BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

Facultad de Ciencias de la Computación

Seguridad en dispositivos móviles: Aplicación del algoritmo Kasumi utilizando PDAs.

TESIS PROFESIONAL
PARA OBTENER EL TÍTULO DE
LICENCIADO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA
C. JAVIER VÁZQUEZ CUCHILLO

ASESOR
DRA. CLAUDIA FEREGRINO URIBE
COORDINACIÓN DE CIENCIAS COMPUTACIONALES, INAOE

COASESOR
M. C. JOSÉ ALFONSO GARCÉZ BÁEZ
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

PUEBLA, PUEBLA FEBRERO 2008

AGRADECIMIENTOS

A la **Dra. Claudia Feregrino Uribe** y **M.C. José Gárces Báez** por su apoyo y asesoría en la realización de esta tesina.

RESUMEN

En este trabajo se muestra la aplicación del algoritmo de encriptación KASUMI, para cifrar los mensajes trasmitidos a través de una red inalámbrica, con el objetivo de evitar la legibilidad de los mensajes en caso de que sean interceptados. Se logró generar una aplicación que muestra la comunicación entre dos pda's a través de un servidor principal mediante una arquitectura cliente servidor.

INDICE

CAPÍTULO I. INTRODUCCIÓN 6

CAPÍTULO II. CONCEPTOS BÁSICOS DE CRIPTOGRAFÍA. 9

2.1. CRIPTOGRAFÍA 9

2.1. Objetivos de la Criptografía. 10

2.3. Criptosistema. 11

2.3.1. Criptosistemas simétricos o de clave privada. 12

2.3.1.1 Criptosistemas simétricos por bloques. 13

2.3.1.2 Criptosistemas simétricos por flujo 14

2.3.2. Criptosistemas Asimétricos o de clave pública. 15

2.3.3. Autenticación. 16

2.2. REDES INALÁMBRICAS. 16

2.2.1. Dispositivos inalámbricos 20

2.2.1.1 PDA (PERSONAL DIGITAL ASSISTANT) 20

CAPÍTULO III. KASUMI Y ALGORITMO DE CONFIDENCIALIDAD 21

3.1. KASUMI 21

3.2 ALGORITMO DE CONFIDENCIALIDAD 29

CAPÍTULO VI. APLICACIÓN Y RESULTADOS. 30

4.1. DETALLES DE IMPLEMENTACIÓN 31

CAPÍTULO V. CONCLUSIONES 37

BIBLIOGRAFIA 38

INDICE DE FIGURAS

Figura 2.3.1.1. Esquema general de Criptosistemas de clave privada. 12
Figura 2.3.1.1.1.Cifrado por bloques Feistel [11]. 14
Figura 2.3.1.2.1. Esquema general de un Criptosistema simétrico por flujo [11]. 15
Figura 2.3.2.1. Esquema general de Criptosistemas de clave pública. 15
Figura 2.2.1. Componentes físicos del 802.11 [20]. 17
Figura 2.2.2. Red Ad Hoc o independiente [20]. 18
Figura 2.2.3. Esquema de un ESS [20]. 19
Figura. 3.1.1. Estructura Feistel del KASUMI. 22
Figura. 3.1.2. Funcion FL. 24
Figura. 3.1.3. Función FO. 25
Figura. 3.1.4. Función FI..... 26
Figura 3.2.1. Algoritmo de confidencialidad f8 [3]. 29
Figura. 4.1. Arquitectura de la aplicación. 31
Figura 4.1.1. Función de envió de mensajes. 32
Figura 4.1.2. Función que implementa el algoritmo de confidencialidad..... 33
Figura 4.1.3. Función KeySchedule. 34
Figura 4.1.4. Implementación del Algoritmo KASUMI, con las funciones FO, FI
Y FL. 36

INDICE DE TABLAS

Tabla 3.1.1. Constantes. 28

Capítulo I. INTRODUCCIÓN

La seguridad informática se ha vuelto un problema muy significativo en la actualidad debido a lo inseguro que llega a ser el medio de transmisión y la necesidad de proteger la información que viaja a través del mismo. Esta inseguridad provocada por diversos factores, como la intercepción de los datos o la inestabilidad del medio de transmisión, causan desconfianza al transferir información debido a la falta de integridad, autenticidad y confidencialidad.

