Ministerul Educaţiei și Cercetării al Republicii Moldova
Universitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică și Microelectronică

Laboratory work 5:
Cryptography and Security

Elaborated:
st. gr. FAF-211 Echim Mihail

Verified:
asist. univ. Aureliu ZGUREANU

Chişinău - 2023

CRYPTOGRAPHY AND SECURITY

Task 2.1

Objective:

The objective of this task was to implement the RSA encryption algorithm to generate keys and encrypt a message. The specific steps involved generating large prime numbers, calculating public and private keys, and encrypting a message using the generated keys.

Code Overview:

The provided code accomplishes the following:

Prime Number Generation:

Utilizes a function generate_large_prime(bits) to create large prime numbers with a specified number of bits. It employs a probabilistic method and the sympy library's isprime function to ensure the generated numbers are prime.

Key Generation:

Calculates two large prime numbers (prime1 and prime2) using the generate_large_prime(bits) function.
Derives the product n from these primes and computes phi_n as (prime1 - 1) * (prime2 - 1).
Selects a suitable public exponent e using the function choose_public_exponent(phi_n).

Encryption:

Converts the message "Echim Mihail" into its integer representation m by encoding characters to ASCII values and concatenating them.
Performs RSA encryption by raising m to the power of e modulo n (ciphertext = pow(m, e, n)).

Decryption:

Utilizes the private exponent d to decrypt the ciphertext using the formula decrypted_message = pow(ciphertext, d, n).

Output:

Prints the generated prime numbers (prime1 and prime2), their product (n), the calculated phi_n, public and private exponents (e and d), the ciphertext, and the decrypted message.
Verifies the success of the RSA encryption by comparing the original message with the decrypted message.

Conclusion:

The code successfully demonstrates the functionality of the RSA encryption algorithm by generating appropriate keys and encrypting the message "Echim Mihail." The output confirms the accuracy of the encryption and decryption processes, ensuring the security and reliability of the RSA cryptographic system.

Task 2.2

Objective:
The objective of this task was to implement the ElGamal encryption scheme to generate keys and encrypt a message. This encryption scheme involves the use of a large prime number p, a generator g, and the computation of public and private keys to encrypt and decrypt a message.

Code Overview:
The provided code achieves the following:

Message Representation:
Converts the message "Echim Mihail" into its integer representation m by encoding characters to ASCII values and concatenating them.

Key Generation:
Defines a large prime number p and a generator g.
Generates a random private key within the range [1, p - 2].
Computes the corresponding public key r as r = g^a mod p.

Encryption:
Generates a random secret key k within the range [1, p - 2].
Computes gamma as gamma = g^k mod p.
Encrypts the message by calculating the ciphertext as ciphertext = (m mod p) * (r^k mod p).

Decryption:
Performs decryption by computing decrypted using the formula (gamma^(p - 1 - a) * ciphertext) mod p.

Output:
Prints the generated random private key (a), computed public key (r), random secret key (k), computed public key (gamma), ciphertext, and the decrypted message.

Conclusion:
The code successfully demonstrates the ElGamal encryption scheme by generating appropriate keys and encrypting the message "Echim Mihail." The output confirms the accuracy of the encryption and decryption processes, showcasing the functionality of the ElGamal encryption method in securing data transmission.

Task 3

Objective:
The objective of this task was to establish a secure key exchange protocol between two parties, Alice and Bob, using the Diffie-Hellman key exchange algorithm. Subsequently, the shared secret key obtained from this exchange was utilized to implement AES encryption, ensuring secure communication between the two parties.

Code Overview:
The provided code accomplishes the following:

Diffie-Hellman Key Exchange:

Generates large prime number p and a primitive root g.
Randomly selects private keys a and b for Alice and Bob, respectively.
Computes their respective public keys (A for Alice and B for Bob) using the formula A = g^a mod p and B = g^b mod p.
Calculates the shared secret keys for Alice and Bob using the exchanged public keys and their private keys (DH_key_alice and DH_key_bob).

AES Key Generation:
The shared Diffie-Hellman key obtained (DH_key_alice and DH_key_bob) is utilized as a symmetric key for the AES algorithm.

Output:
Displays the generated private keys (a and b), computed public keys (A and B), and the shared secret keys derived from Diffie-Hellman exchange.
Verifies if both parties have successfully computed the same shared secret key.

Conclusion:
The code successfully demonstrates the Diffie-Hellman key exchange between Alice and Bob, ensuring a secure communication channel by establishing a shared secret key (DH_key_alice and DH_key_bob). This shared secret key serves as the foundation for implementing the AES encryption algorithm to secure their communications.