CRIPTOANALISIS DEL ALGORITMO RSA CON UN TAMAÑO MAYOR A 330 BITS, CON TECNICA DE FACTORIZACIÓN

Proyecto De Grado

Lina Fernanda Hidalgo López

Director del Proyecto:

Ing. Esp. Siler Amador Donado

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Ingeniería de Sistemas
Línea de Investigación en ……
Popayán, Noviembre de 2016

Contents

CAPITULO 1. INTRODUCCIÓN

1.1. PLANTEAMIENTO DEL PROBLEMA

1.1.1. Definición

El algoritmo RSA (*Rivest, Shamir y Adleman*) creado en 1977 es uno de los más seguros en la actualidad, ya que emplea una clave de ciframiento numérica bastante grande que convierte la información en una cadena de bits o en un archivo binario que solo puede ser descifrada por los integrantes de la comunicación (emisor, receptor) [1]. El funcionamiento del RSA está basado en expresiones exponenciales en aritmética modular, por tanto los mensajes están protegidos por claves generadas al multiplicar dos números primos que tienen como resultado claves mayores a $300\,bits$ o 100 dígitos decimales elegidos al azar. Teniendo en cuenta lo anterior el proyecto pretende realizar un estudio para claves mayores a 330 bits, ya que el tiempo que requiere el criptoanálisis de estos tamaños de clave no supera los cuatro meses de procesamiento de máquina[2].

En la actualidad el problema del RSA se ha desarrollado en torno a cómo fortalecer el algoritmo de seguridad matemáticamente[3], haciendo cada vez más difícil la forma de vulnerar los sistemas que lo utilizan, encontrando números primos muy grandes que permitan cifrar la información, haciéndola cada vez más difícil de robar. Por otra parte países asiáticos han desarrollado investigaciones para vulnerar este algoritmo mediante: mecanismos hardware[4], matemática modular, análisis de potencia consumida por el algoritmo[5] o semejanzas entre el RSA y otros algoritmos de seguridad[6], con el fin de experimentar posibles formas de vulnerarlo, pero hasta ahora aunque algunos casos han sido exitosos por separado, no logran ser tan eficientes como los métodos de factorización, resultado de la convergencia de todos estos estudios. Existen muchos métodos de factorización[7], la diferencia entre cada uno de ellos se encuentra en que no todos son eficientes en el manejo de números primos demasiado grandes, algunos de ellos se quedan cortos para procesar números muy grandes o tardan demasiado tiempo. Por tanto hoy en día logran destacarse tan solo tres de estos

métodos de factorización, como los más útiles y empleados por los investigadores y hackers en el mundo[8].

Dado que no siempre estas nuevas formas de quebrantar el RSA son efectivas y útiles para todos los tamaños de números primos, se ve la necesidad de volver a los métodos de factorización de números primos grandes para lograr tener una solución efectiva para cualquier tamaño de mensaje. Por lo tanto, este trabajo busca resolver el problema del RSA mediante una forma completa, evaluando, caracterizando y ejecutando los mejores métodos de factorización bajo las mismas condiciones computacionales con el fin de encontrar cuál de ellos resuelve con mayor rapidez un tamaño de RSA mayor a 330 bits empleando un mismo tamaño de mensaje. De esta forma se logrará hacer una diferenciación respecto a los trabajos realizados anteriormente, haciendo uso de las investigaciones de RSA realizadas en la actualidad y unificándolas en un solo sistema. Por ahora, esta es una pregunta que no ha sido totalmente resuelta en estudios relacionados al criptoanálisis, por la poca información de cómo se realizó el proceso en su totalidad[9]–[12].

1.1.2. Pregunta de Investigación

Por lo anterior surge la pregunta de esta propuesta de grado: ¿es posible estimar el desempeño de los métodos de factorización con base en las investigaciones realizadas en los últimos años para vulnerar un tamaño de RSA mayor a 330 bits empleando condiciones de entorno predefinidas?

La hipótesis planteada en este trabajo es que es factible vulnerar un tamaño de RSA mayor a 330 bits en menos de 9 meses, empleando un método de factorización que permita encontrar las claves de la comunicación y por ende descubrir el mensaje oculto, por medio de un análisis.

1.1.3. Justificación

Se puede afirmar que en la actualidad la mayor parte de la información esta almacenada en la red, todo se encuentra interconectado o en la nube y cerca de cada 50 de 100 personas usan internet a diario alrededor del mundo [13]. Por tanto la cifra de robo de información por internet va en aumento conforme pasan los años y es de vital importancia tanto para las empresas como para los usuarios garantizar un manejo adecuado de su información sensible. Los mecanismos y

métodos de protección de la información han evolucionado conforme esta cifra ha crecido, enfocando sus esfuerzos a robustecer los algoritmos de cifrado, recurrir a nuevos protocolos de seguridad y tratar de confundir a los que pretenden robar la información.

La investigación alrededor de estos algoritmos de cifrado ha logrado demostrar que no son totalmente seguros, más bien, por el contrario se ha podido demostrar que la tecnología ha sido un puente para los ladrones de la red hacia encontrar los fallos de seguridad y huecos por donde extraer información. El desarrollo de investigaciones alrededor de este tema permiten corregir y descubrir posibles métodos que podrían ser usados o que ya están siendo utilizados para descifrar la información, por ello muchas compañías e incluso el gobierno recompensan los resultados de estas investigaciones y pagan por el conocimiento exclusivo de cada investigación.

Son muchos los tipos de algoritmos de cifrado que se encuentran disponibles en la actualidad [14], pero el RSA especialmente es uno de los mejores y más robustos, razón por la cual es empleado por muchas compañías y entidades de los gobiernos alrededor del mundo. En 1977 los laboratorios RSA lanzaron retos con diferentes tamaños de clave[10], con el fin de fomentar la investigación sobre este algoritmo y la dificultad de factorizar números primos muy grandes[15], el objetivo de esta competencia era tener un indicativo de las longitudes de claves de números primos que aún son seguras y el tiempo estimado en que se demoraran en ser vulneradas. Desarrollar una investigación en torno a este algoritmo es de suma notabilidad dada su importancia, ya que permite conocer su diseño matemático y un poco más a fondo sus fortalezas y debilidad, lo cual permite analizar las variables que intervienen a la hora de realizar un criptoanálisis y una posible extracción de información. Los trabajos de investigación convergen a generar nuevo conocimiento para contrarrestar nuevos tipos de ataques y generar sus contramedidas.

Particularmente esta investigación pretende documentar una forma clara de cómo realizar un criptoanálisis, lo cual hasta el momento no ha sido totalmente documentado. Esta investigación es relevante en este campo dado que se pretende atacar directamente el núcleo de soluciones existentes y comprobar la fortaleza de su línea temporal de vulnerabilidad con los avances tecnológicos disponibles.

1.2. OBJETIVOS

1.1 Objetivo general

Realizar el criptoanálisis del algoritmo RSA de tamaño mayor a 330 bits empleando una configuración computacional predefinida con el fin de evaluar tres métodos de factorización de números primos.

1.2 Objetivos específicos

- ✓ Definir las métricas o medidas con las cuales se puede seleccionar los métodos de factorización más eficientes en la actualidad para criptoanal (2)izar.
- ✓ Diseñar las pruebas para el criptoanálisis del RSA a partir de las métricas de evaluación definidas.
- ✓ Analizar comparativamente el rendimiento de los métodos de criptoanálisis a partir de la evaluación de las métricas de los métodos de factorización.

1.3. METODOLOGÍA

Se empleará una metodología de análisis, síntesis y comprobación con el fin de comprobar la hipótesis planteada anteriormente, a continuación se describe brevemente cada actividad.

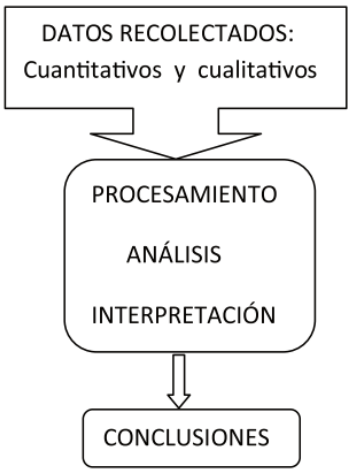

Ilustración 1: Pasos para el desarrollo del proyecto, tomado de **[16]**.

- **Fase 1 Documentación:** Búsqueda de métodos de factorización matemáticos para criptoanálisis del RSA. Trabajos pasados relacionados con la investigación y descripción del funcionamiento del algoritmo matemáticamente y físicamente.
    - Definición de palabras claves
    - Búsqueda bibliográfica
    - Clasificación de la información
- **Fase 2 Clasificación:** Caracterización de las técnicas de factorización de números grandes a partir de modelos existentes empleados anteriormente para el criptoanálisis del algoritmo RSA.
    - Observación de la información
    - Descripción de las características
    - Jerarquización
- **Fase 3 Análisis e interpretación:** Selección de la información a partir de los criterios establecidos en la fase anterior.
    - Filtrado de métodos de factorización
    - Diferenciación entre soluciones matemáticas y computacionales.
- **Fase 4 Comprobación:** Evaluación de un prototipo generado a partir del análisis teórico. Como observación este proyecto no desarrollará el

prototipo funcional, simplemente se enfocará en proponer una posible solución al grupo de investigadores que lo desarrollará con base en la investigación que se está realizando.

    - Selección del tipo de pruebas que se realizarán al prototipo.
    - Aplicación de las pruebas.

- **Fase 5 Conclusión:** Documentación del proceso y resultados del proyecto.

1.4. ESTRUCTURA DE LA TESIS

Capítulo 2: Se describe el estado del arte del algoritmo RSA, funcionamiento, criptoanálisis en el pasado, métodos de factorización existentes y los trabajos actuales alrededor del algoritmo

Capítulo 3: Se describen las métricas de evaluación de los métodos de factorización, junto con los métodos más recientes de vulneración del RSA, el análisis de la información y la posible estimación temporal de la ruptura de RSA mayor a 330 bits.

Capítulo 4: Se expone el diseño de las pruebas propuestas para trabajos futuros.