Aunque la autenticación es una cuestión fundamental, más importante incluso es la confidencialidad de la información que se intercambia, sobre todo teniendo en cuenta lo inestable del medio de transmisión. Estos problemas ya se han atacado de diversas maneras,

mas con el auge de la tecnología inalámbrica donde una nueva generación de dispositivos surgen (como las palm, las pockets, las ipod, los celulares) y que son de uso más común entre la comunidad, nos enfrentamos al renacimiento de los mismos problemas y a una nueva era de la seguridad informática.

El principal problema de las redes inalámbricas era su relativa debilidad desde el punto de vista criptográfico. Como lo habían demostrado diversos estudios, que sustentaban que para descifrar las comunicaciones que viajan a través de estas redes es necesario capturar una cantidad enorme de datos, intercambiados y analizarlos. En entornos de poco tráfico como pequeñas oficinas o casas particulares se pueden tardar muchos días en reunir la información necesaria. En entornos empresariales en los que el volumen de información que se traslada es muy elevado es posible que un atacante preparado rompa las claves en cuestión de unas horas usando un portátil desde muchos metros de distancia y sin ser detectado.

Por lo anterior diferentes métodos criptográficos ya implementados y probados se tomaron para resolver problemas de confidencialidad e integridad, entre los cuales podemos mencionar el RSA, DES, TRIPLE DES, entre otros. El método criptográfico más novedoso es el KASUMI [1-3], basado en el cifrador misty [4].

Este algoritmo ha sido implementado tanto en hardware como en software. Su implementación en hardware se encuentra en el artículo “An Efficient Hardware Implementation of the KASUMI Block Ciphier for Third Generation Cellular Networks” [1]. En cuanto a la implementación en software este trabajo se encarga de ilustrarlo mediante la creación de una aplicación que comunica dos pda´s, a través de un servidor principal, usando el algoritmo criptográfico KASUMI en el cifrado de los mensajes

con la finalidad de evitar la legibilidad en caso de que uno o más mensajes son interceptados.

Capítulo II. Conceptos básicos de Criptografía.

En este capítulo se dan conceptos generales acerca de Criptografía y las redes inalámbricas, así como otros conceptos importantes para la realización de este trabajo de investigación.

2.1. Criptografía

La Criptografía es un arte tan antiguo que tiene sus inicios hace miles de años, esto debido a que el hombre siempre en su comunicación con otras personas ha tenido la necesidad de ocultar información por múltiples razones, tales como evitar que alguien pudiera interpretar la información relacionada a planes bélicos, esconder mensajes a la vista de todos, entre otros. Entre algunas de las civilizaciones pioneras de la Criptografía encontramos a Egipto, Mesopotamia, India, China, Roma, las cuales crearon algunos mecanismos para cifrar (encriptar) y descifrar (desencriptar) mensajes.

La Criptografía se puede definir simplemente como la ciencia de ocultar mensajes de forma segura [5]. Pero, en la actualidad con el auge de la era Informática, se empezaron a necesitar nuevos mecanismos que dieran más seguridad a la información almacenada y trasmitida por los sistemas de computo, por lo que la Criptografía retoma una nueva importancia encontrando definiciones menos simples como la que dice que la Criptografía es el estudio de técnicas matemáticas relacionadas con aspectos de seguridad en la información, tales como autenticación de entidades y de datos, confidencialidad e integridad de datos [6].

2.1. Objetivos de la Criptografía.

La Criptografía al tener que mantener la seguridad en los datos que viajan a través de los medios informáticos tiene que cumplir con ciertos objetivos fundamentales los cuales son la privacidad, la integridad, la autenticidad y el no repudio.

