Podstawy praktycznej kryptografii – rozgrzewka

Raport 1

Zadanie 1

Treść: Napisz w Pythonie skrypt, który będzie w pętli generował kolejne całkowite liczby pseudolosowe z zakresu 0-10000 za pomocą funkcji randint z modułu random. Skrypt ma przyjmować dwa argumenty: 1. wartość seeda (liczba całkowita), która będzie inicjalizować generator tylko raz, na początku pracy skryptu, 2. ilość powtórzeń pętli (liczba całkowita większa od 0).

Kod:

```
import argparse
import random

parser = argparse.ArgumentParser(description="Pseudolosowanie")
parser.add_argument("a")
parser.add_argument("b")
args = parser.parse_args()
lista=[]
i=0
args.b = int(args.b)
random.seed(args.a, version=2)
for i in range (0,args.b):
    lista.append(random.randint(0,10000))
print(lista)
```

Wykonaj eksperymenty uruchamiając skrypt 10 razy z różnymi argumentami. Co się stanie, gdy będziesz podawać te same wartości seeda?

```
PS C:\Users\USER\desktop\studia\2sem> python ./lab1.py 10 10   Seed 10, Powtórzeń 10
[4925, 3638, 1771, 1511, 5868, 2351, 8817, 892, 7318, 6733]
PS C:\Users\USER\desktop\studia\2sem> python ./lab1.py 5 10    Seed 5, Powtórzeń 10
[5776, 4412, 8475, 1015, 8341, 3694, 7580, 9673, 1694, 9471]
PS C:\Users\USER\desktop\studia\2sem> python ./lab1.py 34 10  Seed 34, Powtórzeń 10
[4479, 2916, 6841, 8810, 5609, 1181, 6732, 7864, 807, 3017]
PS C:\Users\USER\desktop\studia\2sem> python ./lab1.py 12 10  Seed 12, Powtórzeń 10
[9702, 2112, 9489, 5050, 490, 6796, 2795, 1799, 8275, 5976]
PS C:\Users\USER\desktop\studia\2sem> python ./lab1.py 12 9   Seed 12, Powtórzeń 9
[9702, 2112, 9489, 5050, 490, 6796, 2795, 1799, 8275]
PS C:\Users\USER\desktop\studia\2sem> python ./lab1.py 15 9   Seed 15, Powtórzeń 9
[1993, 3883, 9096, 1716, 5650, 3241, 9174, 1465, 5953]
PS C:\Users\USER\desktop\studia\2sem> python ./lab1.py 9 9    Seed 9, Powtórzeń 9
[2730, 1408, 484, 3673, 1347, 3251, 1140, 3250, 2001]
PS C:\Users\USER\desktop\studia\2sem> python ./lab1.py 9 10   Seed 9, Powtórzeń 10
[2730, 1408, 484, 3673, 1347, 3251, 1140, 3250, 2001, 5281]
PS C:\Users\USER\desktop\studia\2sem> python ./lab1.py 7 10   Seed 7, Powtórzeń 10
[9230, 8952, 2115, 4885, 3323, 7538, 8858, 5974, 7722, 3972]
PS C:\Users\USER\desktop\studia\2sem> python ./lab1.py 7 9    Seed 7, Powtórzeń 9
[9230, 8952, 2115, 4885, 3323, 7538, 8858, 5974, 7722]
```

Po podaniu konkretnej wartości seeda generują się różne wartości, ale w momencie, gdy wprowadzony zostaje taki sam seed to wygenerowane wartości są takie same.

Przykładowo dla seeda 7 uzyskujemy za każdym razem: [9230, 8952, 2115, 4885, 3323, 7538, 8858, 5974, 7722]

Podobnie dzieje się dla seeda 9 oraz 12. Zmiana ilości powtórzeń również nie wpływa na zmianę otrzymanych wartości

Wniosek: Dla każdego seeda liczby generowane są według konkretnego algorytmu, oznacza to, że te liczby nie są losowe a pseudolosowe.

Zadanie 2

Napisz w Pythonie skrypt, który będzie zwracał hashe dla pliku, którego ścieżka została przekazana jako jedyny argument. Wykorzystaj moduł hashlib. W skrypcie wygeneruj wszystkie podstawowe hashe, które pojawiły się na wykładzie.

Kod:

```
import hashlib
import argparse
parser = argparse.ArgumentParser(description='Hashowanie')
parser.add_argument('sciezka', type=str)
args = parser.parse_args()
plik = open(args.sciezka, "r", encoding='utf-8')
tekst = plik.read()
plik.close()
kodowanie = tekst.encode('utf-8')
print('Podstawowy tekst: ' + tekst)
print("hash_md5 : ")
print(hashlib.md5(kodowanie).hexdigest())
print("hash_sha1 : ")
print(hashlib.sha1(kodowanie).hexdigest())
print("hash_sha224: ")
print(hashlib.sha224(kodowanie).hexdigest())
print("hash_sha256: ")
print(hashlib.sha256(kodowanie).hexdigest())
print("hash_sha384: ")
print(hashlib.sha384(kodowanie).hexdigest())
print("hash_sha512: ")
print(hashlib.sha512(kodowanie).hexdigest())
print("hash_sha3_224: ")
print(hashlib.sha3_224(kodowanie).hexdigest())
print("hash_sha3_256: ")
print(hashlib.sha3_256(kodowanie).hexdigest())
print("hash_sha3_384: ")
print(hashlib.sha3_384(kodowanie).hexdigest())
print("hash_sha3_512: ")
print(hashlib.sha3_512(kodowanie).hexdigest())
print("hash_blake2b: ")
```