Capítulo 5: Contiene las conclusiones, el trabajo futuro y los reconocimientos obtenidos en este trabajo de grado

CAPITULO 2. REVISIÓN BIBLIOGRÁFICA

La evolución del algoritmo RSA ha crecido en los últimos años, actualmente existes nuevos mecanismos de fortalecimiento del algoritmo, así como diferentes contramedidas en favor de prevenir y mitigar posibles ataques de intrusos, sin embargo para realizar una criptoanálisis adecuado es importante hacer un recorrido del RSA, desde sus orígenes hasta su avance en la actualidad. Para ello se realizó una revisión sistemática de la literatura sobre cuatro bases de datos (Elsevier, Springer, ScienceDirect e IEEEXplorer), para obtener artículos en inglés o español, publicados relacionados con el diseño matemático, el fortalecimiento y la evolución del algoritmo, los criptoanálisis realizados en los últimos 20 años, los métodos de factorización del RSA y los trabajos actuales. Las búsquedas se limitaron a los resúmenes, palabras clave y título de los artículos. Los estudios se clasificaron según el tipo de investigación en tres grupos: estudios matemáticos y funcionamiento del RSA, factorización del algoritmo (Criptoanálisis existentes) y ataques pasivos, activos. Se analizaron para cada grupo los siguientes factores: diseño y fortalecimiento matemático, fortalezas y debilidades, tipos de ataques, tiempo en recuperar llaves de la comunicación, tipo de tecnología implementada, tamaños de clave de RSA y comparación entre métodos de factorización.

2.1. REVISIÓN SISTEMÁTICA DEL ALGORITMO RSA

2.1.1. Bases de datos y términos de búsqueda

Se realizó una revisión sistemática sobre cuatro bases de datos (Elsevier, Springer, ScienceDirect e IEEEXplorer), para obtener artículos del RSA se usaron las palabras claves: Broke RSA, RSA, RSA challenges, criptoanálisis RSA, Quadratic Sieve, Factorization RSA, RSA algorithm, Quadratic Root, Attack RSA, asimetric criptosystems, retos RSA, funcionamiento RSA, diseño RSA. Las búsquedas se limitaron a los resúmenes, palabras clave y título de los artículos. Para realizar toda la búsqueda y análisis de esta información se siguió la metodología expuesta en [16].

2.1.1.1. Criterios de selección

Se establecieron los siguientes criterios de inclusión: publicaciones sobre la ruptura del RSA hasta 2007, publicaciones de trabajos actuales (entre 2009 y 2015), en inglés o en español, publicados en revistas científicas. Así mismo, las metodologías usadas para el criptoanálisis y estudios de ataques actuales sobre el algoritmo RSA. Se excluyeron aquellos artículos que no se concentraban en un solo algoritmo de cifrado y que emplearan una modificación matemática brusca sobre el algoritmo RSA.

2.1.1.2. Síntesis de datos

Cada estudio se clasificó de acuerdo al tipo de factorización o método de ataque usado, el tipo de tecnología empleada, el tiempo de criptoanálisis y el tamaño de RSA utilizado. Para ello, se creó una tabla de clasificación de la información en la cual se caracterizó y analizó la información, calificando los métodos de factorización y ataque más efectivos para desarrollar este proyecto de grado, siendo 1 y 2 los más eficientes y 4 y 5 los menos eficientes para factorizar números primos grandes.

N°	Nombre del artículo	Año publicación	Método de criptoanálisis			Otros			Tecnología	Tiempo criptoanálisis	Tamaño del mensaje	Calificación
			Factoriza	Pasivo	Activo	Histo	Matem	Métric				
1	Information Security and Cryptography: Principles and Applications.	2001	_	_	_	_	_	x	análisis cualitativo y cuantitativo	_	_	2
2	Criptosistemas clásicos	2005	_	_	_	x	_	_		_	_	3
3	A method for obtaining digital signatures and public-key cryptosystems	1978	_	_	_	x	x	_	_	_	_	1
4	New Directions in Cryptography	1976	_	_	_	x	_	_	_	_	_	3
5	A Tale of Two Sieves	1996	x	_	_	x	x	x	Computacional	Mayor a 5 meses	Mayor a 47 dígitos	1
6	Investigación, diseño y ejecución	2011	_	_	_	_	_	x	_	_	_	3
7	The Quadratic Sieve Factoring Algorithm	1988	x	_	_	x	x	x	Computacional	Mayor a 5 meses	129 dígitos	2
8	A pipeline architecture for factoring large integers with the quadratic Sieve algorithm	1988	x	_	_	x	x	x	Computacional	1 año	144 dígitos	1

9	Integer Factorization Algorithms	2004	x	_	_	x	x	x	Manual/ computacional	_	(21 :100) dígitos	3
10	*Cryptanalytic Attacks on RSA*	2008	x	_	x	x	x	x	Computacional	varia	200 dígitos	1
11	*Encyclopedia of Cryptography and Security*	2011	_	_	_	x	_	_	_	_	_	4
12	*Information Security Applications: 10th International Workshop, WISA 2009, Busan, Korea, August 25-27, 2009, Revised Selected Papers*	2010	_	_	_	x	_	_	_	_	_	4
13	Implementacion de la Criba Cuadratica (Quadratic Sieve) en C++	–	x	_	_	_	x	_	Computacional	_	_	3
14	Seguridad informática y Criptografía	2004	x	_	_	x	x	_	Computacional	_	(30:200) dígitos	2
15	*Cryptanalysis of ISO/IEC 9796-1*	2009	_	_	_	x	_	_	_	_	_	4

16	On the Security of 1024-bit RSA and 160-bit Elliptic Curve Cryptography	2009	x	_	x	x	x	_	Computacional	_	(80:112) bits	2
17	Energy Consumption Analysis of the Cryptographic Key Generation Process of RSA and ECC Algorithms in Embedded Systems	2014	_	x	x	x	_	_	Computacional/ eléctrica	_	_	4
18	Evaluation of Simple/Comparative Power Analysis against an RSA ASIC implementation	2009	_	x	_	x	_	_	Computacional/ eléctrica	_	_	2
19	An optimized cross correlation power attack of message blinding exponentiation algorithms	2015	_	x	_	_	x	_	Computacional/ eléctrica	_	_	3

20	Chosen-message SPA attacks against FPGA-based RSA hardware implementations	2008	_	x	_	_	x	x	Computacional/ eléctrica (FPGA)	_	_	2
21	Comparative Power Analysis of Modular Exponentiation Algorithms	2010	_	x	_	_	x	x	Computacional/ eléctrica (FPGA)	_	_	2
22	A Practical Fault Attack on Square and Multiply	2008	_	x	x	_	x	x	Computacional y canal	_	_	2
23	Power analysis attack: A vulnerability to smart card security	2015	_	x	x	_	_	_	Computacional/ Tarjetas inteligentes de seguridad	_	128 bits	3
24	Analysis of Attack on RSA through Formal Verification	2011	_	_	x	_	x	_	Computacional	_	_	2
25	Practical Power Analysis Attacks to RSA on a Large IP Portfolio SoC	2009	_	x	_	_	_	_	Computacional/ eléctrica	_	512 bits	3

26	Enhanced power analysis attack using chosen message against RSA hardware implementations	2008	_	x	x	_	_	_	Computacional/ eléctrica	_	_	3
27	Performance and side-channel attack analysis of a self synchronous montgomery multiplier processing element for RSA in 40nm CMOS	2012	_	x	_	_	_	_	Computacional y canal	_	_	4
28	New Directions in Cryptography	1976	_	_	_	x	_	_	_	_	_	4
29	RSA Precomputation Reveals the Secret Exponent.	2007	_	x	x	_	x	_	Computacional/ eléctrico	_	_	3
30	Fault and simple power attack resistant RSA using Montgomery modular multiplication	2010	_	x	_	_	x	_	Computacional/ eléctrica (FPGA)	_	32 bits	2

31	RSA key extraction via low-bandwidth acoustic cryptanalysis	2013	_	x	x	_	_	_	Computacional/ eléctrico	_	4060 bits	2
32												
33	Identity-based ring signatures from RSA	2007	_	_	_	x	x	_	_	_	10^4 bits	3
34	Comparative Power Analysis of Modular Exponentiation Algorithms	2010	_	x	_	_	_	_	Computacional/ eléctrico	_	_	1
35	Fonaments de matemàtica discreta	1999	_	_	_	x	x	_	_	_	_	2
36												
37												
38												
39												
40												
41												
42												
43												
44												
45												
46												
47												

2.1.2. Resultados

Se obtuvieron inicialmente 64 estudios como resultado de las búsquedas de las palabras clave en las bases de datos y páginas con contenido relacionado. De estos 64 estudios, se excluyeron 12 debido a que incurrían en campos de estudio que no se enfocaban específicamente en el algoritmo RSA o en su criptoanalisis; otros estudios no revelaban aportes innovadores contundentes para los objetivos de este trabajo por el contrario describían generalidades de varios algoritmos de seguridad. Finalmente se obtuvieron 52 artículos como base de conocimiento, investigación y análisis.

2.2. Orígenes del algoritmo RSA

Hasta los años 70's el problema de cifrado y firma digital no existía propiamente en el mundo de la informática, hasta que en 1976 Diffie y Hellman introducen el concepto de cifrado de clave simétrica diseñado un algoritmo de seguridad cuya complejidad matemática se basara en el problema del logaritmo discreto. Este algoritmo no permitía firmar digitalmente los mensajes ni cifrar totalmente [17] [18].

