UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS
POSTGRADO EN INFORMÁTICA
MAESTRÍA EN INGENIERÍA DE SOFTWARE

TESIS DE MAESTRÍA

SEGURIDAD CRIPTOGRÁFICA HÍBRIDA DE CLAVE PÚBLICA Y PRIVADA APLICADA A LA MENSAJERÍA SMS

POSTULANTE: ING. RODOLFO VELÁSQUEZ ARANDO

TUTOR: M.Sc. MARCELO PINTO MACEDO

REVISOR: M.Sc. ELIZABETH GARCIA ESCALANTE

La Paz - 2009

Dedicatoria

A mi pequeño Itamar que con su ternura y alegría fue la inspiración para la culminación de esta investigación y así cumplir esta meta en mi vida.

A mis padres que con su ejemplo y apoyo fueron el impulso para seguir adelante.

Agradecimientos

Agradezco a ese ser superior, al que yo llamo Dios, por bendecirme y estar a mi lado para fortalecer mi espíritu en cada momento de mi vida.

Con mucho cariño quiero agradecer a mis padres Freddy y Juana quienes con su amor, orientación y apoyo incondicional han hecho de mi lo que soy.

Un sincero agradecimiento al M.Sc. Carlos Morato por las recomendaciones y observaciones en el desarrollo de mi investigación, al M.Sc. Marcelo Pinto para su culminación y a la M. Sc. Elizabeth García por sus críticas para la mejora del documento.

A mis amigos y familiares les agradezco que siempre hayan contado conmigo y alentado para llevar a cabo esta pequeña meta en mi proyecto de vida.

TABLA GENERAL DE CONTENIDOS

Resumen
Abstract

Pag.

Capítulo 1: Marco referencial.....1
Capítulo 2: Marco conceptual.....7
Capítulo 3: Marco práctico.....32
Capítulo 4: Análisis de resultados y validación de hipótesis.....45
Capítulo 5: Conclusiones y recomendaciones.....48
Referencias bibliográficas
Anexo A
Anexo B

TABLA ESPECÍFICA DE CONTENIDOS

Pag.

CAPÍTULO 1: MARCO REFERENCIAL..... 1
1.1 Introducción..... 1
1.2. Estado del arte..... 2
1.3 Problemática..... 3
1.3.1 Descripción del problema..... 3
1.3.2 Planteamiento del problema..... 3
1.4.1 Objetivo principal..... 4
1.4.2 Objetivos específicos..... 4
1.5 Hipótesis..... 4
1.6 Variables..... 4
1.6.1 Variable dependiente..... 4
1.6.2 Variable independiente..... 4
1.6.3 Variable moderante..... 5
1.6.4 Criterios de evaluación..... 5
1.7 Justificaciones..... 5
1.7.1 Justificación científica..... 5
1.7.2 Justificación técnica..... 5
1.7.3 Justificación económica..... 5
1.8 Métodos y técnicas..... 6
1.9 Alcances..... 6
1.10 Límites..... 6
CAPÍTULO 2: MARCO CONCEPTUAL..... 7
2.1 Relación temática..... 7
2.2 Teoría de números..... 8
2.2.1 Primos fuertes..... 8
2.2.2 Números primos aleatorios..... 8
2.2.2.1 Test de primalidad de Miller-Rabin.....10
2.2.3 Exponenciación binaria..... 11
2.2.4. Máximo común divisor..... 13

2.3 Criptografía pública y privada..... 14
2.3.1 Algoritmos simétricos..... 15
2.3.1.1 Cifrador de Vernam..... 15
2.3.1.2 Justificación de uso del cifrado de Vernam.....18
2.3.2 Algoritmos Asimétricos..... 18
2.3.2.1 RSA.....20
2.3.2.2 Justificación de uso del algoritmo RSA..... 22
2.3.3 Órdenes de complejidad..... 22
2.4 Algoritmos híbridos.....24
2.5 J2ME.....24
2.5.1 Tipos de datos..... 24
2.5.2 Características para compilación con tecnología J2ME.....24
2.5.3 Características principales del lenguaje java en J2ME..... 25
2.6 Teoría de la mensajería SMS.....26
2.6.1 Arquitectura del sistema inalámbrico SMS.....26
2.6.1.1 Capa de Interfase.....
26
2.6.1.2 Capa de implementación..... 26
2.6.1.3 Capa de transporte..... 27
2.6.2 El API de la mensajería SMS.....27
2.6.2.1 Message.....28
2.6.3 Servicios del API.....29
2.6.3.1 MessageObject..... 29
2.6.3.2 Protocolo.....29
2.7 Comunicación inalámbrica.....30
2.7.1 Comandos AT..... 30
CAPÍTULO 3: MARCO PRÁCTICO.....32
3.1 Diagrama arquitectónico del sistema..... 32
3.2 Codificador de mensajes a números por bloques..... 33
3.3 Generador de clave pública.....34
3.4 Encriptador RSA de 32 bits 34
3.5 Encriptador Vernam de 32 bits.....35

3.6 Decriptación del mensaje.....38
3.7 Generador de clave asimétrica..... 39
3.8 Modelo híbrido.....39
3.8.1 Encriptación 40
3.8.2 Decriptación 41
3.8.3 Tiempo de proceso de encriptación..... 41
3.9 Ejemplo de aplicación..... 42
CAPÍTULO 4: ANÁLISIS DE RESULTADOS Y VALIDACIÓN DE HIPÓTESIS.....45
4.1 Análisis de la complejidad algorítmica.....45
4.2 Análisis del tiempo de ataque al algoritmo híbrido.....46
4.3 Comparación con otros algoritmos.....47
CAPÍTULO 5: CONCLUSIONES Y RECOMENDACIONES 48
5.1 Conclusiones 48
5.2 Recomendaciones.....50

Referencias bibliográficas

ANEXO A. Algoritmo de Diffie – Hellman

ANEXO B. Características de procesadores de móviles Symbian

LISTA DE FIGURAS

Pag.

Figura 1. Esquema temático.....7
Figura 2. Esquema de un cifrador Vernam binario.....16
Figura 3. Transmisión de información con algoritmos asimétricos, fuente [9] 19
Figura 4. Comparación de órdenes de complejidad, fuente [21].....23
Figura 5. Arquitectura del sistema de mensajería inalámbrica, fuente [12].....27
Figura 6. Estructura de un mensaje, Fuente [12].....28
Figura 7. Arquitectura que aplica el algoritmo de encriptación híbrida de clave pública y privada.....32
Figura 8. Pantallas del celular con el software en encriptación.....43
Figura 9. Pantalla del software receptor.....44
Figura 10. Metodo de Diffie Hellman, fuente [10].....53

LISTA DE TABLAS

Pag.

Tabla 1. Comandos AT para manejo de SMS en modo PDU……..30
Tabla 2. Lista de comandos AT para manejo de SMS en modo texto……..31
Tabla 3. Codificación del alfabeto español……..33
Tabla 4. Tiempos de proceso de encriptación de mensaje……..42
Tabla 5. Combinaciones posibles……46
Tabla 6. Tiempo de procesamiento……46
Tabla 7. Número de combinaciones generadas para posibles claves en los algoritmos Vigenère, RSA, Vernam……47
Tabla 8. Cálculo del tiempo de procesamiento de ataque por fuerza fruta a los algoritmos Vigenère, RSA y Vernam……47

LISTA DE ALGORITMOS

Pag.

Algoritmo 1. Algoritmo de Jacobi 9
Algoritmo 2. Test de primalidad de Miller-Rabin 10
Algoritmo 3. Pseudocódigo del algoritmo de Euclides 13
Algoritmo 4. Algoritmo RSA 20
Algoritmo 5. Resumen del algoritmo RSA 21
Algoritmo 6. Generación de claves 34
Algoritmo 7. Modelo híbrido de encriptación propuesto 39
Algoritmo 8. Modelo híbrido de decritación propuesto 40

Resumen

La mensajería SMS es una de las formas de comunicación mas popular a nivel mundial pero sólo con fines de uso en que la seguridad no juega un papel importante como en publicidad, entretenimiento y otros; los mensajes cortos no contemplan seguridad alguna actualmente por lo cual la importancia es baja respecto a otras aplicaciones como transacciones bancarias, el pago de un servicio, un voto y similares.

El uso de las transacciones financieras a distancia surgió apenas en la década pasada, en la actualidad son varias las empresas y gobiernos que comienzan a apostar por este modelo de intercambio de datos que poco a poco se empieza a vislumbrar lo que sería el sistema transaccional del futuro.

Con el surgimiento de las redes inalámbricas y los dispositivos móviles ligeros tales como los celulares, el objetivo principal de esta tesis es diseñar un algoritmo híbrido de clave pública y privada con el cual se asegure la privacidad de los usuarios que utilizan la mensajería SMS de forma masiva.

Se diseñó un modelo híbrido criptográfico combinando dos modelos de encriptación RSA y Vernam con infraestructura pública y privada respectivamente en 32 bits que al combinarlos producen una fuerte seguridad híbrida que es demostrada en el capitulo 4.

Así pues para esta investigación se realiza la implementación de un sistema de envío y recepción de mensajería SMS utilizando el modelo hibrido propuesto. Dicho sistema puede enviar desde un celular un mensaje encriptado y luego ser recepcionado por otro que establecerá una comunicación inalámbrica (Bluetooth) con un computador personal para que los mensajes sean decriptados.

Palabras clave: Criptografía, SMS, mensajería, seguridad criptográfica, clave pública, clave privada, RSA, Vernam, algoritmos híbridos, mensajería, llave pública, llave privada.

Abstract

The messaging SMS is one in the communication ways but popular at world level but only with ends of use in that the security doesn't play an important paper as in publicity, entertainment and others; the short messages don't contemplate security at the moment reason why some the importance it is low regarding other applications like bank transactions, the payment of a service, a vote and similar.

The use of the financial transactions at distance hardly arose in last decade, at the present time they are several the companies and governments that begin to bet for this model of exchange of data that little by little you begins to glimpse what would be the transactional system of the future.

With the emergence of the wireless nets and the such slight mobile devices as the cellular ones, the main objective of this thesis is to design a hybrid algorithm of public and private key with which makes sure the privacy of the users that you/they use the messaging SMS in a massive way.

A hybrid model was designed combining two encryptation models respectively RSA and Vernam of public and private infrastructure in 32 bits that produce a strong hybrid security that is demonstrated when combining them in the I surrender 4.

Therefore in this thesis he/she is carried out the implementation of a shipment system and messaging reception SMS using the proposed hybrid pattern. This system can send from a cellular one an encrypted message and then to be received for another that will establish a wireless communication (Bluetooth) with a personal computer so that the messages are decrypt.

Keywords: Cryptography, SMS, messaging, cryptographic security, nail public, nail private, RSA, Vernam, hybrid algorithms, messaging, public key, private key.

CAPÍTULO 1

MARCO REFERENCIAL

1.1 Introducción

A medida que evoluciona la tecnología aumenta la posibilidad de utilización de diferentes modos de comunicación incorporados en los dispositivos móviles actuales, entre ellos la mensajería corta cuya seguridad es de nivel bajo o débil frente a la aparición de nuevas aplicaciones.

Las necesidades del usuario han cambiado, su nivel de requerimientos es otro. Por citar un ejemplo, no hace mas de cinco años enviar mensajes SMS era cosa de unos pocos, hoy existen prestadoras de servicios celulares que permiten enviar mensajes a sólo aquellos "contactos" que se encuentran dentro de las proximidades del usuario, y esto ya es visto como algo habitual, el usuario se acostumbra a este servicio y cada vez resulta mas difícil sorprenderlo con nuevas funcionalidades. Los diseñadores deberán agudizar su ingenio para lograr nuevas aplicaciones, situación que sin lugar a duda sucederá. En este escenario aparecen nuevas necesidades, cada vez más desafiantes y complejas que las anteriores, y que obligan a soluciones que deben ser puestas en servicio cada vez con mayor seguridad sin perder velocidad en las comunicaciones.

A medida que evoluciona la tecnología aumenta la posibilidad de añadir nuevas funcionalidades a los dispositivos inalámbricos y así satisfacer las necesidades de comunicaciones seguras.

La investigación de la seguridad dedicada a equipos móviles, resulta significante ya que puede ser aplicada a pequeños y grandes usuarios en diversidad de proyectos de orden comunicacional transaccional y financiero.

La viabilidad de desarrollar investigación con equipos móviles sólo es limitada por la imaginación y la disposición celulares de última generación.

El presente tema de investigación conlleva a nuevas ideas de aplicaciones ya que el futuro de la programación tiende al desarrollo de sistemas para dispositivos móviles. La

seguridad en futuras aplicaciones juega un papel muy importante y la criptografía resulta la herramienta más adecuada para este fin, considerando criptosistemas fuertes es necesario contar con algoritmos de clave publica y privada adecuados a aplicaciones en terminales móviles (celulares).

1.2. Estado del arte

La investigación actual de seguridad en la mensajería SMS se centra en mejorar los mecanismos por los cuales el usuario puede conseguir la privacidad de sus mensajes. Esto implica una mayor importancia a la mensajería SMS y futuras investigaciones relacionadas a esta forma de comunicación.

Desde el uso de la mensajería SMS, se han implementado numerosas arquitecturas en ambientes como software de alto nivel en diferentes sistemas operativos y en dispositivos de lógica programable.

En cuanto a implementaciones en dispositivos programables, lo cual es la tendencia actual, se destacan varias implementaciones desarrolladas en tesis de maestría e investigaciones que requieren de procesadores con mayor frecuencia de trabajo que la del celular convencional existente en el mercado, éstas son referenciadas en [1] [2] [3] [4].

Cada vez se destaca más la importancia de la investigación de implementaciones eficientes de seguridad en sistemas empotrados, tales como smart cards, cajeros, teléfonos, etc.

Nokia una de las empresas más importantes en el rubro de las telecomunicaciones a finales de febrero de 2007 presenta una tecnología de encriptación de mensajes de clave publica incrustado en el firmware del celular que será puesto al mercado por un costo de 1.3 millones de dólares ya que además incluye en la carcasa con adornos de diamantes [5].

