

**NAME:** Sarvesh Patil

**CLASS:** D15A

**ROLL NO:** 52

**LAB 09**

**LAB 09:** Program to Implementation and analysis of RSA cryptosystem and Digital signature scheme using RSA

<b>ROLL NO</b>	52
<b>NAME</b>	Sarvesh Patil
<b>CLASS</b>	D15A
<b>SUBJECT</b>	Internet Security Lab
<b>LO MAPPED</b>	LO1: To apply the knowledge of symmetric cryptography to implement classical ciphers. LO2: Demonstrate Key management, distribution, and user authentication.

**AIM:**

Program to Implementation and analysis of RSA cryptosystem and Digital signature scheme using RSA

**INTRODUCTION:**

**DIGITAL SIGNATURE:**

A digital signature is a mathematical technique used to validate the authenticity and integrity of a message, software, or digital document. 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, or digital messages. Signers can also use them to acknowledge informed consent.

In many countries, including the United States, digital signatures are considered legally binding in the same way as traditional handwritten document signatures.

***How do digital signatures work?***

Digital signatures are based on public key cryptography, also known as asymmetric cryptography. Using a public key algorithm, such as RSA (Rivest-Shamir-Adleman), two keys are generated, creating a mathematically linked pair of keys, one private and one public.

Digital signatures work through public key cryptography's two mutually authenticating cryptographic keys. The individual who creates the digital signature uses a private key to encrypt signature-related data, while the only way to decrypt that data is with the signer's public key.

If the recipient can't open the document with the signer's public key, that's a sign there's a problem with the document or the signature. This is how digital signatures are authenticated.

Digital signature technology requires all parties to trust that the individual creating the signature has kept the private key secret. If someone else has access to the private signing key, that party could create fraudulent digital signatures in the name of the private key holder.

***What are the benefits of digital signatures?***

Security is the main benefit of digital signatures. Security capabilities embedded in digital signatures ensure a document is not altered and signatures are legitimate. Security features and methods used in digital signatures include the following:

- Personal identification numbers (PINs), passwords, and codes. Used to authenticate and verify a signer's identity and approve their signature. Email, username, and password are the most common methods used.
- Asymmetric cryptography. Employs a public key algorithm that includes private and public key encryption and authentication.

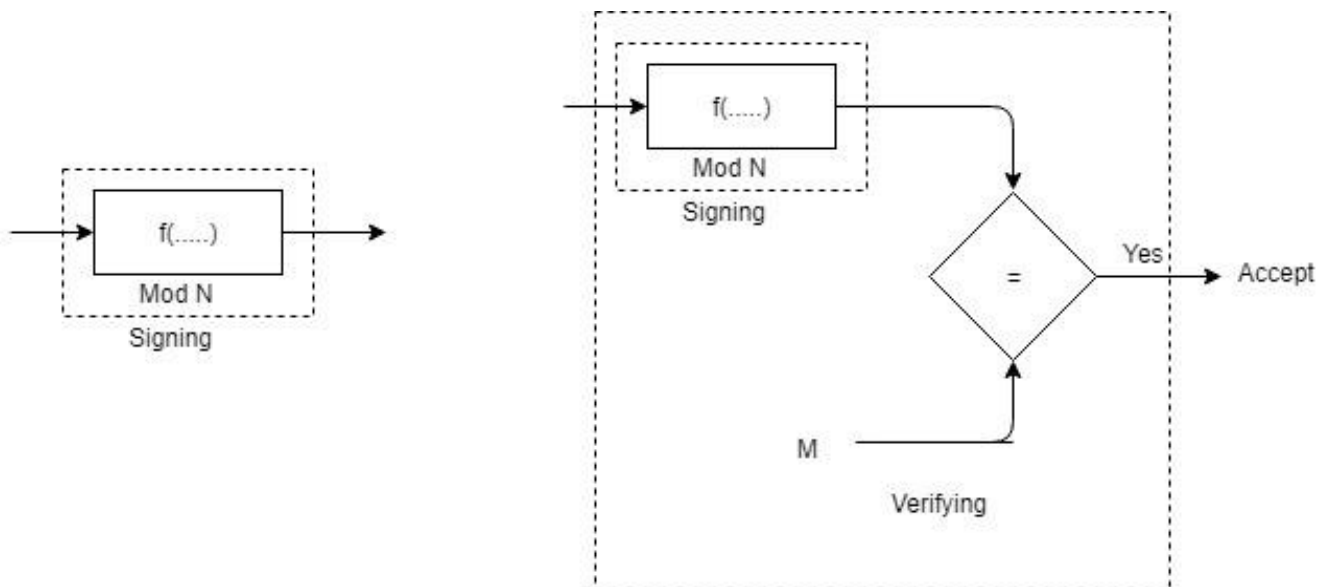
- Checksum. A long string of letters and numbers that represents the sum of the correct digits in a piece of digital data, against which comparisons can be made to detect errors or changes. A checksum acts as a data fingerprint.
- Cyclic redundancy check (CRC). An error-detecting code and verification feature is used in digital networks and storage devices to detect changes to raw data.
- Certificate authority (CA) validation. CAs issue digital signatures and act as trusted third parties by accepting, authenticating, issuing, and maintaining digital certificates. The use of CAs helps avoid the creation of fake digital certificates.
- Trust service provider (TSP) validation. A TSP is a person or legal entity that performs the validation of a digital signature on a company's behalf and offers signature validation reports.