En el año de 1978 Ronald Linn Rivest, Adi Shamir y Leonard Adleman desarrollan el algoritmo RSA, algoritmo que lleva las iniciales de sus apellidos. Este algoritmo basa su dificultad computacional en la factorización de números compuestos muy grandes producto de la multiplicación de dos números primos, bajo la hipótesis: "En los últimos trescientos años muchos matemáticos famosos han trabajado en un método eficiente para factorizar números primos, y en la actualidad la factorización de números primos se basan en el método de Legendre, método con el cual aún no es posible en un tiempo razonable factorizar un número primo de 200 dígitos". De esta forma se pretendía elevar la dificultad algorítmica de factorización de las claves de este algoritmo, usando lo que se conoce como problema de factorización entera no polinomial, es decir que en un sentido el problema matemático es fácil y rápido de realizar, mientras que en el sentido de contrario de la operación es un problema difícil de tratar matemáticamente. Por otra parte la capacidad computacional de la época no daba abasto para factorizar en un tiempo próximo dichos factores dado que se trataba de valores de miles de bits [19]

Se presume que para del año de 1969 a 1973 en Gran Bretaña la GCHQ (Government Communications Headquarters) desarrolló una idea similar a la de los creadores del RSA dirigida por el matemático Clifford Cocks, para distribuir claves a través de una cifra no simétrica llegando a la misma conclusión del

diseño del RSA. Infortunadamente dicho estudio no fue patentado ni publicado ya que fue considerado como información secreta del gobierno Británico. En la actualidad una situación no muy distante de la realidad de los estudios más avanzados del RSA.
En 1983 el MIT (Instituto Tecnológico de Massachusetts) patentó este algoritmo con el número 4.405.829. Esta patente expiró el 21 de septiembre de 2000. Como el algoritmo fue publicado antes de patentar la aplicación, esto impidió que se pudiera patentar en otros lugares del mundo.

El algoritmo

El algoritmo básicamente consta de 4 pasos[20]:

1. Generación de claves:
    - Cada usuario elige un par de números primos distintos *p y q*, para calcular

      n = pq.

    - Los valores p y q no se hacen públicos.
    - Cada usuario calcula $\phi(n) = (p-1)(q-1)$.
    - Cada usuario elige una clave pública e de forma que $1 < e < \phi(n)$ y que cumpla con la condición: $mcd\ [e, \phi(n)] = 1$.
    - Cada usuario calcula la clave privada $d = inv\ [e, \phi(n)]$.
    - Se hace público el grupo n y la clave e.
    - Se guarda en secreto la clave d. También guardará p y q puesto que en la operación de descifrado usará el Teorema del Resto Chino
2. Ciframiento: $C = K^e\ mod\ (nR)$
3. Desciframiento : $K = C^d\ mod\ (nE)$
4. Firma: $C = h(M)^{dE} mod(nR)$

Dónde:

K: es la clave con la que se altera el mensaje
p,q: son dos números primos diferentes
n: es el resultado de la multiplicación de los dos primos
(n,e): son la clave pública de la comunicación

(n,d): son la clave privada de la comunicación
ϕ(n): Función multiplicativa de Euler.

Finalmente en 1978 los laboratorios RSA lanzaron un reto a nivel mundial[15], desafiando a los programadores, criptógrafos, matemáticos y aficionados a descifrar un mensaje numérico de 129 cifras decimales, publicado con su exponente público y módulo. El problema fue bautizado como RSA-129[10], el cual tomó alrededor de 16 años en ser resuelto. En la ilustración 1 se presentan los números de RSA ya factorizados y aquellos que aún faltan.

Reto Critográfico lanzado por los Laboratorios RSA			
		Ya factorizados	Sin factorizar
Números de RSA	Cifras decimales	100, 110, 120, 130, 140, 150, 160, 170, 174, 180,190, 193, 200, 210, 213, 232	240, 250, 260, 270, 280, 290, 300, 309, 310, 320, 330, 340, 350, 360, 370, 380, 390, 400, 410, 420, 430, 440, 450, 460, 470, 480, 490, 500, 620
	Bits	330, 364, 397, 426, 430, 463, 496, 512, 530, 563, 574, 627, 636, 660, 693, 702, 768	795, 829, 862, 895, 896, 928, 962, 995, 1024, 1028,1419, 1485, 1551, 1584, 1617, 1650, 2048

Ilustración 1: Retos RSA con tamaño en dígitos decimales. Fuente: Autor.

En la actualidad el algoritmo RSA es parte de la ISO 9796-1[21]. También muchas empresas reconocidas como: Apple, Microsoft, Google, Oracle, Facebook, entre otros lo emplean como medio de protección de la información.

A continuación se desea hacer una pequeña reseña histórica de los que fueron los retos RSA, mostrando las principales características de cómo se logró resolver en su momento estos retos y el tipo de metodología empleada.

2.3. Retos RSA

RSA 396 bits– 120 dígitos[9]**:**

El 9 de julio de 1993 Thomas Denny, Bruce Dodson, Arjen Lenstra, Walter Lioen, Mark Manasse rompieron el RSA- 120, en colaboración con las universidades de: Saarbruecken, Lehigh, CWI Amsterdam, el Sistemas DEC Centro de Investigación y Bellcore. Usando el método *quadratic sieve* [1] o criba cuadrática en español.

[1] Es un algoritmo de factorización de enteros y, en la práctica, el segundo método más rápido conocido. Es todavía el más rápido para enteros que tienen 100 o menos dígitos decimales, y es considerado mucho más

El tamaño del RSA era de 120 dígitos decimales, lo cual indica que en base binaria se requieren ($2^{3,3}$) 3,3 bits para representar un dígito decimal teniendo en cuenta la codificación ASCII de caracteres imprimibles. Estos números fueron:

n=227010481295437363334259960947493668895875336466084780038173258247009162675779735389791151574049166747880487470296548479

P=32741455569349801575114630374914148806364240324017146340688 3
q=693342667110830181197325401899700641361965863127336680673013

La propuesta de este artículo fue comparar el rendimiento de la factorización comparando los métodos de factorización Sieve. En principio el análisis se hizo empleando el QS (Criba cuadrática) y posteriormente el NFS (criba de cuerpos numéricos). La idea principal del proyecto era realizar una predicción de tiempos de ambos métodos basados en la extrapolación de los mismos, con el fin de encontrar una alternativa para reducir notablemente los tiempos de cómputo a futuro.

La factorización de los dos números primos tomó 825 MIPS[2]. (Millones de instrucciones por segundo). El hardware utilizado para este proyecto fueron máquinas de las universidades mencionadas anteriormente, por tanto las tres cuartas partes del Sieve (búsqueda de relaciones para factorizar n) se realizaron en estaciones de trabajo de las 3 primeras universidades mencionadas anteriormente, mientras que en el centro de computadores de MasPar[3] de Bellcore, realizó la combinación de estas relaciones.

El centro de computadores MasPar produjo un promedio de 480 relaciones de posibles factorizaciones completas de por día utilizando el método QS, es decir 480 posibles números que al multiplicarse daban el valor de n entre las cuales se debían encontrar p y q.

Con esta investigación se concluyó que para los números menores a 100 dígitos era más eficiente el método QS, sin embargo este deja de ser eficiente para números muy grandes, para los cuales es más útil el método NFS.

---

sencillo que la criba de cuerpos numéricos. Es un algoritmo de factorización de propósito general, lo que significa que su tiempo de ejecución únicamente depende el tamaño del entero a ser factorizado.

[2] MIPS o Millones de instrucciones por segundo. Es una forma de medir la potencia de los microprocesadores. Sin embargo, esta medida solo es útil para comparar procesadores con el mismo conjunto de instrucciones y usando técnica para medir el rendimiento de un sistema o componente del mismo en este caso el compilador del equipo.

[3] MasPar es un computador basado en una colección de ALU 's. El énfasis está en la eficiencia de las comunicaciones, y de baja latencia. La arquitectura MasPar está diseñada para escalar, y balancear procesamiento, equilibrar la memoria, y mejorar la comunicación.

A continuación se presenta una tabla comparativa de las relaciones de factorización encontradas por los investigadores, donde se muestra el progreso del número total de polinomios generados durante las pruebas con los métodos NFS con una combinación de tamaño 10^8 o 9 dígitos en cada relación. Lo cual demostró que al usar primos grandes mayores a 10^8 se lograba reducir 2 semanas de tiempo de ejecución usando el método NFS.

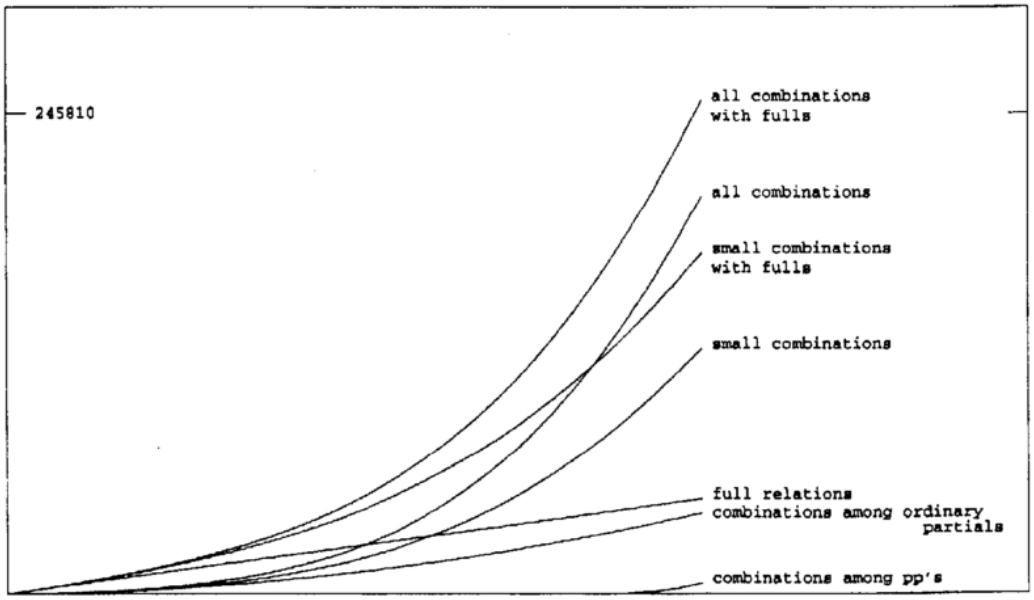

Ilustración 2: Progreso del método Sieve basado en el número de relaciones matemáticas o factores encontrados para encontrar un p y un q[4][9]**.**

En la ilustración se puede apreciar el comportamiento del NFS al combinar números que componen las relaciones matemáticas para factorizar n. Qué pasa si se combinan enteros solo enteros de tamaño grande, enteros de tamaño pequeño o todos los tamaños.