Además se destaca el trabajo de tesis "Método de Encriptación de comportamiento emergente" referenciado en [22], que muestra una alternativa de hibridación de algoritmos criptográficos utilizando inteligencia artificial (algoritmos genéticos) y

criptografía asimétrica (RSA); y cuya aplicación está orientada hacia la encriptación de archivos de texto, pero su fortaleza recae en RSA-512 o superior, modelo que no es factible en cuanto a su implementación en sistemas móviles celulares de solamente poseen un procesador que soporta 32 bits.

También vale mencionar la existencia del famoso Pretty Good Privacy o PGP (privacidad bastante buena) que es un programa desarrollado por Phil Zimmermann y cuya finalidad es proteger la información distribuida a través de Internet mediante el uso de criptografía de clave pública, así como facilitar la autenticación de documentos con firmas digitales; hasta el día de hoy no existe una versión del programa para equipos móviles celulares.

1.3 Problemática

1.3.1 Descripción del problema

El uso de las telecomunicaciones hoy en día es de gran magnitud, expandiéndose rápidamente a todas las áreas de la sociedad. Sin embargo, este crecimiento surge de la urgente necesidad de mejorar la seguridad para mantener la privacidad de los usuarios.

Los mensajes cortos son la solución al envío de información tipo texto y forma masiva a bajo costo pero con la deficiencia de que los mismos pueden ser leídos o auditados por personal de las empresas que prestan el servicio telecomunicacional, es necesario establecer un mecanismo de seguridad en el protocolo para proteger la privacidad de los usuarios.

1.3.2 Planteamiento del problema

La deficiencia de la seguridad que actualmente presentan las aplicaciones en dispositivos móviles es un problema que requiere de algoritmos fuertes y/o híbridos con la cualidad de que puedan ser implementables con las consideraciones de fortaleza y la aplicación de algoritmos criptográficos.

1.4 Objetivos

1.4.1 Objetivo principal

Diseñar un algoritmo híbrido de clave pública y privada con el cual se asegure la privacidad de los usuarios que utilizan la mensajería SMS.

1.4.2 Objetivos específicos

- Desarrollar un programa en tecnología J2ME para el envío de mensajes desde celulares.
- Construir la capa de seguridad a la mensajería con el algoritmo híbrido planteado.
- Desarrollar una interfaz entre el computador y un celular receptor con características inalámbricas (Bluetooth).
- Desarrollar un programa para computadoras personales de modo que funcione como receptor y decriptor de mensajería enviada.
- Obtener el tiempo de criptoanalisis por fuerza fruta del algoritmo hibrido.

1.5 Hipótesis

El envío de mensajería SMS encriptada con un algoritmo híbrido de 32 bits propuesto de clave pública y privada en el celular, mejora la capacidad de implementación en la seguridad criptográfica en dispositivos móviles.

1.6 Variables

1.6.1 Variable dependiente

La mensajería SMS.

1.6.2 Variable independiente

Software de encriptación en el celular.

1.6.3 Variable moderante

Seguridad para los usuarios.

1.6.4 Criterios de evaluación

Comparación con otros algoritmos, tiempo de criptoanálisis por fuerza bruta al algoritmo híbrido y número de instrucciones necesarias para la encriptación y decriptación independiente del procesador.

1.7 Justificaciones

1.7.1 Justificación científica

La investigación de aplicaciones para tecnología móvil con criptografía en la mensajería SMS, científicamente es una interesante combinación de las matemáticas con protocolos telecomunicacionales ya establecidos.

Por otro lado mensajería SMS que hoy en día no es usada en transacciones con alto nivel de seguridad podría ser realzada en cuanto a importancia, y el incremento en futuras investigaciones referidas a la seguridad de comunicaciones en equipos móviles.

1.7.2 Justificación técnica

Con el fin de dar respuesta a la problemática, técnicas como la encriptación de datos resultan aplicables al envío de información (SMS) masiva, sin necesidad de establecer permisos u otros convenios con las empresas que prestan los servicios de telefonía celular, se puede generar con la tecnología actual una capa de seguridad y utilizando algoritmos de criptografía otorgar la seguridad para los mensajes de los usuarios.

1.7.3 Justificación económica

El impacto económico que es posible lograr es inmensurable ya que los servicios con mensajería SMS tomarían importancia en el mundo de las transacciones seguras.

El método de envío de mensajes SMS encriptados que se propone no representa un costo significativo ante las utilidades que se pueden llegar a obtener con este.

1.8 Métodos y técnicas

A continuación se lista los métodos y técnicas que se requerirán para cumplir con el objetivo.

- ✓ Encriptación y Decriptación de texto con la técnica de clave pública y privada.
- ✓ Programación de Midlets bajo la plataforma J2ME.
- ✓ Programación de comandos AT relacionado con tecnología Bluetooth.
- ✓ Enlace de una PC con cualquier celular que soporte tecnología Bluetooth.

1.9 Alcances

La investigación se enmarca en la propuesta de un modelo de seguridad de infraestructura algoritmica hibrida RSA-Vernam de 32 bits dedicada a la aplicación de mensajería SMS. Este modelo es desarrollado en J2ME e instalado en un celular Nokia 6230.

También se desarrolla un software receptor de la mensajería para la respectiva decriptación capaz de correr en un computador personal.

1.10 Límites

El modelo de encriptación desarrollado en software puede ser instalado en celulares que soporten J2ME.

El software receptor es desarrollado en VB 6.0 que puede ser instalado en un computador personal bajo una plataforma Windows 9.X, Me, XP, 2000 o Vista; éste sólo realiza la tarea de configurar al celular vía Bluetooth, recibir los mensajes y decriptarlos.

No se realiza un modelo para el intercambio de claves privadas ya que existen algunos modelos existentes ya planteados, pero se sugiere utilizar el método de Diffie Hellman expuesto en el anexo A.

CAPÍTULO 2

MARCO CONCEPTUAL

2.1 Relación temática

El esquema de la figura 1 muestra la relación temática necesaria para el desarrollo del algoritmo híbrido.

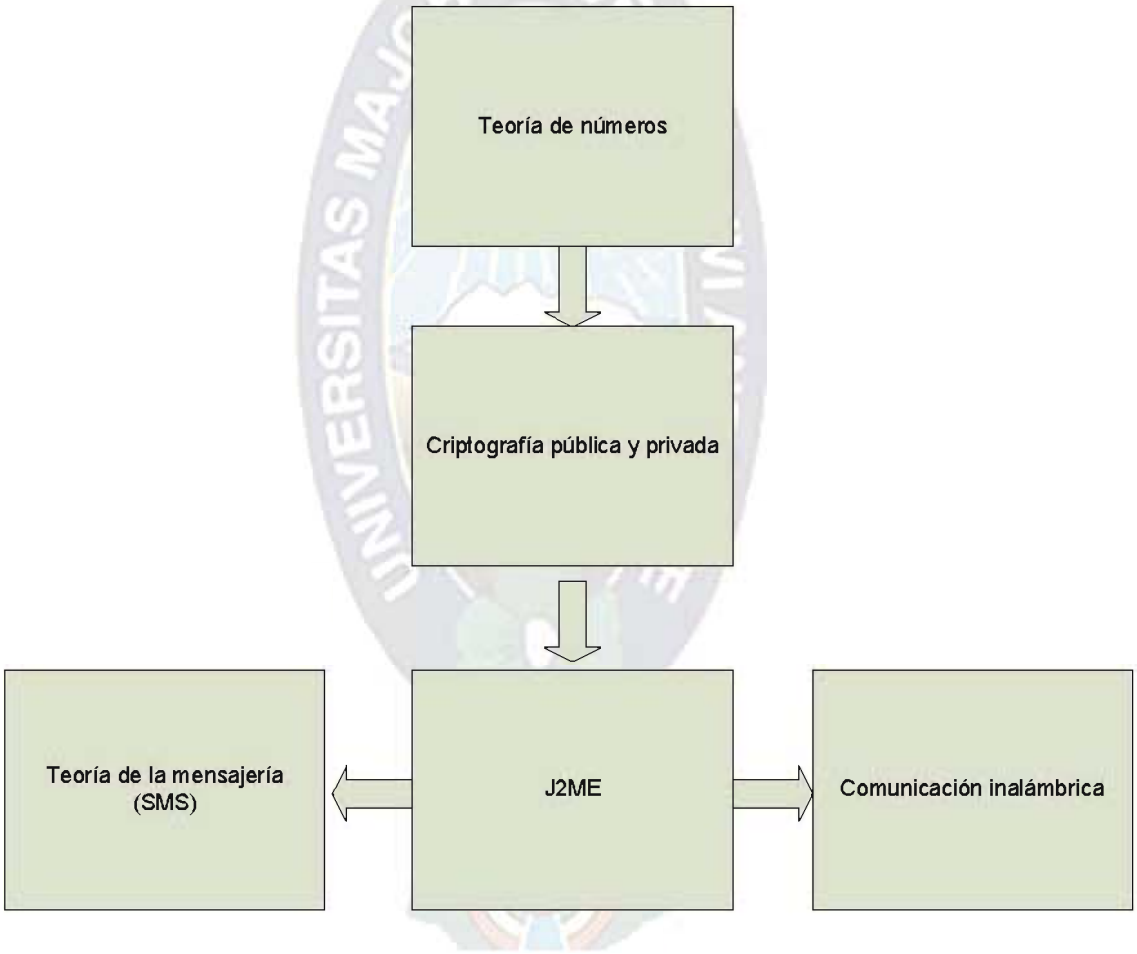

Figura 1. Esquema temático

2.2 Teoría de números

A continuación se explica los principales conceptos y algoritmos de la teoría de números necesarios para respaldar la algoritmia propuesta en esta tesis.

2.2.1 Primos fuertes

Debido a que muchos algoritmos de tipo asimétrico basan su potencia en la dificultad para factorizar números enteros grandes, a lo largo de los años se propusieron diversas condiciones que debían cumplir los números empleados en aplicaciones criptográficas para que no fueran fáciles de factorizar. Se empezó entonces a hablar de números primos fuertes, [8].

Aunque p y q sean primos grandes, existen algunos casos en los que es relativamente fácil factorizar el numero $n=pq$. Se proponen entonces una serie de condiciones para p y q que dificultan la factorización de n. Se dice que p y q son números primos fuertes si cumplen:

- El máximo común divisor de p-1 y q-1 debe ser pequeño
- p-1 y q-1 deben tener algún factor primo grande p' y q'
- Tanto p'-1 como q'-1 deben tener factores primos grandes
- Tanto p'+1 como q'+1 deben tener factores primos grandes

Las dos primeras condiciones se cumplen si tanto (p-1)/2 como (q-1)/2 son números primos.

2.2.2 Números primos aleatorios

Para descubrir un número primo se van probando los números (solo impares) hasta encontrar a uno que sea primo.

Para probar si un número es primo, en general, se utiliza un método probabilístico:

Se escoge un número aleatorio a de distribución uniforme {1,…, b-1} y se observa que:

Si $MCD(a,b)=1$ y $J(a,b) \equiv a(b\text{-}1)/2 \mod b$

Siendo *J(a,b)* el símbolo de Jacobi, la equivalencia es siempre cierta si *b* es primo. Si *b* no lo es será falso con una probabilidad de al menos 1/2. Si se mantiene para 100 valores aleatorios elegidos de a entonces b es casi con toda seguridad primo. Existe una posibilidad de una entre 2^{100} de que *b* sea no primo, pero si incluso aun, si se usa un número no primo, el receptor detectaría esto notando que el desencriptado no funciona correctamente. Cuando *b* es impar, *a<=b y MCD(a,b)=1* el símbolo de Jacobi toma un valor entre *{-1,1}* y puede ser eficientemente computado por el siguiente método obtenido de [9].

```
Int J(a,b){
        If (a= = 1) return 1
        Else
                If a es par
                        J(a/2,b)*((-1)(b^2-1)/8
                Else
                        J(b(mod a),a)* ((-1)(a-1)*((b-1)/4)
}
```

Algoritmo 1. Algoritmo de Jacobi, fuente [9]

Para verificar que el número obtenido es realmente primo no es viable tratar de factorizar el número para saber si es primo, pero existen métodos probabilísticos que pueden decir con un alto grado de certeza si un número es o no compuesto, a continuación se los mencionan los métodos existentes para el análisis de la primalidad.

- Test de primalidad de Fermat
- Test de primalidad de Miller-Rabin
- Test de primalidad de Solovay Strassen
- Método de Lehmann
- Test de primalidad con curvas elípticas
- La prueba de Abril

El problema de la primalidad consiste en determinar, de forma aleatoria, números primos grandes. El problema surge, entre otras razones porque el criptosistema del RSA necesita, para ser implementado, dos números primos grandes *P* y *Q*, Para resolver este problema se recurre a los test de primalidad. De los mencionados se elige

a Miller-Rabin por su efectividad y mayor robustez frente a los otros cuya comparación es expuesta en [18].

2.2.2.1 Test de primalidad de Miller-Rabin

Se elige Un número natural $n>1$, el número k de veces que se ejecuta el test determinará la fiabilidad del test. La salida será COMPUESTO si n es compuesto y POSIBLE PRIMO si n es un posible primo.

Definir r y s tal que r es impar y $(n-1) = r \cdot 2^s$

Para j desde 1 hasta k realizar lo siguiente:

$a \leftarrow$ Función Generar_número_aleatorio_en_intervalo [2, n-2]

$y \leftarrow a^r \bmod n$

Si $y \neq 1 \wedge y \neq n-1$ entonces:

$j \leftarrow 1$

Mientras $j \leq (s-1) \wedge y \neq n-1$ haga lo siguiente:

$y \leftarrow y^2 \bmod n$

Si $y=1$ entonces:

Retorne COMPUESTO

$j \leftarrow j+1$

Si $y \neq n-1$ entonces:

Retorne COMPUESTO

Retorne POSIBLE PRIMO

Algoritmo 2. Test de primalidad de Miller-Rabin, fuente [19]

2.2.3 Exponenciación binaria

Este algoritmo utilizado para calcular de forma rápida grandes potencias de un número x dado. También es conocida como algoritmo de exponenciación por cuadrados o de elevar al cuadrado y multiplicar. Implícitamente utiliza la expansión binaria del exponente. Es de uso bastante regular en aritmética modular, [15].

