📋 TASK 5: FIX KEY REVOCATION MECHANISM

Priority: MEDIUM
Estimated Time: 1 day
Status: PARTIALLY DONE
Dependencies: TASK 2 (Encryption) must be completed first

🎯 WHAT THIS TASK IS ABOUT

The Requirement:

> "These keys can be revoked by EV_Central in the event of a security vulnerability. Clicking this option will delete the keys for a specific CP, which will then be taken offline. This will force EV_CP_M to perform a new authentication, thus obtaining its new encryption keys."

In Simple Words:

Admin can revoke a CP's encryption key (security breach suspected). CP's messages stop working, CP realizes it, and re-authenticates to get a new key.

Scenario:

- Admin suspects CP-001 is compromised
- Admin clicks "Revoke key for CP-001"
- CP-001's encryption key is deleted
- CP-001's next message fails to decrypt
- Central tells CP-001: "Your key is invalid, please re-authenticate"
- CP-001 goes through authentication again
- Gets new encryption key
- Everything works again

❌ WHAT'S WRONG NOW

Currently:

- You have a menu option "revoke key"

- It doesn't actually work properly
- CP doesn't realize its key was revoked
- No re-authentication happens
- It's just a placeholder

✅ WHAT IT SHOULD DO

Full Flow:

```
1. Admin executes: "revoke CP-001"
   ↓
2. Central deletes encryption key for CP-001
   ↓
3. Central marks CP-001 as "KEY_REVOKED"
   ↓
4. CP-001 sends next encrypted message
   ↓
5. Central tries to decrypt → FAILS
   ↓
6. Central sends: "KEY_INVALID#CP-001#REVOKED"
   ↓
7. CP-001 receives error message
   ↓
8. CP-001 Monitor: "Key revoked! Re-authenticating..."
   ↓
9. CP-001 goes through full authentication again
   ↓
10. Gets new encryption key
   ↓
11. Everything works again
```

🔧 WHAT YOU NEED TO DO

Step 1: Add Revocation State to Central

File: `central/ev_central.py`

Add to `__init__`:

```python
self.revoked_cps = set()  # Track CPs with revoked keys
```

In admin command handler:

Current menu option:

```python
elif cmd.startswith("revoke"):
    # TODO: implement properly
```

Change to:

```python
```

```
elif cmd.startswith("revoke"):
    parts = cmd.split()
    if len(parts) < 2:
        print("❌ Usage: revoke <CP_ID>")
        continue

    cp_id = parts[1]

    with self.lock:
        if cp_id not in self.cp_encryption_keys:
            print(f"❌ CP {cp_id} has no key to revoke")
            continue

        # Delete encryption key
        del self.cp_encryption_keys[cp_id]

        # Mark as revoked
        self.revoked_cps.add(cp_id)

        # Update CP state
        if cp_id in self.charging_points:
            self.charging_points[cp_id]["state"] = "KEY_REVOKED"

        print(f"✅ Revoked encryption key for {cp_id}")
        print(f"   CP will need to re-authenticate")

        # Log to audit
        self.audit.log(
            entity="ADMIN",
            event_type="KEY_REVOKED",
            ip_address="localhost",
            parameters=f"target={cp_id}"
        )
```

Step 2: Detect Invalid Keys on Message Receive

File: `central/ev_central.py`

In `_process_message()` or wherever you decrypt:

Add try-catch around decryption:

```python
try:
    # Try to decrypt message
    decrypted_message = decrypt(encrypted_data, cp_encryption_key)

except DecryptionError:
    # Decryption failed - key might be revoked
    print(f"⚠️ Decryption failed for {cp_id}")

    # Check if key was revoked
    if cp_id in self.revoked_cps:
        print(f"⚠️ CP {cp_id} using revoked key")

        # Send error message
        error_msg = Protocol.encode(
            Protocol.build_message(
                "KEY_INVALID", cp_id, "REVOKED"
            )
        )

        try:
            client_socket.send(error_msg)
        except:
            pass

        return  # Don't process message

    else:
        # Some other decryption error
        print(f"❌ Decryption error for {cp_id}: {e}")
        return
```

