RSA

Q1)

Computer Security CW2 > RSA > C rsa_q1.c

```c
#include <stdio.h>
#include <openssl/bn.h>

int main() {
    BN_CTX *ctx = BN_CTX_new();

    BIGNUM *p = BN_new();
    BIGNUM *q = BN_new();
    BIGNUM *n = BN_new();
    BIGNUM *phi = BN_new();
    BIGNUM *e = BN_new();
    BIGNUM *d = BN_new();
    BIGNUM *M = BN_new();
    BIGNUM *C = BN_new();
    BIGNUM *temp1 = BN_new();
    BIGNUM *temp2 = BN_new();

    // Set values
    BN_dec2bn(&p, "71");
    BN_dec2bn(&q, "97");
    BN_dec2bn(&n, "6887");
    BN_dec2bn(&e, "143");
    BN_dec2bn(&M, "1234");

    // phi = (p-1)(q-1)
    BN_sub(temp1, p, BN_value_one()); // temp1 = p - 1
    BN_sub(temp2, q, BN_value_one()); // temp2 = q - 1
    BN_mul(phi, temp1, temp2, ctx);   // phi = (p-1)(q-1)

    // Compute d = e^-1 mod phi
    BN_mod_inverse(d, e, phi, ctx);

    // Compute C = M^e mod n
    BN_mod_exp(C, M, e, n, ctx);

    // Print results
    char *d_str = BN_bn2dec(d);
    char *C_str = BN_bn2dec(C);

    printf("Decryption key d: %s\n", d_str);
    printf("Ciphertext C: %s\n", C_str);

    // Free
    OPENSSL_free(d_str);
    OPENSSL_free(C_str);

    BN_free(p); BN_free(q); BN_free(n); BN_free(phi); BN_free(e); BN_free(d);
    BN_free(M); BN_free(C); BN_free(temp1); BN_free(temp2);
    BN_CTX_free(ctx);

    return 0;
}
```

Output :

```
>>> Compiling rsa_q1.c
>>> Running rsa_q1
---------------------------------
Decryption key d: 47
Ciphertext C: 4347
---------------------------------
```

Q2)

Computer Security CW2 > RSA > C rsa_q2.c

```c
#include <stdio.h>
#include <openssl/bn.h>

int main() {
    BN_CTX *ctx = BN_CTX_new();

    BIGNUM *p = BN_new();
    BIGNUM *q = BN_new();
    BIGNUM *n = BN_new();
    BIGNUM *phi = BN_new();
    BIGNUM *d = BN_new();
    BIGNUM *e = BN_new();
    BIGNUM *M = BN_new();
    BIGNUM *C = BN_new();
    BIGNUM *temp1 = BN_new();
    BIGNUM *temp2 = BN_new();

    // Set known values
    BN_dec2bn(&p, "223");
    BN_dec2bn(&q, "311");
    BN_dec2bn(&n, "69353");
    BN_dec2bn(&d, "29401");
    BN_dec2bn(&M, "12345");

    // phi = (p - 1) * (q - 1)
    BN_sub(temp1, p, BN_value_one()); // temp1 = p - 1
    BN_sub(temp2, q, BN_value_one()); // temp2 = q - 1
    BN_mul(phi, temp1, temp2, ctx);   // phi = (p-1)(q-1)

    // e = d^-1 mod phi
    BN_mod_inverse(e, d, phi, ctx);

    // Encrypt C = M^e mod n
    BN_mod_exp(C, M, e, n, ctx);

    // Print results
    char *e_str = BN_bn2dec(e);
    char *C_str = BN_bn2dec(C);

    printf("Encryption key e: %s\n", e_str);
    printf("Ciphertext C: %s\n", C_str);

    // Free memory
    OPENSSL_free(e_str);
    OPENSSL_free(C_str);

    BN_free(p); BN_free(q); BN_free(n); BN_free(phi); BN_free(e); BN_free(d);
    BN_free(M); BN_free(C); BN_free(temp1); BN_free(temp2);
    BN_CTX_free(ctx);

    return 0;
}
```

Output :

```
>>> Compiling rsa_q2.c
>>> Running rsa_q2
---------------------------------
Encryption key e: 4321
Ciphertext C: 58684
---------------------------------
```

Q3)