Este algoritmo es similar al de la duplicación en la multiplicación, a continuación el algoritmo recursivo calcula x^n para un entero n dado:

$$Potencia(x,n)=\begin{cases} x, & \text{si } n=1 \\ Potencia(x^2, n/2), & \text{si } n \text{ es par} \\ x \ \ Potencia\ (x^2,(n-1)/2), & \text{si } n>2 \text{ es impar} \end{cases}$$

Comparado con el método original de multiplicar x por sí mismo $n - 1$ veces, este algoritmo sólo utiliza $O(log\ n)$ multiplicaciones y acelera el cálculo de x^n tremendamente; más o menos de la misma forma que el algoritmo de la *multiplicación* acelera una multiplicación sobre el método más lento de realizar una suma repetida, [15].

La misma idea permite el cálculo rápido de potencias muy grandes en módulo. Especialmente en criptografía, es útil calcular potencias en el anillo de los enteros módulo q.

La idea puede ser usada también para computar potencias de números enteros en un semigrupo, usando la regla:

$$Potencia(x, -n) = (Potencia(x, n))^{-1}.$$

Este método funciona en cualquier semigrupo, y es usado frecuentemente para calcular potencias de matrices.

Por ejemplo, la evaluación de:

$$13789^{722341} \bmod 2345$$

Tomaría mucho tiempo y espacio de almacenamiento si el método ingenuo es usado: calcular 13789^{722341} y tomar el residuo cuando es dividido por 2345. Incluso usando un método más efectivo tomará tiempo considerable: elevar 13789 al cuadrado, tomar el residuo cuando se divide por 2345, multiplicar el resultado por 13789, y así sucesivamente. Este proceso realizará 722340 multiplicaciones modulares. Este algoritmo está basado en la observación que $13789^{722341} = 13789(13789^2)^{361170}$. Entonces, si se calcula 13789^2, el cálculo completo tomaría 361170 multiplicaciones modulares. Ésta es una ganancia en un factor de dos. Pero como el nuevo problema sigue siendo similar al anterior, se puede aplicar la observación nuevamente, reduciendo a la mitad la cantidad, aproximadamente.

La aplicación sucesiva de este algoritmo es equivalente a descomponer el exponente (convirtiéndolo a base binaria) en una secuencia de cuadrados y multiplicaciones, a continuación el ejemplo muestra la aplicación.

$x^{13} = x^{1101}{}_{bin}$

$= x^{(1*2\wedge3 + 1*2\wedge2 + 0*2\wedge1 + 1*2\wedge0)}$

$= x^{1*2\wedge3} * x^{1*2\wedge2} * x^{0*2\wedge1} * x^{1*2\wedge0}$

$= x^{2\wedge3} * x^{2\wedge2} * 1 * x^{2\wedge0}$

$= x^8 * x^4 * x^1$

$= (x^4)^2 * (x^2)^2 * x$

$= (x^4 * x^2)^2 * x$

$= ((x^2)^2 * x^2)^2 * x$

$= ((x^2 * x)^2)^2 * x$ ← el algoritmo necesita sólo 5 multiplicaciones en vez de

13 - 1 = 12 multiplicaciones

2.2.4. Máximo común divisor

Para calcular el ***MCD*** de dos enteros *a* y *b* se recurre al algoritmo de Euclides donde (ambos mayores que cero, suponemos $a \geq b$) se definen q_i y r_i recursivamente mediante las ecuaciones:

$$a = bq_1 + r_1 \quad (0 \leq r_1 < b)$$

$$b = r_1 q_2 + r_2 \quad (0 \leq r_2 < r_1)$$

$$r_1 = r_2 q_3 + r_3 \quad (0 \leq r_3 < r_2)$$

....

$$r_{k-3} = r_{k-2} q_{k-1} + r_{k-1} \quad (0 \leq r_{k-1} < r_{k-2})$$

$$r_{k-2} = r_{k-1} q_k \quad (r_k = 0)$$

Y de la proposición anterior, se concluye que:

$$MCD(a,b) = MCD(b,r_1) = MCD(r_1,r_2) = \ldots = MCD(r_{k-2}, r_{k-1}) = r_{k-1}$$

Además se demuestra que el número de pasos necesarios para calcular el *MCD* de dos números es menor que 5 veces el número de dígitos del menor de ellos. A continuación, se muestra un pseudocódigo del algoritmo de Euclides para *MCD(a,b)*

```
Si a<b intercambiar (a,b)  // a será el mayor
        Mientras b≠ 0
                r=a mod b
                a=b
                b=r
        fin
retornar a
```

Algoritmo 3. Pseudocódigo del algoritmo de Euclides

Sean a, b enteros no nulos, y sea $d = MCD(a,b)$. Entonces d es el menor entero positivo (no nulo) que puede expresarse en la forma $ax + by$ con $x, y \in Z$.

El algoritmo de Euclides también proporciona un método para calcular dos valores enteros x e y tales que $MCD(a,b) = ax + by$. Este método consiste en ir despejando el resto de la última división, el que nos da el valor $MCD(a,b)$, hacia atrás hasta llegar a los valores a y b de partida.

Para la aplicación criptográfica mediante primos relativos se tiene que si a,b son enteros no nulos. Entonces $MCD(a,b) = 1$ si y sólo si existen enteros s y t tales que $as + bt = 1$. Se dice en este caso que a y b son números primos entre sí, o que son primos relativos, o simplemente que a es primo con b (y viceversa).

2.3 Criptografía pública y privada

La Criptografía se ocupa del diseño de procedimientos para cifrar, es decir, para enmascarar una determinada información de carácter confidencial. Hay varios tipos de seguridad, según el problema computacional en que se base [20]:

- Seguridad incondicional: cuando el sistema es seguro frente a un atacante con tiempo y recursos computacionales ilimitados (por ejemplo, el cifrado de Vernam, basado en secuencias binarias aleatorias).

- Seguridad computacional: cuando el sistema es seguro frente a un atacante con tiempo y recursos computacionales limitados (por ejemplo, el sistema RSA, basado en números primos).

- Seguridad probable: cuando el sistema no puede demostrarse seguro, pero en la práctica aún no ha sido violado (por ejemplo, el sistema DES, basado en "cajas" no lineales).

- Seguridad condicional: cuando el sistema es seguro en tanto el enemigo carece de medios para atacarlo (por ejemplo, métodos clásicos de sustitución, vulnerables a un análisis de frecuencias).

2.3.1 Algoritmos simétricos

La criptografía simétrica se refiere al conjunto de métodos que permiten tener comunicación segura entre las partes siempre y cuando anteriormente se hayan intercambiado la clave correspondiente que llamaremos clave simétrica. La simetría se refiere a que las partes tienen la misma llave tanto para cifrar como para descifrar.

Este tipo de criptografía se conoce también como criptografía de clave privada o criptografía de llave privada.

Existe una clasificación de este tipo de criptografía en tres familias, la criptografía simétrica de bloques (block cipher), la criptografía simétrica de lluvia (stream cipher) y la criptografía simétrica de resumen (hash functions). Aunque con ligeras modificaciones un sistema de criptografía simétrica de bloques puede modificarse para convertirse en alguna de las otras dos formas, sin embargo es importante verlas por separado dado que se usan en diferentes aplicaciones como sugiere en [8].

La criptografía simétrica ha sido la más usada en toda la historia, ésta a podido ser implementada en diferente dispositivos, manuales, mecánicos, eléctricos, hasta los algoritmos actuales que son programables en cualquier computadora. La idea general es aplicar diferentes funciones al mensaje que se quiere cifrar de tal modo que sólo conociendo una clave pueda aplicarse de forma inversa para poder así descifrar.

2.3.1.1 Cifrador de Vernam

En 1917 *Gilbert S. Vernam*, nativo de Brooklyn e ingeniero del MIT que trabaja en los laboratorios de la empresa AT&T, diseña un dispositivo criptográfico para comunicaciones telegráficas basado en los 32 códigos Baudot de los teletipos desarrollados por su compañía. Los códigos Baudot representan los caracteres del lenguaje con cinco elementos que pueden ser el espacio o la marca (el cero y el uno) diseñado para transmisiones telegráficas. Este cifrador, que tuvo una gran aplicación durante la Primera Guerra Mundial, basa su seguridad en el secreto de una clave aleatoria que se supone tan larga como el mensaje y que luego de usarse debería destruirse. Un dato anecdótico: en dicha confrontación algunos encargados del sistema de cifra (llamados *criptocustodios* en el lenguaje militar) hicieron caso omiso de esta recomendación y los códigos fueron rotos por los aliados.

Cada carácter M_i se representa con 5 bits en código Baudot que se suma OR exclusivo (módulo 2) con la correspondiente clave k_i de una secuencia binaria aleatoria. De esta forma, el cifrador de Vernam genera un flujo de bits de texto cifrado de la forma:

$$C = E_K(M) = C_1C_2C_3 \ldots C_N$$

donde:

$$C_i = (M_i + k_i) \bmod 2 \quad para\ i = 1,2,\ldots,N$$

$$C_i = M_i \oplus k_i$$

Para la operación de descifrado, se utiliza el mismo algoritmo por la propiedad involutiva de la operación OR exclusivo. Esto es:

$$C_i \oplus k_i = (M_i \oplus k_i) \oplus k_i$$

Como $k_i \oplus k_i = 0$ para $k_i = 0$ y $k_i = 1$, se obtiene: $C_i \oplus k_i = M_i$

En la Figura 2 se muestra un cifrador de Vernam binario con las respectivas operaciones y flujo de datos.

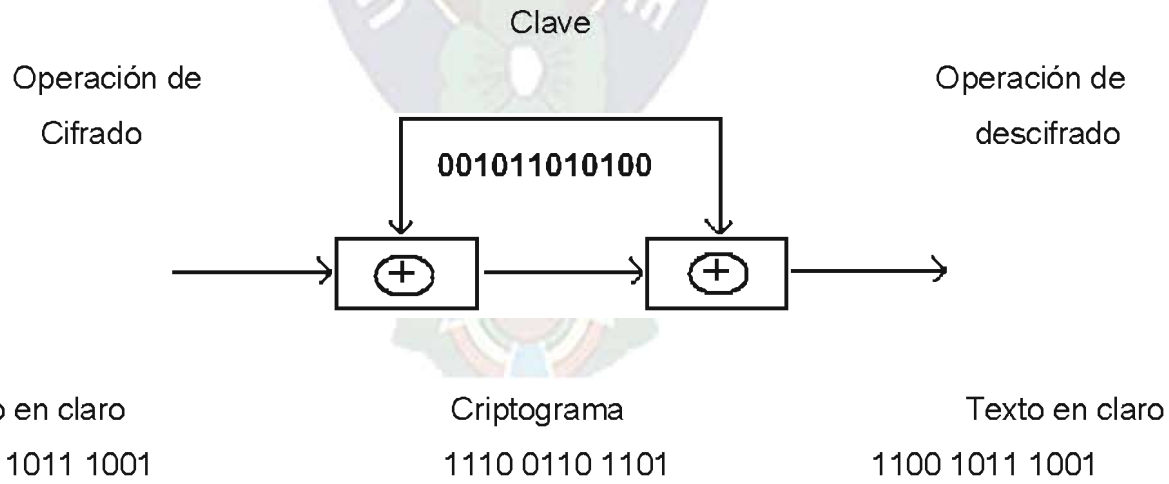

Figura 2. Esquema de un cifrador Vernam binario

A manera de ejemplo se cifra el mensaje M = BYTES con la clave K = VERNAM.

Solución: Haciendo la suma *OR* exclusivo se tiene:

$B \oplus V = 11001 \oplus 11110 = 00111 = U$

$Y \oplus E = 10101 \oplus 00001 = 10100 = H$

$T \oplus R = 10000 \oplus 01010 = 11010 = G$

$E \oplus N = 00001 \oplus 01100 = 01101 = F$

$S \oplus A = 00101 \oplus 00011 = 00110 = I$

Luego, C = UHGFI

Para descifrar se aplica nuevamente la operación del OR exclusivo como se indica a continuación.

Se recibe el siguiente criptograma C = 00110 10100 11100 11010 00000 00010 01110 01011 00110 de un texto que se ha cifrado con clave la K = Gloria Estefan.

El proceso de decriptación recurre a la suma OR exclusivo de C y K (C ⊕ K) como se muestra en los siguientes pasos.

K=G: 00110 ⊕ 11010 = 11100 = M *K*=A: 00010 ⊕ 00011 = 00001 = E

K=L: 10100 ⊕ 10010 = 00110 = I *K*=_: 01110 ⊕ 00100 = 01010 = R

K=O: 11100 ⊕ 11000 = 00100 = *K*=E: 01011 ⊕ 00001 = 01010 = R

K=R: 11010 ⊕ 01010 = 10000 = T *K*=S: 00110 ⊕ 00101 = 00011 = A

K=I: 00000 ⊕ 00110 = 00110 = I Luego *M* = MI TIERRA.

También se puede representar un cifrador de Vernam orientado a caracteres. En este caso la operación de cifra se realiza a través de desplazamientos módulo 27, como si se tratase de un cifrador monoalfabético, con una secuencia de clave compuesta por números aleatorios *NA* = k_i, como se muestra en los siguientes pasos.

M	= C	I	F	R	A	D	O	R	D	E	V	E	R	N	A	M
M_i	= 2	8	5	18	0	3	15	18	3	4	22	4	18	13	0	12
k_i	= 73	12	39	81	07	28	95	52	30	18	32	29	47	20	07	62
M_i+k_i	= 75	20	44	99	7	31	110	70	33	22	54	33	65	33	7	74
C	= U	T	Q	R	H	E	C	P	G	V	A	G	L	G	H	T

Los valores utilizados para la secuencia de clave en el ejemplo anterior están comprendidos entre 00 y 99, si bien pueden reducirse mod 27 y, por tanto, trabajar sólo con el CCR (27). Del ejemplo anterior se puede destacar un aspecto interesante de un cifrador de Vernam: en el criptograma aparecen caracteres iguales que provienen del cifrado de caracteres distintos del texto en claro, al igual que en todos los sistema polialfabéticos, como es el caso de las letras D, E y N que se cifran como el elemento G. Ahora bien, además de utilizar los 27 posibles alfabetos dependiendo del valor de la clave, al ser ésta aleatoria y carecer de periodicidad alguna, hace imposible cualquier tipo de ataque conociendo únicamente el criptograma. Si la secuencia aleatoria de clave usada luego se destruye, entonces el secreto es perfecto.