RSA 425 bits- 129 dígitos[10]**:**

En agosto de 1977 se publicó un mensaje cifrado con RSA-129, el cual desafiaba a las personas a ganar un premio de USD 100 por descifrar este mensaje. Pero solo fue hasta 1994 que el premio fue cobrado y donado a "Free Software Foundation". Este proyecto de RSA recibe un nombre muy especial. "***The Magic Words are Squeamish Ossifrage"*** fue el mensaje que por más de diecisiete años permaneció cifrado y que fue revelado al

[4] Número de relaciones necesarias para factorizar un número primo grande que implica tener el cuenta los dígitos del número y todas las posibles combinaciones de las soluciones, usando números pequeños y grandes, grandes con grandes y pequeños con pequeños, demostrando que el NFS es eficiente para todas las combinaciones de tamaños de números para lograr la factorización, mientras que el QS es útil para números pequeños.

mundo, gracias a la colaboración de 600 personas que aportaron tiempo de cómputo de unas 1600 máquinas (2 eran un fax), durante aproximadamente seis meses y usando por primera vez la internet para comunicarse, esto dio como resultado la ruptura del RSA-129 y la finalización del reto. Como método de factorizar usaron el quadratic sieve factoring (QS) y entre las 1600 máquinas contaron con 16K MasPar MP-1, la cual tardó 45 horas en resolver la matriz, para finalmente entregar el resultado del primer factor que consta de 64 dígitos y el otro factor de 65 dígitos el 2 de abril de 1994 a las 18:15. Finalmente estiman que con 5000 procesadores cada uno a una tasa de 100 MIPS e implementado el NFS, se podría lograr factorizar el RSA 512 bits.

n=114381625757888867669235779976146612010218296721242362562561842935706935245733897830597123563958705058989075147599290026879543541

p=3490529510847650949147849619903898133417764638493387843990820577
q=3276913299326670954996198819083446141317764296799294253979828853

RSA 576 bits – 174 dígitos [11]**:**

El 5 de diciembre de 2003 el equipo de la universidad de Bonn (Bundesamt für Sicherheit in der Informationstechnik) conformado por J. Franke y T. Kleinjung, anuncinciaron la factorización del RSA de 174 dígitos.Cuyo tamaño de mensaje era:

n= 1881 9881292060 7963838697 2394616504 3980716356 3379417382 7007633564 2298885971 5234665485 3190606065 0474304531 7388011303 3967161996 9232120573 4031879550 6569962213 0516875930 7650257059

p= 3980750 8642406493 7397125500 5503864911 9906436234 2526708406 3851895759 4638895726 1768583317

q=4727721 4610743530 2536223071 9730482246 3291469530 2097116459 8521711305 2071125636 3590397527

El método empleado para este proyecto fue el GNFS (Cibra de cuerpo numérico generico) utilizando hardware del Instituto de Computación Científica y Matemática Pura de la Universidad de Bonn, del Instituto Max Planck de Matemáticas en Bonn, y del Instituto de Matemática Experimental de Essen.

Pocos datos se conocen sobre este proyecto, por tanto es un poco difícil establecer detalles del mismo.

RSA 640 bits -193 dígitos [22]**:**

El 2 de noviembre de 2005 nuevamente J. Franke, T. Kleinjung y F. Bahr, M. Boehm, anuncian el haber decifrado el RSA de 193 digitos. Dónde:

n=310 7418240490 0437213507 5003588856 7930037346 0228427275 4572016194 8823206440 5180815045 5634682967 1723286782 4379162728 3803341547 1073108501 9195485290 0733772482 2783525742 3864540146 9173660247 7652346609

p=1634733645809253848443133883865090859841783670033092312181 11085238933310010450815121211816751157 9

q=1900871281664822113126851573935413975471896789968515493666 6385390880271038021044989571912614655 71

Esta vez el reto propuesto por los investigadores fue vulnerar el RSA al menos en 100 días, para ello nuevamente emplearon el GNFS acotando especial entre números primos entre: 28^7 *o* 11 *dígitos decimales y* 77^7 *o* 14 *dígitos decimales*. Produciendo así 166^7 relaciones de factorización en total compatibles para hallar p y q, procesados en 80 CPUs Opteron de 2.2 GHz, lo cual tomó al proyecto meses. Para realizar el proceso matricial del GNFS se utilizó un cluster de 80 computadores 2,2 GHz conectados a través de una red Gigabit, tomando cerca de 1.5 meses en ser reducida y procesada. Arrojando 36^6 resultados posibles de números primos.
Dando un total de 5 meses de procesamiento para obtener los dos números primos.

RSA-768 bits [12] **:**

El 12 de diciembre de 2009 Kleinjung, Thorsten, Aoki, Kazumaro, Franke, Jens, Lenstra, Arjen K, Thomé, Emmanuel, Bos, Joppe W, Gaudry, Pierrick, Kruppa, Alexander, Montgomery, Peter L, Osvik, Dag Arne, Te Riele, Herman, Timofeev, Andrey y Zimmermann, Paul, reportaron la factorización del RSA de 232 dígitos usando el NFS, en su artículo exponen una posible predicción para romper el RSA 1024, que hasta el momento no registra publicación. En este caso:

n=123018668453011775513049495838496272077285356959533479219732245215172640050726365751874520219978646938995647494277406384592519255732630345373154826850791702612214291346167042921431160222124047927473779408066535141959745985690214341 3.

p=33478071698956898786044169848212690817704794983713768568912431388982883793878002287614711652531743087737814467999489
q=36746043666799590428244633799627952632279158164343087642676032283815739666511279233373417143396810270092798736308917

Para ello se utilizó un AMD64 de 2.2GHz. Esta investigación basó su investigación nuevamente al rededor del metodo Sieve. La investigación se enfocó en estudiar el comportamiento del Sieve al generar vectores de posibles números primos para generar relaciones. En la ilustración 4 se condensa la información pertinente a las máquinas usadas así como también el tiempo que tomo cada computador o cluster en país anfitrión .

Cluster location	number of nodes	CPU type	clock speed (GHz)	cores per node	GB RAM per node	interconnect	nodes per job	cores per job	seconds per iteration		communication
									stage 1	stage 3	
Lausanne	56	2×AMD 2427	2.2	12	16	ib20g	12	144	4.3-4.5	4.8	40%
Tokyo	110	2×Pentium 4	3.0	2	5	eth1g	110	220	5.8†, 6.4	7.8	33%†, 44%
Grenoble	34	2×Xeon E5420	2.5	8	8	ib20g	24	144	3.7	n/a	30%
Lille	46	2×Xeon E5440	2.8	8	8	mx10g	36	144	3.1	3.3	31%
							32	256	3.8	n/a	38%
							24	144	4.4	n/a	33%
Nancy	92	2×Xeon L5420	2.5	8	16	ib20g	64	256	2.2	2.4	41%
							36	144	3.0	3.2	31%
							24	144	3.5	4.2	30%
							18	144	n/a	5.0	31%
							16	64	n/a	6.5	19%
Orsay	120	2×AMD 250	2.4	2	2	mx10g	98	196	2.8	3.9	32%
Rennes	96	2×Xeon 5148	2.3	4	4	mx10g	64	256	2.5	2.7	37%
							49	196	2.9	3.5	33%
Rennes	64	2×Xeon L5420	2.5	8	32	eth1g	49	196	6.2	n/a	67%
							24	144	8.4	n/a	67%
							18	144	10.0	n/a	68%
							8	64	n/a	18.0	56%

Ilustración 3: RSA-768, información de configuración de clusters[12]

En este artículo se aclara el desarrollo de cada etapa del GNFS en 3 pasos fundamentales el primero la generación de polinomios, la combinación de relaciones y el álgebra lineal y la eliminación gaussiana por método matricial, a continuación se realiza un breve resumen por etapa:

Selección Polinomial y aplicación del NFS:

Se produjeron tres pares polinomiales utilizando el método Montgomery-Murphy como una mejora al método de Thorsten Kleinjung. El proceso de generación de polinomios se realizó en las universidades de: BSI (Bundesamt für Sicherheit in der Informationstechnik, Bonn) y en la EPFL (Escuela Politécnica Federal de Lausana,Lausana).
En 2007 se empezó a aplicar el método Sieve (búsqueda de posibles relaciones a partir de los polinomios encontrados) hasta el año 2009 en el que finalizo el proceso. El ambiente utilizado para emplear el Sieve fue el siguiente: Se utilizaron varios PCs en BSI, CWI (Centrum Wiskunde & Informatica, Amsterdam,), EPFL, INRIA (Institut National de Recherche en Informatique et en Automatique, Francia), NTT (Nippon Telegraph & Telephone, Japón), la Universidad de Bonn, EGEE (Enabling Grids for E-Science) (El Centro Australiano, AC3 para Computación Avanzada y Comunicaciones), y para PC en el Reino Unido.
El tiempo total de operación se redujo en 1500 MIPS años, usando un AMD64. Para el procesamiento se destinó de la siguiente forma la memoria de las CPUs

- 1GB: 450M para procesos algebraicos, 100M para procesos racionales (raíces de factorización)
- 2GB: 1100M para procesos algebraicos,, 200M para procesos racionales para una q por debajo 450M con el fin de acotar el proceso algebraico

Los parámetros de rendimiento según cada estación fueron los siguientes:
64 334 489 730 relaciones (38% INRIA, el 30% de la EPFL, el 15% de NTT, 8% Bonn, 3,5% CWI, 5.5% otros).

Resolución de la matriz

Para esto se empleó el algoritmo Montgomery, se utilizó un núcleo en cada uno de 12 dual AMD64 hex-core para preparar los datos de 8 posibles soluciones y un dual hex-core AMD64 para la matriz de reducción.

El 12 de diciembre de 2009, se logró encontrar los factores en la primera solución y unos minutos más tarde cuatro de las siete estaciones de trabajo produjeron la factorización también.

