Practical 14

Problem statement : Write a program to implement Digital Signature Algorithms: Elgamal.

#include <stdio.h>

#include <stdlib.h>

#include <gmp.h>

#include <time.h>

void generate\_keys(mpz\_t p, mpz\_t g, mpz\_t x, mpz\_t y) {

    gmp\_randstate\_t state;

    gmp\_randinit\_default(state);

    gmp\_randseed\_ui(state, time(NULL));

    // Generate prime p

    mpz\_urandomb(p, state, 512);

    mpz\_nextprime(p, p);

    // Generate generator g

    mpz\_urandomm(g, state, p);

    // Generate private key x

    mpz\_urandomm(x, state, p);

    // Compute public key y = g^x mod p

    mpz\_powm(y, g, x, p);

    gmp\_randclear(state);

}

void sign\_message(mpz\_t r, mpz\_t s, const mpz\_t p, const mpz\_t g, const mpz\_t x, const char \*message) {

    mpz\_t k, k\_inv, h, temp;

    mpz\_inits(k, k\_inv, h, temp, NULL);

    gmp\_randstate\_t state;

    gmp\_randinit\_default(state);

    gmp\_randseed\_ui(state, time(NULL));

    // Generate random k

    do {

        mpz\_urandomm(k, state, p);

    } while (mpz\_cmp\_ui(k, 0) == 0);

    // Compute r = g^k mod p

    mpz\_powm(r, g, k, p);

    // Compute k\_inv = k^-1 mod (p-1)

    mpz\_sub\_ui(temp, p, 1);

    mpz\_invert(k\_inv, k, temp);

    // Compute hash of the message

    mpz\_import(h, strlen(message), 1, 1, 0, 0, message);

    // Compute s = k\_inv \* (h - x \* r) mod (p-1)

    mpz\_mul(temp, x, r);

    mpz\_sub(temp, h, temp);

    mpz\_mul(temp, k\_inv, temp);

    mpz\_mod(s, temp, p);

    mpz\_clears(k, k\_inv, h, temp, NULL);

    gmp\_randclear(state);

}

int verify\_signature(const mpz\_t r, const mpz\_t s, const mpz\_t p, const mpz\_t g, const mpz\_t y, const char \*message) {

    mpz\_t h, v1, v2, temp1, temp2;

    mpz\_inits(h, v1, v2, temp1, temp2, NULL);

    // Compute hash of the message

    mpz\_import(h, strlen(message), 1, 1, 0, 0, message);

    // Compute v1 = g^h mod p

    mpz\_powm(v1, g, h, p);

    // Compute v2 = (y^r \* r^s) mod p

    mpz\_powm(temp1, y, r, p);

    mpz\_powm(temp2, r, s, p);

    mpz\_mul(v2, temp1, temp2);

    mpz\_mod(v2, v2, p);

    int result = mpz\_cmp(v1, v2) == 0;

    mpz\_clears(h, v1, v2, temp1, temp2, NULL);

    return result;

}

int main() {

    mpz\_t p, g, x, y, r, s;

    mpz\_inits(p, g, x, y, r, s, NULL);

    // Generate keys

    generate\_keys(p, g, x, y);

    // Message to be signed

    const char \*message = "This is a test message";

    // Sign the message

    sign\_message(r, s, p, g, x, message);

    gmp\_printf("Signature: (r: %Zd, s: %Zd)\n", r, s);

    // Verify the signature

    if (verify\_signature(r, s, p, g, y, message)) {

        printf("Signature verified successfully.\n");

    } else {

        printf("Signature verification failed.\n");

    }

    mpz\_clears(p, g, x, y, r, s, NULL);

    return 0;

}