# Kryptografia - Algorytm ElGamal sposób działania i implementacja

na podstawie *"A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, TAHER ELGAMAL"*

Adamski, Wojciech
242359

Górska, Kinga
259505

Mochoń, Filip
259480

19 Maja 2022

## Contents

# 1 Wprowadzenie - Co to jest algorytm ElGamala

Algorytm **ElGamal** to jeden z najważniejszych algorytmów kryptografii asymetrycznej. Opiera sie on o trudność rozwiazania algorytmu dyskretnego w ciele liczb całkowitych modulo dużych liczb pierwszych. Jego nazwa pochodzi od egipskiego kryptografa **Tahera Elgamala** w latach 80. $XX$w. Algorytm ten jest wykorzystywany miedzy innymi do podpisów cyfrowych, ale różne jego modyfikacje moga służyć do wielu innych zastosowań.

# 2 Klucz Publiczny

- $x_A$ - klucz tajny A
- $x_B$ - klucz tajny B
- p - duża liczba pierwsza (znane)
- $\alpha$ - element w ciele p (znane)

Strona A oblicza:

$$y_A \equiv \alpha^{x_A} \, mod \, p$$

i wysyła $y_A$. Podobnie, B oblicza $y_B \equiv \alpha^{x_B} \, mod \, p$ i je wysyła. Wtedy $K_{AB}$ jest obliczane:

$$\begin{aligned} K_{AB} &\equiv \alpha^{x_A x_B} \, mod \, p \\ &\equiv y_A^{x_B} \, mod \, p \\ &\equiv y_B^{x_A} \, mod \, p \end{aligned}$$

Zarówno strona A jak i B potrafi obliczyć $K_{AB}$. Nadal nie udowodniono, że złamanie systemu polega na obliczeniu logarytmu dyskretnego.
Dobranie odpowiedniego p polega na tym, że $p - 1$ posiada przynajmniej jeden duży dzielnik pierwszy.
Przypuśćmy, że A chce wysłać wiadomość $m$ do B (gdzie $0 \leq m \leq p - 1$). Najpierw A wybiera k z przedziału $(0, p - 1)$, a nastepnie oblicza klucz:

$$K \equiv y_B^k \, mod \, p \tag{1}$$

$y_B$ A dostaje od B lub jest publiczne. Zaszyfrowana wiadomość jest krotka $(c_1, c_2)$, gdzie:

$$c_1 \equiv \alpha^k \, mod\, p \qquad c_2 \equiv Km \, mod\, p \tag{2}$$

Deszyfrowanie rozbija sie na dwie cześci. Najpierw trzeba odzyskać $K$

$$K \equiv (\alpha^k)^{x_B} \equiv c_1^{X_B} \, mod\, p$$

$x_B$ jest znane tylko B. Drugim krokiem jest rozszyfrowanie $c_2$
Nierekomendowane jest używanie tej samej wartości $k$ do szyfrowania wiecej niż jednego bloku wiadomości, gdyż w przypadku złamania $m_1$ haker może odszyfrować reszte wiadomości. Załóżmy, że:

$$c_{1,1} \equiv \alpha^k \, mod\, p \qquad c_{2,1} \equiv m_1 K \, mod\, p$$

$$c_{1,2} \equiv \alpha^k \, mod\, p \qquad c_{2,2} \equiv m_2 K \, mod\, p$$

wtedy $\frac{m_1}{m_2} \equiv \frac{c_{2,1}}{c_{2,2}} \, mod\, p$, a $m_2$ jest łatwe do policzenia jeżeli znane jest $m_1$

# 3 Schemat podpisu elektronicznego

- $m$ - dokument do podpisania, gdzie $0 \leq m \leq p-1$
- $y$ - klucz publiczny, gdzie $y \equiv \alpha^x \, mod\, p$
- $x_A$ - klucz prywatny