2.4. METODOS DE ATAQUE ACTIVOS Y PASIVOS:

Poco antes del cierre definitivo del concurso de factorización RSA, muchos investigadores enfocaron sus esfuerzos en hallar formas alternativas de obtener los números primos *p y q* de forma alternativa a la factorización. Con base en ello surgen nuevos métodos, básicamente se clasifican en métodos activos y pasivos, los cuales han sido probados hasta ahora en tamaños de RSA no muy grandes.

Los métodos pasivos son aquellos ataques dirigidos específicamente al hardware del dispositivo electrónico o al canal de transmisión (SCA: “side cannel attacks”), permitiendo analizar el consumo energético ejercido por las compuertas lógicas al procesar una serie de datos e instrucciones. En este caso al ejecutar la generación de llaves aleatorias para cifrar el mensaje RSA o el proceso de cifrado y descifrado del mensaje. Los métodos pasivos permiten realizar un análisis de señales eléctricas, emanaciones electromagnéticas o de radiofrecuencia. Este tipo de análisis da paso a la obtención de los primos que componen el tamaño del mensaje y el tipo de RSA empleado. Los ataques pasivos se clasifican en: DPA (difference power analysis) y SPA simple (power analysis) [6]

Mientras que los métodos activos dependen específicamente de la manipulación software del sistema o del algoritmo de seguridad. Analiza el peso haming de los datos permitiendo identificar el número de ceros y unos en una multiplicación modular, modifica o perturba el algoritmo de procesamiento para obtener comportamientos anormales y/o erróneos en los resultados del proceso de cifrado y descifrado que pueden ser explotados para recuperar información cifrada. Este tipo de ataques se conocen principalmente como “*Fault Attacks*” (FA) y fueron diseñados para contrarrestar los ataques pasivos.[23]

En la actualidad existen combinaciones entre estos métodos, ataques pasivos y activos que en conjunto buscan engañar al sistema haciéndoles creer que están siendo atacados por uno solo mientras el otro recupera la información de las llaves de la comunicación. Este tipo de ataque se conoce como PACA, el cual introduce fallos para perturbar los procesos que se están llevando a cabo con el fin de que el RSA detecte estos fallos al final de ejecutar los comandos. Para entonces las llaves se habrán recuperado con un análisis de potencia y el sistema no lo habrá notado. Este tipo de ataques son diseñados para IPSec, SSL, Smart Cards y procesos de autenticación pequeños.

Por supuesto conforme se han encontrado fallos y fugas de información se han diseñado contramedidas adecuadas que desvían el robo de la información, usando generadores de ruido, relojes de jitter o filtraciones de potencia, que retarden y des sincronicen la señal eléctrica. Del mismo modo para el código malicioso se han desarrollado un relleno aleatorio de bits dentro del tamaño del mensaje y robustecido las firmas digitales.

2.4.1. Ataques Pasivos:

Los ataques pasivos se proponen por primera vez en el año de 1995 por Paul Kocher [24] como resultado de su investigación de filtraciones en hardware para explotar información por medio de fugas en dispositivos durante la ejecución de procesos de criptoanálisis. En 1999 Kocher desarrolla investigaciones enfocadas al estudio de nuevos tipos de ataques relacionados a inyecciones en el hardware, emanaciones de campo eléctrico, fugas de potencia entre otras, que con el tiempo fueron modificadas y mejoradas por otros investigadores. Como resultado en 2004 se da el desarrollo del análisis de correlación de potencias, basado en algoritmos simétricos como: DES, AES y ECC. De igual forma en conjunto con el desarrollo de este tipo de ataques otros autores buscaron documentar ataques contra los exponentes modulares, muchos de ellos implementando el método de multiplicación de Montgomery y sus variaciones para determinar diferentes combinaciones para mejorar la agilidad de procesamiento entre multiplicaciones y raíces en el criptoanálisis [25].

En [26] se explica cómo se emplea una plataforma compuesta por chips, compuertas y un osciloscopio para el criptoanálisis de un IP SoC, utilizando análisis de potencias para un RSA de tamaño 512 bits. La plataforma se basaba en el número de muestras requeridas por el osciloscopio para procesar bit a bit “n” y revelar de 8 a 4 bits de la llave privada RSA. Para ello los investigadores se apoyan en el método Montgomery y en un análisis de correlación de potencia y un análisis diferencial de potencias, de tal forma que al analizar las curvas de

potencia del osciloscopio se pueden observar picos de potencia que al ser parte de un patrón de picos durante el criptoanálisis revelan el información sobre la llave privada.

Por otra parte el análisis de potencia se ha enfocado no solo a Smart Cards[27] sino también a FPGAs como se muestra en: [28] [29][30].

En la actualidad existen 3 tipos de ataques de potencia [26]:

1. **SPA (Simple Power Analysis):** consiste en el análisis de gráficas o trazas de potencia hasta encontrar un comportamiento o patrón entre cada traza, con el fin de hallar posibles valores.

2. **DPA (Differential power analysis):** consiste en estimar un resultado a partir de un resultado de gráficos de potencia seleccionados de tal forma que sean homogéneas y diferentes entre sí. El proceso de estimación consiste dividir los gráficos de potencia en subconjuntos, posteriormente calcular una gráfica promedio diferencial por cada subconjunto. Finalmente los valores que se buscan a través de este tipo de análisis se encuentran en los picos de las gráficas promedio.

3. **CPA (Correlational power analysis):** corresponde al cálculo de la correlación de Pearson entre dos gráficos de potencia y los valores intermedios de la suposición de los posibles valores a encontrar. Los resultados se condensan en una matriz de CPA, con el fin de realizar un análisis gráfico del comportamiento de las gráficas, graficando los valores supuestos en X y los valores de la correlación de Pearson en Y.

2.4.2. Ataques de Tiempo

También existen otro tipo de combinaciones posibles para los métodos pasivos y activos, que buscan no solo explotar las posibilidades que ofrecen estos, sino que junto con otros algoritmos que agilizan el tiempo de máquina, se pretende innovar en el criptoanálisis del RSA. Tales métodos como el de Montgomery (MM) o el teorema del resto chino (CRT) imprimen al proceso mayor velocidad haciéndolo cuatro veces más eficiente a la hora de factorizar y reduciendo el tiempo por cada división o multiplicación modular [31].

Por otra parte el diseño de nuevos algoritmos de ataque y juegos matemáticos para entender mejor el RSA buscan variar parámetros matemáticos del algoritmo mismo para agilizar su criptoanálisis. En [18] se puede apreciar como los autores proponen un nuevo método de obtención de las llaves de la comunicación mediante la variación de los números “e” y “d” que componen las llaves públicas y

privadas, afirmando que a menor valor de “e” y “d” se tendrá una disminución en el número de iteraciones por ciclo, una generación más rápido de las llaves de la comunicación, permitiendo cifrar y descifrar en un tiempo más corto. Sin embargo aunque es novedoso este algoritmo, requiere de ciertas condiciones predeterminadas para su funcionamiento.

2.5. REVISIÓN DE METODOS DE FACTORIZACIÓN

En 1978 calcular la factorización de un tamaño de RSA de 20 dígitos era todo un problema matemático difícil de resolver en corto tiempo, pero en 1981 Carl Pomerance crea el algoritmo de factorización que ayudaría a resolver este problema, el Quadratic Sieve (QS), que hasta la época doblaba los posibles tamaños de RSA factorizables, siendo en 1993 el único algoritmo capaz de factorizar el famosos RSA 129 dígitos, para el cual en el año de 1976 la revista Scientific American estimo un tiempo de factorización cercano a los 40 cuadrillones de años.

En 1996 el QS evoluciona al pollard number field sieve (NFS) publicado en la resvista Springer con la factorización de un RSA de tamaño 130 dígitos decimales. A los esfuerzos de Pomerance se une Kraitchik y Dixon, quienes proponen nuevas variaciones al conocido QS para agilizar la factorización, analizando en principio las descomposiciones de números en factores, posteriormente en polinomios, seguido por vectores y finalmente observando posibles relaciones que permitían encontrar en un menor tiempo p y q [32].

Con base en lo anterior Kraitchik propone una forma simple de explicar lo anterior y de abordar el problema de la factorización de números grandes escribiéndolos como raíces diferenciales de cuadrados, por ejemplo:

$$\text{n=8051= } 8100 - 49 = 90^2 - 7^2$$

Para conseguir los dos números primos que componen un n. Por lo tanto n=ab. Otra forma de verlo matemáticamente es como:

$$n = ab = \left(\frac{(a+b)}{2}\right)^2 - \left(\frac{(a-b)}{2}\right)^2$$

En principio Pomerance describe al QS básicamente como un método capaz de encontrar los factores que componen a n, un “x” y “y” con las siguientes condiciones matemáticas [8]:

$$x \neq \pm y \, mod(n)$$

$$x^2 \equiv y^2 \, mod(n)$$

Lo que implica que: $(x - y)(x + y) \equiv 0 \, mod(n)$

Para ello define una función: $Q(x) = (x + \sqrt{n})^2 - n = X^2 - n$

Pomerance ve la necesidad de encontrar subconjunto de Q(x)=$Q(x_{i1})$ $Q(x_{i2}) \dots Q(x_{in})$ tal que este subconjunto sea una raíz de y^2 por lo tanto para cada x se tendría:

$$Q(x) \equiv X^2 \, mod(n)$$

$$Q(x_{i1})Q(x_{i2}) \dots Q(x_{in}) \equiv (x_{i1}x_{i2} \dots x_{in})^2 \, mod(n)$$

Finalmente si esta condición se cumple se encontrarán los factores de n.