Other benefits to using digital signatures include the following:

- Timestamping. By providing the data and time of a digital signature, timestamping is useful when the timing is critical, such as for stock trades, lottery ticket issuance, and legal proceedings.
- Globally accepted and legally compliant. The public key infrastructure (PKI) standard ensures vendor-generated keys are made and stored securely. Because of the international standard, a growing number of countries are accepting digital signatures as legally binding.
- Time savings. Digital signatures simplify the time-consuming processes of physical document signing, storage, and exchange, enabling businesses to quickly access and sign documents.
- Cost savings. Organizations can go paperless and save money previously spent on physical resources and on the time, personnel, and office space used to manage and transport them.
- Positive environmental impact. Reducing paper use also cuts down on the physical waste generated by paper and the negative environmental impact of transporting paper documents.
- Traceability. Digital signatures create an audit trail that makes internal record-keeping easier for businesses. With everything recorded and stored digitally, there are fewer opportunities for a manual signee or record-keeper to make a mistake or misplace something.

**RSA digital signature scheme**

- RSA idea is also used for signing and verifying a message it is called RSA digital signature scheme.
- Digital signature scheme changes the role of the private and public keys
- Private and public keys of only the sender are used not the receiver
- Sender uses her own private key to sign the document and the receiver uses the sender's public key to verify it.



General Ideal behind RSA Digital Signature Scheme

- The signing and verifying sets use the same function, but with different parameters. The verifier compares the message and the output of the function for congruence. If the result is two true the message is accepted.

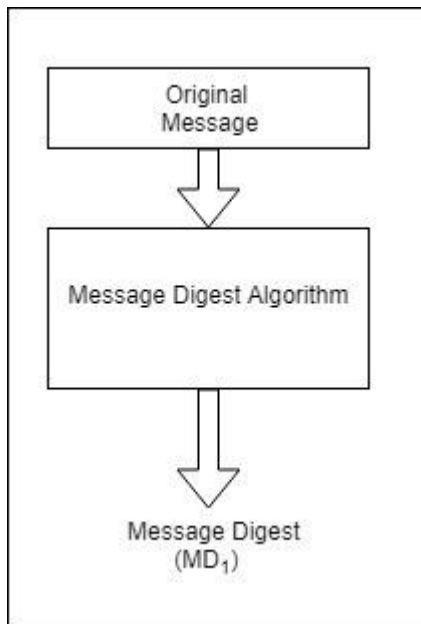
**Key generation in RSA**

Key generation in RSA digital signature scheme is exactly the same as key generation in the RSA cryptosystem.

