Ministerul Educației, Culturii și Cercetării al Republicii Moldova
Universitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică și Microelectronică
Departamentul Ingineria Software și Automatică

Raport
pentru lucrare de laborator Nr. 5 la cursul Criptografia și Securitate "*Criptografia cu cheii publice*"

A efectuat: Arteom KALAMAGHIN, FAF-211
A verificat: Aureliu ZGUREANU

Chișinău - 2023

Subject: Study of asymmetric cryptography

Tasks:

Sarcina 1. Studiați materiale didactice recomandate la temă plasate pe ELSE.

Sarcina 2.1. Utilizând platforma wolframalpha.com sau aplicația Wolfram Mathematica, generați cheile și realizați criptarea și decriptarea mesajului m = Nume Prenume aplicând algoritmul RSA. Valoarea lui n trebuie să fie de cel puțin 2048 biți.

Sarcina 2.2. Utilizând platforma wolframalpha.com sau aplicația Wolfram Mathematica, generați cheile și realizați criptarea și decriptarea mesajului m = Nume Prenume aplicând algoritmul ElGamal (p și generatorul sunt dați mai jos).

Sarcina 3. Utilizând platforma wolframalpha.com sau aplicația Wolfram Mathematica, realizați schimbul de chei Diffie-Hellman între Alisa și Bob, care utilizează algoritmul AES cu cheia de 256 de biți. Numerele secrete a și b trebuie să fie alese în mod aleatoriu în conformitate cu cerințele algoritmului ...

Theoretical notes:

Asymmetric cryptography, also known as public-key cryptography, is a cryptographic approach that uses a pair of mathematically related but distinct keys for securing communication. One key is public and can be shared with anyone, while the other is private and must be kept secret. Messages encrypted with the public key can only be decrypted by the corresponding private key, providing confidentiality and authenticity. Asymmetric cryptography is widely used for tasks like secure data transmission, digital signatures, and key exchange.

RSA is one of the most well-known asymmetric encryption algorithms. It was invented by Ron Rivest, Adi Shamir, and Leonard Adleman in 1977. RSA relies on the mathematical properties of large prime numbers and their difficulty in factoring to provide security. It is used for secure data transmission and digital signatures. RSA's security is based on the challenge of factoring the product of two large prime numbers, a task that becomes computationally infeasible for sufficiently large keys.

The ElGamal encryption and digital signature scheme is another widely used asymmetric cryptographic system. It was developed by Taher ElGamal in the 1980s. ElGamal encryption is based on the mathematical properties of modular exponentiation and the discrete logarithm problem. It is used for secure key exchange, digital signatures, and encryption. ElGamal is often favored for its security properties, particularly in key exchange and digital signature applications.

The Diffie-Hellman key exchange protocol, developed by Whitfield Diffie and Martin Hellman in 1976, is a fundamental component of modern cryptography. It allows two parties to securely exchange cryptographic keys over an untrusted communication channel, such as the internet. The protocol is based on the difficulty of the discrete logarithm problem and modular arithmetic. While Diffie-Hellman itself does not provide encryption or authentication, it forms the basis for creating secure shared keys that can be used for subsequent encryption using symmetric cryptography. It is a fundamental building block in many secure communication systems, including virtual private networks (VPNs) and secure socket layer (SSL) for web security.

Implementation:

To perform this lab, I did not use Wolfram Alpha, but the functionality of the Python libraries: cryptodome and sympy.

T1 - RSA

1. Generate two random prime numbers p1 and p2, each with 1024 bits:
   - use getPrime(1024) from the Crypto.Util.number module.
2. Calculate the product n of p1 and p2 to obtain a 2048-bit modulus:
   - n = p1 * p2.
3. Calculate the Euler's totient function phi(n) for the modulus n:
   - phi(n) = (p1 - 1) * (p2 - 1).
4. Find a suitable public exponent e and its corresponding private exponent d:
   - iterate through prime numbers between 1 and 1000 (inclusive) and select e such that it shares no common factors with n (the modulus);
   - calculate d as the modular inverse of e with respect to phi(n).
5. Convert the string into an integer m according to ASCII decimal representations:
   - concatenate the ASCII values of each character to form an integer.
6. Encrypt the m using the public exponent e and modulus n to obtain the ciphertext c:
   - c = m^e mod n.
7. Decrypt the ciphertext c using the private exp. d and modulus n to recover the original message m:
   - m = c^d mod n.

```
p1 (1024b) - a random prime = 105160955941678...
p2 (1024b) - a random prime = 936150033106928...
          n (2047b) = p1 * p2 = 984464323863583...
Phi(n) = (p1 - 1) * (p2 -1) = 984464323863583...
                            e = 7
                            d = 140637760551940...
              'Nume Prenume' > 781171091013280114101110117109101
                       enc(m) = 177510129205200...
                       dec(c) = 781171091013280114101110117109101
```

