📋 TASK 1: FIX CP AUTHENTICATION FLOW

Priority: CRITICAL
Estimated Time: 2-3 days
Status: NOT STARTED
Dependencies: Should fix BUG-001 first (but can work on this parallel)

🎯 WHAT THIS TASK IS ABOUT

The Requirement from Correction Guide:

> "Prior to authentication and subsequent service provision, CPs must submit a registration request to record their ID and location in the system. EV_Central will use the credentials provided by EV_Registry for this purpose."

In Simple Words:

Right now, your CPs just connect to Central and start working. NO security checks!

What should happen:

1. CP asks Registry: "Give me my username and password"
2. Registry: "Here you go: username=cp_user_abc, password=secret123"
3. CP → Central: "I want to authenticate with these credentials"
4. Central → Registry: "Are these credentials valid?"
5. Registry: "Yes" or "No"
6. If yes: CP can operate
7. If no: CP gets kicked out

❌ CURRENT (BROKEN) FLOW

```
CP_Monitor starts
   ↓
CP_Monitor → Central: "REGISTER#CP#CP-001#40.5#-3.1#0.30"
   ↓
Central: "OK, you're in!"
   ↓
```

```
CP starts working
   ↓
NO SECURITY! Anyone can be a CP!
```

Problems:

- No username/password check
- Registry exists but nobody uses it
- Anyone can pretend to be a CP
- Will fail correction immediately

✅ WHAT IT SHOULD BE

```
1. CP_Monitor starts
  ↓
2. CP_Monitor → Registry API: "POST /register" with CP-001 data
  ↓
3. Registry: Returns { "username": "cp_user_abc", "password": "secret123" }
  ↓
4. CP_Monitor saves these credentials
  ↓
5. CP_Monitor → Central: "AUTHENTICATE#CP-001#cp_user_abc#secret123"
  ↓
6. Central → Registry API: "POST /verify" with username + password
  ↓
7. Registry: Returns { "valid": true }
  ↓
8. Central: "AUTHENTICATED#CP-001#encryption_key"
  ↓
9. CP_Monitor → CP_Engine: "Here's your encryption key"
  ↓
10. CP starts working with verified credentials
```

🔧 FILES YOU NEED TO MODIFY

File 1: `charging_point/ev_cp_monitor.py`

What to add:

Step 1: Add Registry Connection Method

- Before connecting to Central, connect to Registry first
- Call Registry API: `POST http://registry:5001/register`
- Send: `{"cp_id": "CP-001", "latitude": "40.5", "longitude": "-3.1", "price_per_kwh": 0.30}`
- Get back: `{"username": "cp_user_abc", "password": "secret123"}`
- Store these credentials in variables

Step 2: Modify Central Connection

- Don't send `REGISTER` anymore
- Send new message type: `AUTHENTICATE`
- Format: `AUTHENTICATE#CP-001#username#password`
- Wait for Central to respond with `AUTHENTICATED` or `DENIED`

Step 3: Handle Authentication Response

- If `AUTHENTICATED`: Save the encryption key (you'll use this in TASK 2)
- If `DENIED`: Show error and keep retrying every 30 seconds

File 2: `central/ev_central.py`

What to add:

Step 1: Add New Message Handler

- In `_process_message()`, add case for `AUTHENTICATE`
- Extract: cp_id, username, password from message
- Don't just accept it!

Step 2: Verify with Registry

- Call Registry API: `POST http://registry:5001/verify`
- Send: `{"cp_id": "CP-001", "username": "cp_user_abc", "password": "secret123"}`
- Get back: `{"valid": true}` or `{"valid": false}`

Step 3: Generate Encryption Key (for TASK 2)

- If valid: Generate unique encryption key for this CP
- Store it: `self.cp_encryption_keys[cp_id] = encryption_key`
- Send back: `AUTHENTICATED#encryption_key`
- If not valid: Send back: `DENIED#INVALID_CREDENTIALS`

Step 4: Only Allow Authenticated CPs

- Check: Before accepting any charging requests
- If CP not authenticated: Reject the request

File 3: `shared/protocol.py`

What to add:

Step 1: Add New Message Types

```python
# Add to MessageTypes class:
AUTHENTICATE = "AUTHENTICATE"        # CP → Central
AUTHENTICATED = "AUTHENTICATED"      # Central → CP (success)
```

File 4: `registry/ev_registry.py`

What to check:

Already has these endpoints:

- ✅ `POST /register` - Creates CP with credentials
- ✅ `POST /verify` - Checks if credentials are valid
- ✅ `GET /list` - Lists all registered CPs

You DON'T need to modify Registry! It already works correctly.