Podpis dla $m$ jest para $(r, s), 0 \leq r, s < p-1$ w taki sposób, że:

$$\alpha^m \equiv y^r r^s \, mod\, p \tag{3}$$

1. Podpisywanie

   (a) wybierz liczbe $k$ taka, że $0 < k < p-1$, a $NWD(k, p-1) = 1$

   (b) oblicz:

   $$r \equiv \alpha^k \, mod\, p \tag{4}$$

   (c) teraz można zapisać (3):

   $$\alpha^m \equiv \alpha^{xr} \alpha^{ks} \, mod\, p \tag{5}$$

(d) powyższe równanie można rozwiazać dla $s$:

$$m \equiv xr + ks\, mod\, (p-1) \tag{6}$$

2. Procedura weryfikacji
   Posiadajac $m$, $r$ i $s$ łatwo jest zweryfikować autentyczność podpisu poprzez obliczenie obu wartości i sprawdzenie czy sa równe.

# 4 Przykładowe ataki na schemat podpisu

W tej cześci pokazane beda przykładowe możliwości ataku na schemat podpisu. Niektóre ataki polegaja na obliczeniu logarytmu dyskretnego na $GF(p)$. Dotychczas jednak nie udowodniono, że złamanie schematu podpisu jest tożsame z obliczeniem logarytmu dyskretnego. Jednakże żadne z tych ataków nie zakończyły sie jeszcze złamaniem szyfru. Ataki beda podzielone na dwie grupy. Pierwsza z nich zawiera ataki polegajace na odzyskaniu klucza prywatnego $x$, a druga pokazuje niektóre ataki polegajace na podrobieniu podpisu bez posiadania $x$.

1. Ataki polegajace na odzyskaniu $x$: trzeba to dobrze przetłumaczyć bo pisze pierdoły

   (a) Atak 1:
   Posiadajac $\{m_i : i = 1, 2, ..., l\}$ dokumentów z odpowiadajacymi podpisami $\{r_i, s_i : i = 1, 2, ..., l\}$ haker może próbować rozwiazać $l$ równań odzyskujacych z (6). Ponieważ jest $l + 1$ niewiadomych liczba rozwiazań jest ogromna. Powód, dla którego do każdej wartości dla $x$ odpowiada rozwiazanie dla $k_i$ jest taki, że sa to wartości liniowe macierzy współczynników. Jeżeli jakiekolwiek $k$ jest użyte wiecej niż raz do podpisu to system równań jest jasno określony i $x$ może zostać odzyskane.

   (b) Atak 2:
   Próba rozwiazania równań (3) jest równoważna do obliczenia logarytmu dyskretnego na $GF(p)$, ponieważ obie niewiadome $x$ i $k$ pojawiaja sie w wykładniku.

   (c) Atak 3:
   Atakujacy mógłby spróbować rozwinac zależność liniowa pośród niewiadomych $\{k_i, i = 1, 2, ..., l\}$. Wiaże sie to z obliczeniem logarytmu dyskretnego, ponieważ jeżeli $k_i \equiv ck_j\, mod\, (p-1)$, wtedy $r_i \equiv r_j^c\, mod\, p$ i jeżeli $c$ może być obliczone, wtedy rozwiazanie logarytmu dyskretnego jest łatwe.

2. Ataki fałszujace podpis

   (a) Atak 4:
   Posiadajac dokument $m$ fałszerz mógłby spróbować znaleźć $r$, $s$ takie, że (3) zostałoby spełnione. Jeżeli $r \equiv \alpha^j \, mod \, p$ jest ustalone dla danego $j$ (wybranego losowo to obliczenie $s$ jest równoznaczne z rozwiazaniem logarytmu dyskretnego na $GF(p)$.
   Jeżeli fałszerz ustali $s$ najpierw, wtedy $r$ może być obliczone z równania

$$r^s y^r \equiv A \, mod \, p \tag{7}$$

   Rozwiazanie równania (7) dla $r$ jeszcze nie zostało potwierdzone jakoby było równie trudne jak rozwiazanie logarytmu dyskretnego, ale podejrzewa sie, że jest niewykonalne obliczenie (7) w wielomianowym czasie.

