
VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

B.Tech. Winter Semester 2024-25
School Of Computer Science and Engineering (SCOPE)

# Digital Assignment - VI
## Cryptography and Network Security Lab

**Apurva Mishra: 22BCE2791**
**Date:** 28 March, 2025

## Contents

# 1 Diffie Hellman Key Exchange

## 1.1 Code

**Code 0: main.c**

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

// Function for modular exponentiation (a^b mod p)
long long power(long long base, long long exp, long long mod) {
    long long res = 1;
    base = base % mod;
    while (exp > 0) {
        if (exp % 2 == 1) res = (res * base) % mod;
        base = (base * base) % mod;
        exp = exp / 2;
    }
    return res;
}

// Function to find a prime number (for simplicity, not cryptographically strong)
long long findPrime() {
    long long primes[] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97}; //small list of primes
    return primes[rand() % (sizeof(primes) / sizeof(primes[0]))];
}

// Function to find a primitive root modulo p (not fully robust, for simplicity)
long long findPrimitiveRoot(long long p) {
    // For simplicity, using 2. In real implementation, more robust algorithms are needed.
    return 2;
}

int main() {
    srand(time(0));

    // Publicly known parameters
    long long p = findPrime(); // Large prime number
    long long g = findPrimitiveRoot(p); // Primitive root modulo p

    printf("Publicly known parameters:\n");
    printf("Prime (p): %lld\n", p);
    printf("Primitive root (g): %lld\n\n", g);

    // Alice's private key (a)
    long long a = rand() % (p - 2) + 2; // Random number between 2 and p-1

```

```
    // Bob's private key (b)
    long long b = rand() % (p - 2) + 2; // Random number between 2 and p-1

    // Alice computes A = g^a mod p
    long long A = power(g, a, p);

    // Bob computes B = g^b mod p
    long long B = power(g, b, p);

    printf("Alice's private key (a): %lld\n", a);
    printf("Bob's private key (b): %lld\n\n", b);

    printf("Alice computes A: %lld\n", A);
    printf("Bob computes B: %lld\n\n", B);

    // Alice computes shared key K_A = B^a mod p
    long long KA = power(B, a, p);

    // Bob computes shared key K_B = A^b mod p
    long long KB = power(A, b, p);

    printf("Alice's shared key (KA): %lld\n", KA);
    printf("Bob's shared key (KB): %lld\n\n", KB);

    if (KA == KB) {
        printf("Shared keys match! Diffie-Hellman key exchange successful.
\n");
    } else {
        printf("Shared keys do not match! Diffie-Hellman key exchange
failed.\n");
    }

    return 0;
}
```

### 1.2 Output

```
da/ass6/diffi via C v16.0.0-clang took 9s
❯ just run
cc main.c -o main
./main
Publicly known parameters:
Prime (p): 19
Primitive root (g): 2

Alice's private key (a): 18
Bob's private key (b): 17

Alice computes A: 1
Bob computes B: 10

Alice's shared key (KA): 1
Bob's shared key (KB): 1

Shared keys match! Diffie-Hellman key exchange successful.