La privacidad consiste básicamente en mantener en secreto la información excepto para aquellos usuarios que tengan el permiso para poder leerla o interpretarla.

La integridad se refiere al servicio que asegura que los datos no serán alterados por ningún medio, y nos permite detectar algún cambio en la información.

La autenticación esta relacionado a la identificación. Este servicio asegura que alguna entidad (usuario, sistema, computadora, laptop, etc), que se comunica y manda alguna información, es quien dice que ella es.

El no repudio es el servicio que impide a una entidad negar que mando o recibió un mensaje cuando se ha iniciado una comunicación entre dos entidades.

2.3. Criptosistema.

Para lograr los objetivos de la Criptografía se definen y utilizan los Criptosistemas. Un Criptosistema es el conjunto de procedimientos que mediante procedimientos y técnicas criptográficas garantizan la seguridad de la información. Matemáticamente se define en [8] como una quíntupla *(M,C,K,E,D), donde:*

- *M* representa el conjunto de todos los mensajes sin cifrar(lo que se denomina texto claro, o plaintext) que pueden ser enviados).
- *C* representa el conjunto de todos los posibles mensajes cifrados, o criptogramas.
- *K* representa el conjunto de claves (llaves) que se pueden emplear en el criptosistema.
- *E* es el conjunto de transformaciones de cifrado o familia de funciones que se aplica a cada elemento de M para obtener un elemento de *C*. Existe una transformación diferente E_k.
- *D* es el conjunto de transformaciones de descifrado, análogo a *E.*

Todo Criptosistema ha de cumplir la siguiente condición:

$$D_k(E_k(m))=m$$

Lo cual significa, que si tenemos un mensaje *m,* lo encriptamos utilizando la clave *k* y posteriormente lo desencriptamos usando la misma clave, y obtenemos de nuevo el mensaje original *m.*

Existen dos clases de Criptosistemas los cuales son los simétricos o de clave privada y los asimétricos o de clave pública.

2.3.1. Criptosistemas simétricos o de clave privada.

Los Criptosistemas simétricos usan una sola clave para encriptar y desencriptar. Estos tipos de criptosistemas son diseñados generalmente para minimizar la computación requerida para cifrar y descifrar los datos lo cual hace que al implementarlos su operación sea muy rápida [9].

Al utilizar solo una clave para cifrar y descifrar, estos tipos de Criptosistemas no son muy seguros en la comunicación entre dos entidades pues la clave deben poseerla tanto el emisor como el receptor, para lo cual debemos encontrar alguna manera para trasmitir la clave sin que esta pueda ser interceptada por alguna entidad ajena a la comunicación. Un esquema general de este tipo de Criptosistemas de clave privada se muestra en 2.3.1.1.

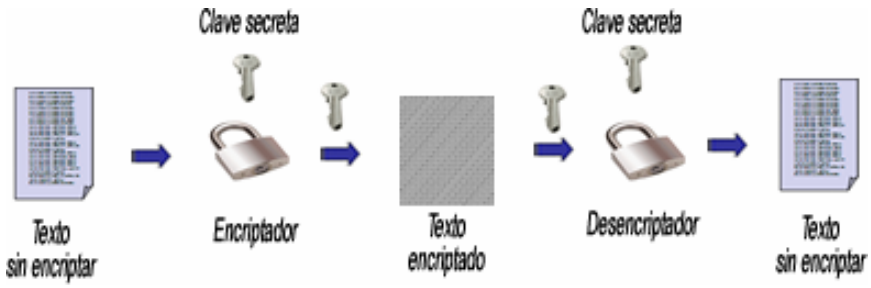

Figura 2.3.1.1. Esquema general de Criptosistemas de clave privada.

Estos Criptosistemas se pueden clasificar en Criptosistemas simétricos por bloques y por flujo de datos.