2.3.1.2 Justificación de uso del cifrado de Vernam

OTP (One Time Pad) También conocido como el cifrado de Vernam, sería técnicamente hablando el algoritmo más seguro.

La llave puede ser de cualquier tamaño y el algoritmo nunca estaría expuesto a tener vulnerabilidades debido a que la llave puede llegar a ser tan larga como los datos y también se debe considerar que la llave debe ser usada sólo una vez [17].

2.3.2 Algoritmos Asimétricos

Los algoritmos de llave pública, o algoritmos asimétricos, han demostrado su interés para ser empleados en redes de comunicación inseguras. Introducidos por Whitfield Diffie y Martin Hellman a mediados de los años 70, su novedad fundamental con respecto a la criptografía simétrica es que las claves no son únicas, sino que forman pares. Hasta la fecha han aparecido multitud de algoritmos asimétricos, la mayoría de los cuales son inseguros; otros son poco prácticos, bien sea porque el criptograma es considerablemente mayor que el mensaje original, bien sea porque la longitud de la clave es enorme. Se basan en general en plantear al atacante problemas matemáticos difíciles de resolver.

Los algoritmos asimétricos emplean generalmente longitudes de clave mucho mayores que los simétricos. Por ejemplo, mientras que para algoritmos simétricos se considera

segura una clave de 128 bits, para algoritmos asimétricos si exceptuamos aquellos basados en curvas elípticas se recomiendan claves de al menos 1024 bits. Además, la complejidad de cálculo que comportan estos últimos los hace considerablemente más lentos que los algoritmos de cifrado simétricos. En la práctica los métodos asimétricos se emplean únicamente para codificar la clave de sesión (simétrica) de cada mensaje o transacción particular [8].

Los algoritmos asimétricos poseen dos claves diferentes en lugar de una, *Kp* y *KP*, denominadas clave privada y clave pública. Una de ellas se emplea para codificar, mientras que la otra se usa para decodificar. Dependiendo de la aplicación que le se de al algoritmo, la clave publica será la de cifrado o viceversa. Para que estos criptosistemas sean seguros también ha de cumplirse que a partir de una de las claves resulte extremadamente difícil calcular la otra.

La figura 3 muestra que cuando *A* tiene el mensaje *m* y quiere enviárselo a *B*; este ultimo envía a *A* su clave pública, *KP*; *A* codifica el mensaje m y envía a *B* el criptograma $E_{KP}(m)$; luego *B* decodifica el criptograma empleando la clave privada *Kp*.

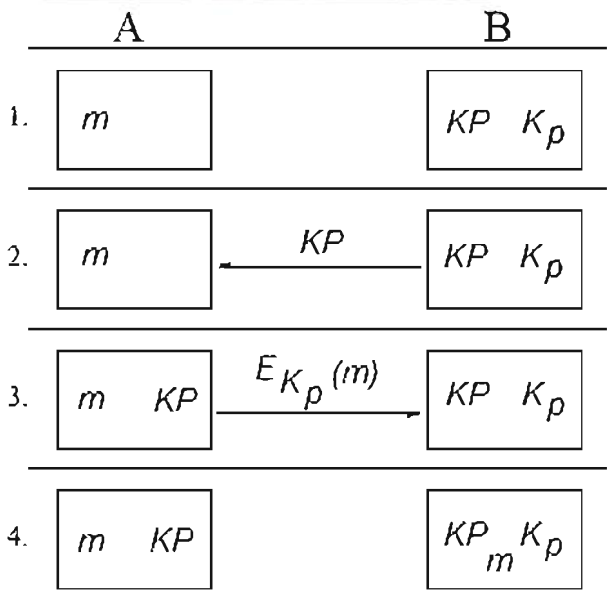

Figura 3. Transmisión de información con algoritmos asimétricos, fuente [9]

2.3.2.1 RSA

El RSA, llamado así por las siglas de sus creadores (*Rivest, Shamir y Adelman*), es el algoritmo de clave pública más popular. El algoritmo se puede usar para encriptar comunicaciones, firmas digitales e intercambio de claves.

La clave es de tamaño variable, generalmente se usan claves entre 512 y 2048 bits en computadores personales, para los equipos móviles celulares el tamaño es diferente pues es limitado por el procesador y los tipos de datos que puedan manejarse (máximo 32 bits). Las claves más grandes aumentan la seguridad del algoritmo pero disminuyen su eficiencia y generan más texto cifrado. Los bloques de texto en claro pueden ser de cualquier tamaño, siempre que sea menor que la longitud de la clave. Los bloques de texto cifrado generados son del tamaño de la clave.

La clave pública del algoritmo tiene la forma (e, n), donde e es el exponente y n el módulo. La longitud de la clave es igual al número de bits de n. El *módulo* se obtiene multiplicando dos números primos grandes, p y q. Los números se seleccionan aleatoriamente y se guardan en secreto. La *clave privada* tiene la forma (d, n), donde d es el producto inverso de e modulo (p-1)(q-1) (es decir, (ed - 1) es divisible por (p -1) (q-1)). El cálculo de d a partir de p y q es sencillo, pero es computacionalmente imposible calcular d sin conocer p y q para valores grandes de n, ya que obtener sus valores es equivalente a factorizar n, que es un problema intratable para un computador. El funcionamiento del algoritmo es como se explica en Algoritmo 4

Paso1: Para encriptar un mensaje un usuario calcula $c=m^e$ módulo n, donde m es el texto en claro, c es el texto cifrado y (e, n) es la clave pública del destinatario.

Paso1: Para decriptar el mensaje el destinatario calcula c^d modulo n = $(m^e)^d$ modulo n = m^{ed} modulo n = m, donde (d, n) es la clave privada del destinatario. Hay que indicar que la última sustitución es posible por el modo en que hemos escogido los números, ya que d es el producto inverso de e modulo n, por lo que m^{ed}= m.

Algoritmo 4. Algoritmo RSA, fuente [9]

El algoritmo es lento, ya que emplea operaciones matemáticas que tienen un coste elevado y trabaja con claves de gran tamaño. Parte del problema está en la elección del exponente *e*, ya que un exponente de 512 bits escogido aleatoriamente precisa 768 multiplicaciones en promedio. Para solucionarlo se suelen escoger los valores 3 ó 65537, que precisan 3 y 17 multiplicaciones respectivamente. La elección de un exponente fijo no disminuye la seguridad del algoritmo si se emplean esquemas de criptografía de clave pública adecuados, como por ejemplo el relleno de mensajes con bits aleatorios [8].

Adicionalmente, el uso de exponentes fijos hace que la encriptación sea más rápida que la decriptación y la verificación más rápida que la firma. Esta última característica es incluso deseable, ya que un usuario firma una vez un mensaje pero es posible que la firma se valide muchas veces.

Comparado con los sistemas de cifrado simétrico como el DES, el algoritmo de RSA es 100 veces más lento en software y de 1000 a 10000 veces más lento en hardware.

A continuación se detalla de forma resumida el algoritmo RSA.

Paso 1: $n = pq$ donde p y q son primos distintos.

Paso 2: phi, $\varphi = (p-1)(q-1)$

Paso 3: $e < n$ tal que $MCD(e, \text{phi})=1$

Paso 4: $d = e^{\wedge -1} \bmod \text{phi}$.

Paso 5: $c = m^e \bmod n$, $1<m<n$.

Paso 6: $m = c^d \bmod n$.

Algoritmo 5. Resumen del algoritmo RSA

2.3.2.2 Justificación de uso del algoritmo RSA

Es el algoritmo asimétrico más usado hoy en día. Está licenciado por la empresa del mismo nombre. Al contrario que los algoritmos simétricos, los algoritmos asimétricos necesitan que la llave sea más larga para proveer la misma seguridad [17].

2.3.3 Órdenes de complejidad

La familia $O(f(n))$ define un Orden de Complejidad. Se elige como representante de este Orden de Complejidad a la función $f(n)$ más sencilla que pertenece a esta familia.

Las funciones de complejidad algorítmica más habituales en las cuales el único factor del que dependen es el tamaño de la muestra de entrada n, ordenadas de mayor a menor eficiencia son:

$O(1)$: Complejidad constante. Cuando las instrucciones se ejecutan una vez.

$O(\log n)$: Complejidad logarítmica. Esta suele aparecer en determinados algoritmos con iteración o recursión no estructural, ejemplo la búsqueda binaria.

$O(n)$: Complejidad lineal. Es una complejidad buena y también muy usual. Aparece en la evaluación de bucles simples siempre que la complejidad de las instrucciones interiores sea constante.

$O(n \log n)$: Complejidad cuasi-lineal. Se encuentra en algoritmos de tipo divide y vencerás como por ejemplo en el método de ordenación quicksort y se considera una buena complejidad. Si n se duplica, el tiempo de ejecución es ligeramente mayor del doble.

$O(n^2)$: Complejidad cuadrática. Aparece en bucles o ciclos doblemente anidados. Si n se duplica, el tiempo de ejecución aumenta cuatro veces.

$O(n^3)$: Complejidad cúbica. Suele darse en bucles con triple anidación. Si n se duplica, el tiempo de ejecución se multiplica por ocho. Para un valor grande de n empieza a crecer dramáticamente.

$O(n^a)$: Complejidad polinómica ($a > 3$). Si a crece, la complejidad del programa es bastante mala.

$O(2^n)$: Complejidad exponencial. No suelen ser muy útiles en la práctica por el elevadísimo tiempo de ejecución. Se dan en subprogramas recursivos que contengan dos o más llamadas internas.

La figura 4 muestra una comparación entre los diferentes órdenes de complejidad expuestos y la relación de número de instrucciones versus la entrada n

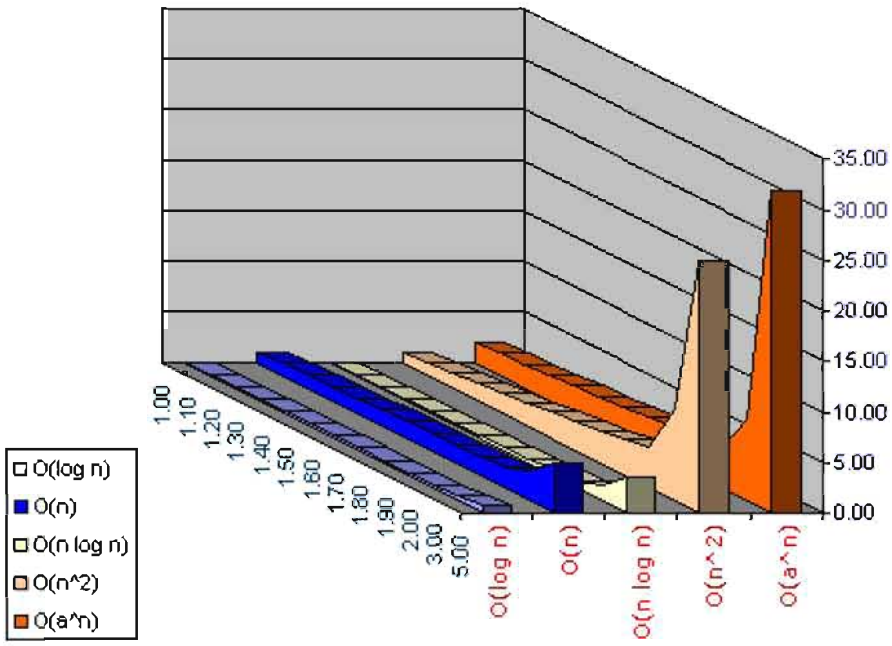

Figura 4. Comparación de órdenes de complejidad, fuente [21]

El cifrado, que utiliza la clave pública (RSA), tiene una complejidad de $O(n^2)$, el descifrado $O(n^3)$ y la generación de claves $O(n^4)$, en donde n es la longitud en bits del módulo.

Mientras que el cifrado Vernam utiliza una complejidad de $O(1)$ es decir constante en el cifrado y descifrado, cabe hacer notar que en esta investigación se utiliza un orden de complejidad $O(n)$ debido a que la cantidad de letras que tenga el mensaje SMS podrá

variar y la limitación de los equipos móviles celulares condicionan a trabajar bajo trozos de 32 bits.

2.4 Algoritmos híbridos

La técnica de hibridación en algoritmos criptográficos se basa en la combinación fuerte de algoritmos y adaptación de los mismos a propósitos que demanden dicha hibridación que para el caso de esta tesis se recurre a dos algoritmos uno de clave privada (Vernam) y otro de clave pública (RSA) los mismos que trabajando a 32 bits presentan una fortaleza adecuada para equipos móviles.

2.5 J2ME

Java2 Micro Edition ha sido creado para adaptarse a las características de los nuevos dispositivos inalámbricos tales como teléfonos móviles y PDAs. Consecuentemente existirán diferencias con la versión estándar de Java destinada a PCs. Algunas de las principales diferencias son las siguientes:

2.5.1 Tipos de datos

J2ME soporta un subconjunto de los tipos de datos de J2SE, los tipos float y double no son soportados por dos razones: la mayoría de dispositivos CLDC no tienen unidad de coma flotante y en segundo lugar es una operación muy costosa.

El máximo tamaño de dato que soporta J2ME es tipo Long de 32 bits.

2.5.2 Características para compilación con tecnología J2ME

Para trabajar con J2ME es necesario conocer algunas características de esta tecnología:

- Preverificación (on-line y off-line). Al contrario que en J2SE dónde se realiza la verificación del código en tiempo de ejecución, en J2ME parte de la verificación se realiza off-line, es decir, fuera del dispositivo. Esto tiene la finalidad de reducir la carga de la máquina, llevando a cabo el resto de la verificación on-line.

- Manifiesto. Es un archivo que sirve cuando se empaquetan varias aplicaciones J2ME (MidLets) en un paquete (MidLet Suite) es necesario incluir un fichero de texto denominado manifiesto. La finalidad del manifiesto es describir el contenido del fichero .JAR con información tal como el nombre, versión, vendedor, etc. también se incluye en este fichero una entrada por cada MIDlet que lo compone.