da/ass6/diffi via C v16.0.0-clang
❯
```

## 2 RSA

### 2.1 Code

**Code 0: main.c**

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <stdbool.h>

// Function to check if a number is prime (using a basic method)
bool is_prime(long long n) {
    if (n <= 1) return false;
    if (n <= 3) return true;
    if (n % 2 == 0 || n % 3 == 0) return false;

    for (long long i = 5; i * i <= n; i += 6) {
        if (n % i == 0 || n % (i + 2) == 0) return false;
    }
    return true;
}
```

```
17
18 // Function to generate a random prime number within a specified range.
19 long long chose_prim() {
20     long long min = 1000; // Example minimum range
21     long long max = 2000; // Example maximum range
22
23     if (min < 2) {
24         min = 2;
25     }
26
27     long long randomNum;
28     srand(time(NULL));
29
30     while (1) {
31         randomNum = (rand() % (max - min + 1)) + min;
32         if (is_prime(randomNum)) {
33             return randomNum;
34         }
35     }
36 }
37
38 // Function to calculate the greatest common divisor (GCD)
39 long long gcd(long long a, long long b) {
40     while (b != 0) {
41         long long temp = b;
42         b = a % b;
43         a = temp;
44     }
45     return a;
46 }
47
48 // Function to choose a valid 'e' (public exponent)
49 long long chose_E(long long phi) {
50     long long e = rand() % (phi - 2) + 2; // Ensure e is between 2
and phi-1
51
52     while (gcd(e, phi) != 1) {
53         e = rand() % (phi - 2) + 2;
54     }
55     return e;
56 }
57
58 // Function to calculate the modular inverse
59 long long mod_inverse(long long e, long long phi) {
60     long long d = 1;
61     while ((e * d) % phi != 1) {
62         d++;
63     }
64     return d;
65 }
66
67 // Function to encrypt the message
68 long long encrypt(long long msg, long long e, long long n) {
```

```
    long long result = 1;
    msg = msg % n;
    while (e > 0) {
        if (e % 2 == 1) {
            result = (result * msg) % n;
        }
        msg = (msg * msg) % n;
        e = e / 2;
    }
    return result;
}

// Function to decrypt the message
long long decrypt(long long ciphertext, long long d, long long n) {
    long long result = 1;
    ciphertext = ciphertext % n;
    while (d > 0) {
        if (d % 2 == 1) {
            result = (result * ciphertext) % n;
        }
        ciphertext = (ciphertext * ciphertext) % n;
        d = d / 2;
    }
    return result;
}

int main(void) {
    srand(time(0));

    long long p = chose_prim();
    long long q = chose_prim();

    while (q == p) {
        q = chose_prim();
    }

    long long n = p * q;
    long long phi = (p - 1) * (q - 1);

    long long e = chose_E(phi);
    long long d = mod_inverse(e, phi);

    long long msg = 0;
    printf("Enter msg to encrypt: ");
    scanf("%lld", &msg);

    printf("p: %lld, q: %lld, n: %lld, phi: %lld, e: %lld, d: %lld\n", p,
q, n, phi, e, d);

    long long ciphertext = encrypt(msg, e, n);
    printf("Cypher Text: %lld\n", ciphertext);

    long long decrypted_msg = decrypt(ciphertext, d, n);
```

```
    printf("Decrypted Text: %lld\n", decrypted_msg);

    return 0;
}

```

### 2.2 Output

```
da/ass6/rsa via C v16.0.0-clang took 12s
❯ just run
zig cc main.c -o main
./main
Enter msg to encrypt: 23
p: 1811, q: 1193, n: 2160523, phi: 2157520, e: 1165913, d: 1027417
Cypher Text: 1704127
Decrypted Text: 23

da/ass6/rsa via C v16.0.0-clang took 3s
❯
```

## 3 Point Doubling in ECC

### 3.1 Code

**Code 0: main.c**

```c
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define N 256

struct Point {
  int x;
  int y;
};

void print_point(struct Point p) {
  printf("Point:\n");
  printf("  X: %d\n  Y: %d\n", p.x, p.y);
}

struct Curve {
  int a;
  int b;
  int p;
};