2.3.1.1 Criptosistemas simétricos por bloques.

Los Criptosistemas simétricos por bloques son los más simples de entender y uno de los más usados. Utilizan un esquema de cifrado que divide el mensaje original o texto en claro en pequeños bloques de longitud n sobre un alfabeto A, cifrando un bloque a la vez [10].

. Estos Criptosistemas no son solamente usados para proteger la privacidad, sino también para construir bloques para crear otros algoritmos.

Este tipo de Criptosistemas esta basado en el diseño propuesto por Horst Feistel [11]. El diseño de Feistel nos dice que un bloque de tamaño de N bits comúnmente (N=64 o 128) se divide en dos bloques de tamaño N/2, X y Y. Una vez realizado esto se inicia el proceso de cifrado que consiste en aplicar una función unidireccional a un bloque (Y) y a una sub-llave k1 generada a partir de la llave secreta. Se mezclan el bloque X con el resultado de la función mediante un XOR. Se permutan los bloques y se repite el proceso n veces. Finalmente se juntan los dos bloques en el bloque original. Como se muestra en la figura 2.3.1.1.1 [11].

Estos Criptosistemas por bloques funcionan con boques de tamaño fijo, generalmente de 64 y 128 bits. Para utilizar mensajes de un tamaño más grande se utilizan otros tipos de operación. Estos tipos de operación son el ECB (Electronic codebook), CBC (Chipre-block chaining), OFB (Output Feedback) y CFB (Cipher Feedback) [12]

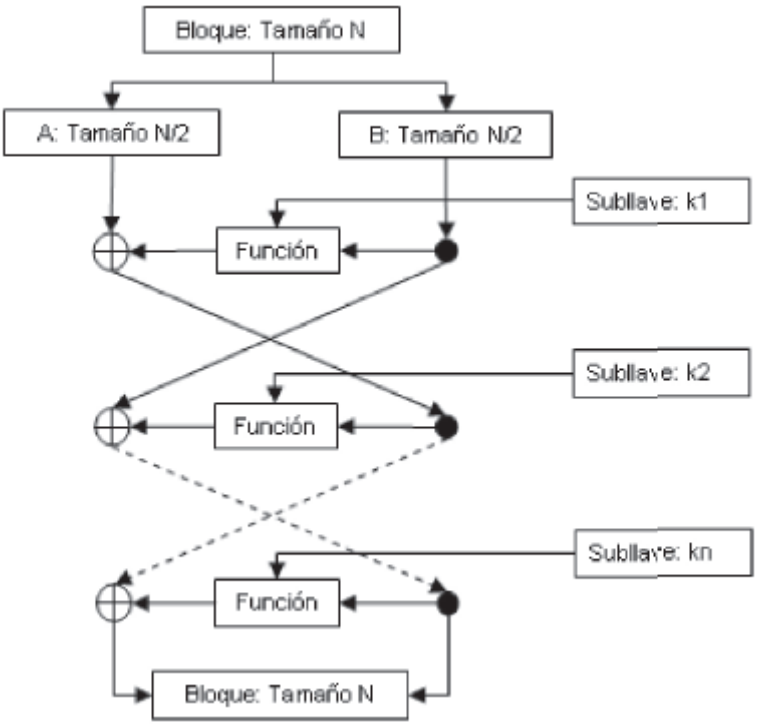

Figura 2.3.1.1.1.Cifrado por bloques Feistel [11].

Ejemplo de los cifradores por bloques son Lucifer, DES [7], AES [14] [22], Blowfish [7] y KASUMI [1-3] el cual se aborda su aplicación en este trabajo.