Un ejemplo de manifiesto podría ser el siguiente:

MIDlet-1: Lista1, /icons/Lista1.png, Lista1
MIDlet-2: Texto1, /Icons/Texto1.png, Texto1
MIDlet-Name: Ejemplos
MIDlet-Vendor: Sun Microsystems
MIDlet-Version: 1.0

- Descriptor Aunque este fichero comparte el mismo formato que el del manifiesto, su finalidad es totalmente diferente. El objetivo de este fichero es proporcionar la información requerida por el Application Management Software (programa que gestiona las descargas de aplicaciones entre otras cosas) para cerciorarse de que el MIDlet suite es adecuado para el dispositivo en el que queremos instalarle. Su extensión es .JAD y al contrario que el manifiesto, éste no se incluye en el paquete.

2.5.3 Características principales del lenguaje java en J2ME

- La *K* Virtual Machine es una máquina virtual java altamente portable y compacta pensada y diseñada para funcionar en dispositivos con recursos limitados y con conexión a red. El objetivo de la tecnología KVM es el de crear la JVM más pequeña y completa que mantuviese los aspectos más importantes del lenguaje de programación Java y que funcionase en dispositivos con procesadores de 16 ó 32 bits y unas capacidades de unos pocos cientos de KiloBytes (originalmente 128 Kbytes de memoria).

- Nueva librería gráfica. Mediante el paquete LCDUI J2ME define un nuevo conjunto de clases para la creación de interfaces gráficas. Estas clases están

adaptadas a dispositivos con memorias muy limitadas y pantallas de tamaño reducido.

- Desaparición del main. Al contrario que las aplicaciones de la edición estándar de Java, en J2ME no existe una función main como punto de entrada para la ejecución del programa. El equivalente a este main sería el método *start app* (será explicado posteriormente).

- Ausencia del recolector de basura. Con el objetivo de reducir la utilización de memoria, desaparece el recolector de basura en J2ME siendo necesario eliminar de forma explícita todos aquellos elementos que no vayan a ser utilizados más.

2.6 Teoría de la mensajería SMS

Servicio de mensajes cortos (Short Messaging Service), es una tecnología que da soporte al envío y recepción de mensajes cortos de texto en dispositivos móviles. Otra característica interesante de SMS es que emplea una mensajería unificada, lo que te permite acceder a mensajes de voz, al correo electrónico y faxes desde un dispositivo móvil.

2.6.1 Arquitectura del sistema inalámbrico SMS

Este sistema puede ser visto en una arquitectura de 3 capas, consistiendo de una capa de interfase, implementación y transporte expuesto en [12].

2.6.1.1 Capa de Interfase

Constituye un genérico conjunto de interfases, independiente de algún protocolo de mensaje. Estas interfases proveen la definición básica de un mensaje, define la funcionalidad básica de envío y recepción de este, y provee un mecanismo para que una aplicación MIDlet sea notificada al convertirse en mensaje.

2.6.1.2 Capa de implementación

Contiene clases las cuales implementan cada capa de interfase para acceder a la mensajería inalámbrica SMS sobre un dispositivo móvil GSM. En una instancia, desde el punto de vista del SMS, esta capa provee una implementación de conexión para los

mensajes SMS's como una implementación de un mensaje SMS con texto o atributos binarios.

La implementación de la capa también realiza segmentación y concatenación de los mensajes para la capa inferior del protocolo. El MIDlet puede entonces especificar el número de segmentos de un mensaje debiendo ser roto dentro un MessageConnection.

2.6.1.3 Capa de transporte

Contiene las clases que son la implementación actual del protocolo que lleva los mensajes al dispositivo móvil.

El mecanismo de las 3 capas se observa en la figura 5.

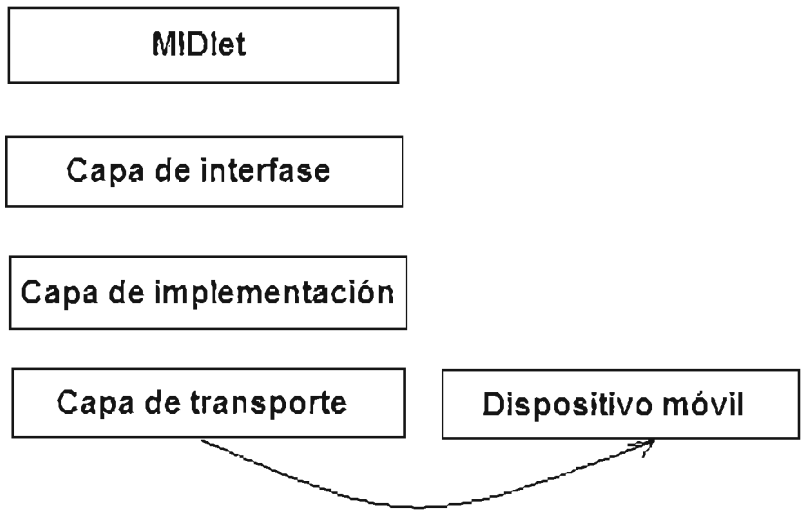

Figura 5. Arquitectura del sistema de mensajería inalámbrica, fuente [12]

2.6.2 El API de la mensajería SMS

El API es definido por el paquete javax.wireless.messaging, el cual define todas las interfaces para el envío y recepción de mensajes SMS, a continuación la lista de interfaces

2.6.2.1 Message

Esto proporciona la definición básica de un mensaje, que actúa la dirección, una carga útil, y banderas para enviar y para bloquear un mensaje. Este es un superinterface para TextMessage, un objeto mensaje con el atributo de carga útil de texto, y a un BinaryMessage, un objeto mensaje con el atributo d binaria de la carga útil. La estructura de un mensaje se demuestra en La figura 6.

Figura 6. Estructura de un mensaje, Fuente [12]

2.6.2.2 MessageConnections

Esto proporciona la funcionalidad básica de recibir y de enviar mensajes. Contiene un método para enviar y recibir mensajes, un método de la fábrica para crear nuevos objetos del mensaje, y un método que calcule el número de los segmentos del protocolo subyacente que son necesarios para enviar un objeto especificado del mensaje. Esta clase es instanciada por una llamada a Connector.open ().

En una conexión del modo del cliente, los mensajes pueden ser enviados solamente. Una conexión del modo del cliente es creada pasando una secuencia que identifica una dirección de destinación al método de Connector.open (). Este método vuelve un objeto de MessageConnection.

clientConn = (MessageConnection) Connector.open ("sms: /+18643630999: 5000");

En una conexión del modo del servidor, los mensajes pueden ser enviados o ser recibidos.

Una conexión del modo del servidor es creada pasando una secuencia que identifique un punto final (identificador dependiente del protocolo, por ejemplo, un número de acceso) en el anfitrión local al método de: Connector.open ().

Seguidamente se muestra la línea de código que se utiliza para la programación en J2ME:

serverConn = (MessageConnection) Connector.open ("sms: /: 5000");

2.6.2.3 MessageListener

Proporciona un mecanismo básico para notificar el uso de MIDlet de un mensaje entrante. Permite que un MIDlet reciba un servicio repetido cuando los nuevos mensajes están disponibles para ser leído.

2.6.3 Servicios del API

El paquete de com.sun.midp.io.j2me.sms proporciona un API para el sistema de mensajería corta y permite que el MIDlet tenga acceso a la funcionalidad del SMS en un dispositivo del móvil del GSM.

Los componentes principales del paquete son: MessageObject y protocol, los cuales dan el soporte para enviar y recibir mensajes SMS.

2.6.3.1 MessageObject

MessageObject es la puesta en práctica de un mensaje de SMS. En la capa de la puesta en práctica, el javax.wireless.messaging. El interfaz de mensaje se pone en ejecución como almacenador intermediario. El MessageObject maneja la creación de los almacenadores intermediarios de mensaje y de las operaciones de la entrada-salida fuera de un almacenador intermediario. Además, tiene dos subclases: TextObject y BinaryObject. Estas clases ponen un mensaje SMS en ejecución con una carga útil, sea texto o binario [12].

2.6.3.2 Protocolo

Pone la conexión del mensaje a bajo nivel transportando el un mecanismo requerido para transmitir un mensaje de SMS. En el proceso, comprueba todos los parámetros de la configuración del runtime y maneja las excepciones relacionadas con la sintaxis de URL inválido, violaciones de la seguridad, violaciones de la entrada-salida, y

argumentos ilegales. El protocolo también maneja el envío y recepción de mensajes usando una conexión del datagrama o del puerto serial.

2.7 Comunicación inalámbrica

La necesidad de comunicación de un celular con un computador requiere un tipo de comunicación que como alternativa se tiene a la tecnología Bluetooth ideal para periféricos de ordenador (ratón, teclado, manos libres,...), dispositivos móviles (teléfonos móviles, PDAs, Pocket PCs,...) que conjuntamente a los comandos AT logran el servicio de comunicación dedicado a trabajar con la mensajería.

Con la desaparición del tan usado puerto serie en los computadores personales y también de escritorio se llega a la necesidad de utilizar un puerto serie inalámbrico que resulta como mejor opción la comunicación vía Bluetooth ya que las configuraciones necesarias y recolección de datos son realizadas por comandos AT en comunicación serial.

2.7.1 Comandos AT

Los caracteres AT (*attention command*) hacen referencia a un comando para llamar la atención y decirle al modem *Modulator-Demodulator* que realice una cierta acción, estos comandos son necesarios para el control del modem integrado del teléfono. Ya sea para control infrarrojo, inalámbrico o por cable.

A continuación se presenta la lista de comandos AT en la tabla 1 y 2 necesarios para la recepción y envío de mensajería SMS obtenido de [13].

Tabla 1. Comandos AT para manejo de SMS en modo PDU, Fuente [13]

Comando	Descripción
AT+CMGL	Lista mensajes
AT+CMGR	Leer mensaje
AT+CMGS	Envía mensaje
AT+CMGW	Escribir mensaje a memoria

Tabla 2. Lista de comandos AT para manejo de SMS en modo texto, Fuente [13]

Comando	Descripción
AT+CSMS	Selecciona el servicio del mensaje
AT+CPMS	Mensaje almacenado preferido
AT+CMGF	Formato del mensaje
AT+CSCA	Dirección del cetro de servicio
AT+CSMP	Fijar parámetros de modo texto
AT+CSDH	Mostrar parámetros de modo texto
AT+CSCB	Selecciona tipos de mensajes
AT+CSAS	Salvar configuración
AT+CRES	Restaura configuración
AT+CNMI	Indicaciones del Nuevo mensaje a TE
AT+CMGL	Lista de mensajes
AT+CMGR	Leer mensaje
AT+CMGS	Envía mensaje
AT+CMSS	Enviar mensaje almacenado
AT+CMGW	Escribir mensaje a memoria
AT+CMGD	Borrar mensaje

CAPÍTULO 3

MARCO PRÁCTICO

3.1 Diagrama arquitectónico del sistema

La arquitectura mostrada en la figura 7 muestra los bloques que intervienen en el proceso y distribución de elementos necesarios para la aplicación del algoritmo híbrido de encriptación de clave pública y privada.

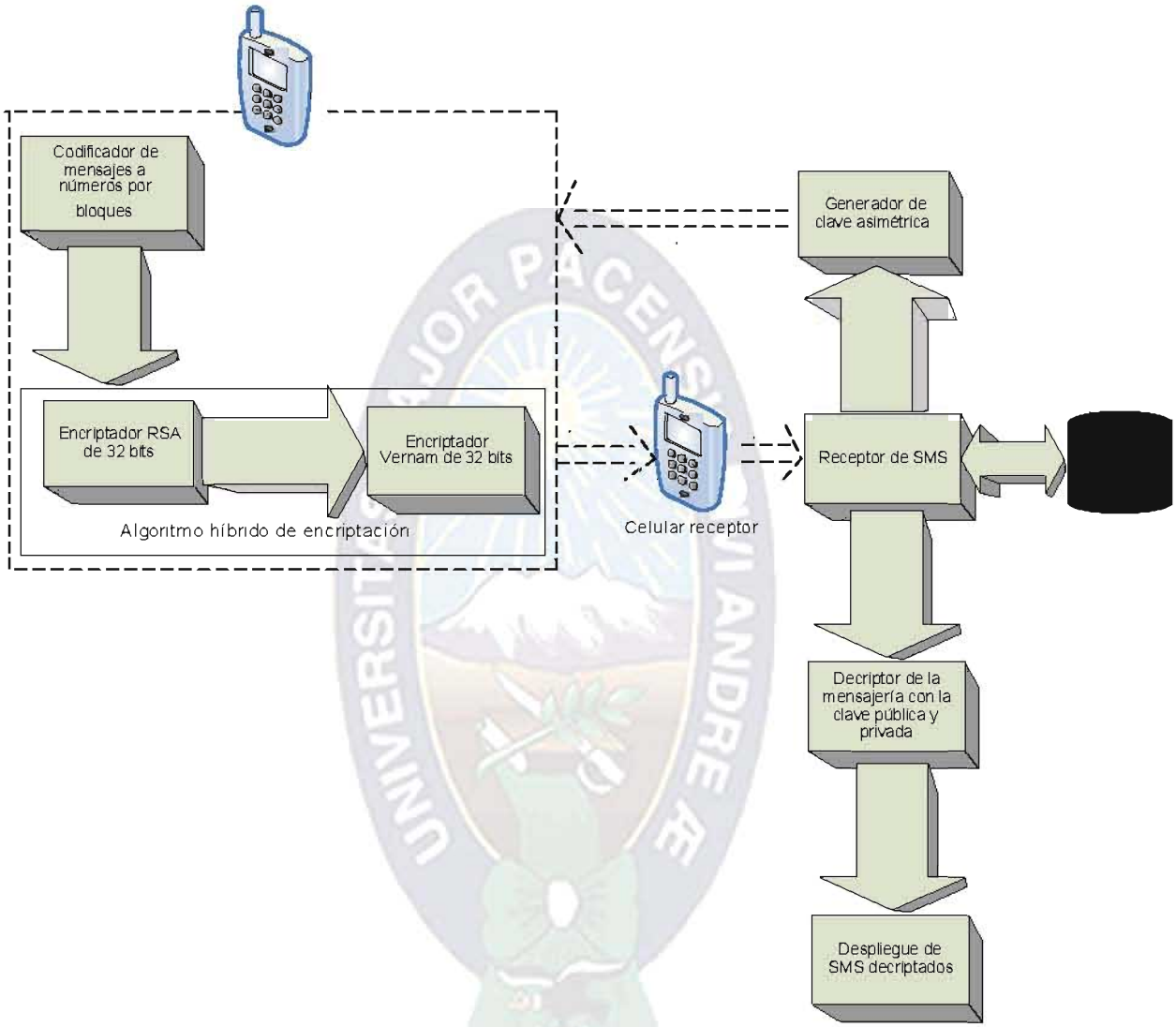

Figura 7. Arquitectura que aplica el algoritmo de encriptación híbrida de clave pública y privada

3.2 Codificador de mensajes a números por bloques

Este bloque se encarga de tomar subcadenas del mensaje y transformarlas a números.

