УДК 681.324.067

В. М. Михалевич
О. І. Тютюнник
Є. С. Дремлюга
К. В. Медведєва

МАТЕМАТИЧНІ МОДЕЛІ ТА ПРОГРАМНІ ЗАСОБИ ГЕНЕРУВАННЯ ПСЕВДОВИПАДКОВИХ ПОСЛІДОВНОСТЕЙ ДЛЯ КРИПТОГРАФІЧНИХ ЗАСТОСУВАНЬ

Вінницький національний технічний університет

Анотація

Розглянуто деякі особливості методів BBS та RSA генерування псевдовипадкових послідовностей для криптографічних застосувань. Розроблено відповідні програмні модулі в середовищах мови програмування високого рівня Python та системи комп'ютерної математики Maple. Наведено деякі результати тестових випробувань розроблених програмних модулів.

Ключові слова: генератор псевдовипадкових послідовностей, BBS, RSA, Python, система комп'ютерної математики Maple.

Abstract

Some features of BBS and RSA methods for generating pseudo-random sequences for cryptographic applications were considered. The corresponding software modules have been developed on the high-level programming language Python and the Maple computer mathematics system. The results of test tests of the developed software modules are presented.

Keywords: pseudorandom generator, Computer Algebra Systems Maple, BBS, RSA, Python.

Вступ

Багато останніх десятиліть увагу наукової спільноти привертають проблеми генерування *псевдовипадкових послідовностей*. Широковідомі основні переваги та недоліки вказаних послідовностей у порівнянні з так званими *істинно випадковими послідовностями*, що можуть бути породжені апаратними або програмно-апаратними засобами. В [2] зазначається, що проектування криптографічно стійких псевдовипадкових послідовностей є важливим елементом побудови надіного криптографічного захисту.

Надалі користуватимемося такою термінологією «… засоби, що забезпечують генерацію випадкових послідовностей чисел або бітів, будемо називати генераторами випадкових послідовностей (ГВП), а генератори, що забезпечують генерацію псевдовипадкових послідовностей (ПВП) – детермінованими генераторами випадкових послідовностей (ДГВП)» [3].

Важливою особливістю ДГВП є те, що за однакових початкових умов ПВП також буде однаковою. Головнй недолік ДГВП полягає в їх періодичності.

Задачі аналізу та дослідження ДГВП наукова спільнота вважає складними та невирішенними до кінця. Для вирішення вказаних задач перш за все необхідно осмислення самої проблематики [3, 4, 5, 6].

Метою роботи є розробка в різних програмних середовищах програмних модулів, що реалізують ДГВП для криптографічних застосувань.

Результати дослідження

Постановка задачі

Може здатися можливим у всіх випадках використовувати, як ДГВП, стандартні процедури, що вбудовані у мови програмування. Проте, як зазначається в [2] такі послідовності не мають властивості криптографічної стійкості, і, як наслідок, непридатні для криптографічних застосувань. Звичайно період таких послідовностей дуже малий. Це зумовлене простими співвідношеннями, що покладено в основу джерела генерування. Як зазначається в [2] з посиланням на [**Error! Reference source not found.**], функція **RAND()** в C++ — мові програмування високого рівня, реалізує обчислення за рекурентною формулою

$$x_{i+1} = (1103515245 \cdot x_i + 12345) \ (\mathrm{mod}\, 4294967296).$$

У довідковій інформації комп'ютерної математики Maple зазначається, що стандартна команда **rand()** у версіях Maple до 9.5 включно, базувалася на функції для створення ДГВП з використанням алгоритму лінійного порівняння

$$x_{i+1} = 427419669081 \cdot x_i \ (\mathrm{mod}\, 999999999989). \qquad (1)$$

В подальших версіях Maple цей алгоритм покладено в основу команди GenerateInteger підпакета RandomTools[LinearCongruence].

Перевіримо, чи дійсно робота команди RandomTools[LinearCongruence] базується на використанні формули (1). Для цього запишемо два невеликих фрагменти програмного коду з відповідними коментарями

```
restart:
with(RandomTools[LinearCongruence]):# Підключення команди
LinearCongruence
s := GetState();# Отримання значення змінної середовища, що відповідає
змінній x[i] в формулі (1)
seq( GenerateInteger(), i=1..5 );# Генерування послідовності декількох
випадкових чисел, x[0]=1
x[0]:=GetState();# Присвоєння змінній x[0] значення змінної середовища,
що дорівнює x[i]
printf(`Генерування послідовності декількох випадкових чисел за допомогою
вбудованої команди Maple`);
''x''[0]=x[0],seq( x[k]=GenerateInteger(), k=1..9 );# Генерування
послідовності декількох випадкових чисел, x[0]=558458718976
```

