Luis Miguel Benitez Ruiz

994361

**Tarea 4**

**Aplicación de Computo Paralelo**

RSA es un algoritmo de encriptación muy común, llamado así por las siglas de sus inventores, Ribest, Shamir y Adleman. La idea del algoritmo es la utilización de dos llaves, una publica y una privada. Un mensaje encriptado con la llave publica solo puede desencriptarse con la llave privada, y uno encriptado con la llave privada solo se desencripta con la llave pública. Generalmente los mensajes se encriptan dos veces, primero con la llave publica del receptor, para que solo este pueda desencriptar y leer el mensaje con su llave privada, y después con la llave privada del emisor, para que quien lo reciba verifique la autenticidad del emisor al desencriptar el mensaje con la llave publica de este.

La fortaleza del algoritmo está en el alto costo computacional de calcular la llave privada a partir de la llave pública. La llave publica está compuesta por dos números, n, calculado como el producto de dos números primos p y q, y un número e. La llave privada por su parte utiliza el mismo modulo n, y un número d tal que

$$d * e \equiv 1 mod((p-1) * (q-1))$$

Como ejemplo, para una llave publica (e,n) y una llave privada (d,n) , una cifra se encripta y desencripta de la siguiente forma:

$$C = M^e mod\ n$$

$$M = C^d mod\ n$$

La dificultad de calcular la llave privada radica en que es muy costoso realizar la factorización de un entero muy grande, así que es muy importante que p y q sean números grandes.
El problema de los números grandes, y el que se aborda en el artículo de Mahajan y Singh, es que también es muy costoso encriptar un mensaje utilizando una llave muy grande, pues hay que dividir el mensaje completo en partes y calcular exponentes y realizar operaciones de modulo por cada una de esas partes. Una forma más eficiente de encriptar utilizando RSA permitiría utilizar llaves de mayor tamaño, lo que a su vez proporcionaría mayor seguridad.
Los autores proponen paralelizar la encriptación con RSA utilizando GPUs con el siguiente algoritmo:

1) CPU recibe mensaje y llaves para encriptación
2) CPU copia el mensaje y las llaves a la memoria de la GPU
3) En el GPU, se encripta cada carácter en el mensaje de forma individual y paralela, utilizando un hilo por cada carácter en el mensaje.
4) El control regresa al CPU y se copian de regreso los valores encriptados.

De esta forma se busca acelerar el procesamiento de RSA utilizando paralelismo SIMD sobre el mensaje, pues la misma instrucción se ejecutara sobre cada carácter en el mensaje.

Los autores realizan una comparación entre dos ambientes para programación en GPU: CUDA y OpenCL.LA prueba se realiza en una GPU Nvidia GeForce GT 630M con 192 cores y 512 MB de memoria.

La primera prueba se realiza con una n = 131*137 = 17947. El tamaño de los mensajes de prueba empieza en 256 bytes y termina en 32784 bytes. El mensaje se divide en bloques de 64 bytes y cada bloque es manejado por un hilo. De acuerdo a los resultados obtenidos, CUDA fue más eficiente, y llego a tener una mejora en velocidad de hasta 4.01, i.e. la tarea se realizó en aproximadamente 25% del tiempo de ejecución para el algoritmo secuencial. La ejecución con OpenCL fue un poco más lenta y la mejora más alta obtenida fue de 2.92.

La segunda prueba se realiza con una n=1009*509 = 513581, y una vez más la implementación en CUDA fue más rápida, aunque es importante notar que la GPU utilizada era NVIDIA, así que puede decirse que estaba diseñada para funcionar con CUDA. En este caso el tamaño de los mensajes de prueba sigue el mismo patrón, y va de 256 bytes a 32784 bytes. Un resultado interesante es que a partir de 1024 bytes, la diferencia en tiempo de ejecución es muy poca, e incluso es más rápida la ejecución con 32784 bytes que con 1024. Los autores consideran que el alto potencial de paralelismo de este problema permite que mientras aumente el tamaño de los datos, mayor sea el provecho que se saque del paralelismo del algoritmo.

Esta es una aplicación relativamente sencilla pero muy útil del cómputo paralelo. La encriptación RSA es muy común, y un método rápido de calcularlo permitirá mejorar la seguridad digital al permitir el uso de llaves más grandes, reduciendo las posibilidades de poder descifrar un mensaje encriptado sin autorización.

## Referencias

Mahajan, S., & Singh, M. (2014). Performance Analysis of Efficient RSA Text Encryption Using NVIDIA CUDA-C and OpenCL. *International Conference on Interdisciplinary Advances in Applied Computing.* Amritapuri,India: ACM.