Working procedure of Digital signature

**CSE 459: Cryptography & Network Security**

Submitted by

Name: D. Tejaswi

Roll No: AP22110011503

Section: CSE Y

Lab Date: 07/03/2025

Submission Date: 13/04/2025

**A picture containing text

Description automatically generated**

**Department Computer Science and Engineering**

**School of Engineering and Sciences**

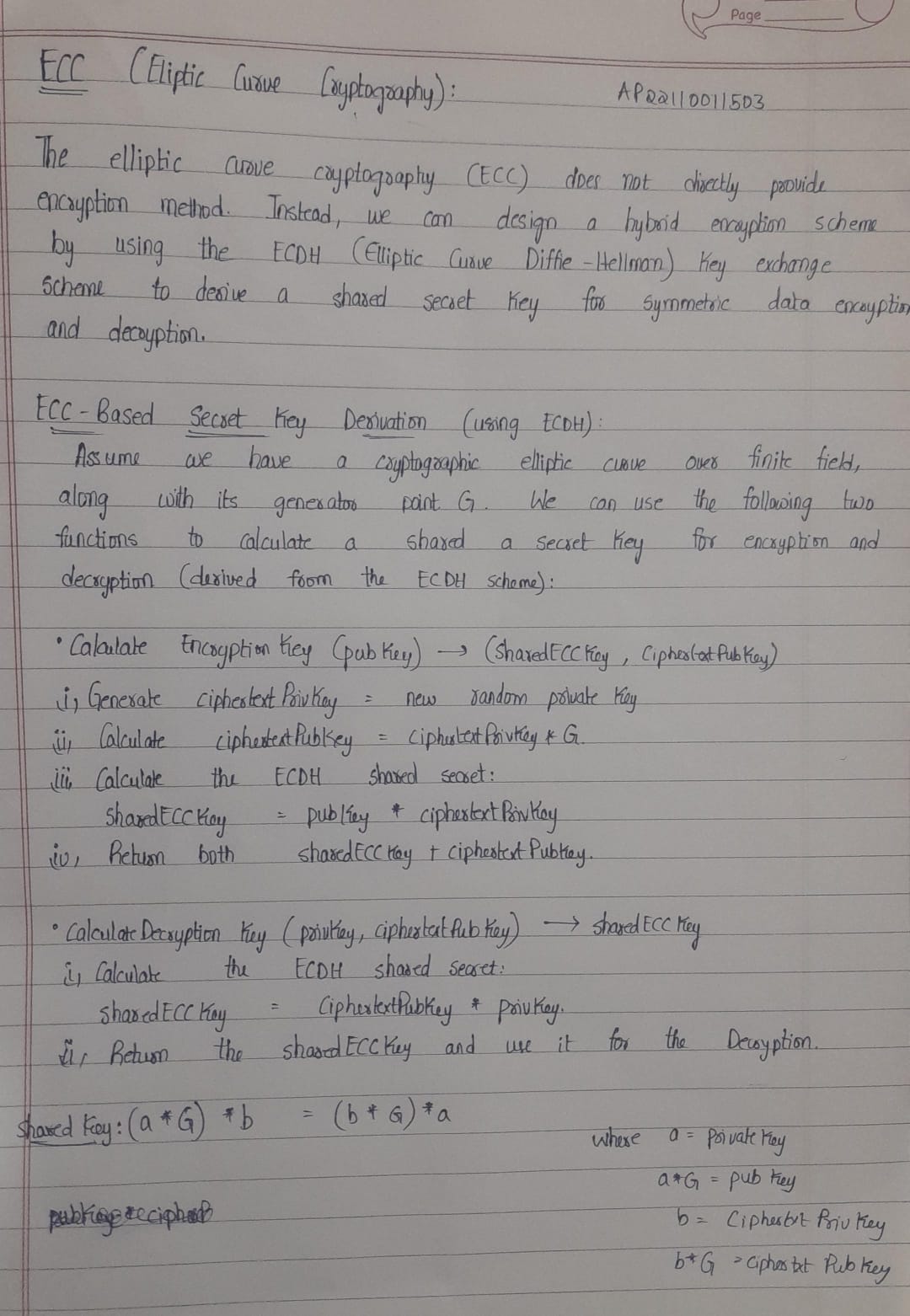
**SRM University–AP**

**Amaravati, Andhra Pradesh – 522 240, India**

1. **Question**

**Write a code to simulate the working procedure of digital signature by  using from py\_ecc.bls.ciphersuites library in multi client environment.**

1. **Algorithm Description**

****

1. **Solution**
2. pip install py\_ecc
3. from py\_ecc.bls import G2ProofOfPossession as bls
4. # Simulate 3 clients
5. NUM\_CLIENTS = 3
6. clients = {}
7. # Step 1: Each client generates a private/public key pair
8. for client\_id in range(1, NUM\_CLIENTS + 1):
9. sk = bls.KeyGen(client\_id.to\_bytes(2, 'big'))  # Simple keygen using client ID
10. pk = bls.SkToPk(sk)
11. clients[client\_id] = {
12. 'sk': sk,
13. 'pk': pk
14. }
15. # Server maintains a list of public keys
16. server\_public\_keys = {cid: data['pk'] for cid, data in clients.items()}
17. # Step 2: Clients sign messages
18. messages = {}
19. signatures = {}
20. print("\n--- Messages and Signatures ---")
21. for client\_id in clients:
22. message = f"Message from Client {client\_id}".encode('utf-8')
23. sk = clients[client\_id]['sk']
24. signature = bls.Sign(sk, message)
25. messages[client\_id] = message
26. signatures[client\_id] = signature
27. print(f"\nClient {client\_id}")
28. print(f"Message: {message.decode('utf-8')}")
29. print(f"Signature: {signature.hex()}")
30. # Step 3: Server verifies each signature
31. print("\n--- Signature Verification ---")
32. for client\_id in clients:
33. pk = server\_public\_keys[client\_id]
34. message = messages[client\_id]
35. signature = signatures[client\_id]
36. is\_valid = bls.Verify(pk, message, signature)
37. print(f"Client {client\_id}: Signature is {'valid' if is\_valid else 'invalid'}")

**Output:**

--- Messages and Signatures ---

Client 1

Message: Message from Client 1

Signature: a76dd52745d8dc093b0b85c2a191bc5444f33c475d2a0c8a588442b8b12a2b72608fff91635e58a8e059505d7c33b89d026a906ce69e48f51de15fe08c462d2efa119b17af0f2e395b1ba2f63f6ace07ae3f01d5140016e5777f913d452b25de

Client 2

Message: Message from Client 2

Signature: b84cf54e7c4bd32dc2ff659992ba88aa68dbb97326cb5aaa1526bb2da607154f09894854bcbdbadec5c4c14245243feb0b313045e0aff4842ff1fa31b1397a1c916e07b2173904c1d41291a859b5f5821d95584694bfc638f6ff5301bc6cfd0d

Client 3

Message: Message from Client 3

Signature: b6c8adab9b59d12fdd6b148369729aa39733f93fe0528e9b72fcddab72a777da34d237715aaf60162344ebb52f11af211201090f94e3e2f753a4839b72b8ca8957258effaf519722cd6c656c64dfcd8872e80e09959d5614cbef7900451ffd53

--- Signature Verification ---

Client 1: Signature is valid

Client 2: Signature is valid

Client 3: Signature is valid

1. **Code Repository:**