$$s := 1$$

$$427419669081,\ 321110693270,\ 343633073697,\ 474256143563,\ 558458718976$$

$$x_0 := 558458718976$$

```
Генерування послідовності декількох випадкових чисел за допомогою
вбудованої команди Maple
```

$$'x'_0 = 558458718976,\ x_1 = 746753830538,\ x_2 = 32062222085,$$
$$x_3 = 722974121768,\ x_4 = 604305613921,\ x_5 = 745580037409,$$
$$x_6 = 259811952655,\ x_7 = 310075487163,\ x_8 = 797179490457,$$
$$x_9 = 39169594160$$

```
printf(`Перевірка тотожності випадкових чисел, що генеруються за
допомогою команди "GenerateInteger()" та за допомогою формули (1)`);
 SetState( state=x[0]);# Присвоєння змінній середовища, що відповідає
змінній x[i] в формулі (1), значення змінної x[0]
```

```
for k to 5 do
 x[k]:=427419669081*x[k-1] mod 999999999989:
 GenerateInteger()-x[k];print(%)
end:
```

```
Перевірка тотожності випадкових чисел, що генеруються за допомогою
команди "GenerateInteger()" та за допомогою формули (1)
```

0

0

0

0

0

Отримання нулів свідчить про успішність вказаної перевірки.

Довжина модуля в лінійному порівнянні для формули, що використовується в СКМ Maple, дорівнює 12, в C++ - 10. Отже, довжина циклу випадкової послідовності, що можна отримати на основі команди RandomTools[LinearCongruence] буде більшою, ніж довжина аналогічного циклу, що може бути породжений на основі функція **RAND()** в C++ .

Слід зауважити, що вказані способи отримання ДГВП цілком задовольняють велику кількість застосувань, зокрема, в комп'ютерних іграх та обчисленнях за методом Монте-Карло.

Що ж стосується криптографічних застосувань, до ДГВП висуваються додаткові умови.

ПВП:

- повинні якнайменше відрізнялися від справді випадкових;
- не повинні бути передбачуваними.

Непередбачуваність ПВП означає, що імовірність передбачення наступного біта не залежить від знання всіх попередніх бітів послідовності. Лінійні й нелінійні поліноміальні конгруентні генератори ПВП цій умові не відповідають, отже й не використовуються для потреб криптографії.

Надалі розглянемо особливості програмної реалізації двох ДГВП, що відносять до найсильніших програмних генераторів ПВП: BBS та RSA.

Як середовище для програмної реалізації та дослідження вказаних генераторів було вибрано СКМ Maple та Python - інтерпретовану об'єктно-орієнтовану мову програмування високого рівня зі строгою динамічною типізацією.

Середовище СКМ Maple вибрано внаслідок великого позитивного власного досвіду розробки прикладних програм різного застосування [8, 10, 11, 12, 13, 14, 14], а також значного поширення серед науковців та викладачів [15, 16, 17, 18, 19].

Python - одна з найпопулярніших мов програмування відповідно рейтингу 2020 року [20] і до того ж має зручні засоби роботи з великими числами у вигляді вбудованої бібліотеки.

Генератор BBS (автори – **B**lum L., **B**lum M., **S**hub M.)

З першого погляду цей метод ДГВП виглядає доволі простим

1. Обираємо два простих числа p та q, що задовольняють умові $p \equiv q \equiv 3 \pmod 4$.
2. Обчислюємо число $n = p \cdot q$ - число Блума.
3. Обираємо просте число x та обчислюємо стартове число генератора $x_0 \equiv x^2 \pmod n$.
4. Обчислення послідовності псевдовипадкових бітів $s_1, s_2, ..., s_i, ...$ за допомогою рекурентної формули $x_{i+1} \equiv x_i^2 \bmod n$ та та знаходження молодшого біта числа x_i: $s_i \equiv x_i \pmod 2$. Ці біти і є результатом, що подається на вихід вказаного генератора.

Цікава особливість цього алгоритму полягає в тому, що для отримання x_n необов'язково обчислювати всі n-1 попередні числа. Якщо відомо початковий стан генератора x_0 та числа p і q, то n - е значення може бути обчислено безпосередньо за формулою:

$$x_k = x_0^{2^k \bmod \lambda(n)} \pmod n. \tag{2}$$

де λ - функція Кармайкла.

Функція Кармайкла $\lambda(n)$ дорівнює найменшому показнику m такому, що

$$a^m = 1 \pmod n. \tag{3}$$

$\forall a$: НСД(a, n)=1. (Для всіх цілих чисел a, взаємнопростих з модулем n.)

Обчислення функції Кармайкла відбувається відповідно таких її властивостей

1. $\lambda(p^\alpha) = \varphi(p^\alpha) = p^{\alpha-1} \cdot (p-1)$ для непарного простого p та $\alpha \in \mathbb{N}$.
2. $\lambda(2) = 1, \lambda(2^2) = \varphi(2^2) = 2, \quad \lambda(2^\alpha) = \frac{1}{2} \cdot \varphi(2^\alpha), \alpha > 2$.
3. $\lambda(n) = lcm\left[\lambda(p_1^{\alpha_1}), \lambda(p_2^{\alpha_2}), ..., \lambda(p_s^{\alpha_s})\right]$.