   (b) Atak 5:
   Wydaje sie możliwe, że (3) może być rozwiazane dla zarówno $r$ i $s$ jednocześnie, ale nie znaleziono dotychczas odpowiedniego algorytmu.

   (c) Atak 6:
   Atak na schemat podpisu pozwala na utworzenie fałszywego-prawdziwego podpisu znajac jeden z poprzednich prawdziwych podpisów. Atak ten jednakże nie łamie systemu szyfrowania, gdyż nie można tak wygenerowanym podpisem podpisać dowolnej wiadomości.
   Jeżeli podpis $(r, s)$ jest prawdziwy dla wiadomości $(m)$, wtedy

$$\alpha^m = y^r r^s \, mod \, p$$

   Wybierzmy liczby całkowite $A, B$ oraz $C$ takie, że $(Ar - Cs)$ jest wzglednie pierwsze do $p-1$. Wtedy:

$$r' \equiv r^A \alpha^B y^C \, mod \, p$$

$$s' \equiv \frac{sr'}{(Ar - Cs) \, mod \, (p-1)}$$

$$m' \equiv \frac{r'(Am + Bs)}{(Ar - Cs) \, mod \, (p-1)}$$

   Wtedy zakłada sie że $(r‘, s‘)$ może podpisać wiadomość $(m')$. Obliczamy (w ciele $p$):

$$y^{r'} r'^{s'} \equiv r^{r'} (r^A \alpha^B y^C)^{\frac{sr'}{(Ar - Cs}}$$

$$\equiv (y^{r'Ar-r'Cs+r'Cs}r^{Asr'}\alpha^{Bsr'})^{\frac{1}{(Ar-Cs)}}$$
$$\equiv ((y^r r^s)^{Ar'}\alpha^{Bsr'}{}^{\frac{1}{(Ar-Cs)}}$$
$$\equiv \alpha^{\frac{mAr'+Bsr'}{Ar-Cs}}$$
$$\equiv \alpha^{m'}$$

W szczególnym przypadku, gdy $A = 0$ prawdziwe podpisy moga być generowane do odpowiadajacych sobie wiadomości nawet bez znajomości żadnego podpisu:

$$r' \equiv \alpha^B y^C \, mod \, p$$

$$s' \equiv \frac{-r'}{C \, mod \, (p-1)}$$
$$m' \equiv \frac{-r'B}{C \, mod \, (p-1)}$$

Widać wiec, że $(r', s')$ podpisze wiadomość $(m')$.

# 5 Właściwości systemu ElGamala i porównanie go do innych schematów podpisu i systemu kluczy publicznych.

Weźmy $m$, które bedzie liczba bitów $p$ dla logarytmu dyskretnego, albo $n$ dla faktoryzacji liczb całkowitych. Wtedy najlepszy znany algorytm do obliczania logarytmów dyskretnych, jak i faktoryzacji liczb całkowitych (który jest funkcja używana w niektórych obecnych systemach, na przykład RSA)

$$O(exp\sqrt{cm \ln m})$$

gdzie najlepszym przybliżeniem dla $c$ jest $c = 0.69$ dla faktoryzacji liczb całkowitych, jak i dla logarytmów dyskretnych na $GF(p)$. Te założenia sugeruja, że musimy używać liczb, które sa rozmiarem zbliżone do tych używanych w RSA w celu osiagniecia podobnego poziomu bezpieczeństwa.

1. Właściwości systemu kluczy publicznych
   Jak wykazano powyżej, system ElGamala różni sie od innych systemów. Po pierwsze, w celu wdrożenia losowości przy kodowaniu, szyfrogram dla danej wiadomości $m$ jest niepowtarzalny, na przykład

jeżeli chcemy zaszyfrować ta sama wiadomość dwa razy, to nie otrzymamy tego samego szyfrogramu $(c_1, c_2)$. To zapobiega atakom polegajacym na odgadywaniu fragmentów tekstu, czyli jeżeli atakujacy podejrzewa, że dany fragment tekstu to $m$, a potem spróbuje zaszyfrować to $m$ to patrzac na szyfrogram nie bedzie w stanie sie upewnić czy miał racje, gdyż oryginalny nadawca może użyć innego $k$ i otrzymać zupełnie inny wynik szyfrogramu.
Ponadto, dzieki strukturze szyfru Elgamala nie ma żadnego oczywistego połaczenia miedzy zaszyfrowaniem $m_1, m_2$ a $m_1 m_2$ ani żadnej innej prostej kombinacji $m_1$ i $m_2$, czego nie można powiedzieć o RSA.

