Problem Statement -

In this project, you must write Python code for the following scenario. Alice has a top-secret document and wants to send it to her partner, Bob, who lives in a different country. Please write Python programs that allow Alice and Bob to transmit this top-secret document to achieve both confidentiality and integrity. You probably need to write two, one for Alice and the other for Bob.

Algo/specs used

RSA -

RSA (Rivest-Shamir-Adleman) is a public-key cryptosystem widely used for secure communication and digital signatures. It was invented by Ron Rivest, Adi Shamir, and Leonard Adleman in 1977 and is named after their last names. RSA is based on the mathematical difficulty of factoring large integers into their prime factors, making it difficult for an attacker to derive the private key from the public key.

RSA has many applications, including secure communication, digital signatures, and key exchange. It is used in many popular protocols, such as SSL/TLS, SSH, and PGP. RSA is also one of the most widely used public-key algorithms, along with Diffie-Hellman and elliptic curve cryptography.

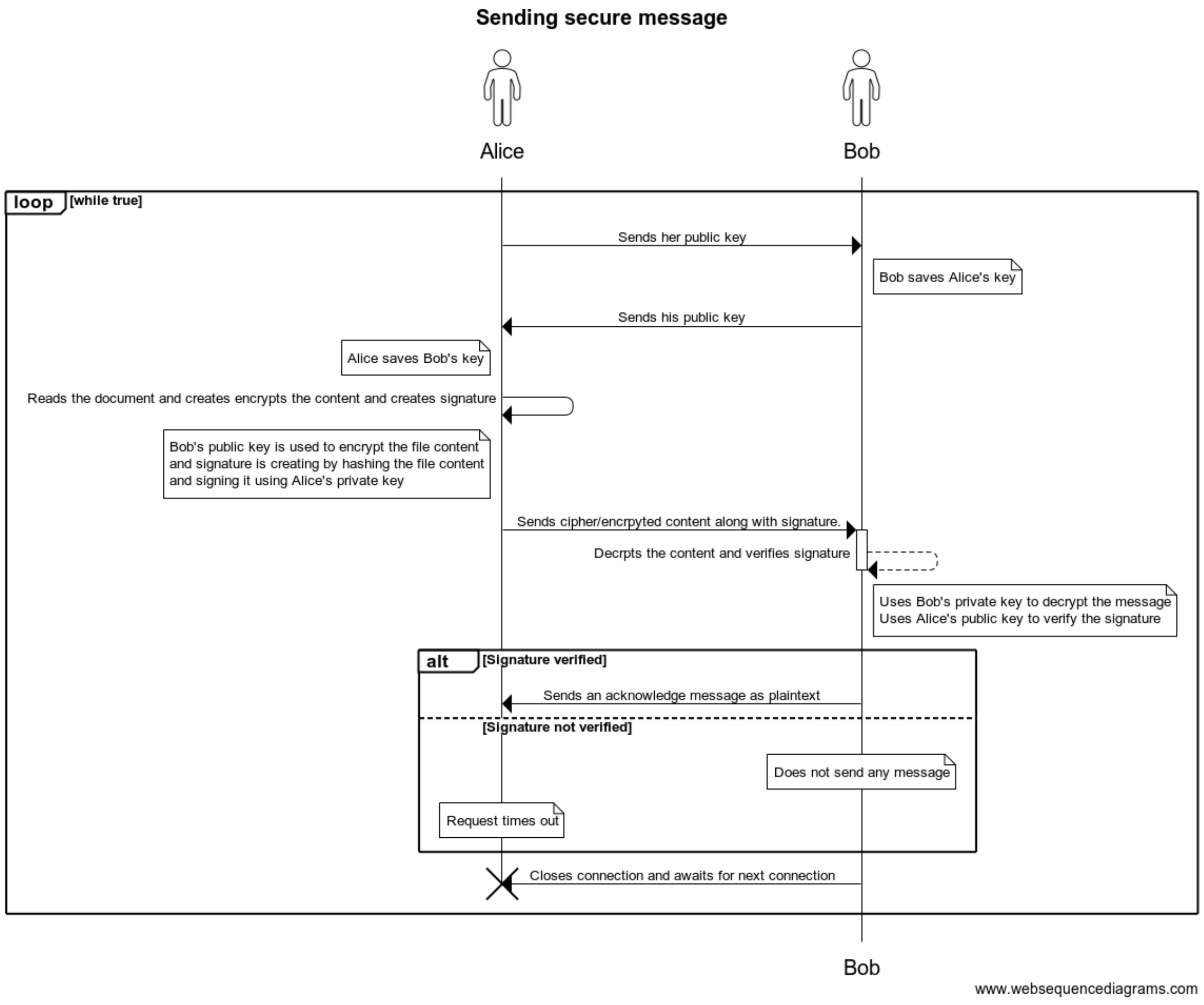
The RSA algorithm works by generating a pair of public and private keys. The public key is used for encryption, and the private key is used for decryption. To generate the keys, we first select two large prime numbers, p, and q. We then calculate n = pq, and we choose a number e that is relatively prime to (p-1)(q-1). The public key is the pair (n, e), and the private key is the pair (n, d), where d is the modular inverse of e modulo (p-1)*(q-1).