**Working of RSA digital signature scheme:**

Sender A wants to send a message M to the receiver B along with the digital signature S calculated over the message M

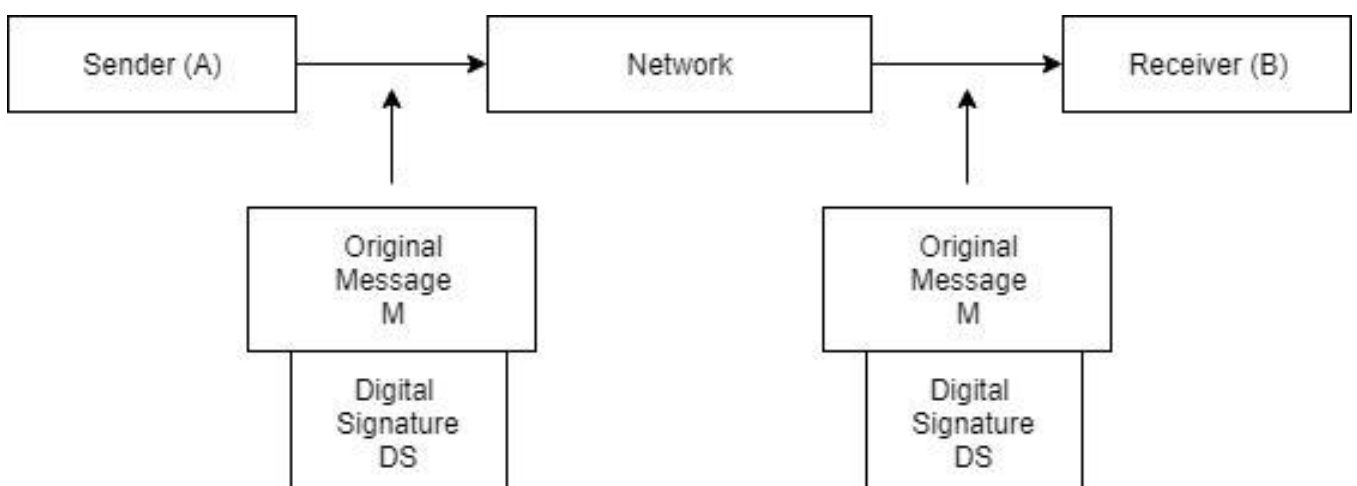
Step1: The sender A uses the message digest algorithm to calculate the message digest MD1 over the original message M



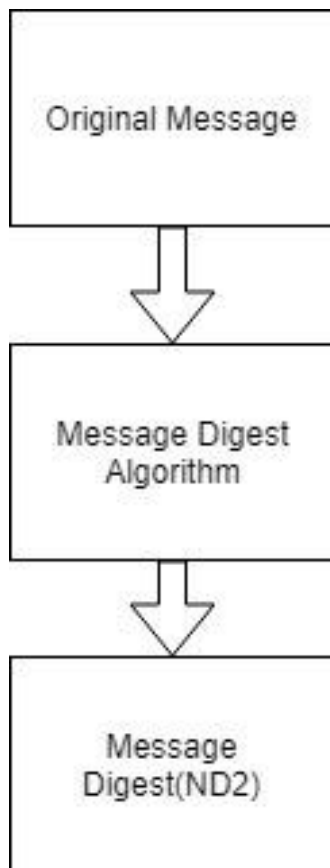
Step 2: The sender A now encrypts the message digest with her private key. The output of this process is called the digital signature.



Step 3: Now the sender A sends the original message M along with the digital signature DS to receiver B



Step 4: After the receiver B receives the original message M and the sender A's digital signature, B uses the same message digest algorithm which was used by A and calculate its own message digest MD2 as shown below.