2. Właściwości schematu podpisu
Zarówno dla omawianego systemu, jak i RSA rozmiary klucza sa podobne. Do podpisania wiadomości wystarczy jedno potegowanie i pare operacji mnożenia. Aby zweryfikować podpis zakłada sie, że potrzebne sa trzy operacje potegowania, ale zostało pokazane, że w optymalnych warunkach wystarczy średnio 1.875 operacji. Jest to robione, poprzez zaprezentowanie każdego działania za pomoca $m, r, s$ w ich binarnej postaci. Na każdym kroku podnosi sie do potegi drugiej liczbe $\alpha^{-1}yr$ i dzieli przez odpowiedni dzielnik, aby uwzglednić różne rozszerzenia $m, r$ i $s$. Różne wielokrotności $\alpha^{-1}, y$ i $r$ moga być przechowywane w tablicy zawierajacej osiem elementów.

# 6 BoB i Alice - teoria

1. Bob: generowanie klucza
Żeby wygenerować swój klucz prywatny i publiczny Bob musi:

   (a) wybierać liczbe pierwsza $p$ i generator $g \in \mathbb{Z}_p^{\otimes}$

   (b) wybierać losowe $b$, takie, że $b \in \mathbb{N}$

   (c) obliczyć $B = g^{b\otimes}$ na $(\mathbb{Z}_p^{\otimes}, \otimes)$

   (d) wysłać swój klucz publiczny $p$,$g, B$ w katalogu kluczy

2. Alice: szyfrowanie
Żeby Alice mogła zaszyfrować wiadomość $m \in \mathbb{Z}_p^{\otimes}$ musi wykonać nastepujace kroki:

   (a) otrzymać klucz Boba $p$, g i B

   (b) wybrać losowe $a \in \mathbb{N}$

   (c) obliczyć klucz prywatny $s = B^{a\otimes}$

(d) obliczyć $A = g^{a\otimes}$

(e) zaszyfrować $m$ poprzez obliczenie $X = m \otimes s$

(f) wysłać Bobowi $(A, X)$

3. Bob: Deszyfrowanie
Informacje, które posiada Bob żeby rozszyfrować wiadomość to jego klucz prywatny $b$ oraz klucz publiczny składajacy sie z liczby pierwszej $p$, generatora $g$ oraz $B = g^b$. Żeby rozszyfrować wiadomość $(A, X)$ Bob musi:

   (a) odebrać wiadomość $(A, X)$ od Alice

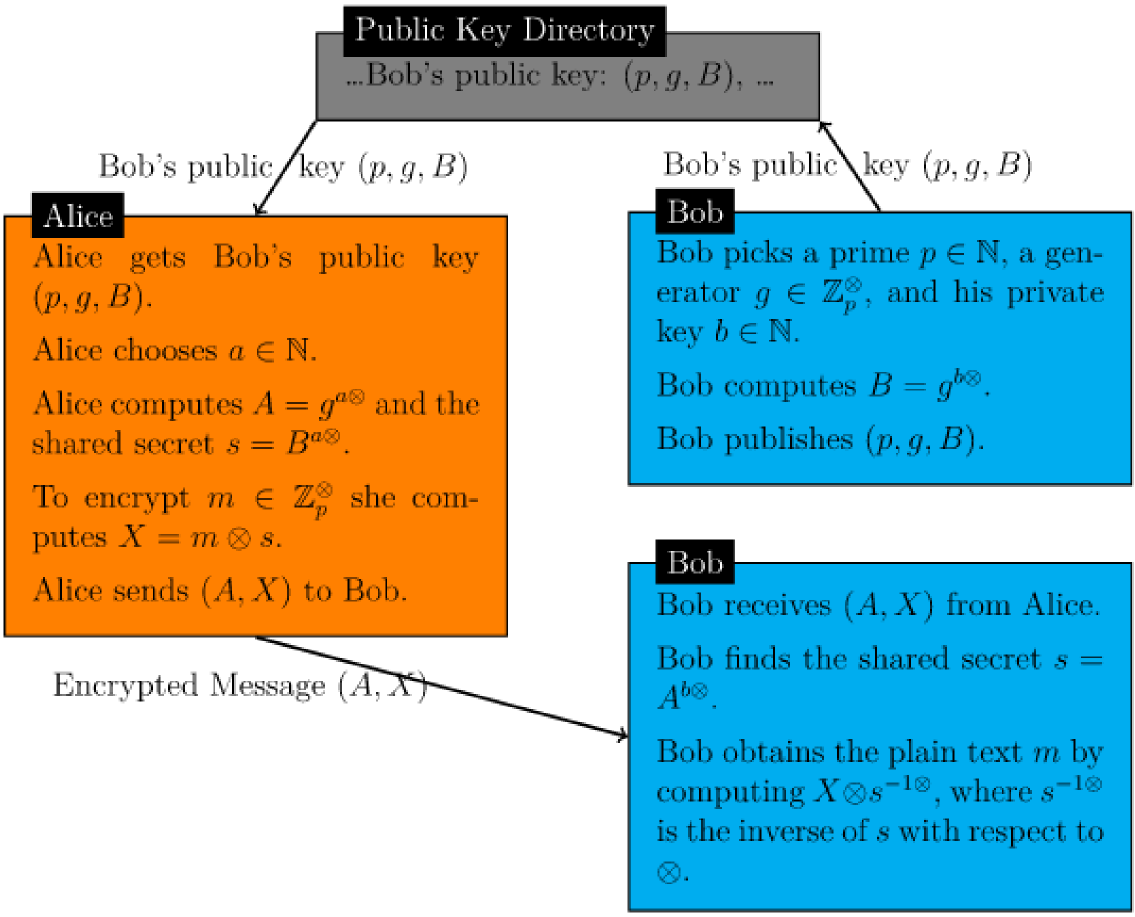
   (b) obliczyć $s = A^{b\otimes}$

   (c) obliczyć odwrotność $s^{-1\otimes}$ z $s$ w $(\mathbb{Z}_p^{\otimes}, \otimes)$

   (d) rozszyfrować wiadomość za pomoca obliczenia $M = X \otimes s^{-1\otimes}$

Aby wykazać że wiadomość M otrzymana przez Boba jest równoznaczna z wiadomościa m od Alice pokażmy:

$$M = X \otimes s^{-1\otimes} = (m)^{-1\otimes} = m \otimes (s^{-1\otimes}) = m \otimes 1 = m$$




# 7 Przykład działania

1. Bob: klucz publiczny $(p, g, B)$, klucz publiczny $d$
   (a) Posiada:
      i. Liczba pierwsza powinna mieć około 300 znaków, ale dla przykładu bedzie ona mała
      $$p = 13$$
      ii. generator - liczba pierwotna dla powyższej liczby pierwszej
      $$g = 2$$
      iii. $NWD(p, g)$ musi wynosić 1, aby operacja została wykonana
      iv. sekretna liczba d musi spełniać warunek $(2 \leq d \leq p - 2)$
      $$d = 3$$