Computer Security CW2 > RSA > C rsa_q3.c

```c
#include <stdio.h>
#include <openssl/bn.h>

int main() {
    BN_CTX *ctx = BN_CTX_new();

    BIGNUM *p = BN_new();
    BIGNUM *q = BN_new();
    BIGNUM *n = BN_new();
    BIGNUM *phi = BN_new();
    BIGNUM *e = BN_new();
    BIGNUM *d = BN_new();
    BIGNUM *C = BN_new();
    BIGNUM *M = BN_new();
    BIGNUM *temp1 = BN_new();
    BIGNUM *temp2 = BN_new();

    // Set values
    BN_dec2bn(&p, "37");
    BN_dec2bn(&q, "67");
    BN_dec2bn(&e, "169");
    BN_dec2bn(&C, "1744");

    // n = p * q
    BN_mul(n, p, q, ctx);

    // phi = (p - 1)(q - 1)
    BN_sub(temp1, p, BN_value_one());
    BN_sub(temp2, q, BN_value_one());
    BN_mul(phi, temp1, temp2, ctx);

    // d = e^-1 mod phi
    BN_mod_inverse(d, e, phi, ctx);

    // M = C^d mod n
    BN_mod_exp(M, C, d, n, ctx);

    // Print results
    char *n_str = BN_bn2dec(n);
    char *phi_str = BN_bn2dec(phi);
    char *d_str = BN_bn2dec(d);
    char *M_str = BN_bn2dec(M);

    printf("n (modulus)      : %s\n", n_str);
    printf("phi(n)           : %s\n", phi_str);
    printf("Decryption key d : %s\n", d_str);
    printf("Decrypted M      : %s\n", M_str);

    // Free memory
    OPENSSL_free(n_str);
    OPENSSL_free(phi_str);
    OPENSSL_free(d_str);
    OPENSSL_free(M_str);

    BN_free(p); BN_free(q); BN_free(n); BN_free(phi); BN_free(e); BN_free(d);
    BN_free(C); BN_free(M); BN_free(temp1); BN_free(temp2);
    BN_CTX_free(ctx);

    return 0;
}
```

Output :

```
>>> Compiling rsa_q3.c
>>> Running rsa_q3
---------------------------------
n (modulus)      : 2479
phi(n)           : 2376
Decryption key d : 2137
Decrypted M      : 1234
---------------------------------
```

Q4)

Computer Security CW2 > RSA > C rsa_q4.c

```c
#include <stdio.h>
#include <openssl/bn.h>

int main() {
    BN_CTX *ctx = BN_CTX_new();

    BIGNUM *S = BN_new();
    BIGNUM *n = BN_new();
    BIGNUM *e = BN_new();
    BIGNUM *M_verify = BN_new();

    // Set known values
    BN_dec2bn(&S, "1459");
    BN_dec2bn(&n, "2479");
    BN_dec2bn(&e, "169");

    // Compute M' = S^e mod n
    BN_mod_exp(M_verify, S, e, n, ctx);

    // Print result
    char *M_str = BN_bn2dec(M_verify);
    printf("Verified Message from Signature: %s\n", M_str);

    // Free
    OPENSSL_free(M_str);
    BN_free(S); BN_free(n); BN_free(e); BN_free(M_verify);
    BN_CTX_free(ctx);

    return 0;
}
```

Output :

```
>>> Compiling rsa_q4.c
>>> Running rsa_q4
--------------------------------
Verified Message from Signature: 1233
--------------------------------
```

Comments : The signature S=1459 was verified using Alice's public key (e=169,n=2479), resulting in M'=1233.

Since the expected message was M=1234, and the recovered message from the signature does **not** match (M'≠M), we conclude that:

This signature was not generated using Alice's private key.
Therefore, it cannot be considered authentic, and may have been forged or created with a different key.

Diffie-Hellman

Q1)