Step 5: The receiver B now uses sender A's public key to decrypt the digital signature. Note that A had used his private key to encrypt the message digest MD1 to form the digital signature. Therefore only A's public key can be used to decrypt it. The output of this process is the original message digest which was calculated by A (MD1) in step 1.

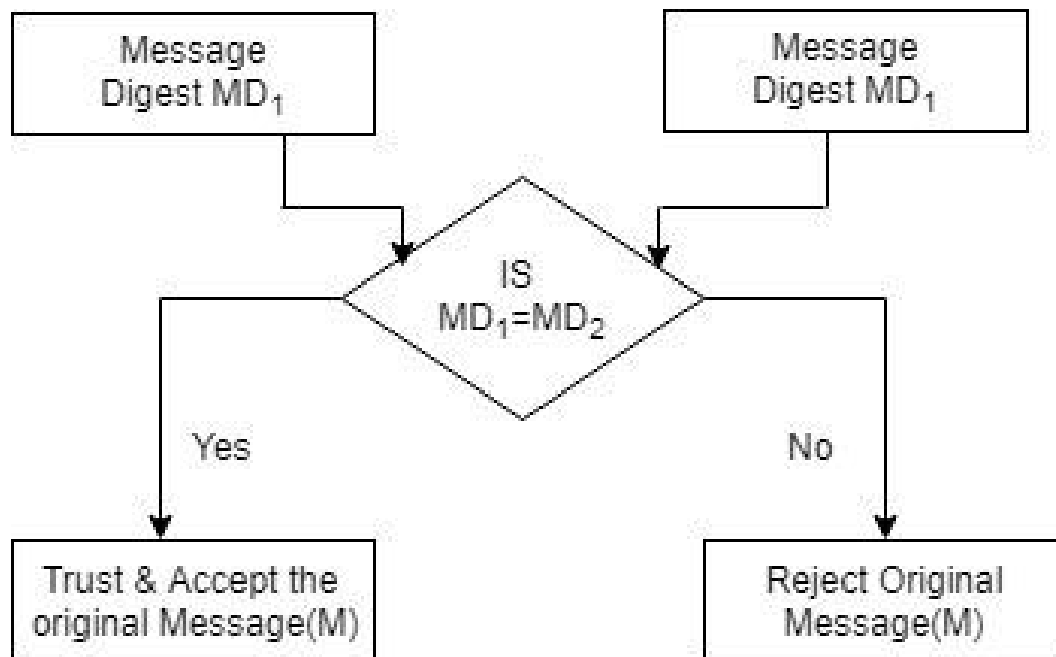


Step 6: B now compares the following two message digests.

1. MD2, which it had calculated in step 4
2. MD1, which is retrieved from A's digital signature in step 5

If  $MD_1 = MD_2$  the following facts are established:

- a. B accepts the original message (M) as the correct, unaltered message from A
- b. B is also assured that the message came from A and not from someone else attached, posing as A



Thus, the principle of digital signature is quite strong, secure and reliable.