(b) Obliczenia:

i. 
$$B = g^d \, mod \, p \rightarrow B = 2^3 \, mod \, 13 \rightarrow B = 3$$

2. Alice:

(a) Posiada:

i. sekretna wiadomość, która jest mniejsza od $p$
$$m = 4$$

ii. losowa liczba
$$k = 7$$

(b) Obliczenia:

i. 
$$y_1 = g^k \, mod \, p \rightarrow 2^7 \, mod \, 13 = 128 \, mod \, 13_1 = 11 \tag{8}$$

ii. 
$$y_2 = me^k \, mod \, p \rightarrow (4 * 8^7) \, mod \, 13 \tag{9}$$
$$= (4 * 2097152) \, mod \, 13 = 8388608 \, mod \, 13 \rightarrow y_2 = 7$$

(c) Proces wysyłania wiadomości od Alice do Boba:

i. 
$$(y_2 * y_1^d)^{-1} \, mod \, p$$

ii. 
$$(7 * 11^3)^{-1} \, mod \, 13$$

iii. 
$$(7 * 8) \, mod \, 13$$

iv. 
$$56 \, mod \, 13 = 4$$

# 8 Implementacja w jezyku Python

1. Generowanie klucza:

```python
#For key generation i.e. large random number
def gen_key(q):
    key= random.randint(pow(10,20),q)
    while gcd(q,key)!=1:
        key=random.randint(pow(10,20),q)
    return key
```

2. Szyfrowanie

```python
#For asymetric encryption
def encryption(msg,q,h,g):
    ct=[]
    k=gen_key(q)
    s=power(h,k,q)
    p=power(g,k,q)
    for i in range(0,len(msg)):
        ct.append(msg[i])
    print("g^k used= ",p)
    print("g^ak used= ",s)
    for i in range(0,len(ct)):
        ct[i]=s*ord(ct[i])
    return ct,p
```

3. Deszyfrowanie

```python
#For asymetric encryption
def encryption(msg,q,h,g):
    ct=[]
    k=gen_key(q)
    s=power(h,k,q)
    p=power(g,k,q)
    for i in range(0,len(msg)):
        ct.append(msg[i])
    print("g^k used= ",p)
    print("g^ak used= ",s)
    for i in range(0,len(ct)):
        ct[i]=s*ord(ct[i])
    return ct,p
```

## 9 Program w pythonie

```python
import random
from math import pow

a=random.randint(2,10)

#To fing gcd of two numbers
def gcd(a,b):
    if a<b:
        return gcd(b,a)
    elif a%b==0:
        return b
    else:
        return gcd(b,a%b)

#For key generation i.e. large random number
def gen_key(q):
    key= random.randint(pow(10,20),q)
    while gcd(q,key)!=1:
        key=random.randint(pow(10,20),q)
    return key

def power(a,b,c):
    x=1
    y=a
    while b>0:
        if b%2==0:
            x=(x*y)%c;
        y=(y*y)%c
        b=int(b/2)
    return x%c

#For asymetric encryption
def encryption(msg,q,h,g):
    ct=[]
    k=gen_key(q)
    s=power(h,k,q)
    p=power(g,k,q)
    for i in range(0,len(msg)):
```

```
        ct.append(msg[i])
    print("g^k used= ",p)
    print("g^ak used= ",s)
    for i in range(0,len(ct)):
        ct[i]=s*ord(ct[i])
    return ct,p

#For decryption
def decryption(ct,p,key,q):
    pt=[]
    h=power(p,key,q)
    for i in range(0,len(ct)):
        pt.append(chr(int(ct[i]/h)))
    return pt


msg=input("Enter message.")
q=random.randint(pow(10,20),pow(10,50))
g=random.randint(2,q)
key=gen_key(q)
h=power(g,key,q)
print("g used=",g)
print("g^a used=",h)
ct,p=encryption(msg,q,h,g)
print("Original Message=",msg)
print("Encrypted Maessage=",ct)
pt=decryption(ct,p,key,q)
d_msg=''.join(pt)
print("Decryted Message=",d_msg)
```

## 10 Bibliografia


1. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, TAHER ELGAMAL
2. Implementacja algorytmu w Pythonie
3. Przykład zastosowania algorytmu
4. Wyjaśnienie wizualne działania
5. Implementing several attacks on plain ElGamal encryption by Bryce Allen

6. sss