XpertBot

BabiBeauty

**A-CIA TRIAD**

The **CIA Triad** is a foundational model in cybersecurity that represents the three core principles of information security:

1. Confidentiality: Ensuring data is accessible only by authorized users,usually enforced by encryption techniques.

Example:When someone logs to their bank account,the website uses SSL/TLS encryption to ensure that their username,password and transaction details remains confidential.

1. Integrity:Ensures the accuracy and completeness of data,assuring the data hasn’t been improperly modified using hashing techniques.

Example:A company sends a contact via email with a digital signature which the recipient can use to verify that the document hasn’t been altered during transit.

1. Avilability:Data must be available when needed by authorized users,usually achieved through implementing redundancy.

Example: A hospital uses **cloud-based patient records** with redundant servers. Even if one server fails (or a **DDoS attack** targets the system), backups ensure doctors can still access critical medical data when needed.

**B-SYMMETRIC ENCRYPTION:**

Symmetric encryption utilizes a single key for both encryption and decryption.key encryption includes:AES,DES,RC5 and Twofish.

Sample run for a python code that encrypt and decrypt using the same key:

Menu:

1. Encrypt a message

2. Decrypt a message

3. Exit

Choose an option (1/2/3): 1

Enter the message to encrypt: the item is buried under the pine tree

Encryption Successful!

Key: txhU13g80srJ8I2ZI7mdS2FcV1PCydHwxrvZyslLmB8=

Encrypted Message: gAAAAABomFiULD2Qj\_hKD7JCrkiyJa2ONSGFbggMiLy6XuSUqzY-P4IwxtN-hGRtsZBWA9UBl7qu7WWxcs2oCxqWwZx\_okqtokrTgRXs\_IJg26eSGR\_rn1R\_9WdgGWBAbF35Km1-LDYl

Menu:

1. Encrypt a message

2. Decrypt a message

3. Exit

Choose an option (1/2/3): 2

Enter the encrypted message: gAAAAABomFiULD2Qj\_hKD7JCrkiyJa2ONSGFbggMiLy6XuSUqzY-P4IwxtN-hGRtsZBWA9UBl7qu7WWxcs2oCxqWwZx\_okqtokrTgRXs\_IJg26eSGR\_rn1R\_9WdgGWBAbF35Km1-LDYl

Enter the secret key: txhU13g80srJ8I2ZI7mdS2FcV1PCydHwxrvZyslLmB8=

Decryption Successful!

Original Message: the item is buried under the pine tree

Menu:

1. Encrypt a message

2. Decrypt a message

3. Exit

Choose an option (1/2/3): 2

Enter the encrypted message: bhwhwbcbxbcwbc

Enter the secret key: hcwbcwbchwhcwcwygwhcwcwhbc

Error: Fernet key must be 32 url-safe base64-encoded bytes. (Invalid key or message!).

C-**Asymmetric Encryption:**

**Asymmetric encryption involves a public and private key for a secure data exchange and digital signatures.Asymmtric encryption algorithems:RSA,ECC and DH-ECDH.**

Sample run for a python code that encrypt and decrypt using the public and private key:

Asymmetric Encryption (RSA) Demo

===================================

Key Principles:

- Encryption uses the PUBLIC KEY (shared with others)

- Decryption uses the PRIVATE KEY (kept secret)

===================================

Public Key (Share this for ENCRYPTION):

-----BEGIN PUBLIC KEY-----

MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAzZAMYwqmyv/wrFCUpXXD

MiqlACHWLxnX7kgUx2zE+FRo1Olp66SvzOaCy160TcBJZQXfXz3tTMxq0VCflYOm

Pf8+cbPWZExhrODdpDUUem71KJ3FKdBMfbwsCbS89D6DziiRwjb8y6zHCAG0trIX

1ECN2eEmT895OXDzRU11ln78DByaLEr4Euh/mLuTEttRwtq0nA1y1AhC6HenWUnz

qmDAEiUqVY7Y0LKbklj7DY+GGE/YMBlET+qkDM+w/ToLoYEqP15E1QboRwB2W1TV

hXFuh962SKtNLrEzVHk/78sUo6dIr5t9nRcCnG/2CujxRYplLHBGKxYY1HYfLr/t

jQIDAQAB

-----END PUBLIC KEY-----

Private Key (Keep this SECRET for DECRYPTION):

