



Academic Year: 2025-2026

Semester: V

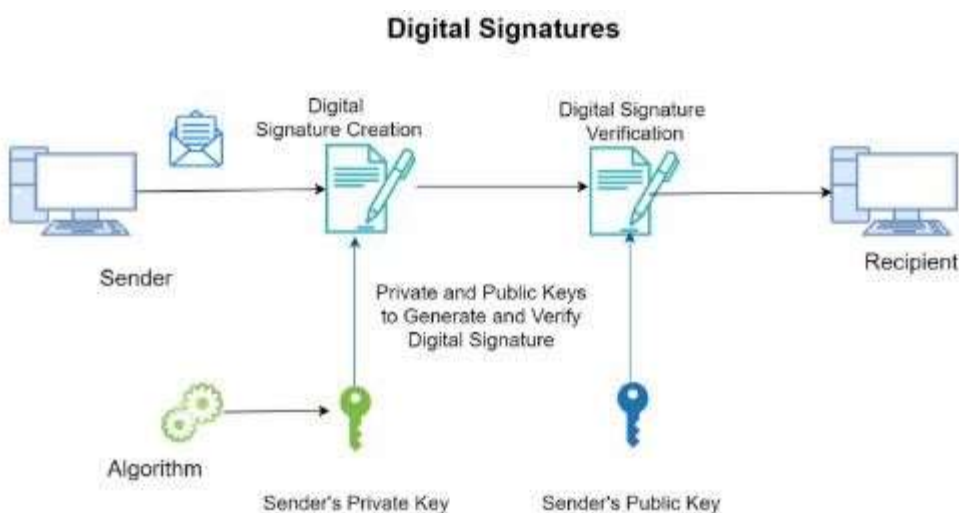
Class / Branch: TE IT

Subject: Security Lab

Subject Incharge : Prof. Apeksha Mohite

Experiment No. 13(ii)

1. **Aim:** To study and analyze RSA cryptosystem and digital signature scheme.
2. **Software Required :** CrypTool 1.4.41
3. **Theory :**



Digital signatures can provide the added assurances of evidence of origin, identity and status of an electronic document, transaction or message and can acknowledge informed consent by the signer.

