Your screenshots demonstrate a **Diffie-Hellman (DH) Key Exchange simulation** between two parties, **Alice (Host.py)** and **Bob (Client.py)**, written in Python. Here's a breakdown of what each image shows and how the system works:

**🖼️ Image 1: Demo\_Key Exchange between Host and Client.png**

This image shows a successful key exchange session between Alice and Bob.

**✅ Flow Explanation:**

1. **Alice starts Host.py (left terminal):**
   * Generates and sends DH parameters (p, g, and g^a mod p) to Bob.
   * Waits for Bob's public key (g^b mod p).
2. **Bob runs Client.py (right terminal):**
   * Enters the password: Password123.
   * Receives DH parameters and computes the shared secret key using Alice’s g^a mod p.
   * Sends back g^b mod p.
3. **Shared Secret Computation:**
   * Both Alice and Bob independently compute the same **shared key**:

makefile

CopyEdit

Alice: 303ff224ac015e12ea0c258a0b0d70ad0069e3d7

Bob: 303ff224ac015e12ea0c258a0b0d70ad0069e3d7

1. **Mutual Authentication using Nonces:**
   * Alice sends nonce a ➝ Bob
   * Bob replies with nonce a + 1 and a new nonce b
   * Alice verifies nonce a + 1, sends back nonce b + 1
   * Bob verifies nonce b + 1
   * ✅ **Handshake complete**
   * ✅ **Login Successful**
2. **Chat begins:**

vbnet

CopyEdit

Alice: Hi

Bob: Hi

Alice: Whats the secret key

Bob: secret is Password123

Alice: okay thanks

Bob: np

**🖼️ Image 2: Demo\_Setup\_Of\_DFKeyExchange.png**

This image shows the **setup process and supporting files** for the key exchange.

**✅ Breakdown:**

1. **Top Left Terminal:**
   * setup.py run with password input: password123
   * Generates DH parameters and hashes the password
   * Saves all of this to Alice/secret.txt
2. **Right Side (Finder Window):**
   * Shows project directory: 84867599\_RohitPanda\_CSCI368\_A1
   * Inside Alice/, we see:
     + Host.py — main server script
     + secret.txt — stores generated DH parameters and password hash
3. **Bottom (README.txt):**
   * Project instructions:
     + Run setup.py first (generates DH values & saves password hash)
     + Run Host.py (waits for connection from Bob)
     + Run Client.py (enters password & starts handshake)

**🔐 Summary of Key Exchange Workflow**

| **Step** | **Party** | **Action** |
| --- | --- | --- |
| 1 | setup.py | Generates DH params + hash(password), saves to secret.txt |
| 2 | Alice (Host.py) | Sends p, g, and g^a mod p |
| 3 | Bob (Client.py) | Receives DH params, computes g^b mod p, sends it |
| 4 | Both | Independently compute the shared key using DH |
| 5 | Alice/Bob | Exchange and verify nonces for mutual authentication |
| 6 | Both | Proceed to secure communication (chat) |

**✅ Security Concepts Demonstrated**

* 🔑 **Diffie-Hellman Key Exchange**: Securely establishes a shared secret over an insecure channel.
* 🔒 **Password Authentication**: Validates user with hashed password.
* 🔁 **Mutual Authentication via Nonces**: Confirms both parties are legitimate.
* 🔐 **Encrypted Communication Potential**: Though the chat is in plaintext now, this setup could be extended to use the shared key for message encryption.

Would you like:

* A diagram to visualize this flow?
* Or suggestions on how to **extend** this to use actual message encryption (e.g., AES with shared key)?