

Emerging Blockchain Models for Digital Currencies

Name: Shaik Aarif

Regd No: 2200033144

Exp 3: Implementation of the Custom Asymmetric Key Encryption Algorithm.

Description:

Key Generation

- The `key_generation` function generates a public-private key pair using predefined values for prime numbers p and q .
- n is the product of p and q , and g is a primitive root modulo n .
- The private key is a (a secret integer), and the public key consists of g , A (where $A = g^a \bmod n$), and n .
- The function returns both the private key (a, n) and public key (g, A, n) .

Encryption

- The `encrypt` function encrypts a message m using the public key.
- A random integer k is selected, and then two values are computed:
 - $c1 = g^k \bmod n$ (first part of the ciphertext)
 - $c2 = (m * A^k) \bmod n$ (second part of the ciphertext)
- It returns the encrypted message as a tuple $(c1, c2)$.

Decryption

- The `decrypt` function decrypts the ciphertext $(c1, c2)$ using the private key (a, n) .
- It first calculates $S = c1^a \bmod n$, which is the shared secret.
- Then, the modular inverse of S (denoted $S^{-1} \bmod n$) is computed.
- Finally, the original message m is recovered as $m = (c2 * S^{-1}) \bmod n$.

Main Program

1. **Key Generation:** It calls `key_generation` to obtain the public and private keys.
2. **Message Conversion:** The original message, "BlockChain", is converted into a numeric format by converting each character to its ASCII value (minus 'A' to shift into a suitable range).

3. **Encryption:** Each numeric value is encrypted using the public key, and the encrypted pairs (c1, c2) are stored in a list res.
4. **Decryption:** The ciphertext (c1, c2) pairs are decrypted using the private key, and the numeric results are stored in resdesc.
5. **Message Reconstruction:** Finally, the decrypted numeric values are converted back to characters (by adding the ASCII value of 'A') and printed as the decrypted message.

```
1. from sympy import mod_inverse
2.
3. def key_generation():
4.
5.     p = 7
6.     q = 11
7.     n = p * q
8.     g = 5
9.     a = 3
10.    A = pow(g, a, n)
11.
12.    private_key = (a,n)
13.    public_key = (g, A, n)
14.
15.    return private_key, public_key
16.
17. def encrypt(public_key, m):
18.     g, A, n = public_key
19.     k = 6
20.
21.     c1 = pow(g, k, n)
22.     c2 = (m * pow(A, k, n)) % n
23.
24.     return c1, c2
25.
26. def decrypt(private_key, c1, c2):
27.
28.     a,n = private_key
29.
30.     S = pow(c1, a, n)
31.     S_inverse = mod_inverse(S, n)
32.
33.     m = (c2 * S_inverse) % n
34.
35.     return m
36.
37. private_key, public_key = key_generation()
38.
39. print("Private Key:", private_key)
40. print("Public Key (g, A, n):", public_key)
41.
42. original_message = "Blockchain"
43. marr= []
44. for i in original_message:
45.     marr.append(ord(i)-ord('A'))
46. # m = 13
47. print("Original Message in number format:", marr)
48.
49. res= []
50. for m in marr:
51.     c1, c2 = encrypt(public_key, m)
52.     res.append((c1, c2))
53.
54. print("Cipher Text: ", res)
```

```

55.
56. resdesc = []
57. for c1, c2 in res:
58.     decrypted_message = decrypt(private_key, c1, c2)
59.     resdesc.append(decrypted_message)
60.
61. print("Decrypted Message:", resdesc)
62.
63. for i in resdesc:
64.     i = i + ord('A')
65.     print(chr(i), end='')

```

Screenshot:

The screenshot displays the Visual Studio Code interface with a Python file named `AsymEnc.py` open. The file contains code for RSA encryption and decryption. The terminal window at the bottom shows the execution of the script, which outputs the private and public keys, the original message 'Blockchain', the cipher text as a list of tuples, and the decrypted message 'Blockchain'.

```

PS C:\Users\Administrator\Desktop\Emerging Blockchain\EBMOC\Exps> python AsymEnc.py
Private Key: (3, 77)
Public Key (g, A, n): (5, 48, 77)
Original Message: Blockchain
Original Message in number format: [1, 43, 46, 34, 42, 2, 39, 32, 40, 45]
Cipher Text: [(71, 15), (71, 29), (71, 74), (71, 48), (71, 14), (71, 30), (71, 46), (71, 18), (71, 61), (71, 59)]
Decrypted Message in number format: [1, 43, 46, 34, 42, 2, 39, 32, 40, 45]
Decrypted Message: Blockchain
PS C:\Users\Administrator\Desktop\Emerging Blockchain\EBMOC\Exps>

```