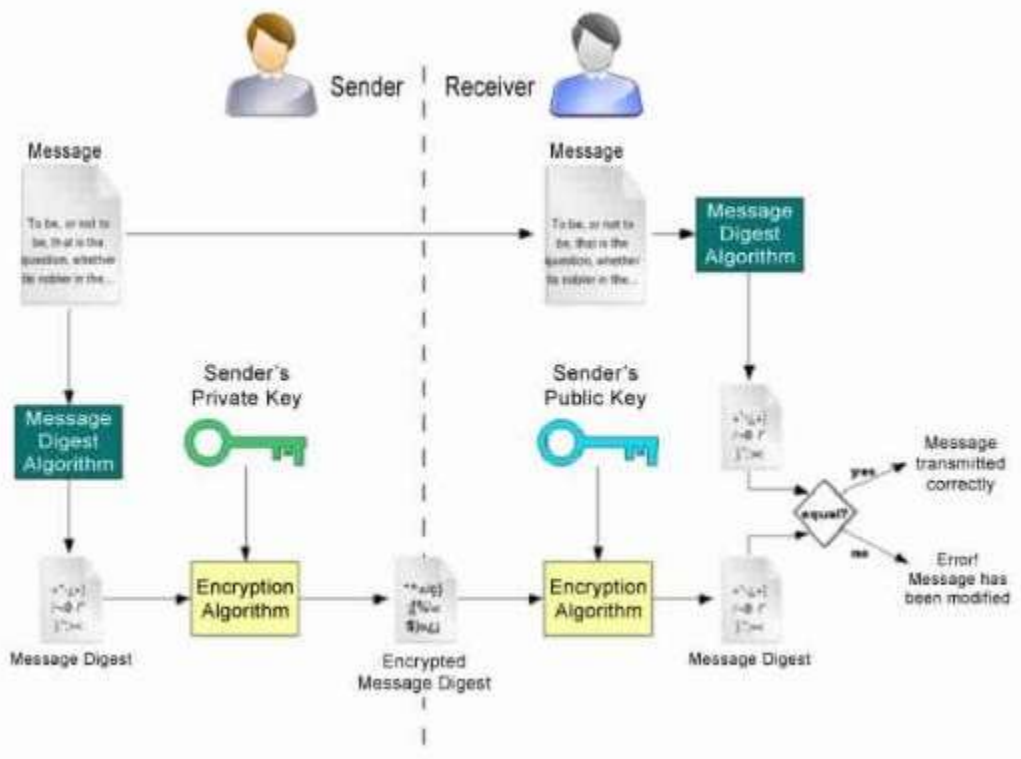
How digital signatures work

Digital signatures are based on public key cryptography, also known as asymmetric



cryptography. Using a public key algorithm, such as RSA, one can generate two keys that are mathematically linked: one private and one public.

Digital signatures work because public key cryptography depends on two mutually authenticating cryptographic keys. The individual who is creating the digital signature uses their own private key to encrypt signature-related data; the only way to decrypt that data is with the signer's public key. This is how digital signatures are authenticated.



How to create a digital signature

To create a digital signature, signing software such as an email program -- creates a one-way hash of the electronic data to be signed. The private key is then used to encrypt the hash. The encrypted hash along with other information, such as the hashing algorithm is the digital signature.



The reason for encrypting the hash instead of the entire message or document is that a hash function can convert an arbitrary input into a fixed length value, which is usually much shorter. This saves time as hashing is much faster than signing.

The value of a hash is unique to the hashed data. Any change in the data, even a change in a single character, will result in a different value. This attribute enables others to validate the integrity of the data by using the signer's public key to decrypt the hash.

If the decrypted hash matches a second computed hash of the same data, it proves that the data hasn't changed since it was signed. If the two hashes don't match, the data has either been tampered with in some way -- integrity -- or the signature was created with a private key that doesn't correspond to the public key presented by the signer -- authentication.

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

An example of asymmetric cryptography :

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

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

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



that 1024 bit keys could be broken in the near future. But till now it seems to be an infeasible task.

We demonstrate RSA with the help of cryptool

RSA Demonstration [X]

— RSA using the private and public key -- or using only the public key

☒ Choose two prime numbers p and q . The composite number $N = pq$ is the public RSA modulus, and $\phi(N) = (p-1)(q-1)$ is the Euler totient. The public key e is freely chosen but must be coprime to the totient. The private key d is then calculated such that $d = e^{-1} \pmod{\phi(N)}$.

☐ For data encryption or certificate verification, you will only need the public RSA parameters: the modulus N and the public key e .

— Prime number entry

Prime number p

Prime number q

— RSA parameters

RSA modulus N (public)

$\phi(N) = (p-1)(q-1)$ (secret)

Public key e

Private key d

— RSA encryption using e / decryption using d [alphabet size: 256]

Input as ☒ text ☐ numbers

Input text

The Input text will be separated into segments of Size 1 (the symbol '#' is used as separator).

Numbers input in base 10 format.

Encryption into ciphertext $c[i] = m[i]^e \pmod{N}$



RSA Demonstration

RSA using the private and public key -- or using only the public key

- ☒ Choose two prime numbers p and q. The composite number $N = pq$ is the public RSA modulus, and $\phi(N) = (p-1)(q-1)$ is the Euler totient. The public key e is freely chosen but must be coprime to the totient. The private key d is then calculated such that $d = e^{-1} \pmod{\phi(N)}$.
- ☐ For data encryption or certificate verification, you will only need the public RSA parameters: the modulus N and the public key e.