To encrypt a message using RSA, we first represent the message as a number m less than n. We then compute c = m^e mod n, where c is the encrypted message. To decrypt the message, we compute m = c^d mod n, where d is the private key.

RSA is a secure algorithm because the factorization of large integers is computationally difficult. It is based on the assumption that it is easier to multiply two large prime numbers than to factor the product of two large prime numbers. However, advances in computing power and factoring algorithms have led to development of more secure public-key cryptosystems, such as elliptic curve cryptography.

In conclusion, RSA is a widely used public-key cryptosystem based on the mathematical difficulty of factoring large integers. It has many applications, including secure communication, digital signatures, and key exchange. While RSA is a secure algorithm, advances in computing power and factoring algorithms have led to the development of more secure public-key cryptosystems.

Flow


Sending secure message
Alice
Bob
loop [while true]
Sends her public key
Bob saves Alice's key
Sends his public key
Alice saves Bob's key
Reads the document and creates encrypts the content and creates signature
Bob's public key is used to encrypt the file content and signature is creating by hashing the file content and signing it using Alice's private key
Sends cipher/encrpyted content along with signature.
Decrpts the content and verifies signature
Uses Bob's private key to decrypt the message
Uses Alice's public key to verify the signature
alt [Signature verified]
Sends an acknowledge message as plaintext
[Signature not verified]
Does not send any message
Request times out
Closes connection and awaits for next connection
Bob
www.websequencediagrams.com

Python Libraries:

We will be using the following libraries in Python for implementing the RSA algorithm.

1. Cryptodome
2. Socket
3. Scapy

Python Code:

We will write three Python programs, one each for Alice, Bob, and Trudy. Alice and Bob will use RSA encryption to transmit the top-secret document securely, while Trudy will act as a man-in-the-middle and try to intercept and alter the message.

Alice's program:

1. Alice generates a pair of public and private keys using the RSA algorithm. She keeps the private key secret and shares the public key with Bob.
2. Alice encrypts the top-secret document using Bob's public key and sends it to him.
3. Alice signs the encrypted document using her private key to ensure its integrity.
4. Alice creates a signature by hashing the file contents and her own private key.
5. Alice sends the signed document to Bob.

Bob's program:

1. Bob generates a pair of public and private keys using the RSA algorithm. He keeps the private key secret and shares the public key with Alice.
2. Bob receives the encrypted and signed document from Alice.
3. Bob decrypts the document using his private key.
4. Bob verifies the signature using Alice's public key to ensure the document's integrity.
5. Bob reads the decrypted document.
6. Bob sends an acknowledgment to Alice.

Trudy's program:

1. Trudy intercepts the encrypted and signed document sent by Alice.
2. Trudy decrypts the document using her private key and reads the content.
3. Trudy alters the document's content and signs it using her private key.
4. Trudy encrypts and sends the altered and signed document to Bob.

