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: Popescu Sabina FAF-233
verificat: Zaica Maia

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 referred to as public-key cryptography, employs a pair of mathematically linked yet distinct keys to ensure secure communication. One key is public and can be shared openly, while the other is private and must be kept confidential. Messages encrypted with the public key can only be decrypted using the corresponding private key, ensuring both confidentiality and authenticity. Asymmetric cryptography finds widespread application in tasks such as secure data transmission, digital signatures, and key exchange.

Among the prominent asymmetric encryption algorithms, RSA stands out. Conceived by Ron Rivest, Adi Shamir, and Leonard Adleman in 1977, RSA leverages the mathematical intricacies of large prime numbers and their challenging factorization to establish security. Widely employed in secure data transmission and digital signatures, RSA's security hinges on the computational infeasibility of factoring the product of two large prime numbers when keys are of sufficient size.

Another widely adopted asymmetric cryptographic system is the ElGamal encryption and digital signature scheme, formulated by Taher ElGamal in the 1980s. ElGamal encryption relies on the mathematical principles of modular exponentiation and the discrete logarithm problem, finding applications in secure key exchange, digital signatures, and encryption. Its preference often stems from robust security features, particularly in key exchange and digital signature scenarios.

The Diffie-Hellman key exchange protocol, pioneered by Whitfield Diffie and Martin Hellman in 1976, constitutes a foundational element in modern cryptography. This protocol enables two parties to securely exchange cryptographic keys across an untrusted communication channel, such as the internet. Built on the complexity of the discrete logarithm problem and modular arithmetic, Diffie-Hellman, while not providing encryption or authentication directly, serves as a cornerstone for establishing secure shared keys. These keys, in turn, can be utilized for subsequent encryption through symmetric cryptography. The protocol plays a pivotal role in numerous secure communication systems, including virtual private networks (VPNs) and the 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.

## *Task 1 - 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 = 116235972195241...
p2 (1024b) - a random prime = 144464063885156...
        n (2047b) = p1 * p2 = 167919209129665...
Phi(n) = (p1 - 1) * (p2 -1) = 167919209129665...
                          e = 5
                          d = 100751525477799...
             'Nume Prenume' > 781171091013280114101110117109101
                    enc(m) = 29089135481387...
                    dec(c) = 781171091013280114101110117109101
```

***Task 2-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) = random.randint(2, p-1) =
62
e (A's public) = g^d mod p =
4611686018427387904
A's public key - (p, g, e):
(323170060713110073001535..., 2, 4611686018427387904)
A's private key - (p, d):
(323170060713110073001535..., 62)
Initial message ~ 'Nume Prenume' ~
781171091013280114101110117109101
k (B's secret) = random.randint(2, 100) =
10
c - cryptogram of the message - (B1, B2):
(1024, 3398939814521058842413...)
dec(c) == m - True and is:
781171091013280114101110117109101
```

### *Task 3 - 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 ~ (random.randint(1, p)) ~ Alice:
270278982047292507941382325...
Private key ~ (random.randint(1, p)) ~ Bob:
664293568874648173699493903 2...
Public key ~ a^(pr_alice) mod p ~ Alice:
259226458672051997106251 2953...
Public key ~ a^(pr_bob) mod p ~ Bob:
259226458672051997106251 2953...
Keys generated are identical - True are 2048

    bits in size and equal to:
839445079810197202148330 1720...
```

***Conclusion:*** In summary, this laboratory investigation offered a comprehensive examination of diverse cryptographic algorithms and their practical applications. The essence of public key cryptography lies in its distinctive approach, utilizing a pair of keys: a public key for encryption and a private key for decryption. This asymmetric key system transforms digital security, enabling secure transmission of sensitive information over public channels without the necessity of sharing secret keys. The foundational principle of public key cryptography finds broad utility, securing online communications, facilitating e-commerce transactions, safeguarding critical infrastructure, and ensuring data integrity.

By executing the assigned tasks, I acquired valuable insights into the principles and methodologies of RSA, ElGamal, and Diffie-Hellman key exchange. I delved into the intricacies of the RSA and ElGamal algorithms, generating encryption keys and observing the encryption and decryption processes. This firsthand experience illuminated the role of these algorithms in preserving information confidentiality. Notably, in the RSA task, we adhered to the crucial requirement of a minimum 2048-bit modulus, underscoring the importance of key length in cryptographic robustness.

The third task introduced the Diffie-Hellman key exchange protocol, a fundamental element in establishing secure keys across various cryptographic systems. Witnessing how two users securely exchanged keys highlighted the protocol's role in enabling the use of the AES algorithm with a robust 256-bit key for data encryption.

Through this laboratory work, I not only deepened my comprehension of cryptographic techniques but also gained practical expertise in their implementation. This knowledge proves indispensable in the realm of information security, where safeguarding sensitive data holds paramount importance. Equipped with these skills, I am prepared to design secure communication systems, protect data, and contribute to the dynamic field of cryptography.