An Efficient Parallel Algorithm for Secured Data Communications Using RSA Public Key Criptography Method

Christofer Fabián Chávez Carazas
Universidad Nacional de San Agustín
Algoritmos Paralelos

25 de abril de 2017

1. Problema

RSA es un algoritmo en encriptación y desencriptación con llave publica basado en exponenciación y factorización de enteros con números muy largo, de unos 1024 bits. El algoritmo secuencial de RSA consume mucho tiempo y energía. Además, es difícil computar enteros muy grandes en la infraestructura GCC.

2. Propuesta

El paper propone un algoritmo paralelo para RSA implementado con la libreria GMP y con la librería OpenMP en la infraestructura GCC.

3. RSA

RSA es uno de los algoritmos más importantes hoy en día en la encriptación y la autenticación de datos transmitidos por toda la internet. RSA esta basado en la factorización de números muy largos, esto provee una fuerte seguridad. RSA esta dividido en tres partes: Generación de las Keys, Encriptación y Desencriptación.

4. Parallelization of RSA

En el paper implementan un algoritmo paralelo para la exponenciación modular usando dos tecnologías: GNU MP; para implementar las tres partes del RSA, y OpenMP para paralelizar el algoritmo y obtener mayor rendimiento.

$g^e \bmod m = (g^{e/2} * g^{e/2}) \bmod m$.

The recursive definition for the repeated square-and-multiply method is described in Figure 1.

$$\text{ModExp}(g,e,m) = \begin{cases} 1, & \text{if } e = 0 \\ (g \times \text{ModExp}(g, (e-1), m)) \bmod m, & \text{if e is odd} \\ \text{ModExp}(g, e/2, m)^2 \bmod m, & \text{if e is even} \end{cases}$$

Figure 1. repeated-square and multiply method

```
1) Result=1
2) N=Power/Number_Of_Cores
3) Divide the whole modular exponentiation and
reduction into N parts
4) For each part execute step 5 PARALLELY
5) IF Power mod Number_of_Cores == 0
      FOR I = 1 to N
        Result = Result * Base;
      END FOR
   ELSE
      FOR I = 1 to N
        Result = Result * Base;
      END FOR
      Result = Result * Base;
   END IF
6) Cipher = Result MOD Modulus
```

Figure 2. Proposed Parallel RSA Algorithm using repeated-square and multiply method

5. Experimentos

5.1. SetUp

- Procesador: AMD FX(tm)-8120 Eight-Core Processor 3.10 GHz
- RAM: 4,00 GB
- O.S.: Ubuntu Linux 11.04
- Platform: GCC Infraestructure

5.2. Test Case Set 1

Tamaño de keys: 128 a 1280 Tamaño de mensaje: 5000 caracteres.

TABLE I. TEST CASE – SET 1: COMAPARATIVE RESULTS OBTAINED USING FIXED MESSAGE SIZE BUT VARIED KEY SIZES

S. No	Key Size (in Bits)	Program Segment Type	Time (in seconds) taken w.r.t. the serial and parallel execution of the same code on varied number of cores					
			Serial Code (Core 1)	*2 cores*	*4 cores*	*6 cores*	*8 cores*	*Speedup on 8 cores w.r.t. to sequential code*
1	128	Encryption Time	0.01025	0.00637	0.00457	0.00486	0.00423	2.43x
		Decryption Time	0.06553	0.0478	0.02566	0.0137	0.01399	4.69x
		Overall Time	0.07928	0.05769	0.03371	0.0221	0.02252	3.53x
2	256	Encryption Time	0.02001	0.01558	0.00889	0.00583	0.00987	2.03x
		Decryption Time	0.29416	0.19298	0.09827	0.07153	0.06138	4.8x
		Overall Time	0.31847	0.21277	0.11134	0.08158	0.0751	4.25x
3	512	Encryption Time	0.04934	0.04435	0.02476	0.01474	0.01247	3.96x
		Decryption Time	1.37108	1.0708	0.51514	0.36241	0.30385	4.52x
		Overall Time	1.4267	1.12223	0.54684	0.38409	0.32289	4.42x
4	768	Encryption Time	0.1126	0.07945	0.03794	0.0296	0.02361	4.77x
		Decryption Time	4.81935	3.22759	1.5884	1.0948	1.19792	4.03x
		Overall Time	4.95449	3.32597	1.64481	1.14298	1.23753	4.01x
5	1024	Encryption Time	0.18261	0.01194	0.06308	0.04888	0.03982	4.59x
		Decryption Time	10.988	7.27724	3.59786	2.47073	2.10255	5.23x
		Overall Time	11.23602	7.4667	3.72418	2.5825	2.19647	5.12x
6	1280	Encryption Time	0.2665	0.18026	0.09486	0.06462	0.05506	4.85x
		Decryption Time	20.12456	13.2836	6.56407	4.55322	3.82693	5.26x
		Overall Time	20.47295	13.5461	6.74019	4.69844	3.96789	5.16x

5.3. Test Case Set 2

Tamaño de keys: 1024 Tamaño de mensaje: 1000 a 6000

TABLE II. TEST CASE – SET 2: COMAPARATIVE RESULTS OBTAINED USING FIXED KEY SIZE BUT VARIED MESSAGE SIZES

S. No	Message size (In characters)	Program Segment Type	Time (in seconds) taken taken w.r.t. the serial and parallel execution of the same code on varied number of cores					
			Sequential Code (Core 1)	*2 cores*	*4 cores*	*6 cores*	*8 cores*	*Speedup on 8 cores w.r.t. to sequential code*
1	1000	Encryption Time	0.03016	0.02689	0.01627	0.01080	0.01039	2.91x
		Decryption Time	1.87211	1.44590	0.72740	0.50758	0.41880	4.48x
		Overall Time	1.95527	1.53493	0.80602	0.58074	0.48479	4.04x
2	2000	Encryption Time	0.06294	0.05387	0.03047	0.01805	0.01985	3.18x
		Decryption Time	3.73262	2.92550	1.43904	0.98343	0.85580	4.37x
		Overall Time	3.85117	3.04296	1.53314	1.06385	0.92902	4.15x
3	3000	Encryption Time	0.09095	0.08032	0.04187	0.02821	0.02807	3.25x
		Decryption Time	5.63559	4.34253	2.16426	1.49051	1.27379	4.43x
		Overall Time	5.18047	4.48739	2.26898	1.58123	1.35528	3.83x
4	4000	Encryption Time	0.12020	0.10032	0.05527	0.04260	0.03130	3.85x
		Decryption Time	7.50242	6.22360	2.89878	1.99339	1.67439	4.49x
		Overall Time	7.67651	6.38835	3.01791	2.09883	1.75975	4.37x
5	5000	Encryption Time	0.15060	0.15593	0.06443	0.04974	0.0405	3.72x
		Decryption Time	9.30783	7.22676	3.58106	2.48179	2.10316	4.43x
		Overall Time	9.51191	7.44727	3.71004	2.59500	2.20005	4.33x
6	6000	Encryption Time	0.19511	0.14662	0.07666	0.05487	0.05249	3.72x
		Decryption Time	11.24749	9.07411	4028718	2.98370	2.51541	4.48x
		Overall Time	11.49780	9.28917	4042824	3.10180	2.62606	4.38x