де за основною теоремою арифметики будь-яке може бути подане як

$$n = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot ... \cdot p_s^{\alpha_s}, \quad \alpha_1, \alpha_2, ..., \alpha_s \in \mathbb{N},$$

$p_1 < p_2 < ... < p_s$ - прості числа; lcm – найменше спільне кратне; $\varphi(n)$ - функція Ейлера, що для довільного натурального числа n може бути обчислена за формулою

$$\varphi(n) = n \cdot \left(1 - \frac{1}{p_1}\right) \cdot \left(1 - \frac{1}{p_2}\right) \cdot ... \cdot \left(1 - \frac{1}{p_s}\right) = n \cdot \prod_{p|n} \left(1 - \frac{1}{p}\right). \tag{4}$$

Наведемо фрагмент програми в Maple та результати його роботи

```
restart:
st:=time():
x[0]:=233:
p:=19:q:=23:
p mod 4;q mod 4;
n:=p*q:
x[0]:=576930249676759584092594 89:
L:=lcm(p-1,q-1):# L=numtheory[lambda](n):
for i to 50 do
x[i]:=x[i-1]^2 mod n;
xk:=x[0]&^(2&^i mod L) mod n;# Обчислення за формулою (2)
if x[i]<>xk then printf("ПОМИЛКА!") end if
end do:
time()-st;
seq([k=i,''x''[i]=x[i],x[i] mod 2],i=1..100);
```

3

3

$$[k=1, 'x'_{1}=404, 0], [k=2, 'x'_{2}=215, 1], [k=3, 'x'_{3}=340, 0],$$
$$[k=4, 'x'_{4}=232, 0], [k=5, 'x'_{5}=73, 1], [k=6, 'x'_{6}=85, 1],$$
$$[k=7, 'x'_{7}=233, 1], [k=8, 'x'_{8}=101, 1], [k=9, 'x'_{9}=150, 0],$$
$$[k=10, 'x'_{10}=213, 1], [k=11, 'x'_{11}=358, 0], [k=12, 'x'_{12}=123, 1],$$
$$[k=13, 'x'_{13}=271, 1], [k=14, 'x'_{14}=25, 1], [k=15, 'x'_{15}=188, 0],$$
$$[k=16, 'x'_{16}=384, 0], [k=17, 'x'_{17}=187, 1], [k=18, 'x'_{18}=9, 1],$$
$$[k=19, 'x'_{19}=81, 1], [k=20, 'x'_{20}=6, 0], [k=21, 'x'_{21}=36, 0],$$
$$[k=22, 'x'_{22}=422, 0], [k=23, 'x'_{23}=225, 1], [k=24, 'x'_{24}=370, 0],$$
$$[k=25, 'x'_{25}=119, 1], [k=26, 'x'_{26}=177, 1], [k=27, 'x'_{27}=302, 0],$$
$$[k=28, 'x'_{28}=308, 0], [k=29, 'x'_{29}=35, 1], [k=30, 'x'_{30}=351, 1],$$
$$[k=31, 'x'_{31}=404, 0], [k=32, 'x'_{32}=215, 1], [k=33, 'x'_{33}=340, 0],$$
$$[k=34, 'x'_{34}=232, 0], [k=35, 'x'_{35}=73, 1], [k=36, 'x'_{36}=85, 1],$$
$$[k=37, 'x'_{37}=233, 1], [k=38, 'x'_{38}=101, 1], [k=39, 'x'_{39}=150, 0],$$
$$[k=40, 'x'_{40}=213, 1], [k=41, 'x'_{41}=358, 0], [k=42, 'x'_{42}=123, 1],$$
$$[k=43, 'x'_{43}=271, 1], [k=44, 'x'_{44}=25, 1], [k=45, 'x'_{45}=188, 0],$$
$$[k=46, 'x'_{46}=384, 0], [k=47, 'x'_{47}=187, 1], [k=48, 'x'_{48}=9, 1],$$
$$[k=49, 'x'_{49}=81, 1], [k=50, 'x'_{50}=6, 0], [k=51, 'x'_{51}=36, 0],$$
$$[k=52, 'x'_{52}=422, 0], [k=53, 'x'_{53}=225, 1], [k=54, 'x'_{54}=370, 0],$$
$$[k=55, 'x'_{55}=119, 1], [k=56, 'x'_{56}=177, 1], [k=57, 'x'_{57}=302, 0],$$
$$[k=58, 'x'_{58}=308, 0], [k=59, 'x'_{59}=35, 1], [k=60, 'x'_{60}=351, 1],$$
$$[k=61, 'x'_{61}=404, 0], [k=62, 'x'_{62}=215, 1], [k=63, 'x'_{63}=340, 0],$$
$$[k=64, 'x'_{64}=232, 0], [k=65, 'x'_{65}=73, 1], [k=66, 'x'_{66}=85, 1],$$
$$[k=67, 'x'_{67}=233, 1], [k=68, 'x'_{68}=101, 1], [k=69, 'x'_{69}=150, 0],$$
$$[k=70, 'x'_{70}=213, 1], [k=71, 'x'_{71}=358, 0], [k=72, 'x'_{72}=123, 1],$$
$$[k=73, 'x'_{73}=271, 1], [k=74, 'x'_{74}=25, 1], [k=75, 'x'_{75}=188, 0],$$
$$[k=76, 'x'_{76}=384, 0], [k=77, 'x'_{77}=187, 1], [k=78, 'x'_{78}=9, 1],$$
$$[k=79, 'x'_{79}=81, 1], [k=80, 'x'_{80}=6, 0], [k=81, 'x'_{81}=36, 0],$$
$$[k=82, 'x'_{82}=422, 0], [k=83, 'x'_{83}=225, 1], [k=84, 'x'_{84}=370, 0],$$
$$[k=85, 'x'_{85}=119, 1], [k=86, 'x'_{86}=177, 1], [k=87, 'x'_{87}=302, 0],$$
$$[k=88, 'x'_{88}=308, 0], [k=89, 'x'_{89}=35, 1], [k=90, 'x'_{90}=351, 1],$$

