Exercise-1

**1)Download the X.509 public certificate from https://weakssl.com; (2 points)**

openssl s\_client -connect weakssl.com:443 -showcerts > weakssl\_cert.pem

(curl -v --insecure https://weakssl.com 2>&1 | tee weakssl\_cert.pem1)

A screenshot of a computer

Description automatically generated

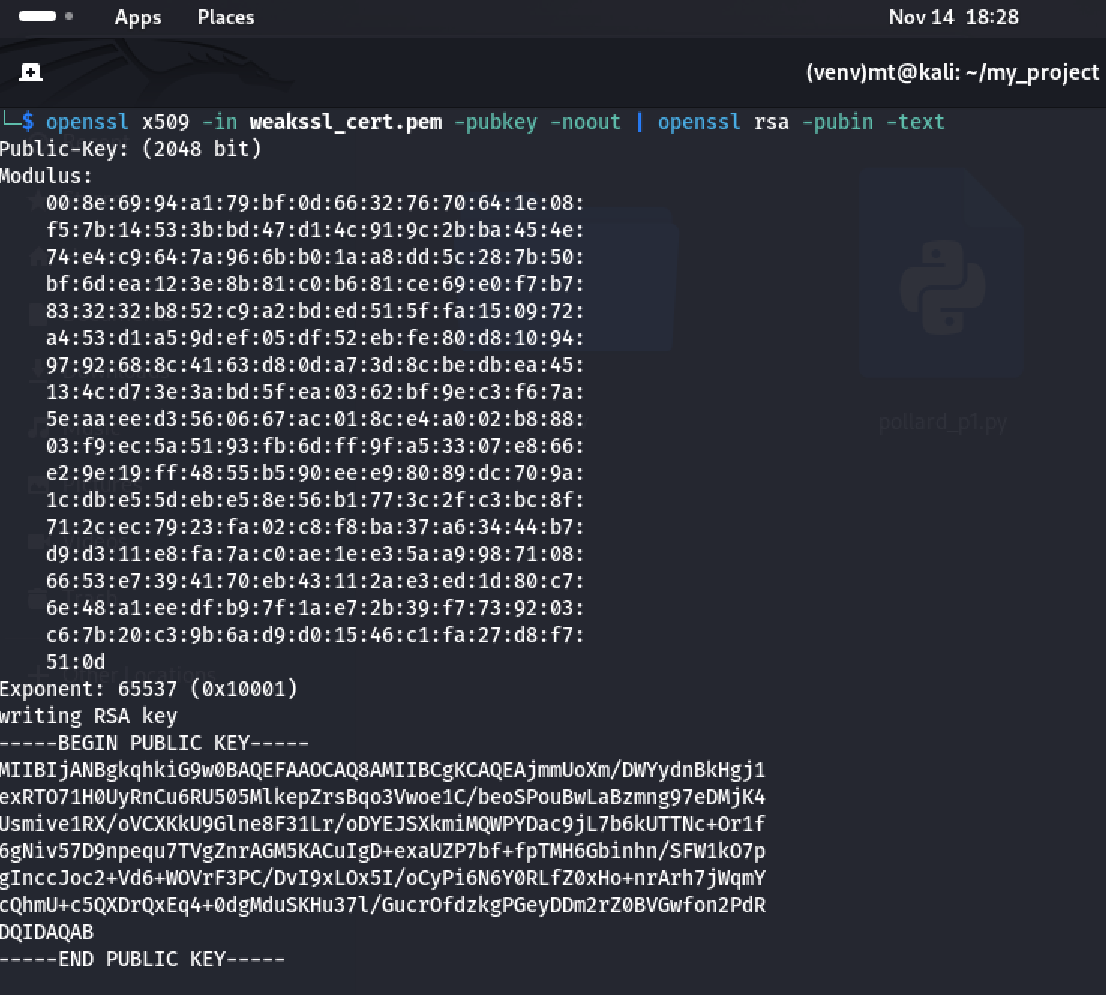
**Explanation**:

* The command openssl s\_client connects to the specified server (in this case, weakssl.com) on port 443 (the default HTTPS port).
* The -showcerts flag tells openssl to display the entire certificate chain.
* The output is saved to a file called weakssl\_cert.pem.