A partir de esto Kraitchik propone encontrar secuencias de posibles valores para p y q. Para ellos matemáticamente toma la propuesta de Pomerance y la describe de la siguiente forma:

$$n = u^2 - v^2$$

$$u^2 \equiv v^2$$

$$u^2 \equiv v^2 mod \, (n)$$

Primero toma la raíz por encima de n, luego toma un rango o cotas por debajo de n, $x^2 - n$ y lo transforma en una función $Q(x) = x^2 - n = v^2$ por tanto Q(x) será igual a u^2 por lo tanto:

$$u^2 = x^2 - n = v^2$$

$$u^2 = x_1^2 \dots \dots . x_k^2$$

$$v^2 = (x_1^2 - n) \dots (x_k^2 - n)$$

$$u^2 = x_1^2 \dots \dots . x_k^2 \equiv (x_1^2 - n) \dots (x_k^2 - n)$$

$$= Q(x_1) . . Q(x_k) = v^2 mod \, (n)$$

Donde cada Q(x) es fácilmente factorizable y por tanto cada valor que tome Q(x) hasta llegar a Q(x_k) será una raíz de n. A esto Kraitchik lo llamó secuencias de Kraitchik. Para verificar que el producto de Q(x) es una raíz es necesario que el conjunto de $x_1^2 \ldots\ldots x_k^2$ lo sean, por tanto es necesario factorizar cada $Q(x_i)$ tomando una base de factorización de números primos pequeños para ello el valor de x debe estar alrededor de cero para ellos se estimará un intervalo [-M, M] en el cual x tomará valores o alternativamente el intervalo tomará la forma $[\sqrt{n} - M, \sqrt{n} + M]$ basad en la expresión $Q(x) = x^2 - n$.

Posteriormente basados en esta secuencia Jhon Brillhart y Michael Morrison proponen una sistematización estratégica para hallar subsecuencias a partir de la secuencia de Kraitchik usando algebra lineal. Todo entero m tiene un vector positivo v(m) en un espacio lineal perteneciente a cada valor de la secuencia anterior.

La información arrojada por cada vector generado a partir de cada Q(x_k) se reduce 1 y 0 con el operador modulo con el fin de encontrar rápidamente las raíces. Al final la suma de todos los vectores v(m) dará cero. Dado que el producto de Kraitchik son productos de raíces.

Para sistematizar el proceso anterior se sugirió seleccionar un límite B dependiente del tamaño de n, introducirlo a la secuencia y observar su comportamiento, generar B+1 vectores a partir de un espacio de B dimensiones, reducirlos y observar si hay dependencia entre cada vector, finalmente descarta aquellos factores en los que “n” no es congruente como raíz. Cada vector se pondrá en una matriz A, en donde las filas representarán todos los Q(x_k) y las columnas serán todos los exponentes $x_1^2 \ldots\ldots x_k^2$ de los primos. El tamaño de B no debe ser muy pequeño porque la posibilidad de encontrar números en la secuencia antes de los primeros B primos sería muy pequeña casi minúscula lo cual bajaría la probabilidad de encontrar al menos un número primo. Gracias a esto poco después se introduciría el concepto de número liso o “smooth number” para denotar a B. Por tanto se puede definir un número liso como: un entero capaz de ser factorizado completamente en números primos pequeños y un número se llamará B-liso si alguno de sus factores primos es mayor que B, un ejemplo sería el siguiente:

$$1620 = 2^2 \times 3^4 \times 5$$

En este ejemplo 5 sería el número liso de 1620 dado que ninguno de sus factores primos es mayor que 5.

Es aquí donde surge la pregunta de qué tiempo podrá tomar calcular p y q utilizando lo anterior junto con la división por tentativa, se llega a la conclusión de

que se requieren números auxiliares que permitan factorizar “n” junto con los números lisos. Pero la probabilidad de encontrar un número B-liso no es sencillo por tanto en 1981 se implementa la criba de Eratosthenes para reconocer rápidamente los valores lisos en la secuencia de polinomios de Kraichik.Este procedimiento se encarga de encontrar todos los primos en un intervalo inicial de números naturales “n”, tachando inicialmente el primer número primo 2 y todos sus múltiplos, posteriormente se devuelve al siguiente número en la secuencia que no haya sido tachado, será 3 y procederá nuevamente a tachar sus múltiplos, así sucesivamente hasta que al final quedarán sin tachar todos los números primos dentro del intervalo “n”. Un ejemplo claro de la criba de Eratosthenes se muestra en a continuación en la ilustración 2 donde se han marcado de diferentes colores todos los números que no son primos del 1 al 120. Se puede observar claramente que hay números que han sido tachados varias veces, pero que a su vez se pueden descomponer en varios factores primos pequeños también.

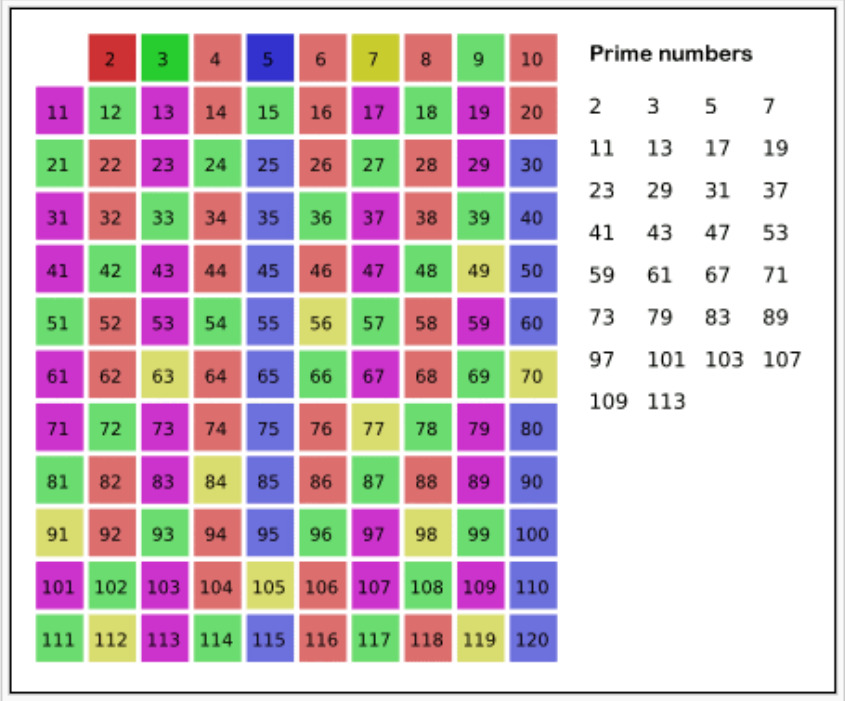

Ilustración 4: Números primos hasta el 120 [33]**.**

Por tanto se concluye que la criba de Eratothenes ayuda a encontrar primos grandes y pequeños en un intervalo de números naturales y que en el recorrido muchos de los números tachados (múltiplos) varias veces tienen correlación entre sí de tal modo que al dividir o descomponer estos números en sus factores primos se obtendrá al final un residuo 1 que indicará que estos números son números lisos. Lo cual lleva a concluir que es mucho más rápido realizar este proceso que una división por tentativa, disminuyendo considerablemente el número de pasos

para hallar un número liso por candidato (secuencia de Kraitchik). Entonces si se tiene un intervalo N el número de pasos para reconocer un número liso en ese intervalo serían:

$$N \log(\log(B))$$

Este procedimiento se repetiría por cada candidato en la secuencia $Q(x) = x^2 - n$. Pero esto aumentaría el tiempo de cálculo de "n" en:

$$exp^{(\sqrt{\log(n)\log(\log(n))}}$$

De esta forma nace la Criba cuadrática o QS a partir de una idea matemática sin algún experimento numérico formal, si no como la concatenación de varias ideas que convergen en un mismo fin que es lograr factorizar n. El QS matemáticamente se ve de la siguiente forma:

$$L(N) = exp^{\{(1+o(1))\sqrt{\ln(N)\ln(\ln(N))}\}}$$

El primero en probar el método QS fue Joseph Gerver con un RSA de 47 dígitos decimales, como una forma de aprender a programar dentro del proyecto Cunningham[5]. Pero para el momento era poco popular este nuevo método de factorización por lo que en 1982 el QS fue probado con éxito en el laboratorio de Sandia[6] en una maquina llamada Cray XMP bajo la supervisión de Gus Simmons y Tony Warnock. Poco después Peter Montgomery presento un nuevo método que aligeraba el cálculo de los polinomios con el fin de factorizar con mayor facilidad tamaños de RSA superior a 100 dígitos decimales, el primero en implementar el QS usando el método de Montgomery fue Silverman el cual lo documento en [34].

Una de las mayores ventajas que presentaba el QS era su capacidad de distribuir tareas entre varios computadores, por tanto la factorización dejo de ser tarea de supercomputadores para pasar a ser parte de la computación distribuida en máquinas comunes. Ventaja de la cual se aprovechó Arjen Lenstra para factorizar

---

[5] El Proyecto de Cunningham pretende encontrar factores de números grandes de la forma $b^n \pm 1$ para *b* = 2, 3, 5, 6, 7, 10, 11, 12 y exponentes grandes.

[6] Es un laboratorio administrado y operado por la Corporación de Sandia (una filial de la Lockheed Martin Corporation) y uno de los mejores laboratorios nacionales de investigación y desarrollo del Departamento de Energía de los Estados Unidos con dos localizaciones, unaen Albuquerque (Nuevo México) y otra en Livermore (California). Su principal misión es el desarrollo de ingeniería, y la prueba de componentes no nucleares de armas atómicas.

un tamaño de RSA de 129 dígitos decimales, pidiendo tiempo de cómputo por internet a millones de usuarios alrededor del mundo para terminar la factorización, tomó un tiempo aproximadamente de 8 meses en ser finalizado con un total de 10^{17} instrucciones o pasos.

El funcionamiento del QS como algoritmo se puede resumir en la siguiete secuencia de pasos según [35]:

Paso 1: seleccionar una base de números que son residuo cuadrático de n. Para hacer esto sólo se debe aplicar el algoritmo de Jacobi, el cual devuelve 1 si es residuo cuadrático y -1 si no lo es, y se seleccionan los números hasta el rango indicado. Por ejemplo, si el rango es 30 se escogerán los números que cumplen ser residuo cuadrático hasta ese número.

Paso 2: se obtiene la raíz cuadrada del número n.

Paso 3: se obtiene un vector de x números, los cuales serán la suma de la raíz cuadrada más el rango de aleatorios.

Paso 4: se halla un vector de números que sean función de x, donde se elevara x al cuadrado y se restara el número.