Scapy's program:

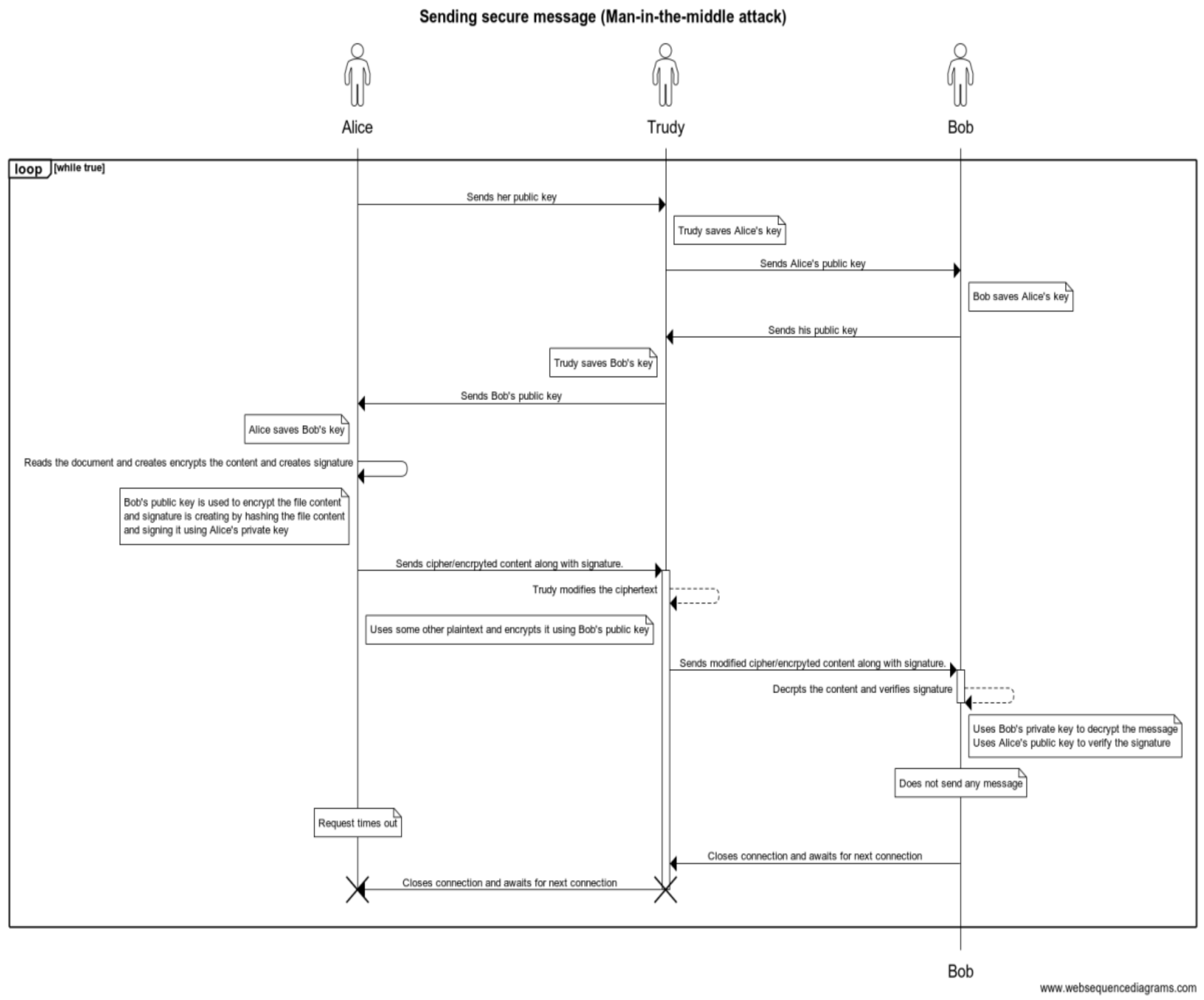
1. This program uses Scapy to sniff packets sent from Alice to Bob and prints out the source IP and message payload of each packet.
2. The program sets up a packet sniffer using the sniff() function from Scapy, which captures packets that match the specified filter.
3. The prn parameter of the sniff() function specifies the function to be called for each captured packet.
4. Overall, this program allows us to use Scapy to intercept and analyze the network traffic between Alice and Bob, providing valuable insights into the contents of the transmitted packets.