**Result**: This file contains the **server's SSL certificate** in PEM format, which includes the public RSA key.

**2)Extract the modulus *N* and attempt to factorize it using the Pollard P-1 algorithm; (8 points)**

openssl x509 -in weakssl\_cert.pem -pubkey -noout | openssl rsa -pubin -text

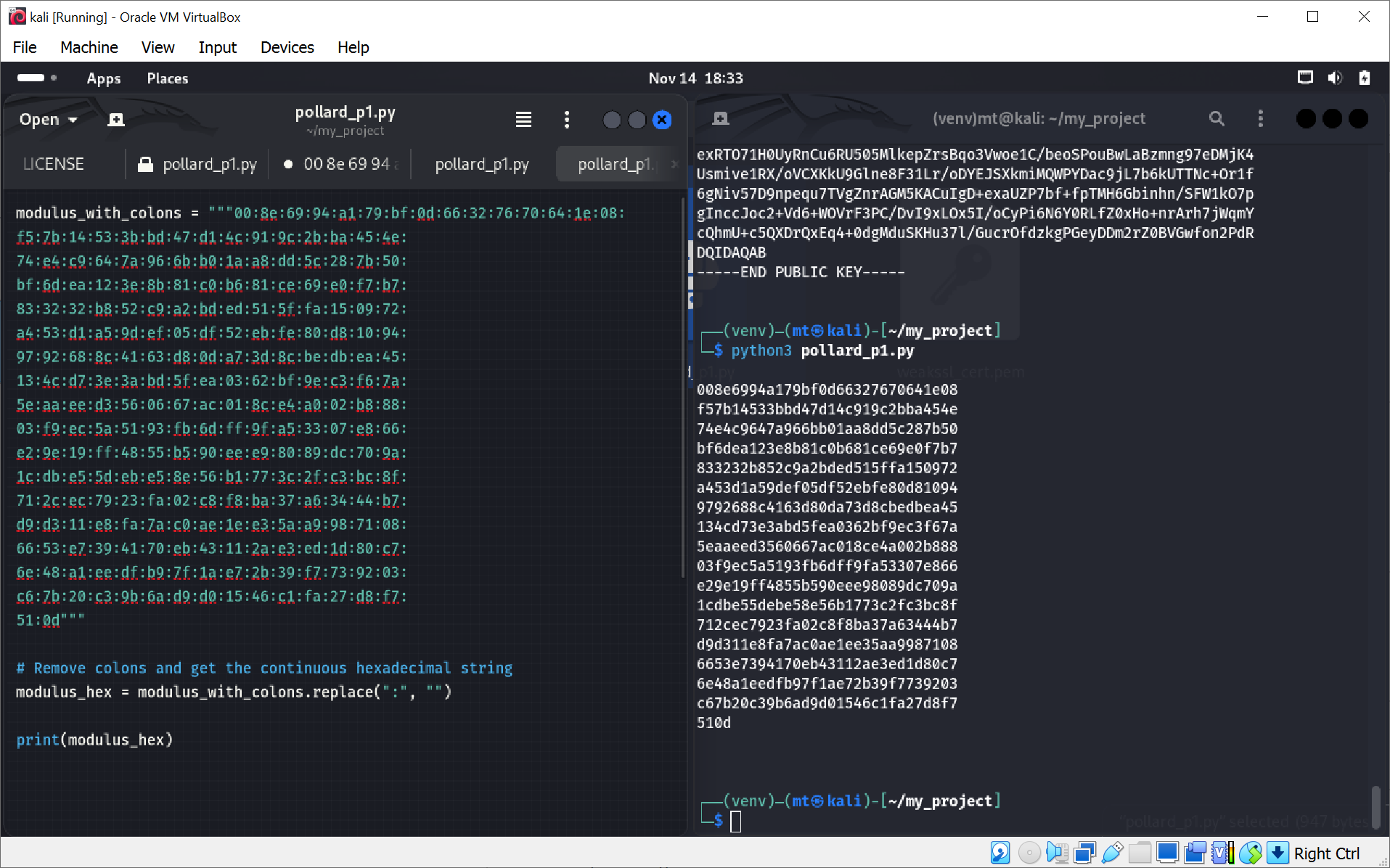


**Explanation**:

* The openssl x509 -in weakssl\_cert.pem -pubkey -noout extracts the public key from the PEM file weakssl\_cert.pem.
* The openssl rsa -pubin -text command takes this public key and prints it in readable format, which includes the modulus N, the public exponent e, and other key information.

**Result**: The modulus N is printed in hexadecimal format, with colons separating the bytes.

* The modulus extracted from the certificate will have colons (:) separating the hex values. To process it, remove the colons and get a continuous hexadecimal string:



**Explanation**: The modulus N extracted from the certificate is in hexadecimal form with colons separating the bytes. This code removes the colons to get a continuous hexadecimal string.

**Result**: This gives the **cleaned-up modulus** as a continuous hexadecimal string.

* Convert the cleaned-up hexadecimal string into an integer that will represent the modulus N

A screenshot of a computer

Description automatically generated

**Explanation**: The int() function is used to convert the cleaned hexadecimal string to an integer. This gives us the modulus N, which is required for RSA key computations.

**Result**: This prints the **integer value of N**.

A screenshot of a computer

Description automatically generated

**Code Documentation:**

1. **Purpose**: This code implements Pollard's P-1 algorithm to factorize a given modulus N into its prime factors p and q.
2. **Pollard's P-1 Algorithm**: The function pollard\_p1() computes ai mod  Nfor increasing values of iii and checks the **GCD** of ai−1and N to find a non-trivial factor of N.
3. **Factorization**: The factorize\_modulus() function applies Pollard’s P-1 repeatedly to find factors of N. It continues dividing N by the found factors until it reaches a prime number.
4. **Input**: The input is a modulus N for which you want to find the prime factors.
5. **Output**: The function prints the prime factors p and q of N if the factorization is successful. If Pollard's P-1 fails to find factors, it will indicate that factorization was unsuccessful.

**3)Reconstruct the original private key using the obtained prime numbers; (5 points)**

A screenshot of a computer

Description automatically generated

**Code Documentation:**

1. **Purpose**: The code factors the modulus N of an RSA public key using Pollard's P-1 algorithm and then reconstructs the private key exponent d using the prime factors p and q.
2. **Pollard's P-1 Algorithm**: The function pollard\_p1() uses Pollard's P-1 method to find a factor of N. It computes ai mod   for increasing values of i and checks the GCD of ai −1and N. If a non-trivial factor is found, it returns the factor.
3. **Factorization**: The factorize\_modulus() function repeatedly calls pollard\_p1() to find factors of N, then divides N by the found factors. It continues until the remaining number is prime, at which point the factors are returned.
4. **Private Key Calculation**: The function compute\_private\_key() calculates the private exponent d from the prime factors p and q using the formula:

d=e−1mod  φ(N)

where φ(N)=(p−1)(q−1) is Euler's Totient.

1. **Input**: You provide the modulus N and the public exponent e (usually 65537).
2. **Output**: The code prints the private key exponent d after factoring N into its prime components p and q.

All codes are in the LINK