$$[k=91, 'x'_{91}=404, 0], [k=92, 'x'_{92}=215, 1], [k=93, 'x'_{93}=340, 0],$$
$$[k=94, 'x'_{94}=232, 0], [k=95, 'x'_{95}=73, 1], [k=96, 'x'_{96}=85, 1],$$
$$[k=97, 'x'_{97}=233, 1], [k=98, 'x'_{98}=101, 1], [k=99, 'x'_{99}=150, 0],$$
$$[k=100, 'x'_{100}=213, 1]$$

```
seq(x[i] mod 2,i=1..1000):
print(`+`(%));
parse(cat(%));
```

567

```
1001111010111001110001011001101001111010111001110001011001\
    1101001111010111001110001011001101001111010111001110\
    0101100110100111101011100111000101100110100111101011\
    0011100010110011010011110101110011100010110011010011\
    1010111001110001011001101001111010111001110001011001\
    0100111101011100111000101100110100111101011100111000\
    0110011010011110101110011100010110011010011110101110\
    1110001011001101001111010111001110001011001101001111\
    1011100111000101100110100111101011100111000101100110\
    0011110101110011100010110011010011110101110011100010\
    1001101001111010111001110001011001101001111010111001\
    1000101100110100111101011100111000101100110100111101\
    1110011100010110011010011110101110011100010110011010\
    1111010111001110001011001101001111010111001110001011\
    0110100111101011100111000101100110100111101011100111\
    0010110011010011110101110011100010110011010011110101\
    1001110001011001101001111010111001110001011001101001\
    1101011100111000101100110100111101011100111000101100\
    1010011110101110011100010110011010011110
```

В наведеному програмному коді міститься перевірка працездатності формули (2). Слід звернути увагу, що для забезпечення можливості передбачення результату на виході генератора за допомогою цієї формули потрібно мати значення $\lambda(n)$, що може бути визначене тільки на основі відомих значень *p* та *q*.

Наведемо фрагмент програми в Python та результати його роботи

```
import sympy
import time
start_time = time.time()
min_prime = 1290771951205109750917295791212908105185081125912059152319258309532095902309839523875023852395792385758239 8
max_prime = 1351512895791025901229510151510957935093905230950923711335235235239823752312453502239852389535823952389591 15

p = sympy.randprime(min_prime, max_prime)  # межі генерування min and max prime number
```

```
q = sympy.randprime(min_prime, max_prime)

        # перевірка умов ББС
while p % 4 != 3:
    p = sympy.randprime(min_prime, max_prime)
while q % 4 != 3:
    q = sympy.randprime(min_prime, max_prime)

print("p = ", p, "\n q = ", q)
n = p*q
print("n = ", n)
x = sympy.randprime(min_prime, max_prime)
print("x = ", x)
x_bit = []
s = []
x_bit.insert(0, pow(x,2) % n)
s.insert(0, x_bit[0] % 2)
for i in range(1, 8):
    x_bit.insert(i, pow(x_bit[i-1], 2) % n)
    s.insert(i, x_bit[i] % 2)

for i in range(len(s)):
    print(i, x_bit[i], s[i])

print('-----------------')
print('The length of the min_prime is: \n')
print(len(str(min_prime)))
print('-----------------')
print('The length of the max_prime is: \n')
print(len(str(max_prime)))
print('The execution time is :')
print("--- %s seconds ---" % (time.time() - start_time))
```