Step 3: Handle Re-Authentication After Revocation

Still in `central/ev_central.py`

In `_handle_authenticate()` method:

Add at the start:

```python
```

```
# Check if this CP was revoked
if cp_id in self.revoked_cps:
    print(f"🔄 {cp_id} re-authenticating after key revocation")

    # Remove from revoked set
    self.revoked_cps.remove(cp_id)

    # Log re-authentication
    self.audit.log(
        entity=cp_id,
        event_type="RE_AUTHENTICATION",
        ip_address=client_ip,
        parameters="after_key_revocation"
    )
```

Then continue with normal authentication:

- Verify credentials with Registry
- Generate NEW encryption key
- Send AUTHENTICATED with new key
- CP is back in business!

Step 4: Modify CP to Handle Key Revocation

File: `charging_point/ev_cp_engine.py` or `ev_cp_monitor.py`

Add message handler for KEY_INVALID:

In message receiving loop:

```
python
```

```python
if msg_type == "KEY_INVALID":
    reason = fields[2] if len(fields) > 2 else "UNKNOWN"

    print(f"\n⚠⚠⚠ KEY REVOKED BY CENTRAL ⚠⚠⚠")
    print(f"Reason: {reason}")
    print(f"Re-authenticating in 5 seconds...")

    # Mark key as invalid
    self.encryption_key = None

    # Wait a bit
    time.sleep(5)

    # Re-authenticate
    self._re_authenticate()
```

Add method `_re_authenticate()`:

```python
```

```
def _re_authenticate(self):
    """Re-authenticate with Central after key revocation"""
    print(f"[{self.cp_id}] Starting re-authentication...")

    # Get credentials from Registry again
    credentials = self._fetch_credentials_from_registry()

    if not credentials:
        print(f"[{self.cp_id}] ❌ Failed to get credentials")
        return False

    # Authenticate with Central
    success = self._authenticate_with_central(
        credentials['username'],
        credentials['password']
    )

    if success:
        print(f"[{self.cp_id}] ✅ Re-authentication successful!")
        print(f"[{self.cp_id}] Got new encryption key")
        return True
    else:
        print(f"[{self.cp_id}] ❌ Re-authentication failed")
        return False
```

Step 5: Add New Message Type

File: shared/protocol.py

Add to MessageTypes class:

python

```
KEY_INVALID = "KEY_INVALID"  # Central → CP (key revoked)
```

🧪 HOW TO TEST

Test 1: Revoke Key

1. Start full system
2. Create CP-001
3. Start charging (to verify encryption works)
4. Attach to Central admin: docker attach evcharging_central
5. Type: revoke CP-001
6. Should see: "✅ Revoked encryption key for CP-001"

Expected: Key deleted, CP marked as revoked

Test 2: CP Detects Revocation

1. After revoking key (from Test 1)
2. CP tries to send next message (HEARTBEAT or SUPPLY_UPDATE)
3. Watch CP logs:
   - Should see: "⚠⚠⚠ KEY REVOKED BY CENTRAL ⚠⚠⚠"
   - Should see: "Re-authenticating in 5 seconds..."
4. Watch Central logs:
   - Should see: "Decryption failed for CP-001"
   - Should see: "Sent KEY_INVALID to CP-001"

Expected: CP detects revocation immediately

Test 3: Re-Authentication

1. After CP detects revocation
2. Wait 5 seconds
3. Watch CP logs:
   - Should see: "Starting re-authentication..."
   - Should see: "Connecting to Registry..."
   - Should see: "Authenticating with Central..."
   - Should see: "✅ Re-authentication successful!"
4. Watch Central logs:
   - Should see: "🔄 CP-001 re-authenticating after key revocation"
   - Should see: "New encryption key generated"

Expected: CP successfully re-authenticates

Test 4: Normal Operation Resumes

```
1. After re-authentication complete
2. Try to charge vehicle at CP-001
3. Should work normally
4. Check logs show encrypted messages working
```