On the next page you’ll find everything about the ElGamal crypto process …

T2 - ElGamal

1. Define a large prime number p and a generator g for the ElGamal encryption system.
2. Generate a random secret key d for Alice (A) (the secret key must be smaller than p but greater than 1).
3. Calculate Alice's public key e by raising the generator g to the power of her secret key d modulo p:
   - e = g^d mod p
4. Define Alice's public and private keys as (p, g, e) and (p, d) respectively.
5. Convert the message "Nume Prenume" into a numeric representation m using ASCII decimal values.
6. Generate a random secret key k for Bob (B) (this secret key must also be smaller than p but greater than 1).
7. Compute two values, B1 and B2, for the cryptogram:
   - B1 = g ^ k mod p;
   - B2 = m * e^k mod p.
8. Assemble the cryptogram c as a tuple of B1 and B2: (B1, B2).
9. To decrypt the message, calculate dec_c by performing the following steps:
   - m = dec(c) = B2 * (B1^d)^(-1) mod p.

```
  d (A's secret) = 22
             e (A's public) = 4194304
A's public key - (p, g, e) : (1700607131..., 2, 4194304)
  A's private key - (p, d) : (3231700607..., 22)
             'Nume Prenume' > 781171091013280114101110117109101
             k (B's secret) = 76
               c - (B1, B2) : (7555786372..., 1640207951...)
                     dec(c) : 781171091013280114101110117109101
```

And the following page will conclude this report with comments on the implementation of the Diffie-Hellman key exchange process …

T3 - Diffie-Hellman

1. Define a large prime number p and a generator a for the Diffie-Hellman key exchange.
2. Generate random private keys for both Alice and Bob. These private keys must be integers between 1 and p.
3. Calculate the corresponding public keys for Alice and Bob by raising the generator a to the power of their respective private keys, modulo p:
   - pub_alice = a^(pr_alice) mod p;
   - pub_bob = a^(pr_bob) mod p.
4. Calculate the common secret key k for both Alice and Bob by raising each other's public keys to the power of their own private keys, modulo p:
   - common_secret = pub_bob^pr_alice mod p = pub_alice^pr_bob mod p

```
           Private key (Alice) : 509742672535874...
             Private key (Bob) : 358869281324595...
Pub. key ~ a^(pr_alice) mod p (A) : 234071545487472...
  Pub. key ~ a^(pr_bob) mod p (B) : 234071545487472...
                 Common secret : 231241809343845...
```

Conclusion:

In conclusion, this laboratory work provided a comprehensive exploration of various cryptographic algorithms and their practical applications. The significance of public key cryptography is rooted in its unique principle of using a pair of keys: a public key for encryption and a private key for decryption. This asymmetric key system revolutionizes digital security by allowing individuals and organizations to securely transmit sensitive information over public channels without the need to share secret keys. The fundamental principle of public key cryptography has widespread applications, from securing online communications and e-commerce transactions to protecting critical infrastructure and ensuring data integrity. By following the assigned tasks, I gained valuable insights into the principles and techniques of RSA, ElGamal, and Diffie-Hellman key exchange. I delved into the RSA and ElGamal algorithms. Were generated encryption keys, encrypted and decrypted messages, I saw firsthand how these algorithms are employed to ensure the confidentiality of information. It's crucial to note that for RSA, we adhered to the requirement of using a minimum 2048-bit modulus, reflecting the significance of key length in cryptographic strength. The third task introduced us to the Diffie-Hellman key exchange protocol, which serves as the foundation for secure key establishment in various cryptographic systems. I observed how two users can securely exchange keys, enabling them to use the AES algorithm with a robust 256-bit key for data encryption. Through this laboratory work, I've not only deepened my understanding of cryptographic techniques but also gained hands-on experience in implementing them. This knowledge is crucial in the realm of information security, where safeguarding sensitive data is of paramount importance. It has equipped me with the skills to design secure communication systems, protect data, and contribute to the ever-evolving field of cryptography.

* Check out my GitHub repo for the source code of these project:
https://github.com/Starlight-Crusader/CS-Lab