## PROGRAM:

```
# Function to find gcd
# of two numbers
from traceback import print_tb

def euclid(m, n):
    if n == 0:
        return m
    else:
        r = m % n
        return euclid(n, r)

# Program to find
# Multiplicative inverse
def exteuclid(a, b):
    r1 = a
    r2 = b
    s1 = int(1)
    s2 = int(0)
    t1 = int(0)
    t2 = int(1)

    while r2 > 0:

        q = r1 // r2
        r = r1 - q * r2
        r1 = r2
        r2 = r
        s = s1 - q * s2
        s1 = s2
        s2 = s
        t = t1 - q * t2
        t1 = t2
        t2 = t
```



```
    if t1 < 0:
        t1 = t1 % a

    return (r1, t1)

# Enter two large prime
# numbers p and q
p = 823
q = 953
n = p * q
Pn = (p - 1) * (q - 1)

# Generate encryption key
# in range 1 < e < Pn
key = []

for i in range(2, Pn):
    gcd = euclid(Pn, i)

    if gcd == 1:
        key.append(i)

# Select an encryption key
# from the above list
e = int(313)

# Obtain inverse of
# encryption key in Z_Pn
r, d = exteuclid(Pn, e)
if r == 1:
    d = int(d)
    print("Decryption key is: ", d)
else:
```

```
print(
    "Multiplicative inverse for the given encryption key does not
exist. Choose a different encryption key "
)

# Enter the message to be sent
M = int(input("Enter the message that Alice sends to Bob(M): "))

print("Public key of Alice is: ", e)
print("Private key of Alice is: ", d)

# Signature is created by Alice
S = (M**d) % n
print("Signature generated by Alice is: ", S)

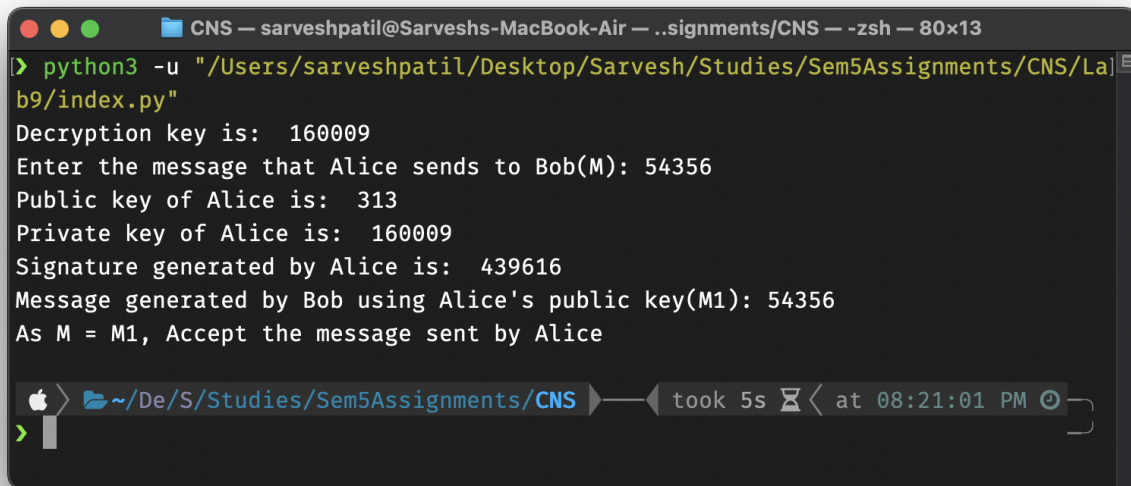
# Alice sends M and S both to Bob
# Bob generates message M1 using the
# signature S, Alice's public key e
# and product n.
M1 = (S**e) % n
print("Message generated by Bob using Alice's public key(M1): " +
      str(M1))

# If M = M1 only then Bob accepts
# the message sent by Alice.

if M == M1:
    print("As M = M1, Accept the message sent by Alice")
else:
    print("As M is not equal to M1, Do not accept the message sent by
Alice ")
```