Paso 5: a este vector de f(x) se seleccionarán solo los que su factorización en números de la base cumplen con el rango de números, se le asigna a cada factor que cumple el número de su exponente y al que no cero.

Paso 6: a la matriz anterior obtenida se le calcula el módulo 2 para obtener solo una matriz binaria.

Paso 7: se selecciona las filas de la matriz que cumple que su suma en módulo 2 da cero.

Paso 8: en los vectores de x se multiplicarán todos los números que cumplieron el paso anterior y se procederá a sacar el módulo 2 obteniendo una variable que llamaremos X.

Paso 9: en los vectores de f(x) se multiplicará a todos los números que cumplieron el paso 7 y se obtendrá una variable y a la cual se le calculará la raíz para obtener Y.

Paso 10: se obtienen X y Y. Primero se debe hallar el máximo común divisor de la suma (X+Y) teniendo en cuanta para este cálculo el valor de n, de este modo se obtendrá un factor no trivial de n. Luego se debe calcular nuevamente el máximo común divisor de de la resta (X-Y) teniendo en cuenta n, para conseguir el otro factor no trivial. Finalmente, estos dos factores o números obtenido serán los factores de n.

En el congreso de Teoría de números en Vancouver de 1989 Pomerance habla por primera sobre su idea de una mejora para el QS lo denominó Pollard's Number Field Sieve y aunque aún el nuevo algoritmo no existía, habló sobre su posible funcionamiento general y su diferencia drástica con el QS que según sus supociones algebraicas y posibles resultados matemáticos esperados que se daría específicamente en la forma de su exponente, tomando la siguiente forma:

$$exp^{(c(\log(n))^{\frac{1}{3}}(\log(\log(n)))^{\frac{2}{3}})}$$

Pomerance explicaba que al reducir la constante en el exponente de $\frac{1}{2}$ a $\frac{1}{3}$ se produciría un fuerte impacto al pasar del método de fracciones continuas al QS.

Hay que mencionar que más adelante Pomerance y Lenstra mostrarían 3 variaciones para el NFS basándose en su expresión matemática. El coeficiente "c" denotaría el tipo de NFS que se estaría empleando.

- Si c≈1,523 se estaría empleando el NFS especial muy parecido al Pollar Sieve original. Este tipo de NFS es muy empleado para factorizaciones de 100 ≤ n ≤ 200.
- Si c≈1,923 se estaría empleando el Generic Number Field Sieve (GNFS) generalmente empleado para números impares compuestos.

Con el ánimo de dar forma al nuevo algoritmo Pomerance une esfuerzos con Joe Buhler y Hendrik Lenstra, para empezar definen un polinomio mónico[7] f(x) para los enteros irreducibles o enteros m donde $f(m) \equiv 0 mod(n)$. El grado de estos polinomios estará dado por un coeficiente "d" que a su vez estará dado por la factorización de número de Fermat[8], útil para n≥100. Como se muestra a continuación se tiene un número Fermant de la forma:

$$n = 2^{2^n} + 1$$

Se podrá ver como el polinomio $f(x) = 2^{515} + 8$

Puede tomar la forma $f(x) = (2^{103})^5 + 8$

De modo que se al final tendrá la forma de un polinomio Mónico de la forma:

$$f(x) = x^5 + 8$$

Donde m= 2^{103} y d= 5

$$f(x) = (m)^d + 1$$

Una vez se tiene la forma del polinomio, se supone una raíz compleja α de f(x) y a Z[α] como el conjunto de todas las expresiones polinomicas de α con coeficientes

[7] Un polinomio Mónico es de la forma: $a_0 + a_1x + a_2x^2 + \cdots a_nx^n$ se dice que es mónico si $a_n = 1$. Es decir, si el coeficiente del término de mayor grado es 1. Un ejemplo de este polinomio es: $x^3 - 6x^2 + 2x + 100$ donde el coeficiente del termino mayor que es x^3 es 1

[8] Es un número natural de la forma: $F_n = 2^{2^n} + 1$ primo. Desde 2009 se conocen la factorización completa para tamaños de n=11

enteros. Cada ocurrencia de α tendrá una proyección desde de Z[α] a Z/(nZ) llamada $\emptyset$. Luego se supone un S como un conjunto finito de pares enteros primos coprimos $\langle a,b \rangle$ con dos propiedades fundamentales:

1. El producto algebraico de enteros a-αb en S es una raíz de Z[α], lo que Pomerance llamará γ^2.
2. El producto de todos los números a-mb en S es una raíz en Z, lo que Pomerance llamará v^2.

Con lo anterior Pomerance pretenderá forma nuevamente la expresión de los polinomios de Kraitchik $Q(x) = x^2 - 1$ pero esta vez basado en α, por tanto reemplaza cada ocurrencia de α con el entero m y a esto le llamará u. De modo que:

$$u^2 = \emptyset(\gamma)^2 = \emptyset(\gamma^2) = v^2 \; mod \; (n)$$

Con estas dos propiedades pretendían generar nuevos vectores que tuvieran coordenadas sobre todos los números primos a-αb en Z[α]. De esta forma Pomerance quería realizar un Sieve mejorado con un espectro más amplio de primos.

Dado que suena confuso hallar f(x), Pomerance lo resume de la siguiente forma:

1. Definir el grado “d” del polinomio f(x)
2. Hacer a m la parte entera de $n^{\frac{1}{d}}$
3. Para generar el polinomio f(x) se debe escribir a n en base m.
4. Factorizar el polinomio mónico f(x) si este es reducible.

Para detallar un poco más la forma en que se factorización el NFS Lenstra y Adleman publican en 1993 [36] donde muestran como lo hicieron a partir de los polinomios mencionados anteriormente y que métodos emplearon según el tipo de polinomio obtenido al final. Dado que es diferente a la factorización usada para el QS.

Finalmente se puede concluir en que llamaremos X a un vector que contendrá el tamaño de los números auxiliares que se esperan combinar dentro de cada raíz de factorización. Por tanto X dará al proceso la complejidad y la rapidez de la factorización. En el Number Field Sieve se selecciona el polinomio f(x) el entero m de modo que se cumpla (a-mb)N(a-αb) para conseguir los posibles números lisos para generar la base de factorización y que a su vez se encuentra asociado al vector X. Por tanto el número de dígitos de los números con los que se hace el Sieve usando la base de números lisos es equivalente $n^{\frac{2}{3}}$ en contraste al QS que es $n^{\frac{1}{2}}$. Por lo tanto al reducir el número de dígitos de los números auxiliares se reducirá también el tiempo de cómputo, el número de posibilidades o relaciones y

el número de operaciones necesarias para factorizar n, haciendo al NFS el método hasta el momento más eficiente de factorización.

2.5.1. Software de factorización

Actualmente se encuentra a disposición algunos software para factorizar números grandes como se muestra en [37]. En el caso de este trabajo de tesis se propone explorar algunos de ellos tales como Msieve, GGNFS y Yafu con el fin de probar principalmente el QS y el NFS en sus generalidades tales como GNFS y Pollard Sieve.

Msieve es hasta ahora uno de los software libres de factorización de binarios más completos en la red. Este software no solo emplea el algoritmo Sieve para la factorización de números, contrariamente clasifica el tamaño de los números con el fin de aplicar a cada rango de intervalos el método más eficaz para factorizar. En el caso de los números menores a 25 dígitos decimales el software aplica división por tentativa y el algoritmo Rho de Jhon Pollard[9]. En el caso de números de tamaño intermedio mayores a 25 dígitos el software conmutará entre el GMP-ECM y el ECM (Curva elíptica de Lenstra) y en el caso de que la factorización requiera mayor robustez en el proceso el software cambiará al Quadratic Sieve si el número no supera los 100 dígitos decimales, de los contrario implementará inmediatamente el Number Field Sieve y sus generalidades para resolver la factorización.

Este software se caracteriza por ser rápido en comparación a otros dada su programación. El código está escrito en C y sus mejoras han sido codificadas en Visual Basic para ofrecer al usuario su uso sistemas operativos tales como Linux o Windows, por ota parte este software contiene una serie de bibliotecas estáticas autónomas encargadas de la factorización (lib msieve.a) y de su interfaz (msieve.h). La intención del grupo de programadores fue que el programa fuera fácil de usar, para ello solo se requiere ingresar el número que se desea factorizar por línea de comando o por un archivo de texto plano, posteriormente el programa se encargará de realizar todas las operaciones de factorización almacenando todos los datos en diferentes archivos durante el proceso, en caso de que si en un momento es suspendido el proceso este pueda reactivar su proceso desde el punto en el que quedo. El consumo de memoria del software depende básicamente del tamaño del número que se este factorizando, en el caso de

[9] Algoritmo usado para números compuestos que tengan factores pequeños, basado en el algoritmo de la liebre y la tortuga y la paradoja de cumpleaños

números menores a 100 dígitos se estima que requiere entre 55MB y 65MB, mientras que para tamaños mayores a 100 dígitos decimales puede tomar de 100MB a 3GB si el número llega a la barrera de los 512 bits. Es claro mencionar que el programa consumirá más memoria casi al finalizar la factorización.

Por otra parte el software por si solo no puede factorizar tamaños de números mayores a 110 dígitos decimales, por tanto se recomienda instalar la librería de GGNFS [38]. En principio el programa ejecutará el GGNFS y para finalizar empleará el Msieve para desarrollar el proceso de algebra lineal que requiere.