Результат виконання програми:

p = 341024892101917615179475634968952401778195304351949311405463546918689622797821391813745436212222758709338643

q = 690854091654238925391533073228239594992318341389735558598508453641689411463967221766777825966689869380207183

n = 235598442064555132332226536698699862558214008625289237118428575772374375169195680312385137387591390762915254319362747076435651021966498933241337461009593582030523935876414699444492496191494435943407267412824807266 9

x = 873879416727431718179381390711659227319056900548020205131478499509216467983131508176562090614126377186851970 0

5686990878621086965742608043005254272393606601963737620060615650037361918285593268359712933472511033452874483904226482950729777377888797616683212999242994652656829252472838705062632679605253645195397550056427108020 1

2272714967311902509854087241764773204528101998645299014077128536194603894927193426142574709100824116926834297841684744330958702738601935749376937528349092862463118092935394195985620175380664034537137069080052466356 0

2
157321786501101035845760705478292747122200023057742426632671189178864992384627203153560039850425804428287083634812648216316425395624419332622705703013706209532162841740038222806472618092610353142072113381228988248 0
3
99209722512095348420953592621442199815968867641075610712482565401944150832530432292952159531181005215244133738263822129290748499525002982885317016112974392948739837292420033309506082291475682056644018007955268128 0
4
20082597315318286876096310830206613272731106851597142495300460513389478964194899220411406041534289827320672208611505982312842811836692503024579126281895696045994787953124940592966201036430826813751308410792644837991
5
233263114090969854277277834540194112869108405973494908044220548681313549265855954569440607300374233460655499038061221965489237969754223866018672997087990564812816818796310900354383771064339489408281273101987431629 6
6
35835742030902615397887273301397813734046626559715174880642881617808150625736767642184947395376063012651092292513116240140906174729444155473721473869664477392039102906004196841889088644415388856476532726553823519 1
7
12972389013699132331959584275444590484955407847582341477471209302149983321133112716039817032876240240971403149411580584120762909073001594921321151371645976474415456514775006410824998903897846705475226737302695652180

The length of the min_prime is:

107

The length of the max_prime is:

108
The execution time is :
--- 0.02400517463684082 seconds ---

Process finished with exit code 0

Отже, безпека методу BBS ґрунтується на складності розкладання n на множники. В той же час для генерування бітів на основі рекурентних обчислень достатньо тільки самого числа n.

Аналогічно будується генератор псевдовипадкових двійкових послідовностей за допомогою інших однобічних функцій з лазівкою.

Генератор RSA

Для генерування необхідна пара чисел (n, e), що є відкритою частиною ключа в криптографічному алгоритмі RSA, а також стартове випадкове число $x_0, x_0 < n = p \cdot q$, p, q - два великих простих числа (секретні числа). Як відомо, $1 < e < (p-1)\cdot(q-1)$ і НСД(e, $(p-1)\cdot(q-1)$)=1. Генерування числової послідовності відбувається на основі рекурентного співвідношення $x_{i+1} \equiv x_i^e \bmod n$. Виходом генератора слугує молодший біт числа x_i.

Наведемо фрагмент програми в Maple та результати його роботи

```
restart:
st:=time():
n:=65815671868057:
e:= 7423489:
x[0]:=715671868:
```

```
for i to 1000 do
x[i]:=x[i-1]&^e mod n:
end do:
seq([k=i,''x''[i]=x[i],x[i] mod 2],i=1..20);
time()-st;
```

$$[k=1, 'x'_1 = 2082776429908, 0], [k=2, 'x'_2 = 15356494740989, 1],$$
$$[k=3, 'x'_3 = 55026867148802, 0], [k=4, 'x'_4 = 63500460954479, 1],$$
$$[k=5, 'x'_5 = 32749047177013, 1], [k=6, 'x'_6 = 49688430876803, 1],$$
$$[k=7, 'x'_7 = 35087976054080, 0], [k=8, 'x'_8 = 4103698789974, 0],$$
$$[k=9, 'x'_9 = 32314447310806, 0], [k=10, 'x'_{10} = 29881221788221, 1],$$
$$[k=11, 'x'_{11} = 55708784081080, 0], [k=12, 'x'_{12} = 708032648219, 1],$$
$$[k=13, 'x'_{13} = 49292480622227, 1], \quad [k=16, 'x'_{16} = 21216227641186, 0],$$
$$[k=14, 'x'_{14} = 29423858819396, 0], \quad [k=17, 'x'_{17} = 22522531867023, 1],$$
$$[k=15, 'x'_{15} = 26702143671710, 0], \quad [k=18, 'x'_{18} = 24439068283399, 1],$$
$$[k=19, 'x'_{19} = 24527990099116, 0],$$
$$[k=20, 'x'_{20} = 49238504677660, 0]$$