Prime number entry

Prime number p	<input type="text" value="211"/>	<input type="button" value="Generate prime numbers..."/>
Prime number q	<input type="text" value="233"/>	

RSA parameters

RSA modulus N	<input type="text" value="49163"/>	(public)
$\phi(N) = (p-1)(q-1)$	<input type="text" value="48720"/>	(secret)
Public key e	<input type="text" value="2^16+1"/>	<input type="button" value="Update parameters"/>
Private key d	<input type="text" value="44273"/>	

RSA encryption using e / decryption using d [alphabet size: 256]

Input as ☐ text ☒ numbers

Ciphertext coded in numbers of base 10

Decryption into plaintext $m[i] = c[i]^d \pmod{N}$

Output text from the decryption (into segments of size 1; the symbol '#' is used as separator).

Plaintext



A digital signature scheme typically consists of 3 algorithms;

- A *key generation* algorithm that selects a *private key* uniformly at random from a set of possible private keys. The algorithm outputs the private key and a corresponding *public key*.
- A *signing* algorithm that, given a message and a private key, produces a signature.
- A *signature verifying* algorithm that, given the message, public key and signature, either accepts or rejects the message's claim to authenticity.

Two main properties are required. First, the authenticity of a signature generated from a fixed message and fixed private key can be verified by using the corresponding public key. Secondly, it should be computationally infeasible to generate a valid signature for a party without knowing that party's private key. A digital signature is an authentication mechanism that enables the creator of the message to attach a code that acts as a signature.

One digital signature scheme (of many) is based on RSA. To create signature keys, generate a RSA key pair containing a modulus, N , that is the product of two random secret distinct large primes, along with integers, e and d , such that $e d \equiv 1 \pmod{\phi(N)}$, where ϕ is the Euler phi-function. The signer's public key consists of N and e , and the signer's secret key contains d .

To sign a message, m , the signer computes a signature, σ , such that $\sigma \equiv m d \pmod{N}$. To verify, the receiver checks that $\sigma e \equiv m \pmod{N}$.



Digital Signatures using Cryptool

MD5 digital signature example

Cryptool 1.4.41 - RSA (MD5) signature of <startingexample-en>

File Edit View Encrypt/Decrypt Digital Signatures/PKI Individ. Procedures Analysis Options Window Help

startingexample-en

HELLOWORLD

RSA (MD5) signature of <startingexample-en>

```
00000000 53 69 67 6E 61 74 75 72 65 3A 20 20 20 20 20 20 F6 Signature:
00000012 11 D6 E4 78 DD 09 FA 73 96 33 90 E3 C8 BF 72 27 87 FD ...x...s.3...r'...
00000024 C8 3B 84 7F D2 53 33 04 41 30 D2 88 16 C2 84 BD EE F0 ...S3.A0...
00000036 0A 5F 67 64 C6 D6 EA 30 85 80 DE 25 6A 9D 43 16 71 F3 ...gd...0...%j.C.q...
00000048 42 90 DD E9 34 63 90 44 FE E0 CA DA 23 C4 75 97 6A A1 B...4c.D...#u.j...
0000005A 3A 84 81 74 48 64 B4 AE 00 D0 AA 2C D8 87 F4 59 06 75 ...tHd...Y.u...
0000006C 07 FE BC 64 86 48 4B 98 AD 25 89 81 1E 8A BD 34 C2 95 ...d.HK...4...
0000007E 18 62 F3 2E 56 39 A5 F4 6B E7 70 4B 66 F0 49 7F 88 AD ...b...V9...k.pKf.I...
00000090 7A 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 z
000000A2 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
000000B4 6E 61 74 75 72 65 20 6C 65 6E 67 74 68 3A 20 20 31 30 nature length: 10
000000C6 32 34 20 20 20 20 20 20 20 20 20 20 20 20 20 20 24
000000D8 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
000000EA 74 68 6D 3A 20 20 20 20 20 20 20 20 20 20 20 20 20
000000FC 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0000010E 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00000120 6E 3A 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00000132 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00000144 20 4B 65 79 3A 20 20 20 20 20 20 20 20 20 20 20 20
00000156 61 5D 5B 4D 6F 68 69 74 65 5D 5B 52 53 41 2D 31 30 32
00000168 34 5D 5B 31 35 33 38 36 31 34 37 31 33 5D 5B 41 54 4D
0000017A 5D 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0000018C 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0000019E 45 4C 4C 4F 57 4F 52 4C 44 Message: H
HELLOWORLD
```