```

```c
void print_curve(struct Curve c) {
  printf("Curve:\n");
  printf("  a: %d\n  b: %d\n  p: %d\n", c.a, c.b, c.p);
}

int gcd(int a, int b)
{
    // Find Minimum of a and b
    int result = ((a < b) ? a : b);
    while (result > 0) {
        // Check if both a and b are divisible by result
        if (a % result == 0 && b % result == 0) {
            break;
        }
        result--;
    }
    // return gcd of a nd b
    return result;
}

int modInverse(int a, int p) {
  if (p < 0) {
    perror("The prime cna not be negative");
    exit(0);
  }

  int flag = 0;
  if (a < 0) {
    a = -a;
    flag = 1;
  }

  int i = 0;
  while (i < 100) {
    if ((a*i) % p == 1) {
      break;
    }

    i += 1;
  }

  if (flag) {
    return p - i;
  } else {
    return i;
  }
}

int mod(int num, int p) {
  if (p < 0) {
    perror("The prime can not be negative");
    exit(0);
  }
```

```
  if (num < 0) {
    int tmp = (-num) % p;
    return p - tmp;
  } else {
    return num % p;
  }
 }

 int mod_frac(int num, int den, int p) {
   int a = mod(num, p);
   int b = modInverse(den, p);

   return (a*b) % p;
 }

struct Point ecc_double(struct Point P, struct Point Q, struct Curve E) {
  int lambda = 0;
  if (P.x == Q.x && P.y == Q.y) {
   lambda = mod_frac((3*P.x*P.x)  + E.a, (2*P.y), E.p);
  } else {
    lambda = mod_frac((Q.y - P.y), (Q.x - P.x), E.p);
  }


  struct Point R = {0};

  int tmp_x = (lambda*lambda) - P.x - Q.x;
  R.x = mod(tmp_x, E.p);

  int tmp_y = (lambda*(P.x - R.x)) - P.y;
  R.y = mod(tmp_y, E.p);

  return R;
}

int main(void) {
  struct Curve E = {0};
  struct Point P = {0};
  struct Point Q = {0};

  printf("Inputs: \n");
  printf("Curve: a b p: ");
  scanf("%d %d %d", &E.a, &E.b, &E.p);

  printf("Point P: x y: ");
  scanf("%d %d", &P.x, &P.y);
  printf("Point Q: x y: ");
  scanf("%d %d", &Q.x, &Q.y);

  print_curve(E);
  print_point(P);
  print_point(Q);

```

```
130   struct Point R = ecc_double(P, Q, E);
131
132   printf("P + Q = \n");
133   print_point(R);
134   return 0;
135 }
136
```

### 3.2 Output

```
da/ass6/doub_ecc via C v16.0.0-clang took 18s
❯ just run
cc main.c -o main
./main
Inputs:
Curve: a b p: 1 1 23
Point P: x y: 3 10
Point Q: x y: 3 10
Curve:
  a: 1
  b: 1
  p: 23
Point:
  X: 3
  Y: 10
Point:
  X: 3
  Y: 10
P + Q =
Point:
  X: 7
  Y: 12
```

## 4 Negative Point in ECC

### 4.1 Code

**Code 0: main.c**

```c
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define N 256

struct Point {
  int x;
```

```c
  int y;
};

void print_point(struct Point p) {
  printf("Point:\n");
  printf("  X: %d\n  Y: %d\n", p.x, p.y);
}

struct Curve {
  int a;
  int b;
  int p;
};

void print_curve(struct Curve c) {
  printf("Curve:\n");
  printf("  a: %d\n  b: %d\n  p: %d\n", c.a, c.b, c.p);
}

int mod(int num, int p) {
  if (p < 0) {
    perror("The prime can not be negative");
    exit(0);
  }
  if (num < 0) {
    int tmp = (-num) % p;
    return p - tmp;
  } else {
    return num % p;
  }
 }

struct Point ecc_neg(struct Point P, struct Curve E) {
  struct Point R = {0};

  R.x = P.x;
  R.y = mod(-P.y, E.p);

  return R;
}

int main(void) {
  struct Curve E = {0};
  struct Point P = {0};

  printf("Inputs: \n");
  printf("Curve: a b p: ");
  scanf("%d %d %d", &E.a, &E.b, &E.p);

  printf("Point P: x y: ");
  scanf("%d %d", &P.x, &P.y);

  print_curve(E);
```

```
63    print_point(P);
64
65    struct Point R = ecc_neg(P, E);
66
67    printf("\n-P = \n");
68    print_point(R);
69    return 0;
70 }
71
```

### 4.2 Output

```
da/ass6/neg_ecc via C v16.0.0-clang took 2m8s
❯ just run
cc main.c -o main
./main
Inputs:
Curve: a b p: 1 1 23
Point P: x y: 3 10
Curve:
  a: 1
  b: 1
  p: 23
Point:
  X: 3
  Y: 10
-10
23 10

-P =
Point:
  X: 3
  Y: 13
```