Expected: Everything works with new key

Test 5: Audit Log

```
1. Check data/audit_log.txt
2. Should see entries for:
   - KEY_REVOKED (admin action)
   - RE_AUTHENTICATION (CP action)
3. Timestamps should be ~5 seconds apart
```

Expected: All events logged

📋 STEP-BY-STEP IMPLEMENTATION

Hour 1-2: Modify Central for Revocation

- Add `revoked_cps` set
- Implement revoke command
- Test key deletion works
- Test CP state changes

Hour 3-4: Add Decryption Error Detection

- Add try-catch in message processing
- Detect revoked keys
- Send KEY_INVALID message

- Test message is sent

Hour 5-6: Modify CP for Re-Authentication

- Add KEY_INVALID handler
- Implement `_re_authenticate()` method
- Test CP detects revocation
- Test re-authentication works

Hour 7-8: Integration & Edge Cases

- Test complete revoke→detect→re-auth flow
- Test multiple CPs with different keys
- Test revoking key during charging (should complete, then re-auth)
- Test revoking already revoked key
- Fix any bugs
- Add audit logging

📋 CHECKLIST

Before marking complete:

- [ ] Added revoked_cps tracking to Central
- [ ] Implemented revoke command in admin menu
- [ ] Key is deleted when revoked
- [ ] Decryption errors are detected
- [ ] KEY_INVALID message sent to CP
- [ ] CP detects KEY_INVALID message
- [ ] CP waits 5 seconds before re-auth
- [ ] CP re-authenticates automatically
- [ ] New encryption key generated
- [ ] Normal operation resumes
- [ ] Works with multiple CPs
- [ ] Audit log entries created
- [ ] Tested edge cases
- [ ] Ready to demonstrate to teacher

💡 WHAT TO TELL TEACHER

Teacher says: "Show me key revocation"

Your demonstration:

1. Start system with CP running
2. Show charging works (proves encryption working)
3. Attach to Central admin console
4. Type: "revoke CP-001"
5. Show Central logs: key deleted
6. Wait ~5 seconds
7. Show CP logs: detects revocation, re-authenticating
8. Show Central logs: accepts re-authentication
9. Try charging again: works with new key
10. Show audit log: KEY_REVOKED and RE_AUTHENTICATION events

Your explanation: "Key revocation is a critical security feature for when we detect potential compromise. The administrator can immediately invalidate a CP's encryption key through the Central console. The next encrypted message from that CP will fail to decrypt, triggering the detection mechanism. Central sends a KEY_INVALID notification, causing the CP Monitor to initiate an automatic re-authentication sequence. The CP retrieves fresh credentials from the Registry and obtains a new encryption key from Central. This entire process happens transparently without manual intervention, ensuring both security and service continuity. The operation is fully audited with timestamps and parameters."

⚠️ COMMON PROBLEMS

Problem 1: "CP doesn't detect revocation"

- **Fix:** Make sure KEY_INVALID message is actually sent
- Check CP is listening for that message type
- Verify socket connection still alive

Problem 2: "Re-authentication fails"

- **Fix:** Check CP can still reach Registry
- Verify credentials still valid in Registry
- Make sure Central isn't rejecting re-auth

Problem 3: "CP gets stuck in loop"

- **Fix:** Add counter to prevent infinite re-auth attempts
- After 3 failures, give up and wait for manual intervention

Problem 4: "Revoke during charging breaks everything"

- **Fix:** Don't revoke immediately if CP is charging
- Queue revocation for after charge completes
- Or: Complete charge with old key, then revoke

✅ SUCCESS CRITERIA

This task is complete when:

1. ✅ Admin can revoke CP key via command
2. ✅ Key is immediately deleted from Central
3. ✅ CP's next message fails to decrypt
4. ✅ Central sends KEY_INVALID notification
5. ✅ CP detects revocation automatically
6. ✅ CP re-authenticates without manual intervention
7. ✅ New key is generated and used
8. ✅ Normal operation resumes
9. ✅ All events logged in audit log
10. ✅ Can demonstrate to teacher smoothly

Status: Not started
Depends on: TASK 2 (Encryption) must be done first
Final Task: This completes all critical security requirements!