0.016

```
seq(x[i] mod 2,i=1..1000):
print(`+`(%));
parse(cat(%));
```

497

```
1011100010110001100110101001110101100010011100101001011\
0
  0101010001001101100100111101111010010100011110100001\
0
  1101101110011100111110001100110111011101011000110010\
0
  0000110100110001011110110011101010110110100110011\
1
  0011101100001110011010101001101010001010001111101011\
1
  0011011000000010000010101010101000000000100000110010\
0
  1110111010111110101001001101010111010101101010100011\
0
  0010000001000011100010111100101010010111011001010011\
1
  1100101000011010011101110001101100111101010101100\
0
  0011111001000001001000111101100011001101101100101110\
0
```

```
01110011011101111010111000110010101101100111001001011\
11000101101110101101100101110000111101011011100011110\
11001001010011101111010111111000100110011100001010101\
10001010000010110011000110100000100011010100011011111\
01110010001001001110011011000001100110100111110101000\
01011001000011111100010110110000010111011100010111000\
10011101011000000100111010000010111000100010000001010\
00101011110001110000110101001100010000100001111100010\
10101111010010101001011011100011101111000
```

Наведемо фрагмент програми в Python та результати його роботи:

```
def encrypt(pk, plaintext):
    # Unpack the key into it's components
    key, n = pk
    # Convert each letter in the plaintext to numbers based on the character using a^b mod m
    cipher = [pow(ord(char), key, n) for char in plaintext]
    # Return the array of bytes
    return cipher


def decrypt(pk, ciphertext):
    # Unpack the key into its components
    key, n = pk
    # Generate the plaintext based on the ciphertext and key using a^b mod m
    aux = [str(pow(char, key, n)) for char in ciphertext]
    # Return the array of bytes as a string
    plain = [chr(int(char2)) for char2 in aux]
    return ''.join(plain)



'''
Detect if the script is being run directly by the user
'''
print("============================================================
==========================================")
print("================================== RSA Encryptor / Decrypter
==========================================")
print(" ")

#p = int(input(" - Enter a prime number (17, 19, 23, etc): "))
#q = int(input(" - Enter another prime number (Not one you entered above): "))


min_prime = 124241834982344423
max_prime = 224241834982344423
p = sympy.randprime(min_prime,max_prime )
q = sympy.randprime(min_prime,max_prime )
```

```
    print(" - Generating your public / private key-pairs now . . .")

    public, private = generate_key_pair(p, q)

    print(" - Your public key is ", public, " and your private key is ", private)

    print(' - The length of the min_prime is: ')
    print(len(str(min_prime)))
    print(' - The length of the max_prime is:')
    print(len(str(max_prime)))


    print('The execution time is :')
    print("--- %s seconds ---" % (time.time() - start_time))

    message = input(" - Enter a message to encrypt with your public key: ")
    encrypted_msg = encrypt(public, message)

    print(" - Your encrypted message is: ", ".join(map(lambda x: str(x), encrypted_msg)))
    print(" - Decrypting message with private key ", private, " . . .")
    print(" - Your message is: ", decrypt(private, encrypted_msg))
    print(" ")
    print("=========================================== END ============================================================")
    print("==============================")
```

Безпека генератора **RSA**, так само, як і BBS, грунтується на складності розкладання *n* на множники.

Результат виконання програми :

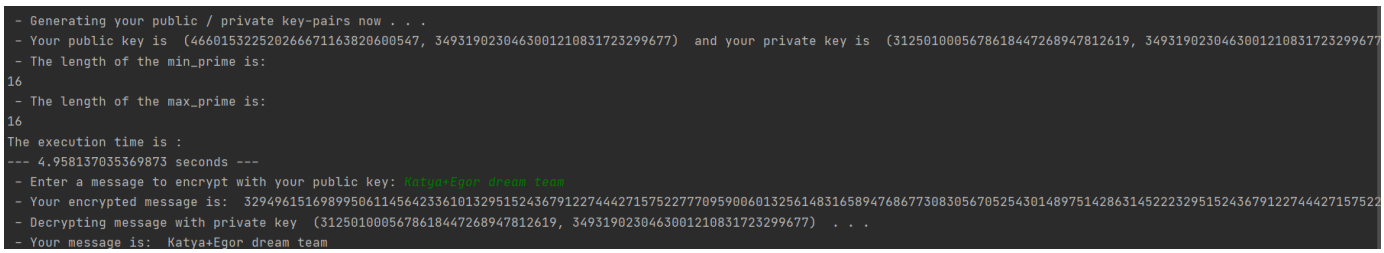
```
 - Generating your public / private key-pairs now . . .
 - Your public key is  (865268199477263364754223155392671 9, 3105490386578310375537100474071 4313)  and your private key is  (220586849625007445277144356490 1
 - The length of the min_prime is:
18
 - The length of the max_prime is:
18
The execution time is :
--- 25.237709999084473 seconds ---
 - Enter a message to encrypt with your public key: Math is cool
 - Your encrypted message is:  138995236385718326230027442244289862517981758052818807395390255143687218371215627530686392351742045378577266264704001787474 7
 - Decrypting message with private key  (220586849625007445277144356490137 99, 3105490386578310375537100474071 4313)  . . .
 - Your message is:  Math is cool
```

```
================================================================================================================
================================= RSA Encryptor / Decrypter ====================================================

 - Generating your public / private key-pairs now . . .
 - Your public key is  (412114976212810648577397551, 4431120483338935949902207 69)  and your private key is  (264807262625832912722970527, 4431120483338935949902207 69)
 - The length of the min_prime is:
14
 - The length of the max_prime is:
14
The execution time is :
--- 0.5541393756866455 seconds ---
 - Enter a message to encrypt with your public key: I am a scientist

 - Your encrypted message is:  1995319776984776358592153151229248047051469224436852641427606065817047835863198691967976312885543682919048831229248047051469224436852641427606065817
 - Decrypting message with private key  (264807262625832912722970527, 4431120483338935949902207 69)  . . .
 - Your message is:  I am a scientist

========================================== END ============================================================
==============================
```



