```
        1. Check file for malware using hash
        2. Hash a string using SHA-256
        3. Encrypt a message
        4. Decrypt a message
        5. Exit

Enter your choice: 1
Enter the path to the file to check: /Users/danie/Desktop/python/WORK/CyberOps.py
The file /Users/danie/Desktop/python/WORK/CyberOps.py seems clean (based on our limited hash list).

        1. Check file for malware using hash
        2. Hash a string using SHA-256
        3. Encrypt a message
        4. Decrypt a message
        5. Exit

Enter your choice: 2
Enter string to hash: This is a test
Hashed String: c7be1ed902fb8dd4d48997c6452f5d7e509fbcdbe2808b16bcf4edce4c07d14e

        1. Check file for malware using hash
        2. Hash a string using SHA-256
        3. Encrypt a message
        4. Decrypt a message
        5. Exit

Enter your choice: 3

            a. Use a specific key
            b. Generate a strong password as key

Choose an option: b
Generated Key (in base64 format): UGdnYntxXnYzRlZsUTcsIQ==
Enter message to encrypt: This is supposed to be a secret.
Nonce: 0Kf/zLuI/q/ZQGKOXenL0A==
Encrypted Message: ++46+7vqtebPK6e1VEzCe9ZDATVaTpukfGnKZ36HSCk=
Encrypted Message saved to file: HflSQRbRVoMTWPKe.txt

        1. Check file for malware using hash
        2. Hash a string using SHA-256
        3. Encrypt a message
        4. Decrypt a message
        5. Exit

Enter your choice: 4
Enter the nonce: 0Kf/zLuI/q/ZQGKOXenL0A==
Enter the encrypted message (the gibberish string, not the original text): ++46+7vqtebPK6e1VEzCe9ZDATVaTpukfGnKZ36HSCk=
Enter the key (in base64 format): UGdnYntxXnYzRlZsUTcsIQ==
Decrypted Message: This is supposed to be a secret.

        1. Check file for malware using hash
        2. Hash a string using SHA-256
        3. Encrypt a message
        4. Decrypt a message
        5. Exit
```