A continuación se explica el proceso de transformación de caracteres a números.

Se reemplaza la letra por un equivalente numérico según su posición en el alfabeto español incluyendo los números del 0 al 9 y un caracter de *espacio* lo cual hace 38 caracteres por lo cual se trabaja con modulo 38, en caso de un alfabeto inglés se suprime la letra "Ñ" y trabaja con modulo 37.

Se toma bloques de 3 caracteres en base 38 usando la codificación mostrada en la tabla 3.

Tabla 3. Codificación del alfabeto español

CARACTER	CODIGO	CARACTER	CODIGO	CARACTER	CODIGO
A	0	Ñ	14	1	28
B	1	O	15	2	29
C	2	P	16	3	30
D	3	Q	17	4	31
E	4	R	18	5	32
F	5	S	19	6	33
G	6	T	20	7	34
H	7	U	21	8	35
I	8	V	22	9	36
J	9	W	23	Espacio	37
K	10	X	24	-	-
L	11	Y	25	-	-
M	12	Z	26	-	-
N	13	0	27	-	-

Un ejemplo de cómo se realiza la transformación para las tres primeras letras del texto "POSTGRADO EN INGENIERIA DEL SOFTWARE" se muestra a continuación:

Desglosando en su equivalente la primera terna del mensaje de acuerdo a la tabla 3 se tiene:

$POS = 16 \times 38^2 + 15 \times 38^1 + 19 \times 38^0 = 23693$

3.3 Generador de clave pública

El modelo de encriptación híbrida propuesto requiere de un generador de claves que es integrado como software en un computador personal, el cual tiene por misión generar 2

números aleatorios coprimos que representan a la clave publica y también otro para la clave privada siguiendo los pasos siguientes.

Paso 1: $N = p * q$ donde p y q son dos números coprimos distintos

Paso 2: $\varphi = (p-1)(q-1)$

Paso 3: $e<n$ tal que $MCD(e,phi)=1$

Paso 4: $d=e^{-1} \bmod phi$

Algoritmo 6. Generación de claves

Por tanto el computador deberá generar y publicar una cadena con el siguiente formato "*n-e*" que la denominaremos de aquí en adelante clave pública; y "*d*" no necesitará ser publicada sólo servirá en la decriptación después de la recepción de los mensajes.

Es importante generar números aleatorios y no así seudo-aleatorios, por lo cual se utiliza la entrada del micrófono del computador para obtener y generar números aleatorios

3.4 Encriptador RSA de 32 bits

Este nivel de encriptación utiliza la subcadena publicada "*n*" obtenida de la cadena "*n-e*" siguiendo los siguientes pasos

Para la encriptación se debe realizar la potenciación del mensaje codificado en módulo *n* que para el caso es 38.

$$c = m^e \bmod n, \text{ donde } 1<m<n.$$

Considerando e=13, n=55919 y phi=55440

Luego el número resultante de la primera terna encriptada es:

$c=23693^{13} \bmod 55919=17959$

3.5 Encriptador Vernam de 32 bits

Este encriptador basado en el OR Exclusivo es realizado en forma secuencial al cifrador RSA de 32 bits.

Siguiendo con el mensaje anteriormente usado:

"POSTGRADO EN INGENIERIA DEL SOFTWARE"

Se aplica a continuación el modelo de encriptación.

El numero 17959 resultado de la primera encriptación será operado en un OR exclusivo con la clave privada definida por el usuario para el código Vernam.

Transformando el código se obtiene: $17959_d = 100011000100111_b$

Ahora se necesita determinar una clave privada que sea compartida tanto por el receptor como el emisor.

Ejemplo: "**RODOLFITO**".

Realizando la transformación con la tabla 1 se tiene:

RODOLFITO = 18, 15, 3, 15, 11, 5, 8, 20, 15 (en decimal)

Y en binario: 10010, 1111, 11, 1111, 1011, 101, 1000, 10100, 1111

Uniendo todos los paquetes de bits se tiene la siguiente clave:

Clave privada: $100101111111111101110110001010011111_b$

Se puede notar que la cantidad de letras no determina la cantidad de bits por lo cual el software advierte al usuario acerca de la cantidad de bits que tiene su clave.

Esta clave tiene 35 bits lo cual garantiza que mejora los 32 bits predefinidos inicialmente, pero se debe tomar en cuenta que la clave no deberá exceder en cantidad de bits al mensaje, es decir:

Cantidad de bits (clave privada compartida)<= Cantidad de bits (mensaje)

Esta relación se debe a que en el caso en que sea mayor la cantidad de bits de la clave compartida a la del mensaje, puede no llegar a usarse los bits sobrantes y el mensaje sería encriptado en una cantidad menor a 32 bits lo cual incrementaría la vulnerabilidad del modelo.

Seguidamente es necesario emparejar las longitudes de ambos textos, es decir la clave privada compartida deberá tener la misma cantidad de bits que el mensaje RSA-32 expresado en binario, para lo cual se debe repetir la clave tantas veces sea necesaria hasta emparejar dichas longitudes.

Clave privada:

$10010111111111110111011000101001111_b$

Mensaje RSA-32:

$10001100001001110011001101010111000001110010100010000010000010000000001$
$10110011000011101011110001100101011110111011100100010111111011001010101$
$01010101010110010111011011100100111100110010110011111_b$

Clave privada repetida:

$10010111111111110111011000101001111100101111111111101110110001010011111$
$10010111111111110111011000101001111100101111111111101110110001010011111$
$10010111111111110111011000101001111100101111111111110_b$

Seguidamente se requiere realizar la operación lógica OR exclusivo entre el mensaje y la clave privada compartida repetida y ambas expresadas en sus equivalentes binarios.

Mensaje RSA-32:

$10001100001001110011001101010111000001110010100010000010000010000000001$
$10110011000011101011110001100101011110111011100100010111111011001010101$
$01010101010110010111011011100100111100110010110011111_b$

⊕ (OR EXCLUSIVO)

Clave privada repetida:

$10010111111111110111011000101001111100101111111111110111011000101001111$
$10010111111111110111011000101001111100101111111111110111011000101001111$
$10010111111111110111011000101001111100101111111111110_b$

Mensaje encriptado:

$00011011110110000100010101111110111101011101011101100110101010011 10$
$00100100111100011100101001001100100010001001001101100101001100110110 10$
$11000010100101100000000001100110100000001110100110001$

Para el envío del SMS es necesario reducir la cantidad de texto a su mínima expresión dado que el mensaje encriptado crece a medida que los niveles de encriptación se aumentan, por tal razón se codifica a hexadecimal tomando grupos de 4 bits.

Por tanto el ordenando en grupos de 4 bits el mensaje quedaría de la siguiente forma:

0001_b, 1011_b, 1101_b, 1000_b, 0100_b, 0101_b, 0111_b, 1110_b, 1111_b, 0101_b, 1101_b, 0111_b, 0110_b, 1100_b, 1101_b, 0101_b, 0011_b, 1000_b, 1001_b, 0011_b, 1100_b, 0111_b, 0010_b, 1001_b, 0011_b, 0010_b, 0010_b, 0010_b, 0100_b, 1101_b, 1001_b, 0100_b, 1100_b, 1101_b, 1010_b, 1100_b, 0010_b, 1001_b, 0110_b, 0000_b, 0000_b, 1100_b, 1101_b, 0000_b, 0001_b, 1101_b, 0011_b, 0001_b

Convirtiendo el código binario a hexadecimal queda el siguiente texto encriptado:

1BD8457EF5D76CD53893C72932248D94CDAC2A600CD01D31

3.6 Decriptación del mensaje

Partiendo de: 1BD8457EF5D76CD53893C72932248D94CDAC2A600CD01D31

La decriptación sigue un proceso matemático en la primera etapa (decriptación Vernam) utilizando la operación OR exclusiva entre el mensaje encriptado y la clave privada compartida.

Transformando cada dígito a código binario se tiene:

0001_b, 1011_b, 1101_b, 1000_b, 0100_b, 0101_b, 0111_b, 1110_b, 1111_b, 0101_b, 1101_b, 0111_b, 0110_b, 1100_b, 1101_b, 0101_b, 0011_b, 1000_b, 1001_b, 0011_b, 1100_b, 0111_b, 0010_b, 1001_b, 0011_b, 0010_b, 0010_b, 0010_b, 0100_b, 1101_b, 1001_b, 0100_b, 1100_b, 1101_b, 1010_b, 1100_b, 0010_b, 1001_b, 0110_b, 0000_b, 0000_b, 1100_b, 1101_b, 0000_b, 0001_b, 1101_b, 0011_b, 0001_b

Por tanto:

00011011110110000100010101111110111101011101011101101100110101010011100010010011111000111001010010011001000100010010011011001010011001101101010110000101001011000000000110011010000000011101001100001

Utilizando la clave privada compartida "RODOLFITO" se tiene:

RODOLFITO = 18, 15, 3, 15, 11, 5, 8, 20, 15 (en decimal).

Y en binario: 10010, 1111, 11, 1111, 1011, 101, 1000, 10100, 1111

Repitiendo la clave hasta igualar en tamaño al mensaje se tiene:

10010111111111110111011000101001111100101111111111110111011000101001111100101111111111111011101100010100111110010111111111111011101100010100111110010111111111110$_b$

Aplicando el *O* exclusivo entre el mensaje y la clave se tiene:

10001100001001110011001101010111000001110010100010000010000100000000011011001100001110101111000110010101111011110111001000101111111011001010101010101100101110111011100100111100110010110011111$_b$

Tomando grupos de 16 bits y convirtiendo a su equivalente decimal se tiene:

35879-13143-1832-33296-1740-15089-38383-29231-55637-21911-28239-13007

Realizando la exponenciación 35879^{34117} mod 55919 =23693

De igual forma se procede con el resto de los números para llegar a obtener:

23693;29126;129-53593;53745;8829;11722;11589;4495;54165;8003;688

Transformando los números decimales a base 38 se obtiene la siguiente cadena:

16-15-19;20-6-18;0-3-15;37-4-13;37-8-13;6-4-13;8-4-18;8-0-37;3-4-11;37-19-15;5-20-23;0-18-4

Utilizando la Tabla 3 de codificación se obtiene para la primera terna: 16-15-19=POS

De igual manera se procede con el resto de ternas numéricas para encontrar el mensaje, seguidamente se muestra el texto decriptado.

POS-TGR-ADO- EN- IN-GEN-IER-IA -DEL- SO-FTW-ARE

3.7 Generador de clave asimétrica

Para generar los números primos *p* y *q*, se genera números aleatoriamente de una longitud de *b*/2 donde *b* se requiere que tenga una longitud de *n*, se coloca el bit más bajo a uno (esto asegura que el número sea impar) y a uno los dos bits mas altos (esto asegura que el bit mas alto de *n* también sea fijo).

A partir de los números primos se debe generar la clave pública y privada.

3.8 Modelo híbrido

La conjunción del algoritmo Vernam y RSA-32 conforman la encriptación híbrida desde el paso 1 al 6 mismos que son referidos al algoritmo RSA-32 utilizando paquetes de 3 letras, el paso 6 y 7 es dedicado al algoritmo Vernam y su aplicación sobre la anterior encriptación RSA-32. El algoritmo 7 detalla los pasos que comprende el modelo.

3.8.1 Encriptación

Paso 1: Escoger *n* donde $n = p * q$ donde *p*, *q* son coprimos y no deben ser publicados.

Paso 2: Calcular $\varphi = (p-1)(q-1)$ Este valor debe cumplir con el paso 3 para la decriptación.

Paso 3: Escoger $e<n$ tal que $MCD(e, \varphi)=1$ donde e es la clave pública que será publicada

Paso 4: *Calcular* $d=e^{-1} \bmod \varphi$, d es la clave privada que no debe ser publicada

Paso 5: For $i=1$ to longitud del mensaje en pasos de 3

$c_i = m_i^{\wedge e} \bmod n$, donde $1< m_i <n$, donde c_i es un trozo del mensaje encriptado

end for

Paso 6: Convertir c_i en binario

For $i=1$ to longitud del mensaje en pasos de 3

c_i= *convertir_en_binario*(c_i)

end for

Convertir la clave k_i en binario

For $i=1$ to longitud del mensaje en pasos de 3

k_i= *convertir_en_binario*(k_i)

end for

Paso 7: Aplicar el OR exclusivo a cada trozo

For $i=1$ to longitud del mensaje/3

$ME_i = c_i \oplus k_i$, donde c_i es un trozo del mensaje encriptado y k_i la clave privada compartida

end for

Algoritmo 7. Modelo híbrido de encriptación propuesto

3.8.2 Decriptación

Paso 1: OR exclusivo entre el mensaje encriptado y la clave

For $i = 1$ to longitud del mensaje/3

$C_i = ME_i \oplus k_i$

end for

Paso 2: Aplicación del modelo RSA

For $i = 1$ to longitud del mensaje en pasos de 3

$m_i = c_i^{\wedge d} \bmod n$. donde m_i es un trozo de mensaje decriptado

end for

Paso 3: Unir los paquetes encontrados

Algoritmo 8. Modelo híbrido de decritación propuesto

El modelo de decriptación mostrado en el algoritmo 8 muestra que la aplicación del los algoritmos Vernam y RSA deben ser aplicados de forma secuencial respectivamente.

