Homework 6

Task 1 - RSA Modulus Generation (10 points)

(a)

Given that $N = PQ$, $P = R - i$ and $Q = R + j$, where R is a k-bit integer and (since primes are dense) i, j are relatively small integers, we have two scenarios:
1. If R is not a prime, then P and Q are neighboring primes in the set of k-bit numbers.
2. If R is a prime, then P and Q are immediate prime neighbors of R.

In both scenarios, since P and Q are close to each other, the value of N can be efficiently factorized. One approach can be to search for primes near the square root of N, which would be approximately $\sqrt{N} \approx R$. This search is feasible in polynomial time relative to k, which makes the RSA keys generated using this method insecure.

The proximity of P and Q ($|P - Q|$ is small) significantly reduces the complexity of the factorization problem. Since RSA security relies on the difficulty of factoring N, the method mentioned above of generating P and Q compromises the security of RSA modulus.

(b)

$$P = 351230145912301391230119331203122231987162381239182311193820 61$$
$$Q = 351230145912301391230119331203122231987162381239182311193824 47$$

```
from sympy import isprime
from timeit import default_timer as timer
import math

# The RSA modulus N
N = 123362615397576525683206910571962544945300500765564700092323336712076729023858866739705216165335280143754047119747057008326 7

def factor(N):
    # Approximate square root of N
    approx_sqrt_N = int(math.isqrt(N))

    # Search for prime factors near the square root of N
    for i in range(approx_sqrt_N, 1, -1):
        print("Trying i = ", i)
        if N % i == 0 and isprime(i):
            # double check
            if isprime(N // i) and i * (N // i) == N:
                return i, N // i

    print("Error: no factors found")
    return None, None

# Find P and Q
timer_start = timer()
P, Q = factor(N)
timer_end = timer()
print("P = ", P, "Q = ", Q, "Time = ", timer_end - timer_start)

```

Task 2 - ElGamal and DDH (15 points)

(a)

(b)

(c)

Task 3 - Chosen-Ciphertext Security (10 points)

(a)

(b)

(c)

Task 4 - AES-Based Signatures (15 points)

(a)

(b)

(c)