RNG: Random Number Generator

Una de las grandes fortalezas de RSA es la dificultad que tienen las computadoras en factorizar números grandes tratando de llegar a 2 factores que sean primos. Sería infinitamente más fácil para una computadora tratar de replicar la periodicidad de una pseudo generador de números aleatorios que tratar de calcular los dos números primos. El último experimento que se hizo demoró 3 años en factorizar números que no llegaban a los 1024 bits de longitud, por lo cual tenemos que asegurarnos que el generador de primos aleatorios sea verdaderamente fuerte y no se convierta en una vulnerabilidad para el RSA.

Existen 2 tipos de generadores de un número aleatorio, los pseudo generadores y los generadores verdaderos. Los pseudo generadores son cadenas extremadamente largas de datos que se repiten, y uno de estos números es elegido en orden, pero por la naturaleza tan larga de la cadena se podrían ver ligeros patrones que podrían confundir a un atacante haciéndolo pensar que ha llegado al final de la cadena, cuando en realidad este patrón en tan solo un a coincidencia.

Podríamos asumir que la cadena 2,3,1,4 es una lista de números aleatorios.

Esta sería una lista extremadamente débil dado que después de 8 iteraciones podríamos tener por seguro cual es la lista y podríamos atacar el RSA con éxito dado que ya sabemos cuál será el próximo número en generarse.

Podríamos agregar una dimensión de complejidad y decir de la lista 4,1,3,2 se sume corra el número n cantidad de espacios para generar un número.

Esto nos daría una lista de 16 combinaciones, y no 4, se puede hacer más complicado, y así hay muchos métodos para expandir la lista y combinaciones, pero al ser pseudo aleatorio siempre existirá la duda que alguien pueda romper la manera en la que se presentan los números y explotar un a vulnerabilidad en el código.

Pero por otro lado los generadores de números reales toman en consideración variables físicas imposibles de medir lo cual hace que esta generación de números sea absolutamente impredecible. Un ejemplo sería lanzar un dado, es verdaderamente imposible calcular cuál será el número que salga, a menos que alguna máquina pueda calcular todas las variables físicas involucradas en el lanzamiento de este y simular una lanzamiento prediciendo qué número saldrá antes que el dado se deje de mover. Aun así, se tiene que lanzar el dado antes de predecir qué número saldrá, así que es completamente imposible saber qué número saldrá con 100% de certeza antes que se haga el tiro, lo cual hace los algoritmos verdaderos impenetrables.

No obstante, existe una debilidad en los algoritmos verdaderos, la cual es la cantidad de variables físicas que pueden variar. A lo que me refiero es que si alguien lanza un dado a 1 milímetro del suelo sin darle ningún tipo de giro es prácticamente imposible que el dado cambie de cara de la que tiene en ese momento mirando hacia arriba. Por otro lado si el dado se tira desde 1 metro de

altura dándole giro en una superficie que no sea lisa se incrementan las variables que afectarán el resultado haciéndolo “más aleatorio”.

Entonces, en la búsqueda de un algoritmo de generación de primos grandes aleatorios para el RSA debemos optar por un algoritmo aleatorio verdadero porque es más seguro que un pseudo aleatorio. Lamentablemente los algoritmos verdaderos tienen desventajas a comparación de los algoritmos pseudo aleatorios. Los 2 principales problemas que presentan los algoritmos verdaderos son que por un lado necesitan algún tipo de medición del mundo físico mediante un hardware y que su generación es mucho más lenta y tiene un costo computacional mucho mayor al del algoritmo pseudo aleatorio. Tocaremos este tema a continuación.