2.3.1.2 Criptosistemas simétricos por flujo.

Los Criptosistemas simétricos por flujo pueden realizar el cifrado incrementalmente, convirtiendo el texto claro en texto cifrado bit a bit. Para realizar esto se construye un generador de flujo de clave (secuencia clave). El flujo de clave es una secuencia de bits de tamaño variable que se usa para mezclarla con el contenido de un flujo de datos mediante una función XOR, con la finalidad de tener ilegibilidad en el flujo de datos. Un esquema general de este tipo de Criptosistemas se muestra en la figura

2.3.1.1.1 [11]. Ejemplos de estos Criptosistemas son el RC4 (utilizado en redes inalambricas) [14] y el A5 (utilizado en telefonía celular) [15][23].

Figura 2.3.1.2.1. Esquema general de un Criptosistema simétrico por flujo [11].

2.3.2. Criptosistemas Asimétricos o de clave pública.

Los Criptosistemas asimétricos usan dos claves las cuales son llamadas clave privada y clave pública. Una se usa para la transformación de cifrado y la otra para la transformación de descifrado, las cuales pueden ser usadas indistintamente, mientras una sola sirva para cifrar y la otra para descifrar. Algo importante que deben tener estos criptosistemas asimétricos es que el conocer la clave pública no permita obtener la clave privada. Este tipo de Criptosistemas permiten tener comunicaciones seguras por canales inseguros, pues solo viaja a través del canal la clave pública, o para ser usados en autentificación.

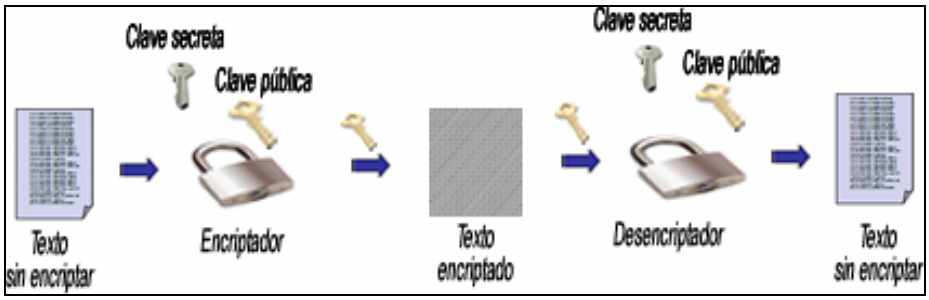

Figura 2.3.2.1. Esquema general de Criptosistemas de clave pública.

Ejemplos de estos son el RSA [16] y las curvas elípticas [17].

2.3.3. Autenticación.

La autenticación es una forma mediante la cual una entidad A identifica a otra entidad B, para asegurar que la entidad B es quien dice ser y viceversa [18]. Esto nos permite cumplir dos de los objetivos principales de la criptografía que son la privacidad y la integridad de la información. Uno de los métodos de autenticación es la firma digital El proceso de firma se basa en un Criptosistema asimétrico, cuya diferencia esta en que se utiliza la clave privada para encriptar el resultado de la función hash (Una función hash transforma una cantidad de información de tamaño variable en un bloque de tamaño especifico) y se usa la clave pública para desencriptar el mensaje.

2.2. Redes Inalámbricas.

El concepto “inalámbrico” refiere a la tecnología sin cables que permite interconectar varias maquinas usando frecuencias de radio u ondas infra-rojas [19]. Existen tres categorías principales de redes inalámbricas que son:

1. WAN(WIDE AREA NETWORK)
2. WLAN(WIRELES LOCAL AREA NETWORK)
3. PAN(PERSONAL AREA NETWORK)

Las WAN son utilizadas para transmitir la información en espacios que pueden variar desde una misma ciudad o hasta varios países

circunvecinos, generalmente se utiliza en el servicio de telefonía móvil. Las WLAN son redes con un tamaño enfocado a un área de trabajo en especifico (Universidades, Empresas) donde las oficinas se encuentran en uno o varios edificios que no están muy alejados entre si. Las PAN son redes que se utilizan en distancias pequeñas (Un rango de 10 metros), tales como el Bluetooth.