Press F1 to obtain help. L:1 C:1 P:1 OVR NUM






IIT VIRTUAL LAB FOR CRYPTOGRAPHY

CRYPTOGRAPHY LAB

[Home](#) [Cryptography Lab](#)









Welcome to Cryptography lab

 [INTRODUCTION](#)  [EXPERIMENTS](#)  [TARGET AUDIENCE](#)  [COURSES ALIGNED](#)  [PRE-REQUISITE COURSES](#)  [FEEDBACK](#)

Introduction

Welcome to the Cryptography lab. In this lab, we will do virtual experiments to understand the basic mathematical foundations of cryptography, to gain insightful experience by working with fundamental cryptographic applications and to train in the art of design and analysis of information security protocols.

Digital Signatures Scheme

 [INTRODUCTION](#)  [THEORY](#)  [OBJECTIVE](#)  [EXPERIMENT](#)  [MANUAL](#)  [QUIZZES](#)  [PROCEDURE](#)  [FURTHER READINGS](#)

Introduction

A Digital Signature is an authentication mechanism that enables the creator of the message to attach a code that acts as a signature. The signature is formed by taking the hash of the message and encrypting the message with the creator's private key. The signature guarantees the source and integrity of the message.

About the experiment:

In Public key setting, it becomes difficult to verify for a receiver whether message is originated from claimed source. In this experiment, we show how can a receiver verify integrity of the message in public key setting. Your task is to verify, whether digital signature scheme really works and why it works?



Digital Signatures Scheme



Manual

Step 1: Enter the input text to be encrypted in the 'Plaintext' area and generate hash value for message by clicking on the **SHA-1** button.

Step 2: Copy content of **Hash Output(hex)** field and paste it in **Input to RSA(hex)** field.

Step 3: Select keysize of public key from **RSA Public key** section by clicking on any key button.

Step 4: Click on **Apply RSA** button to generate a digital signature.

Digital Signatures Scheme



Experiment

Digitally sign the plaintext with Hashed RSA.

Plaintext (string):

first digital certificate

Hash output(hex):

4982952581d7cc58a42e85e92bbc58b74092f4f6

Input to RSA(hex):

4982952581d7cc58a42e85e92bbc58b74092f4f6

Digital Signature(hex):

9ad81803e31cc30f64858d5673c123b8c41035ec43789c4a2e9f007ca777
3e590948618247bf9250819b44a49769dd114b37d58e37ed1a7785592614a12
478a6d1815a752884e93d53f6581420801cac97016ed52b727bba7e13233d99d
132577b784e5f55b100bf5bc7a6df6b505636e0008a00ee412ed58049b71c89

Digital Signature(base64):

mtgYA+McnwUhw1Wu8EpsQONcz9N4nP5i6BAK9lc+WQlgYJhV5JOgZIEpJdp
3fEUs31W437Rp3hVkmFKCkeKbRgVp1KTPVPV2WBQggByslwFu1Sye7p+EyM&md
EYV3i4T8VsQCu8em32tQVjgAlaA7kEu1YBJuz2k=

Status:

Time: 18ms

RSA public key

Public exponent (hex, F4=0x10001):

10001

Modulus (hex):

ca5261939975940bb7a5d4ff6f54e65f04989175f5a09288610b8975671e99
a3b5e5d94057b0fcd7535f6f7444504fa351694461d0d30cd0182c307727cd6
5168c786771c561a9400fb45175e9e6aa4c23e11af89e9412ds23b0cb8664c4
c2429bce138e848ab26d0629073351f4acd36074eaf4036a5eb93359d2a698d3

1024 bit | 1024 bit (e=3) | 512 bit | 512 bit (e=3)