**RESULTS:****TESTCASE 1:**

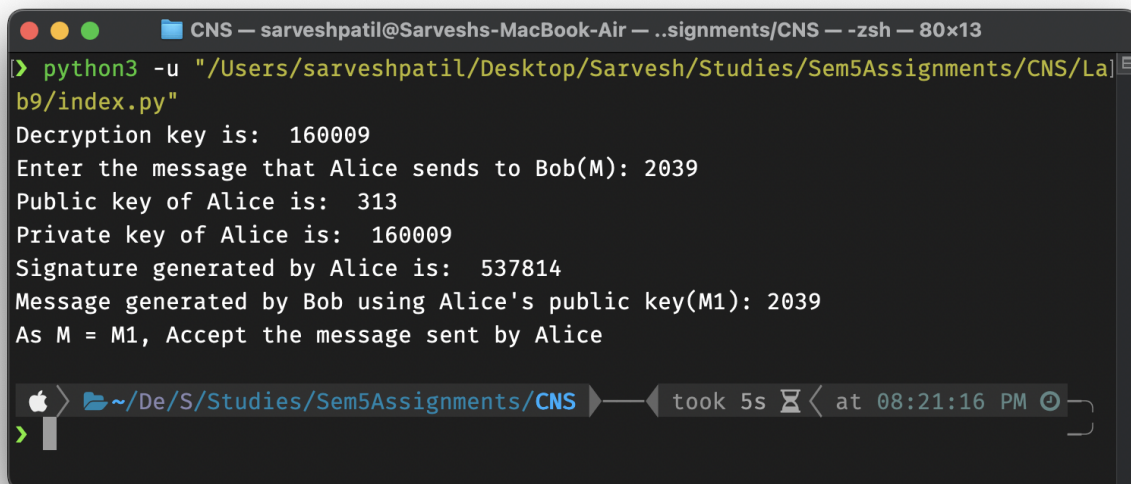
M = 54356

A terminal window titled 'CNS — sarveshpatil@Sarveshs-MacBook-Air — ..signments/CNS — -zsh — 80x13'. The command 'python3 -u "/Users/sarveshpatil/Desktop/Sarvesh/Studies/Sem5Assignments/CNS/La]b9/index.py"' is entered. The output shows: 'Decryption key is: 160009', 'Enter the message that Alice sends to Bob(M): 54356', 'Public key of Alice is: 313', 'Private key of Alice is: 160009', 'Signature generated by Alice is: 439616', 'Message generated by Bob using Alice's public key(M1): 54356', and 'As M = M1, Accept the message sent by Alice'. The terminal bar at the bottom shows the path '~/De/S/Studies/Sem5Assignments/CNS', a duration of 'took 5s', and the time 'at 08:21:01 PM'.

```
CNS — sarveshpatil@Sarveshs-MacBook-Air — ..signments/CNS — -zsh — 80x13
> python3 -u "/Users/sarveshpatil/Desktop/Sarvesh/Studies/Sem5Assignments/CNS/La]
b9/index.py"
Decryption key is: 160009
Enter the message that Alice sends to Bob(M): 54356
Public key of Alice is: 313
Private key of Alice is: 160009
Signature generated by Alice is: 439616
Message generated by Bob using Alice's public key(M1): 54356
As M = M1, Accept the message sent by Alice
>
```

**TESTCASE 2:**

M = 2039

A terminal window titled 'CNS — sarveshpatil@Sarveshs-MacBook-Air — ..signments/CNS — -zsh — 80x13'. The command 'python3 -u "/Users/sarveshpatil/Desktop/Sarvesh/Studies/Sem5Assignments/CNS/La]b9/index.py"' is entered. The output shows: 'Decryption key is: 160009', 'Enter the message that Alice sends to Bob(M): 2039', 'Public key of Alice is: 313', 'Private key of Alice is: 160009', 'Signature generated by Alice is: 537814', 'Message generated by Bob using Alice's public key(M1): 2039', and 'As M = M1, Accept the message sent by Alice'. The terminal bar at the bottom shows the path '~/De/S/Studies/Sem5Assignments/CNS', a duration of 'took 5s', and the time 'at 08:21:16 PM'.

```
CNS — sarveshpatil@Sarveshs-MacBook-Air — ..signments/CNS — -zsh — 80x13
> python3 -u "/Users/sarveshpatil/Desktop/Sarvesh/Studies/Sem5Assignments/CNS/La]
b9/index.py"
Decryption key is: 160009
Enter the message that Alice sends to Bob(M): 2039
Public key of Alice is: 313
Private key of Alice is: 160009
Signature generated by Alice is: 537814
Message generated by Bob using Alice's public key(M1): 2039
As M = M1, Accept the message sent by Alice
>
```

**CONCLUSION:**

Thus we have successfully implemented and analyzed RSA cryptosystem and Digital signature scheme using RSA.