Computer Security CW2 > Diffie-Hellman > C dh_q1.c

```c
#include <stdio.h>
#include <openssl/bn.h>

int main() {
    BN_CTX *ctx = BN_CTX_new();

    BIGNUM *p = BN_new();
    BIGNUM *g = BN_new();
    BIGNUM *xa = BN_new();
    BIGNUM *xb = BN_new();
    BIGNUM *ya = BN_new();
    BIGNUM *yb = BN_new();
    BIGNUM *kab1 = BN_new();
    BIGNUM *kab2 = BN_new();

    // Set values
    BN_dec2bn(&p, "773");
    BN_dec2bn(&g, "200");
    BN_dec2bn(&xa, "333");
    BN_dec2bn(&xb, "603");

    // y_a = g^x_a mod p
    BN_mod_exp(ya, g, xa, p, ctx);

    // y_b = g^x_b mod p
    BN_mod_exp(yb, g, xb, p, ctx);

    // K_ab = y_b^x_a mod p
    BN_mod_exp(kab1, yb, xa, p, ctx);

    // (Optional: K_ab from other side to double-check)
    BN_mod_exp(kab2, ya, xb, p, ctx);

    // Print results
    char *ya_str = BN_bn2dec(ya);
    char *yb_str = BN_bn2dec(yb);
    char *kab_str = BN_bn2dec(kab1);
    char *kab2_str = BN_bn2dec(kab2);

    printf("Alice's public key (y_a): %s\n", ya_str);
    printf("Bob's public key (y_b): %s\n", yb_str);
    printf("Shared secret key (K_ab): %s\n", kab_str);
    printf("Verification (K_ab from other side): %s\n", kab2_str);

    // Cleanup
    OPENSSL_free(ya_str);
    OPENSSL_free(yb_str);
    OPENSSL_free(kab_str);
    OPENSSL_free(kab2_str);
    BN_free(p); BN_free(g); BN_free(xa); BN_free(xb);
    BN_free(ya); BN_free(yb); BN_free(kab1); BN_free(kab2);
    BN_CTX_free(ctx);

    return 0;
}
```

Output:

```
>>> Compiling dh_q1.c
>>> Running dh_q1
---------------------------------
Alice's public key (y_a): 164
Bob's public key (y_b): 405
Shared secret key (K_ab): 476
Verification (K_ab from other side): 476
---------------------------------
```

Q2)

Computer Security CW2 > Diffie-Hellman > C dh_q2.c

```c
#include <stdio.h>
#include <openssl/bn.h>

int main() {
    BN_CTX *ctx = BN_CTX_new();

    BIGNUM *p = BN_new();
    BIGNUM *g = BN_new();
    BIGNUM *xa = BN_new();
    BIGNUM *xb = BN_new();
    BIGNUM *ya = BN_new();
    BIGNUM *yb = BN_new();
    BIGNUM *kab1 = BN_new();
    BIGNUM *kab2 = BN_new();

    // Set input values for DH Q2
    BN_dec2bn(&p, "1553");
    BN_dec2bn(&g, "307");
    BN_dec2bn(&xa, "1333");
    BN_dec2bn(&xb, "807");

    // Compute public keys
    BN_mod_exp(ya, g, xa, p, ctx); // y_a = g^x_a mod p
    BN_mod_exp(yb, g, xb, p, ctx); // y_b = g^x_b mod p

    // Compute shared keys
    BN_mod_exp(kab1, yb, xa, p, ctx); // K_ab = y_b^x_a mod p
    BN_mod_exp(kab2, ya, xb, p, ctx); // K_ab (from other side)

    // Output
    char *ya_str = BN_bn2dec(ya);
    char *yb_str = BN_bn2dec(yb);
    char *kab1_str = BN_bn2dec(kab1);
    char *kab2_str = BN_bn2dec(kab2);

    printf("Alice's public key (y_a): %s\n", ya_str);
    printf("Bob's public key (y_b): %s\n", yb_str);
    printf("Shared secret key (K_ab): %s\n", kab1_str);
    printf("Verification (K_ab from other side): %s\n", kab2_str);

    // Cleanup
    OPENSSL_free(ya_str);
    OPENSSL_free(yb_str);
    OPENSSL_free(kab1_str);
    OPENSSL_free(kab2_str);
    BN_free(p); BN_free(g); BN_free(xa); BN_free(xb);
    BN_free(ya); BN_free(yb); BN_free(kab1); BN_free(kab2);
    BN_CTX_free(ctx);

    return 0;
}
```

Output:

```
>>> Compiling dh_q2.c
>>> Running dh_q2
---------------------------------
Alice's public key (y_a): 1143
Bob's public key (y_b): 124
Shared secret key (K_ab): 245
Verification (K_ab from other side): 245
---------------------------------
```

El-Gamal

Q1)