3.8.3 Tiempo de proceso de encriptación

Teniendo la consideración que el software desarrollado bajo la tecnología J2ME es instalado en un celular modelo 6230 cuya frecuencia de trabajo del procesador es de 123 MHz obtenido del anexo B, se muestra el cálculo de tiempo de procesamiento mostrado en la tabla 4.

Tabla 4. Tiempos de proceso de encriptación de mensaje

A	B	C	D	E	F	G

Caso peor	123	1	462	20	9240	75.122	0.0751
Caso mejor	123	60	462	20	554400	4507.3	4.5073

Donde:

A: Frecuencia del celular modelo Nokia 6230.

B: Cantidad de letras enviadas en un mensaje SMS.

C: Número de líneas de código de programa instalado en el celular

D: Ciclos estimados de maquina por instrucción para tecnología RISC.

E: Total de ciclos necesarios para correr el programa, es decir (C x D) .

F: Cociente entre el total de ciclos y la frecuencia del celular, es decir (E/A) expresado en milisegundos.

G: Cociente entre el total de ciclos y la frecuencia del celular, es decir (E/A) expresado en segundos.

3.9 Ejemplo de aplicación

El programa desarrollado con tecnología J2ME fue instalado en un celular Nokia 6230, se presenta las pantallas de la aplicación en la figura 8 y se muestra las casillas con datos reales.

En el ejemplo la figura 8-A y 8-B se muestra que el usuario debe introducir los campos de **Destino, Clave pública, Clave privada** y texto a enviar para luego en (**Opc**.) proceder a la escritura del mensaje y su posterior envío con encriptación.

La figura 8-C se observa la casilla donde se debe introducir el mensaje que será enviado, como ejemplo se utilizará el texto: "POSTGRADO EN INGENIERIA DEL SOFTWARE".

A B C

Figura 8. Pantallas del celular con el software en encriptación

El prototipo implementado en el lado del receptor de mensajería posee una interfaz desarrollada en VB 6.0 y comunicación inalámbrica Bluetooth hacia el celular que recibe los mensajes, a través del cual los mensajes son colectados y decriptados por un computador personal.

El interfaz del software colector y decriptor de mensajes se muestra en la figura 9 y los parámetros necesarios, la clave pública que es conocida por todos, la clave privada compartida conocida por el emisor y el receptor del mensaje; y la clave privada no compartida conocida sólo por el receptor.

En el recuadro superior de la figura 9 se muestra el mensaje puro, es decir, con todos los detalles que llegan del celular como la fecha, hora, remitente y el respectivo mensaje sin decriptación.

En la parte central del formulario el botón "Cortar mensaje" realiza una separación de datos para su respectiva decriptación.

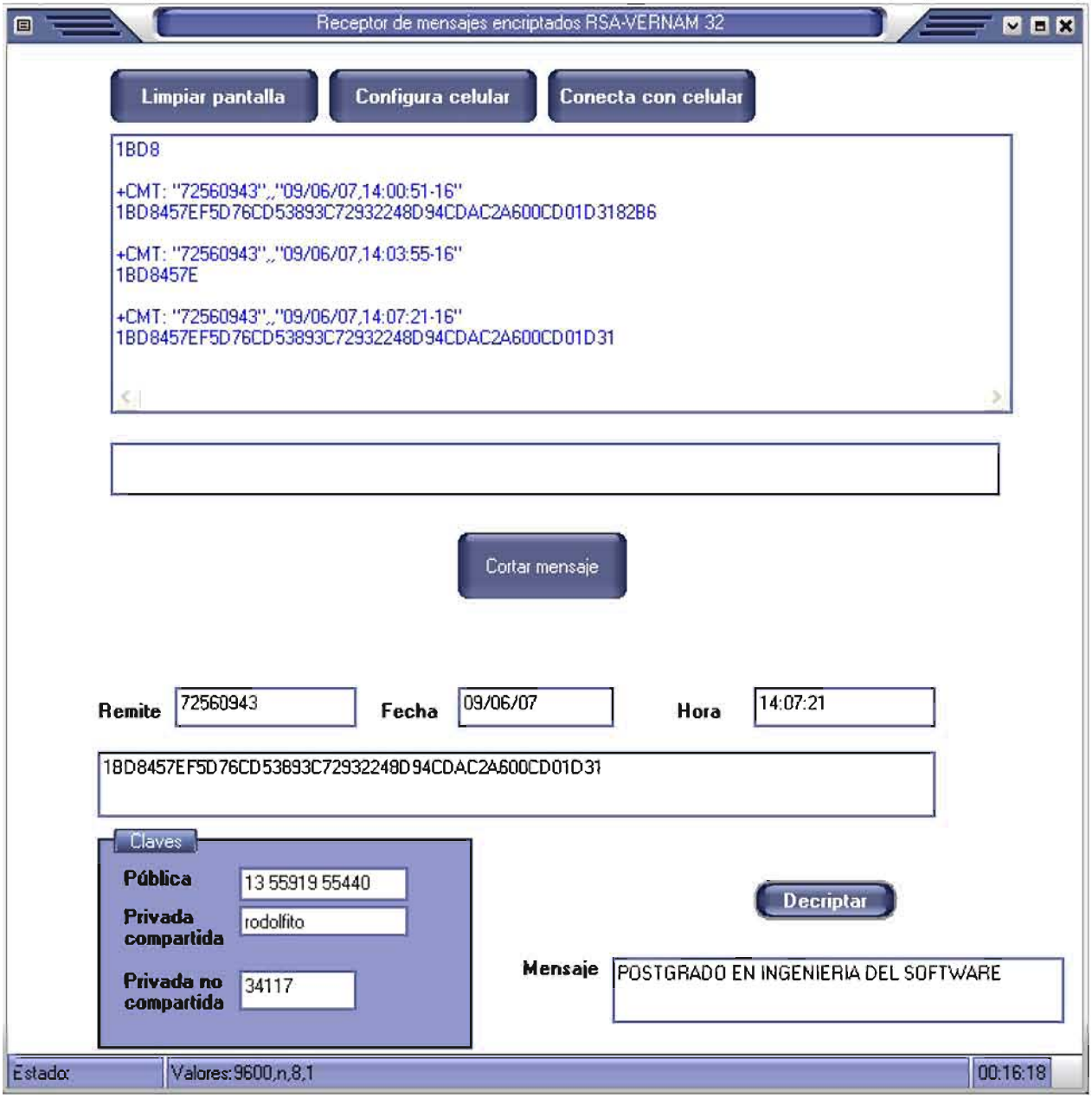

Figura 9. Pantalla del software receptor

CAPÍTULO 4

ANÁLISIS DE RESULTADOS Y VALIDACIÓN DE HIPÓTESIS

4.1 Análisis de la complejidad algorítmica

Por una parte es necesario analizar la potencia de los algoritmos independientemente del desempeño de la máquina que los ejecute e incluso de la habilidad del programador que los codifique.

La consecuencia de hacer seguro al algoritmo hibrido, surge el principal problema: su lentitud. Se debe elegir como clave pública el menor de los dos exponentes a fin de conseguir que el proceso de cifrado sea el más rápido (más que el descifrado, el cual, a su vez es más que el generador de claves).

Sean "$g(n)$" diferentes funciones que determinan el uso de recursos. Lo que se realizara es identificar "familias" de funciones, usando como criterio de agrupación su comportamiento asintótico.

A un conjunto de funciones que comparten un mismo comportamiento asintótico se lo denomina un orden de complejidad. Habitualmente estos conjuntos se denominan O, existiendo una infinidad de ellos.

El cifrado, que utiliza la clave pública, tiene una complejidad de $O(k^2)$ y combinado con el algoritmo Vernam con $O(k)$ se tiene una complejidad de $O(k^3)$, el descifrado RSA tiene $O(k^3)$ y combinado con Vernam $O(k)$ se tiene $O(k^4)$ y la generación de claves públicas $O(k^4)$ es mantenido debido a que sólo afecta al algoritmo RSA, en donde k es la longitud en bits del módulo.

Por tanto el algoritmo hibrido presenta los siguientes órdenes de complejidad y el número de instrucciones que necesitan ser ejecutadas.

Cifrado: si k =32 entonces $O(k^3)$ = 32768 instrucciones.

Descifrado si k =32 entonces $O(k^4)$= 1048576 instrucciones.

Generación de claves públicas si k =32 $O(k^4)$= 1048576 instrucciones.

4.2 Análisis del tiempo de ataque al algoritmo híbrido

La complejidad algorítmica en términos de frecuencia del reloj de un procesador es como sigue:

Tabla 5. Combinaciones posibles

	Base	N° de bits	Posibles combinaciones
Vernam	2	32	4294967296
RSA	2	32	4294967296

Producto combinaciones Vernam-RSA=1.84467E+19

Número de instrucciones necesarias=1.84467E+19

Frecuencia del procesador= 20000000000 Hz

Tabla 6. Tiempo de procesamiento

	Ciclos por instrucción	Tiempo (Segundos)	Tiempo (Años)
Caso peor	15	13835058055	438.706813
Caso mejor	1	922337203.7	29.24712087

La tabla 5 muestra el cálculo de las posibles combinaciones que se generan para el algoritmo Vernam y RSA, el producto cartesiano que se generan entre las posibles combinaciones es 4294967296^2 =1.84467E+19; considerando que se necesiten la misma cantidad de instrucciones y a una frecuencia de procesador igual a 2 0000000000 Hz lo cual se estima como estándar en un par de años o utilizando computación paralela en la tabla 6 se muestra el tiempo estimado en un caso peor y mejor para la decriptación del mensaje por un tercero sin conocer las claves privadas.

4.3 Comparación con otros algoritmos

Se realizó la implementación del algoritmo Vigenère, RSA y Vernam independientemente, los cuales son implementables en móviles celulares.

Considerando un ataque por fuerza bruta se tiene la siguiente tabla comparativa.

Tabla 7. Número de combinaciones generadas para posibles claves en los algoritmos Vigenère, RSA, Vernam

	Nº de bits en la clave	Nº de combinaciones
Vigenère	32	4294967296
RSA	32	4294967296
Vernam	32	4294967296
Alg. Híbrido RSA-Vernam	32	1.84467E+19

Tabla 8. Cálculo del tiempo de procesamiento de ataque por fuerza fruta a los algoritmos Vigenère, RSA y Vernam

		Nº de ciclos por instrucción	Tiempo de procesamiento(seg)	Tiempo de procesamiento(años)
Caso mejor	Vigenère	15	3.22	1.02144E-07
	RSA	15	3.22	1.02144E-07
	Vernam	15	3.22	1.02144E-07
	Alg. Híbrido RSA-Vernam	15	13835058055.28	438.706813
Caso peor	Vigenère	1	0.21	6.80963E-09
	RSA	1	0.21	6.80963E-09
	Vernam	1	0.21	6.80963E-09
	Alg. Híbrido RSA-Vernam	1	922337203.69	29.24712087

La tabla 7 muestra el número de combinaciones generadas para posibles de claves en los algoritmos Vigenère, RSA y Vernam. Se detalla la cantidad de combinaciones posibles utilizando 32 bits (2^{32}) y se puede notar la diferencia con las combinaciones del algoritmo híbrido ($2^{32} \cdot 2^{32}$).

La tabla 8 resume los tiempos de procesamiento para un ataque a los diferentes algoritmos por fuerza bruta considerando un procesador genérico corriendo a 20GHz.

CAPÍTULO 5

CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

- Acorde al objetivo principal se logró plantear un modelo de seguridad criptográfica "RSA-VERNAM", que permite a sistemas de telefonía móvil de segunda generación la mejora de su oferta de servicios de seguridad, incorporando características tan importantes como pueden ser la aplicación de seguridad de la mensajería extremo a extremo, disponibilidad del servicio de integridad y la capacidad de adaptación a nuevos mecanismos y servicios. Es destacable la relevancia de la seguridad extremo a extremo que facilita la posibilidad de que el usuario gestione el nivel de seguridad deseado sin estar a disposición del operador de telefonía móvil.

- Se realizó cumplir a cabalidad los cuatro objetivos específicos con el desarrollo del software receptor de mensajería con conectividad Bluetooth y la programación respectiva de los algoritmos de encriptación y decriptación tanto en el celular como en la PC.

- En el desarrollo de la tesis y resultado de los estudios realizados se gestó el modelo de seguridad "RSA-VERNAM 32", este modelo es práctico, es decir es aplicable en el mercado local e internacional.

- La fortaleza del modelo propuesto requiere en el mejor de los casos de 29 años en procesamiento por un computador corriendo a 20 GHz para ser atacado, lo cual hace que sea una buena opción de seguridad para la mensajería sobre equipos móviles celulares.

- La aplicación de un computador para la decriptación del mensaje enviado ayuda en gran forma, dado que los tiempos de procesamiento de un computador personal son mayores que el de un teléfono celular ya que 220 MHz es lo máximo que puede correr un celular existente hoy en día (Ver. Anexo B).

- Matemáticamente existen formas para poder reducir el tiempo en el ataque al algoritmo RSA pero el problema sea hace grande cuando se presenta el producto

cartesiano entre los posibles valores generados por ambos algoritmos RSA-VERNAM.

- También resultó conveniente plantear el sistema con un recolector de mensajes corriendo en un computador ya que en un celular la cantidad de mensajes llegaría a un limite dependiendo de su memoria, mientras que un computador personal de estos tiempos puede almacenar gran cantidad de mensajería dependiendo del gestor de bases de datos, la configuración del computador y las características físicas del computador.

- Los algoritmos de clave pública hoy en día son aplicados en programas que corren en computadoras de gran velocidad y solamente son utilizados para contraseñas u otros campos de datos cortos, entonces hacer correr un algoritmo relativamente pesado fue el reto en esta tesis para un equipo móvil con poca velocidad de procesamiento.

- Ambos algoritmos son vulnerables por separado, pero al ser unidos secuencialmente mejora enormente su fortaleza ante las técnicas de ataques matemáticas conocidas para cada uno gracias al producto cartesiano de las posibilidades que presentan por separado.

- Es necesario utilizar un computador como receptor debido a que utilizando el procesador de otro celular la decriptación se hace pesada para cualquier procesador de equipo móvil actual que no rebasa los 220 MHz.

- La comunicación Bluetooth no tiene mucha relevancia dado que podría ser reemplazado por una comunicación serial RS232, RS485 o USB que son las más comerciales hoy en día.