Hashing Algorithm:

We have used SHA-256(Secure Hash Algorithm 256-bit) for hashing the top_secret.txt file contents. SHA-256 is a cryptographic hash function that produces a 256-bit (32-byte) hash value. It is one of the most commonly used cryptographic hash functions in the world.
SHA-256 is a one-way function, which means that it takes an input (message or data) and produces a fixed-size output, which cannot be reversed to get the original input. It is designed to be a secure way to verify the integrity and authenticity of data, as even a small change in the input will produce a completely different output.
SHA-256 is used for a wide range of security applications, including digital signatures, password hashing, and file verification. It is also commonly used in blockchain technology to secure transactions and create unique identifiers for blocks.

Integrity Check:

To check the integrity, we have simulated a Man in The Middle Attack using Trudy as an attacker who tries to alter the contents of the top-secret document that is sent by Alice to Bob. In this scenario, Alice is unaware of the correct IP address and connects to Trudy(attacker) IP address. Trudy acts as the middleman in this case and forwards all the messages from Alice to Bob and vice versa. We have simulated this attack on the assumption that the public key is not compromised and that Trudy only wants to alter the contents of the text.
Once Alice sends the ciphertext to Trudy, thinking he's Bob, Trudy won't be able to decrypt the message, but he will send the altered text document to Bob.
Upon receiving the ciphertext, when Bob tries to verify the signature using Alice's public key, the verification will fail, and Bob will realize that the communication is compromised.
If no response is received from Bob, the socket will be timed out after 15 seconds, and Alice will either get no response or a warning stating that the message is compromised.


Sending secure message (Man-in-the-middle attack)
Alice
Trudy
Bob
loop [while true]
Sends her public key
Trudy saves Alice's key
Sends Alice's public key
Bob saves Alice's key
Sends his public key
Trudy saves Bob's key
Sends Bob's public key
Alice saves Bob's key
Reads the document and creates encrypts the content and creates signature
Bob's public key is used to encrypt the file content and signature is creating by hashing the file content and signing it using Alice's private key
Sends cipher/encrpyted content along with signature.
Trudy modifies the ciphertext
Uses some other plaintext and encrypts it using Bob's public key
Sends modified cipher/encrpyted content along with signature.
Decrpts the content and verifies signature
Uses Bob's private key to decrypt the message Uses Alice's public key to verify the signature
Does not send any message
Request times out
Closes connection and awaits for next connection
Closes connection and awaits for next connection
Bob
www.websequencediagrams.com

Confidentiality Check:

Our project is based on the assumption that the public key of Alice & Bob is not compromised. Without the public key, Trudy won't be able to decrypt the top secret document sent by Alice to Bob. This ensures that the message is confidential and not compromised.

Additional Scope

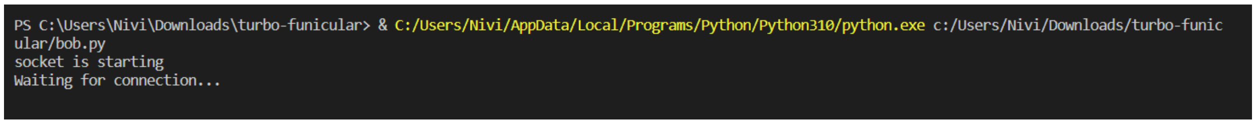
Packet Sniffing:

Packet sniffing is the process of intercepting and analyzing network traffic. In this project, we will be using Scapy to sniff packets sent between Alice and Bob to analyze the network traffic and understand the contents of the packets. We have tried to implement the Scapy module to sniff the packets sent by Alice to Bob. Scapy successfully sniffed and monitor the packets sent by Alice to Bob. Our project has further scope wherein scapy can manipulate the sniffed packages sent by Alice to Bob.

