Aleatoriedad en Criptografia

Valentina S. Vispo

Octubre 2023

Índice

1 Aleatoriedad

1.1 Criptografia y la aleatoriedad

Uno de los aspectos importantes para la seguridad de los sistemas criptograficos es que la generacion aleatoria sea impredecible y rapida.

La aleatoriedad puede ser real o estadistica. Dado un evento probabilistico, el resultado de esto se le llama variable aleatoria que tiene una distribucion de probabilidad.

Luego, la suma de las probabilidades de cada evento es siempre 1.

Para medir la aleatoriedad se utiliza la entropia. El concepto de la entropia es medida generalmente en bits con la siguiente formula:

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i)$$

Existen 2 tipos de entropia:

1. distribucion uniforme
2. distribucion no uniforme

Decimos que una variable tiene una distribucion uniforme cuando todos los eventos son igualmente probables. Y distribucion no uniforme cuando la probabilidad de todos los eventos tienen diferentes probabilidades de ocurrir.

Con respecto a la maxima entropia que se puede obtener de una cadena de n bits, esta corresponde a la distribucion uniforme. Es exactamente de n bits.

1.2 PRNG (Pseudo Random Number Generator)

Es un generador de numeros pseudoaleatorios que produce una secuencia de numeros estaditicamente aleatorios.

Componentes

- Seed
- Estado con ciclos

Pasos:

1. Se inicializa el seed, si esta ya fue inicializada go to (2).
2. A partir del seed, se genera una secuencia de numeros pseudoaleatorios.
3. Se devuelve esa secuencia creada.

Propiedades:

- Deterministico: dada una seed, si se conocen los valores generados, al reutilizar dicha seed se obtendra los mismos valores (secuencia producida por una seed es deterministica).
- Repeticion de secuencias: dado que el estado interno es finito, cuando se supera el ciclo de dicho estado interno, las secuencias a partir de ese momento se repetiran.

2 Desafio: Generador de numeros aleatorios de Java

2.1 Enunciado

El desafío consiste en adivinar el siguiente número de 32 bits a ser producido por un generador de números pseudoaleatorios congruencial lineal.

Para obtener el numero producido se realiza una peticion al servidor.

Al encontrar el siguiente numero debe ser enviado al servidor. En caso de ser correcto el numero, se recibe **¡Ganaste!**, en caso contrario, si el formato es incorrecto se dara una breve descripcion o si el numero es incorrecto se recibira **Lo siento, siga participando**.

2.2 Analisis

Cada resultado de multiplicar el valor anterior por una constante a, sumarle una constante b y calcular el resto de dividirlo por una constante m.

$$X_i = \left(aX_{i-1} + b\right) \mod m \text{ con } 1 \leq i$$

X_0: seed

a: multiplicador

b: incremento

m: modulo — notar el periodo de una secuencia esta acotado por el modulo

En particular, en este desafio tenemos los siguientes valores:

Multiplicador	**Incremento**	**Modulo**
$a = 25214903917$	$b = 11$	$m = 2^{48}$

El resultado del numero generado sera un numero entero de 32 bits, con signo, expresado como valor decimal.

Como se menciono en las propiedades, el ciclo de estado del generador esta dado por m.

2.3 Solucion

Notas a tener en cuenta

1. El int en Java es de 32 b
2. El long en Java es de 64 b
3. Python no posee un desplazamiento a derecha sin signo.

Pasos

1. Obtener 2 numeros generados, ambos estan expresados en bytes. Estos son los estado del generador.
2. Transformarlos de bytes a int y luego a c_int32
3. Obtengo los posibles valores para los 16 b

4. Con el primer numero generado hago un rshift con los posibles valores de 16b (parte desconocida del estado)
5. Con el valor obtenido en el punto 4 realizo la operacion para el seed
6. Comparo el numero generado con la seed obtenida con el segundo numero generado
   (a) Si son iguales, puede ser que obtuve el seed, lo guardo
   (b) Si no lo son, sigo iterando
7. Si encontre la seed, realizo el envio al servidor

2.3.1 Como correr la solución

```
python3 challenges/2_Java.py
```

2.4 Evitar el problema

Una buena elección de a, b y M

Las buenas propiedades dependen de una elección apropiada de a, b y m, y en algunos casos X_0

- La elección de m se relaciona con: longitud de la secuencia y velocidad computacional.

- La elección de a y b, en función de m, se relacionan con la aleatoriedad.

Por otra parte:

- Tener mas bits secretos.

- No utilizar valores predecibles para inicializar un PRNG.

2.5 Herramientas

Se utilizaron los recursos dados en la pagina de la catedra.

Particularmente la seccion de operaciones con enteros, desplazamiento.

Se creo una clase especial para esto, se puede revisar en el siguiente archivo challenges/customCrypt.py *(requiere permisos)*

```
1  from customCrypt import Krypth
2  # ...
3  # Inicilizacion de la clase que contiene funciones de desplazamiento
4  global c_krypth
5  c_krypth = Krypth()
6  # ...
7  # 4. rshift con el numero generado con los posibles valores de 16b
8  seed_base = c_krypth.lshift_64(a=f_int64b, n=16)
9  # ...
```

Otra de las principales utilidades que se utlizaaron, fue la libreria cytpes de Python3

```
1  from ctypes import c_int32, c_int64
2  # ...
3  first_decoded = c_int32(int(encoded[0])).value
4  second_decoded = c_int32(int(encoded[1])).value
5  # ...
```

2.6 Referencias

1. Enunciado ejercicio https://ciberseguridad.diplomatura.unc.edu.ar/cripto/doc/javarand.html
2. https://famaf.aulavirtual.unc.edu.ar/pluginfile.php/25894/mod_resource/content/1/Aleatoriedad.pdf
3. https://wiki.archlinux.org/title/Random_number_generation
4. https://es.wikipedia.org/wiki/Aleatoriedad
5. https://www.famaf.unc.edu.ar/~jgimenez/Modelos%20y%20Simulacion/2012/clase5_pr.pdf
6. Desplazamintos https://ciberseguridad.diplomatura.unc.edu.ar/cripto/doc/integers.html#desplazamientos-2
7. https://rubenfcasal.github.io/simbook/gen-cong.html
8. https://es.wikipedia.org/wiki/Generador_lineal_congruencial
9. Libreria ctypes de Python3 https://docs.python.org/3/library/ctypes.html
10. https://es.wikipedia.org/wiki/Aleatoriedad_estad%C3%ADstica