## 5 Elgamal Cryptography

### 5.1 Code

**Code 0: main.c**

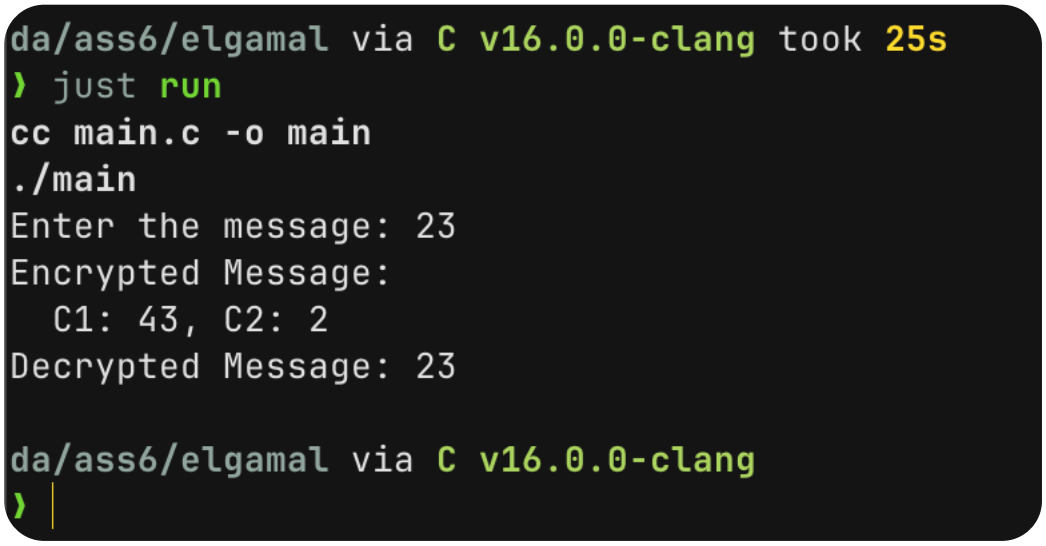
```
1  #include <math.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <time.h>
```

```c

struct pub_key {
  int q;
  int alpha;
  int Y;
};

struct priv_key {
  int x;
  int q;
};

struct enc_msg {
  int c1;
  int c2;
};

void print_enc_msg(struct enc_msg E) {
  printf("Encrypted Message:\n");
  printf("  C1: %d, C2: %d\n", E.c1, E.c2);
}

int gcd(int a, int b) {
  if (a < b)
    return gcd(b, a);
  else if (a % b == 0)
    return b;
  else
    return gcd(b, a % b);
}

int gen_key(int q) {
  int key = rand() % q;
  while (gcd(q, key) != 1) {
    key = rand() % q + 91;
  }
  return key;
}

int power(int a, int b, int c) {
  int x = 1;
  int y = a;
  while (b > 0) {
    if (b % 2 != 0) {
      x = (x * y) % c;
    }
    y = (y * y) % c;
    b = b / 2;
  }
  return x % c;
}

struct enc_msg encrypt(int msg, struct pub_key P) {
```

```
  if (msg > P.q) {
    perror("Msg cant not be more than q");
    exit(0);
  }

  int k = rand() % P.q;

  int K = power(P.Y, k, P.q);

  int c1 = power(P.alpha, k, P.q);
  int c2 = (K * msg) % P.q;

  struct enc_msg E = {c1, c2};

  return E;
}

int modInverse(int a, int p) {
  if (p < 0) {
    perror("The prime cna not be negative");
    exit(0);
  }

  int flag = 0;
  if (a < 0) {
    a = -a;
    flag = 1;
  }

  int i = 0;
  while (i < 100) {
    if ((a*i) % p == 1) {
      break;
    }

    i += 1;
  }

  if (flag) {
    return p - i;
  } else {
    return i;
  }
}

int decrypt(struct enc_msg E, struct priv_key P) {
  int K = power(E.c1, P.x, P.q);
  int tmp = modInverse(K, P.q);

  int msg = (E.c2 * tmp) % P.q;

  return msg;
}
```

```c

int main() {
  int msg;
  printf("Enter the message: ");
  scanf("%d", &msg);
  int q = rand() % 91;
  int alpha = rand() % (q - 2) + 2;
  int X = gen_key(q);

  struct priv_key priv_key = {X, q};

  int Y = power(alpha, X, q);
  struct pub_key pub_key= {q, alpha, Y};

  struct enc_msg E = encrypt(msg, pub_key);

  int dec_msg = decrypt(E, priv_key);

  printf("Decrypted Message: %d\n", dec_msg);
  return 0;
}

```

### 5.2 Output




## 6 RC4

*Done in Assessemnt 5*