Output:

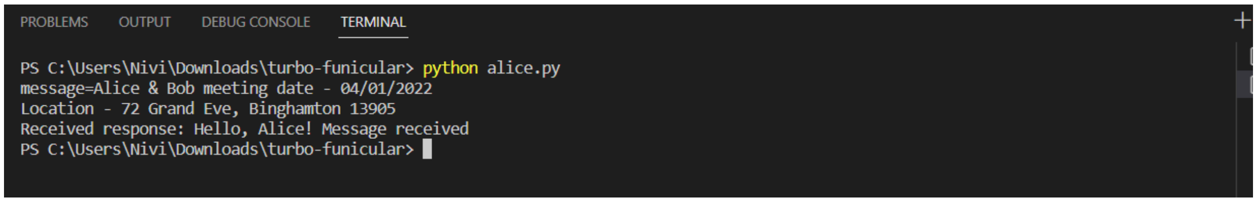
1. Alice sending secure message to Bob using RSA

Bob: Server socket starting, waiting for a connection



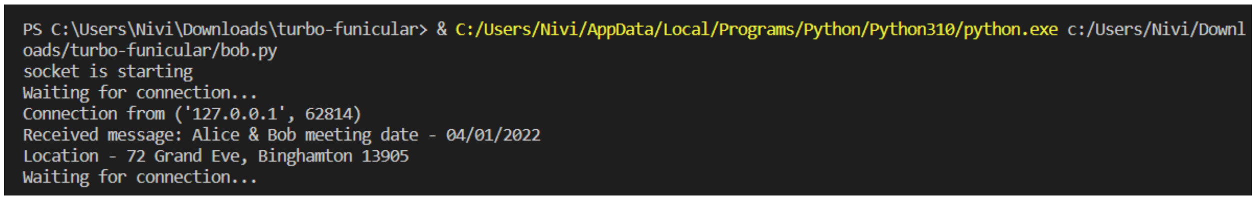
```
PS C:\Users\Nivi\Downloads\turbo-funicular> & C:/Users/Nivi/AppData/Local/Programs/Python/Python310/python.exe c:/Users/Nivi/Downloads/turbo-funic
ular/bob.py
socket is starting
Waiting for connection...
```

Alice: Reading the top_secret.txt file on her end and receiving acknowledgement from Bob



```
PS C:\Users\Nivi\Downloads\turbo-funicular> python alice.py
message=Alice & Bob meeting date - 04/01/2022
Location - 72 Grand Eve, Binghamton 13905
Received response: Hello, Alice! Message received
PS C:\Users\Nivi\Downloads\turbo-funicular>
```

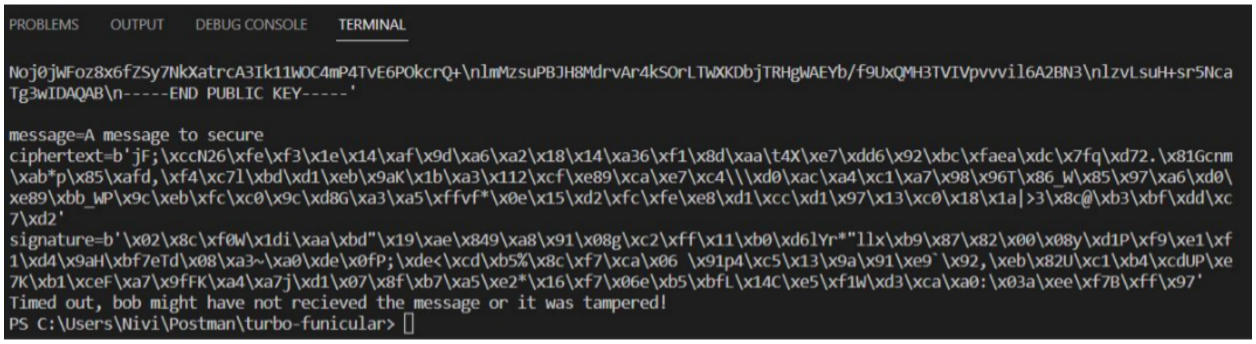
Bob: Decrypting the encrypted message on his end and reading the message contents



```
PS C:\Users\Nivi\Downloads\turbo-funicular> & C:/Users/Nivi/AppData/Local/Programs/Python/Python310/python.exe c:/Users/Nivi/Downl
oads/turbo-funicular/bob.py
socket is starting
Waiting for connection...
Connection from ('127.0.0.1', 62814)
Received message: Alice & Bob meeting date - 04/01/2022
Location - 72 Grand Eve, Binghamton 13905
Waiting for connection...
```