-----BEGIN PRIVATE KEY-----

MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQDNkAxjCqbK//Cs

UJSldcMyKqUAIdYvGdfuSBTHbMT4VGjU6WnrpK/M5oLLXrRNwEllBd9fPe1MzGrR

UJ+Vg6Y9/z5xs9ZkTGGs4N2kNRR6bvUoncUp0Ex9vCwJtLz0PoPOKJHCNvzLrMcI

AbS2shfUQI3Z4SZPz3k5cPNFTXWWfvwMHJosSvgS6H+Yu5MS21HC2rScDXLUCELo

d6dZSfOqYMASJSpVjtjQspuSWPsNj4YYT9gwGURP6qQMz7D9OguhgSo/XkTVBuhH

AHZbVNWFcW6H3rZIq00usTNUeT/vyxSjp0ivm32dFwKcb/YK6PFFimUscEYrFhjU

dh8uv+2NAgMBAAECggEAA/gFWomDcY/szvw599KE6PG3KyrRaSoP5+MvDi/6GEFT

n34SNCEt3mrk7kBEguvKKaUqDDtfMDMSLOW9mbQSVZ1upeCvAai9NLVIEjo5dv5a

xWSmazRBBdHIQo0MHjscGF2NfY505bR5nfH1J0Hbq+wYe7MZJkP/Ny4IbwzGecKu

VaUto9kR5HZCHxBTr8WUCMgod4uIlPBZpYr3fkBW5f7xbzPZA2qI3HEjYAE0Oe2m

O7jdkiA/8jNP/a1kRbBKvOPwn8EZBryqZ7DF2T3dQnuA1gRrKurZ8q9lse4G5BUy

xVD1FDDp8lq2wapxDAq/Ab4C98Ha1x2DYRw1ciMlEQKBgQD4R0ZeJONvVATJaUjw

ZvhkUmeAgWo67Kb6+O8Uuk+R1IVqej3uGdL/GQm/fMgdDRAs/aXsCwnWwDF9eZuz

c3iwwklRuxpRQ/GoAvmkLgH43ydA74I8ykwppB7LEse23sJFgo30BFO5tORxM580

Pc8mZIkYvUFqqw6FiTYbzM6cswKBgQDT9K65h0LDyEzvG0KaBOuZVj6eLcpOvNbp

aELEAo1v7gg6uRgEit0+i0jc7pqIH/MKmNKp/QmdFSIuvLxc8ti80xOpXvXVYybS

8RuM+zTzx1DKSVaR5/pYGlmuZRaHSnmHrJ5gTlmtICxomWYxhJPqV34Nfh9+xueC

TtcZkUzsvwKBgHNj1O2OYXHtb6gYWcD5anpd3iLYgwHN2TGNnrhYewDA8BzzoXLW

57yN63zmxq7R0nHGlWXmH3WCr72sVlQJ1873DVKCDhI07Mx0TYgjpeD//eNtLW+J

aep0j6c8HDv+vfXGXqlJlYwa6Y0/O7/w9iTW1uz7kLq9s6sWgWJcRq6HAoGBAICE

qrVAONFx0IRjKR0uwpR1YP6UkUjw4/cfCuzDB3PJByBkPxXyBXj5B7M+0mCbzsis

JYCBN6URvWpNcoGBlr3X1uIp/DHpLw9UtvnjEBaI0on1shJCcN+fCqXeEazzEYGc

v2Nx3ppFGehKq1Z3JyE+7+XKhdpfTN0MZQzRt60NAoGALCNFx+1xZqKUzxuIOG0a

4/ks9uJpKM/6LQeCiQG6Zhbe3QL/FNmB/E/0bUWWcGD3nJ2EfvQ/UY4PTSm/kt9w

MyR+XxDkXFJoLP2tfenoC+tMHIkrCRxjBcLsLd0Fain8L4/ub4eHdJ+uXz7+8qoU

9jSaSzMAU0oRyhSn2wvUYNY=

-----END PRIVATE KEY-----

Enter a message to encrypt (using PUBLIC key): hi my name is sajed

Message encrypted with PUBLIC KEY:

Encrypted Bytes: b"5\x14c\x1f<\xb1\x8ao\x1d\x19dW\xa7\x99\xe9\*\xaa\x99W\xccN\xf1GR\n\xcf\x82\xee\x85[s\xc9L\x80\xeb\xbd&c\xbfHAw\xbdK\xd2\*M5\xdc\xec\x9a\xd4\xf8\xf1Q/\x92\x94\x7f?\x01\x96x\xbb\x04\x88\x81\xbd\xc4\x8f\xf5d\x07\x8d\xa7~\xc4\xee\x10\xca\xd5\\\xc3r\xfa\xca\xe1\x99o\x94\xda<\xa8\x87\xbcp\xc6\x834\xcc0\x91\xec\xd4\xe0'\xca\xa3\x06\x83HG\x93\xe0\x1e\x96\xfbd\x98z\xbe3D\x89\x93\xc9D-e\xdcM\x8c6\xc6\xa9\x9f\x8d\x1c-&~2=Sp\x04\xb1\x9b\x88M\xf5i\xc1\xb0\xb9K-w\xd82\xf6\x01\x07\xbfBe\x1a\xe0\x06\xc6+\xd06\xfd\xfd\xba\xc9\r\x18tN\x1a\x95Pp\xfcv\xf2\xcf\xf2\xa8(\x03\\37`U\xe9\x8f\x93l\xb0\xe93\xe1\xf9\xf3\x7f&\x9d\xc1ZE:p\xf0\xfa\x04\_\xc3;\xae\x12GhH5\xeb\xbc E\xbc\xe1l%\x94\xc8\xd9Q\xedb\xbds\xc1\x06N\x94\x155;\xc3S\xb3\x9c\xb5"

Press Enter to decrypt (using PRIVATE key)...

Message decrypted with PRIVATE KEY:

Original Message: hi my name is sajed

**D-DIGITAL SIGNATURE:**

A **digital signature** is a cryptographic technique that binds a person's identity to a digital message or document, ensuring:

* **Integrity** – The message has not been altered in transit.
* **Authentication** – The sender is who they claim to be.
* **Non-repudiation** – The sender cannot deny sending the message.

Digital signatures use **asymmetric cryptography** (public/private key pairs) to create a unique "signature" that can be verified by anyone with the public key.

**How Digital Signatures Works:**

**Step 1: Signing (Sender's Side)**

1. **Hashing** – The sender generates a **hash** (fixed-length fingerprint) of the message using a cryptographic hash function (e.g., SHA-256).
2. **Encryption** – The sender **encrypts the hash** with their **private key**, creating the **digital signature**.
3. **Transmission** – The sender sends the **original message + signature** to the recipient.

**Step 2: Verification (Recipient's Side)**

1. **Decryption** – The recipient decrypts the signature using the sender’s **public key**, recovering the original hash.
2. **Recomputing Hash** – The recipient independently hashes the received message.
3. **Comparison** – If the decrypted hash matches the recomputed hash, the message is **authentic and unaltered**.

**How Digital Signatures Ensure Security**

**✔ Integrity**

* If even **1 bit** of the message changes, the **hash will differ**, causing verification to fail.
* Ensures the message was **not tampered with** during transmission.

**✔ Authentication**

* Only the sender’s **private key** can produce a signature that decrypts correctly with their **public key**.
* Proves the message came from the claimed sender.

**✔ Non-Repudiation**

Since only the sender has their private key, they **cannot deny** signing the message.

**Common Digital Signature Algorithms:** RSA-PSS, ECDSA, EdDSA, DSA.

Sample run for a python code that signs/verifies the digital signature of a message:

Digital Signature Demo (RSA-PSS)

----------------------------------

1. Sign a message (with private key)

2. Verify a signature (with public key)

3. Exit

Choose action (1/2/3): 1

Enter message to sign: Hello World!

Message signed successfully!

Message: Hello World!

Signature (Base64): MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAwL9W...

Public Key (Share for verification):

-----BEGIN PUBLIC KEY-----

MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAwL9Wz8YwZ3X7K2Z8K1YJ

...

-----END PUBLIC KEY-----

Private Key (Keep secret!):

-----BEGIN PRIVATE KEY-----

MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQDAv1bPxjBndfsr

...

-----END PRIVATE KEY-----

Choose action (1/2/3): 2

Enter original message: Hello World!

Enter signature (Base64): MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAwL9W...

Paste sender's public key (PEM):

-----BEGIN PUBLIC KEY-----

MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAwL9Wz8YwZ3X7K2Z8K1YJ

...

-----END PUBLIC KEY-----

Signature is VALID. Message is authentic and unaltered.

Choose action (1/2/3): 3

Exiting...