Висновки

Розроблені в середовищах Python та Maple програмні модулі надають можливість усвідомити та дослідити особливості отримання ПВП для криптографічних застосувань на основі використання однобічних функцій з лазівкою. Ці модулі можуть бути покладено в основу створення, як додатків для наукових досліджень, так і навчально-методичних матеріалів для аудиторної та самостійної роботи студентів під час лекцій, практичних занять та лабораторних робіт з різних дисциплін, зокрема, «Математичні основи криптографії», «Теорія ймовірностей та математична статистика», «Вища математика» та ін.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Михалевич В. М. Навчально-контролюючий Maple — комплекс з вищої математики / В. М. Михалевич // Інформаційні технології та комп'ютерна інженерія. — 2004. — № 1. — С. 74–78.
2. Остапов С. Е. Основи криптографії [Текст] : навч. посіб. / С. Е. Остапов, Л. О. Валь; Чернівці: Книги - XXI, 2017. – 188 с. ISBN 978-966-2147-35-3.
3. Горбенко Ю. І. Методи та засоби генерування псевдовипадкових послідовностей / Ю. І. Горбенко, Н. В. Шапочка, Т. О. Гріненко, А. В. Нейванов, Р. І. Мордвінов // Прикладна радіоелектроніка: наук.-техн. журнал. Харьков. – 2011. – Том 10. № 2. – С. 141-152.
4. Яремчук Є. В. Оцінка періоду псевдовипадкових послідовностей на основі р-чисел Фібоначчі [Текст] / Є. В. Яремчук, О. Д. Азаров // Вимірювальна та обчислювальна техніка в технологічних процесах. – 1999. – № 2. – С. 100-103. http://ir.lib.vntu.edu.ua/handle/123456789/9257
5. Лужецький В. А. Рекурентні Vk-послідовності / В. А. Лужецький, Ю. Є. Яремчук // Вісн. Вінниц. політехн. ін-ту. - 1999. - № 6. - С. 53-59.
6. Лужецький В. А. Щільність заповнення ряду натуральних чисел членами окремої зворотної послідовності другого порядку/Лужецький В. А., Михалевич В. М., Михалевич О. В., Каплун В. А. // Інформаційні технології та комп'ютерна інженерія. – 2010. – №1(17) – С. 46-51.
7. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си = Applied Cryptography. Protocols, Algorithms and Source Code in C. – М.: Триумф, 2002. – 816 с.
8. Михалевич В. М. Постановка та розв'язання задачі знаходження найменших та найбільших значень основних характеристик окремого класу дволанкового деформування / В. М. Михалевич, О. В. Краєвський // Вісник машинобудування та транспорту. – Вінниця: ВНТУ – 2019. – № 10(2). – С. 40 47; https://doi:https://doi.org/10.31649/2413-4503-2019-10-2-40-47
9. Михалевич В. М. Розвиток системи Maple у навчанні вищої математики майбутніх інженерів-механіків : монографія / В. М. Михалевич, Я. В. Крупський. — Вінниця: ВНТУ, 2013. — 236 с. ISBN. — 978-966-641-539-7.
10. Михалевич В. М. Використання систем комп'ютерної математики у процесі навчання лінійного програмування студентів ВНЗ: монографія / В. М. Михалевич, О. І. Тютюнник. – Вінниця: ВНТУ, 2016. – 279 с. ISBN 978-966-641-670-7.
11. Михалевич В. М. Розробка електронних освітніх ресурсів в середовищі СКМ Maple [Текст] / В. М. Михалевич, Я. В. Крупський, Ю. В. Добранюк // Математика та інформатика у вищій школі: виклики сучасності : зб. наук. праць за матеріалами Всеукр. наук.-практ. конф., 18-19 травня 2017 р. / М-во освіти і науки України, Вінницький державний педагогічний університет імені Михайла Коцюбинського [та ін.]. - Вінниця : ФОП Рогальська І. О., 2017. - С. 69-72. Режим доступу: https://conferences.vntu.edu.ua/index.php/pmovc/index/pages/view/zbirn2018 Дата звернення: Черв. 2018