🧪 HOW TO TEST

Test 1: Registration with Registry

```
1. Start only Registry: docker-compose up registry
2. Use Postman or browser to test:
   POST http://localhost:5001/register
   Body: {"cp_id": "CP-999", "latitude": "40.5", "longitude": "-3.1", "price_per_kwh": 0.30}
3. Should return: {"username": "cp_user_xxx", "password": "yyy"}
4. Call: GET http://localhost:5001/list
5. Should see CP-999 in the list
```

Expected: Registry works independently

Test 2: Credential Verification

```
1. Use credentials from previous test
2. Call: POST http://localhost:5001/verify
   Body: {"cp_id": "CP-999", "username": "cp_user_xxx", "password": "yyy"}
3. Should return: {"valid": true}
4. Try wrong password:
   Body: {"cp_id": "CP-999", "username": "cp_user_xxx", "password": "WRONG"}
5. Should return: {"valid": false}
```

Expected: Registry correctly validates credentials

Test 3: Full Authentication Flow

```
1. Start full system
2. Create CP using CP Manager
3. Watch CP_Monitor logs:
   - Should see: "Connecting to Registry..."
   - Should see: "Got credentials: username=xxx"
   - Should see: "Authenticating with Central..."
   - Should see: "Authentication successful!"
4. Check Central logs:
   - Should see: "Received AUTHENTICATE from CP-001"
   - Should see: "Verifying credentials with Registry..."
   - Should see: "Credentials valid, sending AUTHENTICATED"
5. CP should now be ACTIVATED (green) in dashboard
```

Expected: Complete authentication chain works

Test 4: Failed Authentication

```
1. Manually modify CP_Monitor code to send wrong password
2. Start CP
3. Should see: "Authentication DENIED by Central"
4. CP should NOT appear as ACTIVATED
5. Central should log: "Invalid credentials for CP-001"
```

Expected: Bad credentials are rejected

📋 STEP-BY-STEP IMPLEMENTATION PLAN

Day 1: Modify CP_Monitor

Hour 1-2: Add Registry Connection

- Import requests library
- Create method `_register_with_registry()`
- Call Registry `/register` endpoint
- Parse response and save username/password
- Add error handling

Hour 3-4: Modify Central Authentication

- Change `connect_to_central()` method
- Send `AUTHENTICATE` instead of `REGISTER`
- Wait for response
- Handle `AUTHENTICATED` vs `DENIED`

Hour 5-6: Testing

- Test Registry connection works
- Test credentials are received

- Test authentication message is sent
- Debug any connection issues

Day 2: Modify Central

Hour 1-2: Add Authentication Handler

- Add `_handle_authenticate()` method
- Extract cp_id, username, password from message
- Print them to verify they arrive

Hour 3-4: Add Registry Verification

- Call Registry `/verify` endpoint
- Check if response is valid
- Handle connection errors

Hour 5-6: Generate Response

- If valid: Generate encryption key (just random for now)
- Send `AUTHENTICATED` message back
- If invalid: Send `DENIED` message
- Update CP state in memory

Day 3: Testing & Integration

Hour 1-3: End-to-End Testing

- Start full system
- Create CP and watch logs
- Verify authentication completes
- Test charging works after authentication

Hour 4-6: Edge Cases

- Test what happens if Registry is down
- Test wrong credentials
- Test multiple CPs authenticating simultaneously
- Fix any bugs found

📋 CHECKLIST

Before marking this task as complete:

- [ ] CP_Monitor connects to Registry first
- [ ] CP_Monitor receives username and password
- [ ] CP_Monitor sends AUTHENTICATE to Central
- [ ] Central receives authentication request
- [ ] Central calls Registry /verify endpoint
- [ ] Central correctly handles valid credentials
- [ ] Central correctly rejects invalid credentials
- [ ] Central sends AUTHENTICATED response
- [ ] CP can only operate after successful authentication
- [ ] All this works with multiple CPs
- [ ] Tested with Registry down (error handling)
- [ ] Tested with wrong credentials (rejection)
- [ ] Code is commented and clean
- [ ] Ready to explain to teacher

💡 WHAT TO TELL TEACHER

Teacher asks: "Explain the authentication process"

Your answer: "The authentication follows a three-party verification model. First, the CP Monitor registers with the Registry service via REST API, receiving unique credentials - a username and hashed password. These credentials are then used to authenticate with the Central system through our socket protocol. Central validates these credentials by making a verification request to the Registry. Only after successful verification does Central authorize the CP to begin operations and provides it with an encryption key for secure communication. This ensures that only registered, verified charging points can access the network, preventing unauthorized devices from operating."

⚠️ COMMON PROBLEMS

Problem 1: "Registry API not responding"

- **Fix:** Make sure Registry container is running
- Check: `docker ps | grep registry`

Problem 2: "CP_Monitor can't reach Registry"

- **Fix:** Use Docker network name: `http://registry:5001` not `localhost:5001`

Problem 3: "Credentials work in Postman but not in code"

- **Fix:** Check you're sending JSON with correct Content-Type header
- Use: `headers={'Content-Type': 'application/json'}`

Problem 4: "Central can't verify with Registry"

- **Fix:** Same as problem 2 - use Docker network names

✅ SUCCESS CRITERIA

This task is complete when:

1. ✅ CP can't operate without valid credentials
2. ✅ Registry generates and stores credentials
3. ✅ Central verifies credentials before allowing operation
4. ✅ Invalid credentials are rejected
5. ✅ You can explain the entire flow confidently
6. ✅ Teacher can see it working in logs

Status: Not started
Next Task: TASK 2 (Encryption)