Finalmente este programa no fue creado pensando para trabajarse en redes grandes, puede ser empleado en pequeñas LAN, en el caso de trabajarse en Cluster se recomienda distribuir algunas de sus librerías. Para conocer un poco más sobre el proyecto Msieve puede consultarse [39]

Yafu es uno de los software de factorización más reconocidos en la red, este software que también es de libre distribución y se caracteriza al igual que Msieve en hacer uso de librerías que implementen métodos de factorización como GNFS, SNFS y ECM. Yafu fue creado en 2010 por Benjamin Chaffin como resultado del proyecto Hobby enfocado en la factorización de enteros grandes, Chaffin aún continúa trabajando en el proyecto, una de las mejoras de Yafu es MyFu que pretende mejorar algunos procesos y mejorar los tiempos de factorización. Yafu utiliza algoritmos de factorización de forma automática, por otro lado el número a factorizar puede ser ingresado por línea de comando o por medio de una interfaz gráfica similar a la de MATLAB que a su vez permite varias funcionalidades que aunque son limitadas invitan al usuario a interactuar con el código y el algoritmo en sí autorizando al usuario a personalizar el funcionamiento del software según su necesidad. Para conocer más sobre este proyecto se puede obtener más documentación en [40]

2.6. CONCLUSIONES

En la revisión bibliográfica se encontró mucha documentación en torno a las multiples investigaciones realizadas actualmente y en años pasados del algoritmo RSA donde se muestra claramente su funcionamiento, posibles debilidades y fortalezas y así mismo los avances obtenidos hasta el momento en su criptoanálisis. Dado que el campo que busca explorar este proyecto no es netamente matemático se encuentran varios posibles campos de exploración en la optimización de los algoritmos de factorización mediante procesos matemáticos o en la generación de estrategias probabilísticas que enfoquen la investigación a

encontrar con mayor rápidez los primos que componen “n”. Por otra parte en el área de tratamiento de señales se encuentra un amplio campo de exploración empleando FPGAs o Raspberrys para atacar un sistema que emplee al RSA como algoritmo de seguridad o analizando las fugas de campo electromagnético generadas por los aparatos electrónicos tales como tarjetas electrónicas, chips y computadoras, como también se propone un estudio de hardware a nivel físico y lógico de los mismos para vulnerar el sistema y acceder a la información. Cabe aclarar que muchas de estas investigaciones no han logrado romper tamaños de RSA mayores a 100 dígitos decimales, razón por la cual este trabajo pretende explorar mejoras a los tiempos de criptoanálisis empleando métodos de factorización en un arreglo de Cluster, ya que los métodos de factorización son hasta ahora los que registran respuestas contundentes frente a tamaños de RSA por debajo de los 229 dígitos decimales alrededor de 760 bits. Tamaños tales como 1024 y 2040 aún no se registran como factorizados. Algunas investigaciones como [4] aseguran haberlo logrado sin embargo no es un campo de profundización que pretenda tomar este trabajo, pero encuentra una utilidad en el aporte que genera esta investigación.

BIBLIOGRAFIA:

[1] “MOOC Crypt4you UPM.” [Online]. Available: http://www.criptored.upm.es/crypt4you/temas/RSA/leccion8/leccion08.html. [Accessed: 19-Oct-2015].

[2] D. M. Bressoud, “Factorization and Primality Testing,” in *Springer*, Board, Ed. New York ,USA, 1989.

[3] M. Kaminaga, H. Yoshikawa, and T. Suzuki, “Double Counting in <inline-formula> <tex-math notation=‘LaTeX’>2^{t} </tex-math></inline-formula>-ary RSA Precomputation Reveals the Secret Exponent,” *IEEE Trans. Inf. Forensics Secur.*, vol. 10, no. 7, pp. 1394–1401, Jul. 2015.

[4] D. Genkin, A. Shamir, and E. Tromer, “RSA key extraction via low-bandwidth acoustic cryptanalysis,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8616 LNCS, no. PART 1, pp. 444–461, 2014.

[5] A. P. Fournaris, “Fault and simple power attack resistant RSA using Montgomery modular multiplication,” in *Proceedings of 2010 IEEE*

International Symposium on Circuits and Systems, 2010, pp. 1875–1878.

[6] F. Amiel, K. Villegas, B. Feix, and L. Marcel, “Passive and Active Combined Attacks: Combining Fault Attacks and Side Channel Analysis,” in *Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC 2007)*, 2007, pp. 92–102.

[7] H. C. A. van Tilborg and S. Jajodia, *Encyclopedia of Cryptography and Security, Volume 1*, vol. 0. Springer Science & Business Media, 2011.

[8] E. Landquist, “The Quadratic Sieve Factoring Algorithm,” 2001.

[9] T. D. B. D. A. K. L. M.S. Manasse, “On the factorization of RSA-120,” pp. 166–174, 1993.

[10] D. Atkins, M. Graff, a Lenstra, and P. Leyland, “The magic words are squeamish ossifrage,” *Adv. Cryptology— …*, pp. 263–277, 1995.

[11] “RSA Laboratories - RSA-576 is factored!” [Online]. Available: http://www.emc.com/emc-plus/rsa-labs/historical/rsa-576-factored.htm. [Accessed: 27-Nov-2015].

[12] T. Kleinjung, K. Aoki, J. Franke, A. K. Lenstra, E. Thomé, J. W. Bos, P. Gaudry, A. Kruppa, P. L. Montgomery, D. A. Osvik, H. Te Riele, A. Timofeev, and P. Zimmermann, “Factorization of a 768-Bit RSA modulus,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6223 LNCS, pp. 333–350, 2010.

[13] “Usuarios de Internet (por cada 100 personas) | Datos | Tabla.” [Online]. Available: http://datos.bancomundial.org/indicador/IT.NET.USER.P2. [Accessed: 27-Mar-2016].

[14] “Criptografía - Wikipedia, la enciclopedia libre.” [Online]. Available: https://es.wikipedia.org/wiki/Criptograf%C3%ADa#Algoritmos. [Accessed: 27-Mar-2016].

[15] “RSA Laboratories - The RSA Challenge Numbers.” [Online]. Available: http://www.emc.com/emc-plus/rsa-labs/historical/the-rsa-challenge-numbers.htm. [Accessed: 02-Dec-2015].

[16] M. Ni, *Investigación Diseño y ejecución*. .

[17] W. Diffie, W. Diffie, and M. E. Hellman, “New Directions in Cryptography,” *IEEE Trans. Inf. Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[18] S. J. Aboud, “An efficient method for attack RSA scheme,” in *2009 Second*

International Conference on the Applications of Digital Information and Web Technologies, 2009, pp. 587–591.

[19] R. L. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[20] J. R. Aguirre, “Seguridad informática y Criptografía.” Universidad Politécnica de Madrid, Madrid, 2004.

[21] D. C. F. G. J. S. C. S. Halevi, “Cryptanalysis of ISO/IEC 9796-1,” Paris, Francia, 2009.

[22] “RSA Laboratories - RSA-640 is factored!” [Online]. Available: http://www.emc.com/emc-plus/rsa-labs/historical/rsa-640-factored.htm. [Accessed: 02-Dec-2015].

[23] P. Roberto de Oliveira, V. Delisandra Feltrim, L. Andreia Fondazzi Martimiano, and G. Brasilino Marcal Zanoni, “Energy Consumption Analysis of the Cryptographic Key Generation Process of RSA and ECC Algorithms in Embedded Systems,” *IEEE Lat. Am. Trans.*, vol. 12, no. 6, pp. 1141–1148, Sep. 2014.

[24] P. C. Kocher, “Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems,” *Adv. Cryptol. — CRYPTO ’96*, vol. LNCS 1109, pp. 104–113, 1996.

[25] B. Devlin, H. Ueki, S. Mori, S. Miyauchi, M. Ikeda, and K. Asada, “Performance and side-channel attack analysis of a self synchronous montgomery multiplier processing element for RSA in 40nm CMOS,” in *2012 IEEE Asian Solid State Circuits Conference (A-SSCC)*, 2012, pp. 385–388.

[26] G. M. Bertoni, L. Breveglieri, A. Cominola, F. Melzani, and R. Susella, “Practical Power Analysis Attacks to RSA on a Large IP Portfolio SoC,” in *2009 Sixth International Conference on Information Technology: New Generations*, 2009, pp. 455–460.

[27] H. J. Mahanta, A. K. Azad, and A. K. Khan, “Power analysis attack: A vulnerability to smart card security,” in *2015 International Conference on Signal Processing and Communication Engineering Systems*, 2015, pp. 506–510.

[28] A. Miyamoto, N. Homma, T. Aoki, and A. Satoh, “Chosen-message SPA attacks against FPGA-based RSA hardware implementations,” in *2008 International Conference on Field Programmable Logic and Applications*,

2008, pp. 35–40.

[29] A. Pellegrini, V. Bertacco, and T. Austin, “Fault-based attack of RSA authentication,” in *2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010)*, 2010, pp. 855–860.

[30] N. Homma, A. Miyamoto, T. Aoki, A. Satoh, and A. Samir, “Comparative Power Analysis of Modular Exponentiation Algorithms,” *IEEE Trans. Comput.*, vol. 59, no. 6, pp. 795–807, Jun. 2010.

[31] C. Chen, T. Wang, and J. Tian, “Improving timing attack on RSA-CRT via error detection and correction strategy,” *Inf. Sci. (Ny).*, vol. 232, pp. 464–474, 2013.

[32] C. Pomerance, “A Tale of Two Sieves,” pp. 1473–1485, 1996.

[33] W. M. F. and W. M. F., “Criba de Eratóstenes,” *Rev. Digit. Matemática Educ. e Internet*, vol. 7, no. 2.

[34] R. Silverman, “The multiple polynomial quadratic sieve,” *Math. Comput.*, vol. 48, no. 177, pp. 329–339, 1987.

[35] J. Fernandez, “Implementacion de la Criba Cuadratica (Quadratic Sieve) en C++ ~ Code Botic.” [Online]. Available: http://www.codebotic.com/2015/07/implementacion-de-la-criba-cuadratica.html. [Accessed: 27-Nov-2015].

[36] A. K. H. W. L. J. Lenstra, *The development of the number field sieve*, vol. 1554. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993.

[37] Jeff Gilchrist, “Windows Factoring Software Binaries (64bit & 32bit),” 2013.

[38] A. K. Andrei Belenko, “GGNFS suite,” 2007. [Online]. Available: https://sourceforge.net/p/ggnfs/wiki/Home/.

[39] Jason Papadopoulos, “Msieve,” 2009. [Online]. Available: https://sourceforge.net/p/msieve/wiki/Home/.

[40] “yafu download | SourceForge.net,” 2010. [Online]. Available: https://sourceforge.net/projects/yafu/. [Accessed: 04-Aug-2016].