- Although unlikely, hash collisions (where two different messages produce the same hash) can occur.
- The security of HMAC depends on the length of the secret key. Short keys are more vulnerable to brute-force attacks.

Applications of HMAC

- Verification of e-mail address during activation or creation of an account.
- Authentication of form data that is sent to the client browser and then submitted back.
- HMACs can be used for Internet of things (IoT) due to less cost.
- Whenever there is a need to reset the password, a link that can be used once is sent without adding a server state.
- It can take a message of any length and convert it into a fixed-length message digest. That is even if you got a long message, the message digest will be small and thus permits maximizing bandwidth.

Explain secure hash algorithm? (16 marks)

Knapsack Encryption Algorithm is the first general public key cryptography algorithm. It was developed by **Ralph Merkle** and **Mertin Hellman** in 1978. As it is a Public key cryptography, it needs two different keys. One is the Public key which is used for the Encryption process and the other one is the Private key which is used for the Decryption process.

Example:

We will choose a **super-increasing problem**. Super increasing knapsack is a sequence in which every next term is greater than the sum of all preceding terms.

{1, 2, 4, 10, 20, 40} is a super increasing as

1<2, 1+2<4, 1+2+4<10, 1+2+4+10<20 and 1+2+4+10+20<40.

Derive the Public key:

Step 1: Choose a super increasing knapsack {1, 2, 4, 10, 20, 40} as the private key.

Step 2: Choose two numbers **n** and **m**. Multiply all the values of the private key by the number **n** and then find modulo **m**. The value of **m** must be greater than the sum of all values in the private key, for example, 110. The number n should have no common factor with **m**, for example, 31.

Step 3: Calculate the values of the Public key using **m** and **n**.

1x31 mod(110) = 31
2x31 mod(110) = 62
4x31 mod(110) = 14
10x31 mod(110) = 90
20x31 mod(110) = 70
40x31 mod(110) = 30

Thus, our public key is **{31, 62, 14, 90, 70, 30}**
And Private key is **{1, 2, 4, 10, 20, 40}**

Now take an example for understanding the process of encryption and decryption.

Example: Let our plain text be **100100111100101110**.

1. Encryption: As our knapsacks contain six values, so we will split our plain text into groups of six:

100100 111100 101110

Multiply each value of the public key with the corresponding values of each group and take their sum.

100100 {31, 62, 14, 90, 70, 30}
1x31+0x62+0x14+1x90+0x70+0x30 = **121**
111100 {31, 62, 14, 90, 70, 30}
1x31+1x62+1x14+1x90+0x70+0x30 = **197**
101110 {31, 62, 14, 90, 70, 30}
1x31+0x62+1x14+1x90+1x70+0x30 = **205**

So, our cipher text is **121 197 205**

2. Decryption: The receiver receives the cipher text which has to be decrypted. The receiver also knows the values of **m** and **n**.
So, first, we need to find the $\mathbf{n^{-1}}$, which is the multiplicative inverse of **n** mod **m** i.e.,

$$\mathbf{n \times n^{-1} \bmod(m) = 131 \times n^{-1} \bmod(110) = 1n^{-1} = 71}$$

Now, we have to multiply **71** with each block of cipher text and take modulo **m**.

121 x 71 mod(110) = 11

Then, we will have to make the sum of 11 from the values of private key {1, 2, 4, 10, 20, 40} i.e., 1+10=11 so make the corresponding bits 1 and others 0 which is 100100. Similarly,

197 x 71 mod(110) = **17**
1+2+4+10=17 = **111100**

And, 205 x 71 mod(110) = **35**
1+4+10+20=35 = **101110**

After combining them we get the decoded text.
100100111100101110 which is our plain text.

These notes are prepared by **Suhail Abass Hurrah** (S.P College Srinagar, batch 2019)
For any queries, you can email me at Hurrahsuhail1@gmail.com