Przeprowadź prosty eksperyment:

1. stwórz plik tekstowy

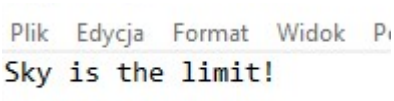
2. wygeneruj dla niego wszystkie hashe (zapisz je sobie)

3. zmień jeden znak w pliku

4. wygeneruj ponownie wszystkie hashe

5. porównaj hashe z obu uruchomień.

Plik tekstowy:




```
PS C:\Users\USER> cd desktop
PS C:\Users\USER\desktop> cd studia
PS C:\Users\USER\desktop\studia> cd 2sem
PS C:\Users\USER\desktop\studia\2sem> python ./parsowanie.py C:\Users\USER\Desktop\studia\2sem\dohashy.txt
Podstawowy tekst: Sky is the limit.
hash_md5 :
7342b36b32595025988acd2b74e43b00
hash_sha1 :
87b94b9e791582029b10d127d1c7fe5160eea931
hash_sha224:
f49dcbb84fc58f6a71b6e158c02e9ee3909b71ba1c6517fe01fd4651
hash_sha256:
45dc7a8b9593b316015b96b24c28a88a76deacc75cc5a5dd44c9daebb2ae4d32
hash_sha384:
7910d1fbd9610e6ca2bfdc8b62e963b43481e9a06c1e51a6df120f2ff2aa14a929440944ca04e7f9c5bb4b30143c76c1
hash_sha512:
9eebccc4c20c3b1614ed56fe97324a2942803bbd857bf91fa90969fd0f285e726cd477c433e23ce64bfb8281ad3d972c4680598df093c407885868ffee56e130
hash_sha3_224:
1b6e165631e3d457e41ac9f7db8458724263ee34e158fd1e3e777832
hash_sha3_256:
31f5b06ccde75bd1e60169b872940f8f7f7de688f30b464ea570911bb54385f2
hash_sha3_384:
0dfb9fc59ba66e7774bf112731bf6b44b0a1ac2b90423d97fdc9901e6d49f2803c832914b0acaaaac069159a45e5f452
hash_sha3_512:
56e8d32cbc89d244d761ba630cf85ced1622e8b4a7994c618e51afa6edb46736c0ea7486a96c8dd5ff3d7188e7dcc31a9dd3dc7aefb8f33eb4eaac116b13c38a
hash_blake2b:
PS C:\Users\USER\desktop\studia\2sem>
```

W pliku tekstowym zmieniono kropkę na wykrzyknik i uzyskano takie hashe:

```
PS C:\Users\USER\desktop\studia\2sem> python ./parsowanie.py C:\Users\USER\Desktop\studia\2sem\dohashy.txt
Podstawowy tekst: Sky is the limit!
hash_md5 :
3ebbd635f849e0a9afa1fc46077017fd
hash_sha1 :
1c433a19cbfd19360b8fe0744057587183c78576
hash_sha224:
1f1ddfb47d93effccf196142fdadddadaa64982c81f11568bcbb6139
hash_sha256:
33368f65d2de56cac800d34e8977827ae74d87b5796931f722a3fb9a59947ab6
hash_sha384:
8fae1a52d25f147522aa0d14d0c1c539377f0034e49ef0ef5f82fc87f6b2498a6cb016e37050195c66f88da51cd2f480
hash_sha512:
260028b8eafd1a4ef680c0871f822ebc1c67d72220e8546610f134908ca31c80020379e871b65ecce784686c1e7f36fcabcf70d5968f2107a9ce7c90865fef6b
hash_sha3_224:
a5fc80ace492c5ba7b1d95b20f84432c26721022970e66cd635990ab
hash_sha3_256:
3b9abe639b5a192cccc82e05c3ebf85dcc42808a3e60bf30673f78797fee6853
hash_sha3_384:
ae1c2daada79295944be73d12c23252a04fce03525ee589eda9916e70462facb28f47f79c60944d10341b20360b890cb
hash_sha3_512:
0c05bbec5ddc959c8fac29a3b1dea6d696939d976c3aa81cba99ca123e14f67aa441cc06dce3f31bd65b347dcd2aa2e14b63efe6707b137d282f316fb6db40b2
hash_blake2b:
PS C:\Users\USER\desktop\studia\2sem>
```

Porównanie wygenerowanych hashy:

Po zmianie jednego znaku hashe zostają zaktualizowane i całkowicie się różnią od poprzednich. Ciężko zaobserwować jakiekolwiek podobieństwa.