a) Periodicidad: Queremos que nuestro algoritmo no tenga periodicidad, esto es único a los algoritmos pseudo aleatorios dado que en algún punto se quedará sin números que generar y tendrán que regresar al comienzo de la lista de todos los números que puede generar, probando que tienen un periodo. Es por esta vulnerabilidad que no queremos esta periodicidad y queremos usar un algoritmo verdadero.
b) Predictibilidad: Como ya se comentó, un buen algoritmo verdadero no puede ser predicho, que es justamente lo que queremos en nuestro generador de números aleatorios para el RSA.
c) Irreproducible: Guiándonos de lo previamente dicho, tampoco queremos que esté generado pueda ser producido, dado que si se pudiese producir vulneraría la generación de primos y se podría romper el RSA.
d) Independencia de valores: Cuando se habla de la independencia de valores estamos hablando de qué valores se están midiendo para generar una semilla, como ya mencioné entre más valores se están tomando en cuenta de manera independiente es más “aleatoria” la generación de números.
e) Complejidad: Como el ejemplo del dado, entre más factores se tomen en cuenta más complejo se vuelve el algoritmo. Por ejemplo, si se usara una imagen dependiente de la densidad de píxeles en la imagen y la cantidad de bits se que se tiene que procesar se puede hacer un variar los valores de qué píxeles se obtendrán en base de los bits que se necesitan y los píxeles disponibles en la imagen.
f) Propiedades estadísticas: En cuanto a estadística queremos que cada valor tenga la misma probabilidad de salir que cualquier otro valor. Es decir, si compilamos una distribución de valores deberíamos tener una línea recta teórica, dado que para que todos los valores salgan la misma cantidad de veces necesitaríamos una cantidad infinita de tiempo e iteraciones del algoritmo.
g) Número de bits: El algoritmo debe trabajar con una cantidad variable de bits, entonces la semilla que se generará también se adecuará a pedir 1 bit como pedir 2048 bits.
h) Semilla: La semilla es el número que se genera en base a los estímulos físicos detectados por el hardware que termina siendo el número del cual se generan los números primos grandes. 2 semillas iguales deben dar el mismo

par de primos grandes, pero generar esta semilla 2 veces debería ser prácticamente imposible y se haría por chance.

Se nos ha recomendado usar 7 diferentes estímulos físicos para generar nuestra semilla y por consiguiente los primos grandes para el RSA. De estos he decidido descartar la mayoría dado que piden variables físicas que no todos los dispositivos tienen. El RSA se presenta en la gran mayoría de dispositivos desde computadoras hasta Smartwatches, es por estos que debemos usar algún estímulo físico que todos estos dispositivos tengan. Por ejemplo, una pc podría usar la luz del ambiente para generar una semilla, pero debe tener una webcam para poder digitalizar esta variable física y luego poder analizarla para generar la semilla. Lo mismo para con el sonido, se necesita un micrófono. La información TCP necesita una conexión a una red, como también el RTT para poder medir la congestión de esta. Otro ejemplo del que se habló en clase fue la posición del mouse, pero no todos los dispositivos tienen mouse, es por esto que tenemos que elegir un método que sea una variable física que todos los dispositivos tengan. Mi propuesta es elegir entre el tiempo interno de un dispositivo, o la actividad de su CPU.

Reloj interno: Las ventajas del reloj interno es que todos los dispositivos modernos tienen algún tipo de reloj interno que siempre está corriendo dado que se tiene un batería (usualmente de reloj) que hace que ese contador nunca pare, todos los dispositivos tienen relojes diferentes porque a todos se le ha puesto esta bateria en momentos diferentes así sean muy parecido incluso en una línea de ensamblaje es diferente. Esto nos da mucha variabilidad de dispositivo a dispositivo, y la gran mayoría de estos datos estarán guardados en una forma similar, con segundos, minutos, meses y años lo cual lo haría más fácil poder implementar en varios dispositivos.

CPU: Nuevamente todos los dispositivos tienen un CPU integrado, no obstante, estos pueden variar mucho en su capacidad de procesamiento como cores y como delay en estos. Esto podría ocasionar problemas dado que un aparato con un cpu muy débil tendría menos factores que tomar en cuanto haciéndolo menos aleatorio mientras que el cpu de una cpu con varios núcleos tiene muchas más variables que incrementa la fuerza de la generación de la semilla.

Por defecto C++ tiene una función de generación verdaderamente aleatoria de semillas que usa el reloj interno de la computadora para generar estas semillas, lamentablemente por la naturaleza cronológica del reloj y del tiempo estas semillas son prácticamente no aleatorias en lo absoluto, dado que el tiempo incrementa linealmente y por esto la generación de sedes se parecen.