2. Alice sending a secure message to Bob (Man-in-the-middle attack)

   Alice: Receiving the error message that the message is compromised by Trudy



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Noj0jWFoz8x6fZSy7NkXatrcA3Ik11WOC4mP4TvE6POkcrQ+\nlmMzsuPBJH8MdrvAr4kSOrLTWXKDbjTRHgWAEYb/f9UxQMH3TVIVpvvvil6A2BN3\nlzvLsuH+sr5Nca
Tg3wIDAQAB\n-----END PUBLIC KEY-----'

message=A message to secure
ciphertext=b'jF;\xccN26\xfe\xf3\x1e\x14\xaf\x9d\xa6\xa2\x18\x14\xa36\xf1\x8d\xaa\t4X\xe7\xdd6\x92\xbc\xfaea\xdc\x7fq\xd72.\x81Gcnm
\xab*p\x85\xafd,\xf4\xc7l\xbd\xd1\xeb\x9aK\x1b\xa3\x112\xcf\xe89\xca\xe7\xc4\\\xd0\xac\xa4\xc1\xa7\x98\x96T\x86_W\x85\x97\xa6\xd0\
xe89\xbb_WP\x9c\xeb\xfc\xc0\x9c\xd8G\xa3\xa5\xffvf*\x0e\x15\xd2\xfc\xfe\xe8\xd1\xcc\xd1\x97\x13\xc0\x18\x1a|>3\x8c@\xb3\xbf\xdd\xc
7\xd2'
signature=b'\x02\x8c\xf0W\x1di\xaa\xbd"\x19\xae\x849\xa8\x91\x08g\xc2\xff\x11\xb0\xd6lYr*"llx\xb9\x87\x82\x00\x08y\xd1P\xf9\xe1\xf
1\xd4\x9aH\xbf7eTd\x08\xa3~\xa0\xde\x0fP;\xde<\xcd\xb5%\x8c\xf7\xca\x06 \x91p4\xc5\x13\x9a\x91\xe9`\x92,\xeb\x82U\xc1\xb4\xcdUP\xe
7K\xb1\xceF\xa7\x9fFK\xa4\xa7j\xd1\x07\x8f\xb7\xa5\xe2*\x16\xf7\x06e\xb5\xbfL\x14C\xe5\xf1W\xd3\xca\xa0:\x03a\xee\xf7B\xff\x97'
Timed out, bob might have not recieved the message or it was tampered!
PS C:\Users\Nivi\Postman\turbo-funicular> []
```

Conclusion:

In conclusion, we have successfully implemented the RSA algorithm using Python for secure document transmission between Alice and Bob. The algorithm ensures both the confidentiality and integrity of the transmitted document. Alice encrypts the document using Bob's public key, ensuring that only Bob can read the document. She signs the encrypted message using her private key, which provides integrity and authenticity to the document. Bob receives the encrypted message and signature from Alice, decrypts the message using his private key, and verifies the signature using Alice's public key.
However, we also considered the scenario of a man-in-the-middle attack where Trudy tries to intercept and alter the message. Trudy intercepts the encrypted and signed message from Alice, alter the content, and signs it using her private key. She then sends the altered message to Bob, which Bob receives and decrypts, assuming it is from Alice. This scenario highlights the importance of using additional security measures, such as digital certificates and SSL/TLS protocols, to prevent man-in-the-middle attacks.
In conclusion, the RSA algorithm is an effective and widely used public-key cryptosystem for secure document transmission. However, it is essential to consider the potential security threats and implement additional security measures to ensure the confidentiality and integrity of transmitted documents.