12. Михалевич В. М. Фрагменти електронних освітніх ресурсів з функції двох змінних в середовищі СКМ Maple [Текст] / В. М. Михалевич, Ю. В. Добранюк, Я. В. Крупський //http://ir.lib.vntu.edu.ua//handle/123456789/15474
13. Михалевич В. М. Курс математики для слухачів-іноземців в середовищі СКМ Maple. Алгебраїчні рівняння і системи рівнянь: Електронний освітній ресурс / В. М. Михалевич, Н. Б. Дубова, І. А. Клеопа – Вінниця : ВНТУ, 2019. – 64 с.
14. Михалевич В. М. Електронний освітній ресурс з курсу математики для слухачів-іноземців в середовищі СКМ Maple [Текст] / В. М. Михалевич, Н. Б. Дубова, І. А. Клєопа // Збірник наукових праць за матеріалами дистанційної всеукраїнської наукової конференції «Математика у технічному університеті XXI сторіччя», м. Краматорськ, 15–16 травня 2019 р. – Краматорськ : ДДМА, 2019. – С. 193-195.
15. Краєвський В. О. Спецкурс математичного аналізу. Диференціальні рівняння з частинними похідними та їх аналіз в системі Maple [Текст] : навч. посіб. / В. О. Краєвський, Н. В. Сачанюк-Кавецька ; Вінниц. нац. техн. ун-т. - Вінниця : ВНТУ, 2017.
16. Бедратюк, Л. П. Системи комп'ютерної алгебри Maple в елементарній теорії чисел / Л. П. Бедратюк, Г. І. Бедратюк // Восточно-Европейский журнал передовых технологий. – 2013. – № 6/4 (66). – С. 10–13.
17. Клочко В. І. Вища математика з комп'ютерною підтримкою. Теорія функцій комплексної змінної : навч. посіб. / В. І. Клочко, С. А. Кирилащук — Вінниця: ПП «Торговий дім Едельвейс і К», 2010. – 128 с.
18. Дерець Є. В. Дослідження готовності студентів до використання комп'ютерно-орієнтованих систем навчання математики / Є. В. Дерець // Збірник наукових праць Дніпровського державного університету. - 2017. - Режим доступу: http://sj-dstu.dp.ua/index.php/sjdstu/article/view/153/163
19. Мороховець Г. Ю. Використання системи комп'ютерної математики MAPLE у навчанні медичній і біологічній фізиці / Г. Ю. Мороховець, М. С. Саєнко, Ю. В. Лисанець, О. В. Сілкова // Проблеми екології та медицини. – 2018. – Т. 22, № 1-2. – С. 60–62.
20. https://dou.ua/lenta/articles/language-rating-jan-2020/

Оксана Іванівна Тютюнник *— кандидат педагогічних наук, доцент кафедри вищої математики, Вінницький національний технічний університет, м. Вінниця, e-mail:* ***tutunnik.oksana@gmail.com***

Єгор Сергійович Дремлюга *– студент групи 1БС-20Б, факультет інформаційних технологій та комп'ютерної інженерії. Вінницький національний технічний університет, Вінниця, email: tatlan@meta.ua*

Катерина Вікторівна Медведєва *– студентка групи 1БС-18Б, факультет інформаційних технологій та комп'ютерної інженерії. Вінницький національний технічний університет, Вінниця, email: tatlan@meta.ua*

Науковий керівник: ***Володимир Маркусович Михалевич*** — д-р техн. наук, професор, завідувач кафедри вищої математики, Вінницький національний технічний університет, м. Вінниця, e-mail: vmykhal@gmail.com

Oksana I. Tiytiynnyk — Candidate of Pedagogical Sciences (Eng.), Docent of the Chair for Higher Mathematics, Vinnytsia National Technical University, Vinnytsia, e-mail: ***tutunnik.oksana@gmail.com***

Yehor S. Dremliuha — student of group 1BS-20B, Faculty of Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: ***western.ant2@gmail.com***

Kateryna Medvedieva— student of group 1BS-18B, Faculty of Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail ***medvedieva.katya@gmail.com***

Supervisor: ***Mykhalevych Volodymyr M.*** — Dr. Sc. (Eng.), Professor, Head of the Chair for Higher Mathematics, Vinnytsia National Technical University, Vinnytsia, vmykhal@gmail.com.