- Para una ampliación de aceptación de caracteres, es decir, que la estructura del algoritmo pueda funcionar con caracteres especiales como ser: coma, guión y otros el valor del modulo puede ser ampliado según la necesidad.

5.2 Recomendaciones

- Se recomienda no repetir claves privadas compartidas con alta frecuencia debido a que se arrastra la debilidad del código Vernam que puede sufrir un ataque con entradas y salidas conocidas.

- Se recomienda tomar en cuenta las características del procesador del celular para lograr óptima velocidad en la ejecución algoritmo híbrido que corre en el mismo.

- Se pudo notar que un computador P4 de 2.4 GHz tarda como 2 segundos aproximadamente en la decriptación de un mensaje, se recomienda usar un computador de buena velocidad en procesador para decriptar mensajería masiva.

- Nunca transmita La clave privada compartida sobre canales inseguros, tampoco utilice la misma clave pública muchas veces porque se expone la información a un ataque por comparación aumentando la probabilidad de ser decriptado en menor tiempo al señalado en la tabla 6.

- Se debe de utilizar claves superiores al valor de n porque produciría repeticiones con otras combinaciones de letras, tampoco mayores a $38^3+38^2+38=56354$ debido a que es el número máximo que puede generar la combinación de los caracteres más altos de la tabla 3 (caracter: *espacio*).

Referencias bibliográficas

[1] G. Orlando. *Efficient elliptic curve processor architectures for field programmable logic*. Tesis, Worcester Polytechnic Logic, 2002.

[2] J.Lutz and A. Hasan, *High performance FPGA based elliptic curve co-processor*, Proc. IEEE ITCC´04.

[3] N. Telle, W. Luk and R.C.C. Cheung, *Customizing hardware design for elliptic curve cryptography*, SAMOS 2004, LNCS 3133, pp. 274-283, 2004.

[4] M. Ernst, S. Klupsch, O. Hauck and S. A. Huss, *Rapid Prototyping for Hardware Accelerated Elliptic Curve Public-KeyCryptosystems*, *Proc. 12th IEEE Workshop on Rapid System Prototyping (RSP01)*, Monterey, CA, 2001.

[5] El Blog de las tecnologías móviles [en linea]. Febrero 2008 [fecha de consulta: 5 mayo 2008]. Disponible en: http://www.mas34.net/index.php?category_name=cursiosidades/&paged=2

[6] Departamento de Publicaciones de la Escuela Universitaria de Informática de la Universidad Politécnica de Madrid, *Aplicaciones criptográficas*, Segunda edición, 1999.

[7] Departamento de Publicaciones de la Escuela Universitaria de Informática de la Universidad Politécnica de Madrid, *Libro electrónico de seguridad informática y criptografía*, Quinta edición v4.0 en Internet, 2005.

[8] Manuel José Lucena López, *Criptografía y seguridad en computadores*, Departamento de Informática Escuela Politécnica Superior Universidad Jaén, Tercera Edición, 2002.

[9] QUINTANA, Marcos. *Criptosistema* [en linea]. Septiembre 2007 [fecha de consulta: 7 marzo 2008]. Disponible en: <http://www.webtaller.com/maletin/articulos/criptosistema-rsa-2.php>

[10] MORENO, Luciano. *Criptografía (IX)* [en linea]. Febrero 2005 [fecha de consulta: 7 septiembre 2007]. Disponible en: <http://www.terra.es/personal6/morenocerro2/seguridad/cripto/cripto_9.html>

[11] WIKILEARNING, Comunidades de wikis libres para aprender. *Método de Lehmann* [en línea]. Diciembre 2004 [fecha de consulta: 10 diciembre 2006]. Disponible en: <http://www.wikilearning.com/metodo_de_lehmann-wkccp-15516-27.htm>

[12] SOMA, Ghosh. Extend *J2ME to Wireless Messaging* [en línea]. marzo 2006 [fecha de consulta: 10 diciembre 2006] Disponible en: <http://www.ibm.com/developerworks/wireless/library/wi-extendj2me/>

[13] EVELIUX. *Comandos AT* [en línea], enero 2006 [fecha de consulta: 10 febrero 2007]. Disponible en: <http://www.eveliux.com/mx/comandos-at.php>

[14] TORRANO, Emilio. *Algoritmo de Euclides*, [en línea]. España: Universidad Politécnica de Madrid, 2005 [fecha de consulta: 5 abril 2006] Disponible en: <http://www.dma.fi.upm.es/java/matematicadiscreta/aritmeticamodular/divisibilidad.html>

[15] WIKIPEDIA, La enciclopedia libre. *Exponenciación binaria*, [en línea]. 2005 [fecha de consulta: 12 mayo 2006]. Disponible en: <http://es.wikipedia.org/wiki/Exponenciaci%C3%B3n_binaria>

[16] WINNER_MAN. *Procesadores de moviles symbian,* [en línea]. Octubre 2006 [fecha de consulta: 3 abril 2007]. Disponible en: <http://foro.powers.cl/viewtopic.php?t=208577>

[17] GUTIÉRREZ, Rodrigo. *Secretos del PC* [en linea]. Chile: Santiago, 12 abril 2008 [fecha de consulta: 15 enero 2009]. Disponible en: <http://www.terra.cl/tecnologia/index.cfm?id_cat=1719&id_reg=612148>

[18] WIKIPEDIA. La enciclopedia libre. *Test de primalidad*, [en línea]. 2005 [fecha de consulta: 22 febrero 2008]. Disponible en: <http://es.wikipedia.org/wiki/Test_de_primalidad>

[19] WIKIPEDIA. La enciclopedia libre. *Test de primalidad*, [en línea]. 2005 [fecha de consulta: 23 febrero 2008]. Disponible en: <http://en.wikipedia.org/wiki/Miller-Rabin_primality_test>

[20] SEMINARIO. Informática, Ciencia y Tecnología(1º, 2005, Segovia, España). *Criptología*, [en línea]. 2005 [fecha de consulta: 5 diciembre 2008]. Disponible en: <http://wmatem.eis.uva.es/~ignfar/crypto.html>

[21] PINTO, Rolf. Monografias.com, *Teoría de la complejidad algorítmica*, [en línea]. 2006 [fecha de consulta: 10 diciembre 2008]. Disponible en: <http://www.monografias.com/trabajos27/complejidad-algoritmica/complejidad-algoritmica.shtml>

[22] AGUILAR JAVIER, Clara. *Método de encriptación de comportamiento emergente*. Tesis (Licenciatura en Informática). La Paz, Bolivia, Universidad Mayor de San Andrés, 2001. 118 p.

ANEXO A

ALGORITMO DE DIFFIE - HELLMAN

Este algoritmo de encriptación de Whitfield Diffie y Martin Hellman supuso una verdadera revolución en el campo de la criptografía, ya que fué el punto de partida para los sistemas asimétricos, basados en dos claves diferentes, la pública y la privada, vió la luz en 1976, [10].

Matemáticamente se basa en las potencias de los números y en la función mod (módulo discreto). Uniendo estos dos conceptos se define la potencia discreta de un número como $Y = X^a \bmod q$. Si bien el cálculo de potencias discretas es fácil, la obtención de su función inversa, el logaritmo discreto, no tiene una solución analítica para números grandes.

En la figura se observa un esquema explicativo acerca del método para compartir claves según Diffie Hellman

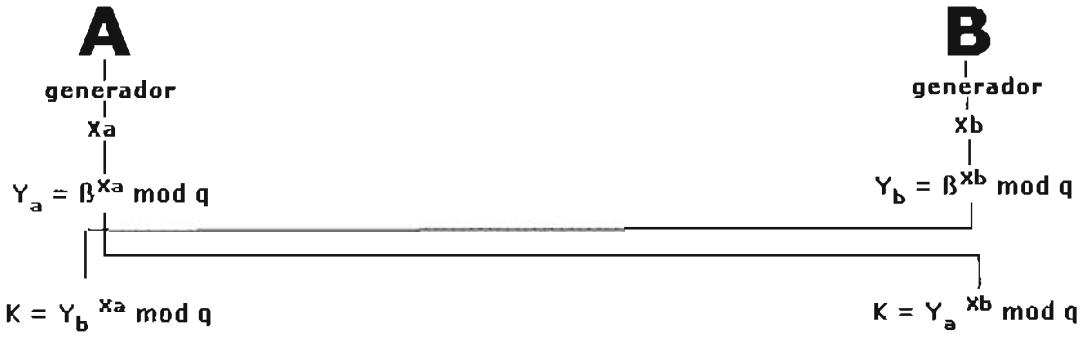

Figura 10. Metodo de Diffie Hellman, fuente [10]

Para implementar el sistema re realizan los siguientes pasos [10]:

Paso 1: Se busca un número primo muy grande, q.

Paso 2: Se obtiene el número _, raíz primitiva de q, es decir, que cumple que _ mod q, $_^2 \bmod q$,...., $_^{q-1} \bmod q$ son números diferentes.

Paso 3: _ y q son las claves públicas.

ANEXO B

CARACTERISTICAS DE PROCESADORES DE MOVILES SYMBIAN

Nokia 6600, 3650, Ngage Qd, Ngage: 104 mhz ARM9 Siemens Sx1: 120 mhz Nokia 6260, 6670: 123 mhz ARM9 Nokia 6620: 150 mhz ARM9 nokia 7610: 123 mhz ARM9 Nokia 6630: 220 mhz Sony ericcson P800, P900, P910: 156 mhz ARM9 Nokia 6682: 220 mhz TI OMAP 1710 Nokia 9500, 9300, 7710: 150 mhz TI OMAP 1510		
Nokia 3230 RM-51 32-bit RISC CPU ARM-9 123 MHz	Nokia 3600 NHM-10 32-bit RISC CPU ARM-9 104 MHz	Nokia 3620 NHM-10(X); 32-bit RISC CPU ARM-9 104 MHz
Nokia 3650 NHL-8 32-bit RISC CPU ARM-9 104 MHz	Nokia 3660 NHL-8X 32-bit RISC CPU ARM-9 104 MHz	Nokia 6230 RM-25 32-bit RISC CPU ARM-9 123 MHz
Nokia 6600 NHL-10 32-bit RISC CPU ARM-9 104 MHz	Nokia 6620 NHL-12 32-bit RISC CPU ARM-9 150 MHz	Nokia 6630 RM-1 32-bit RISC CPU ARM-9 220 MHz
Nokia 6670 RH-67 32-bit RISC CPU ARM-9 123 MHz	Nokia 6670B RH-68 32-bit RISC CPU ARM-9 123 MHz	Nokia 6680 RM-36 32-bit RISC CPU ARM-9 220 MHz
Nokia 6681 RM-57 32-bit RISC CPU ARM-9 220 MHz	Nokia 6682 RM-58 32-bit RISC CPU ARM-9 220 MHz	Nokia 7610 RM-51 32-bit RISC CPU ARM-9 123 MHz
Nokia 7610B RH-52 32-bit RISC CPU ARM-9 123 MHz	Nokia 7650 NHL-2(NA); 32-bit RISC CPU ARM-9 104 MHz	Nokia 770 SU-18 TI OMAP 1710 ARM-926 220 MHz
Nokia 7710 RM-12 32-bit RISC CPU ARM-9 150 MHz	Nokia 9210 RAE-3(N); 32-bit RISC CPU ARM-9 52 MHz	Nokia 9210c RAE-3(N); 32-bit RISC CPU ARM-9 52 MHz
Nokia 9210i RAE-5(N); 32-bit RISC CPU	Nokia 9290 RAB-3(N); 32-bit RISC CPU	Nokia 9300 RAE-6(N); TI OMAP 1510

ARM-9 52 MHz	ARM-9 52 MHz	ARM-925 150 MHz
Nokia 9300B RA-4 TI OMAP 1510 ARM-925 150 MHz	Nokia 9300i RA-8 TI OMAP 1510 ARM-925 150 MHz	Nokia 9500 RA-2 TI OMAP 1510 ARM-925 150 MHz
Nokia 9500B RA-3 TI OMAP 1510 ARM-925 150 MHz	Nokia E50 TI OMAP 1710 ARM-926 220 MHz	Nokia E60 RM-49 TI OMAP 1710 ARM-926 220 MHz
Nokia E61 RM-89 TI OMAP 1710 ARM-926 220 MHz	Nokia E62-1 RM-88 TI OMAP 1710 ARM-926 220 MHz	Nokia E62-1 RM-88A TI OMAP 1710 ARM-926 220 MHz
Nokia E70-1 RM-10 TI OMAP 1710 ARM-926 220 MHz	Nokia E70-2 RM-24 TI OMAP 1710 ARM-926 220 MHz	Nokia N-Gage NEM-4 32-bit RISC CPU ARM-9 104 MHz
Nokia N-Gage QD RH-29 32-bit RISC CPU ARM-9 104 MHz	Nokia N-Gage QDA RH-47 32-bit RISC CPU ARM-9 104 MHz	Nokia N70-1 RM-84 TI OMAP 1710 ARM-926 220 MHz
Nokia N70-5 RM-99 TI OMAP 1710 ARM-926 220 MHz	Nokia N71-1 RM-67 TI OMAP 1710 ARM-926 220 MHz	Nokia N71-5 RM-112 TI OMAP 1710 ARM-926 220 MHz
Nokia N72-5 RM-180 TI OMAP 1710 ARM-926 220 MHz	Nokia N73-1 RM-133 TI OMAP 1710 ARM-926 220 MHz	Nokia N73-? RM-132 TI OMAP 1710 ARM-926 220 MHz
Nokia N80-1 RM-92 TI OMAP 1710 ARM-926 220 MHz	Nokia N80-3 RM-91 TI OMAP 1710 ARM-926 220 MHz	Nokia N90-1 RM-42 TI OMAP 1710 ARM-926 220 MHz
Nokia N91-1 RM-43 TI OMAP 1710 ARM-926 220 MHz	Nokia N91-5 RM-158 TI OMAP 1710 ARM-926 220 MHz	Nokia N92 TI OMAP 1710 ARM-926 220 MHz
Nokia N93-1	Nokia N93-5	

RM-55 TI OMAP 1710 ARM-926 220 MHz	RM-153 TI OMAP 1710 ARM-926 220 MHz	

Obtenido de [16]