Computer Security CW2 > El-Gamal > C eg_q1.c

```c
#include <stdio.h>
#include <openssl/bn.h>

int main() {
    BN_CTX *ctx = BN_CTX_new();

    BIGNUM *p = BN_new();
    BIGNUM *g = BN_new();
    BIGNUM *xa = BN_new();  // Alice's ephemeral message key
    BIGNUM *xb = BN_new();  // Bob's secret key
    BIGNUM *yb = BN_new();  // Bob's public key
    BIGNUM *K = BN_new();   // Shared secret
    BIGNUM *C1 = BN_new();
    BIGNUM *C2 = BN_new();
    BIGNUM *M = BN_new();
    BIGNUM *temp = BN_new();

    // Set values
    BN_dec2bn(&p, "773");
    BN_dec2bn(&g, "200");
    BN_dec2bn(&xa, "333");  // Ephemeral key generated by Alice (sender)
    BN_dec2bn(&xb, "603");  // Bob's private key (receiver)
    BN_dec2bn(&M, "321");

    // Compute Bob's public key: y_b = g^x_b mod p
    BN_mod_exp(yb, g, xb, p, ctx);

    // C1 = g^x_a mod p
    BN_mod_exp(C1, g, xa, p, ctx);

    // K = y_b^x_a mod p
    BN_mod_exp(K, yb, xa, p, ctx);

    // C2 = (M * K) mod p
    BN_mul(temp, M, K, ctx);
    BN_mod(C2, temp, p, ctx);

    // Output
    char *yb_str = BN_bn2dec(yb);
    char *C1_str = BN_bn2dec(C1);
    char *C2_str = BN_bn2dec(C2);
    char *K_str = BN_bn2dec(K);

    printf("Bob's public key (y_b): %s\n", yb_str);
    printf("Ephemeral key (x_a): 333\n");
    printf("Ephemeral message key (K): %s\n", K_str);
    printf("Ciphertext (C1, C2): (%s, %s)\n", C1_str, C2_str);

    // Cleanup
    OPENSSL_free(yb_str);
    OPENSSL_free(C1_str);
    OPENSSL_free(C2_str);
    OPENSSL_free(K_str);
    BN_free(p); BN_free(g); BN_free(xa); BN_free(xb);
    BN_free(yb); BN_free(K); BN_free(C1); BN_free(C2); BN_free(M); BN_free(temp);
    BN_CTX_free(ctx);

    return 0;
}
```

Output:

```
>>> Compiling eg_q1.c
>>> Running eg_q1
---------------------------------
Bob's public key (y_b): 405
Ephemeral key (x_a): 333
Ephemeral message key (K): 476
Ciphertext (C1, C2): (164, 515)
---------------------------------
```

Q2)

Computer Security CW2 > El-Gamal > C eg_q2.c

```c
#include <stdio.h>
#include <openssl/bn.h>

int main() {
    BN_CTX *ctx = BN_CTX_new();

    BIGNUM *p = BN_new();
    BIGNUM *xb = BN_new();      // Bob's private key
    BIGNUM *C1 = BN_new();
    BIGNUM *C2 = BN_new();
    BIGNUM *K = BN_new();       // Shared key
    BIGNUM *Kinv = BN_new();    // Inverse of K
    BIGNUM *M = BN_new();       // Decrypted message
    BIGNUM *temp = BN_new();

    // Set values from question
    BN_dec2bn(&p, "6469");
    BN_dec2bn(&xb, "4127");
    BN_dec2bn(&C1, "3533");
    BN_dec2bn(&C2, "3719");

    // K = C1^x_b mod p
    BN_mod_exp(K, C1, xb, p, ctx);

    // K^-1 mod p
    BN_mod_inverse(Kinv, K, p, ctx);

    // M = (C2 * K^-1) mod p
    BN_mul(temp, C2, Kinv, ctx);
    BN_mod(M, temp, p, ctx);

    // Output
    char *K_str = BN_bn2dec(K);
    char *Kinv_str = BN_bn2dec(Kinv);
    char *M_str = BN_bn2dec(M);

    printf("Recovered shared key (K): %s\n", K_str);
    printf("Inverse of shared key (K^-1): %s\n", Kinv_str);
    printf("Decrypted Message (M): %s\n", M_str);

    // Cleanup
    OPENSSL_free(K_str);
    OPENSSL_free(Kinv_str);
    OPENSSL_free(M_str);
    BN_free(p); BN_free(xb); BN_free(C1); BN_free(C2);
    BN_free(K); BN_free(Kinv); BN_free(M); BN_free(temp);
    BN_CTX_free(ctx);

    return 0;
}
```

Output:

```
>>> Compiling eg_q2.c
>>> Running eg_q2
--------------------------------
Recovered shared key (K): 4125
Inverse of shared key (K^-1): 1758
Decrypted Message (M): 4312
--------------------------------
```