Para implementar una red inalámbrica, se utiliza el estándar IEEE 802.11 [20], donde se explican todo los requerimientos necesarios para crear una red inalámbrica. La norma IEEE 802.11 fue diseñada para sustituir a la capa física y MAC de la norma 802.3 (Ethernet), así, la única diferencia entre ambas es la manera en la que los dispositivos acceden a la red, por lo que ambas normas son perfectamente compatibles.

Las redes 802.11 consisten de cuatro principales componentes físicos los cuales se muestran en la figura 2.2.1 [20] y son:

1. Sistema de distribución.
2. Punto de acceso(PA).
3. Medio inalámbrico.
4. Estaciones

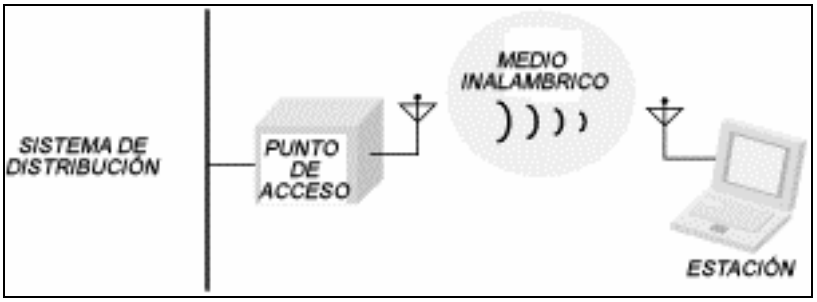

Figura 2.2.1. Componentes físicos del 802.11 [20].

El bloque básico y principal de una red inalámbrica es el llamado Conjunto Básico de Servicios generalmente llamado BBS. Un BBS es un conjunto de estaciones que se comunican entre si. Dicha comunicación se realiza en un área con un limite llamada área básica de servicios [21]. Existen dos tipos de BBS:

- Redes Ad Hoc o independientes
- Redes de infraestructura.

Las redes independientes no utilizan un punto de acceso. La figura 2.2.2 [20] muestra una red independiente.

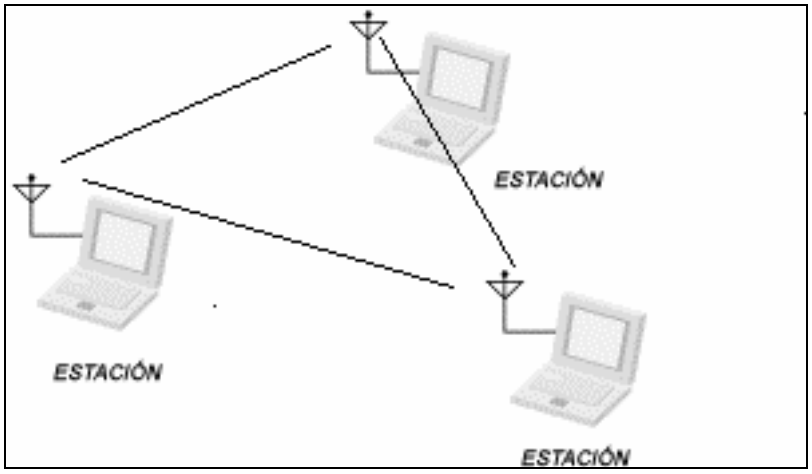

Figura 2.2.2. Red Ad Hoc o independiente [20].

La red de infraestructura utiliza un dispositivo llamado Punto de Acceso. La comunicación entre dos estaciones dentro de una misma área de servicios requiere dos saltos, uno hacia el punto de acceso y otro hacia la otra estacion. Un BBS se define por la distancia hacia el punto de Acesso que es el encargado de comunicar a las estaciones entre si.

Los BBS solo pueden cubrir áreas limitadas. Así que para abarcar mayores áreas se utiliza el conjunto extendido de servicios (ESS).

Utilizando un mismo ESS, las estaciones pueden comunicarse unas con otras dentro de un mismo ESS. La figura 2.2.3 [20] muestra la configuración de un ESS.

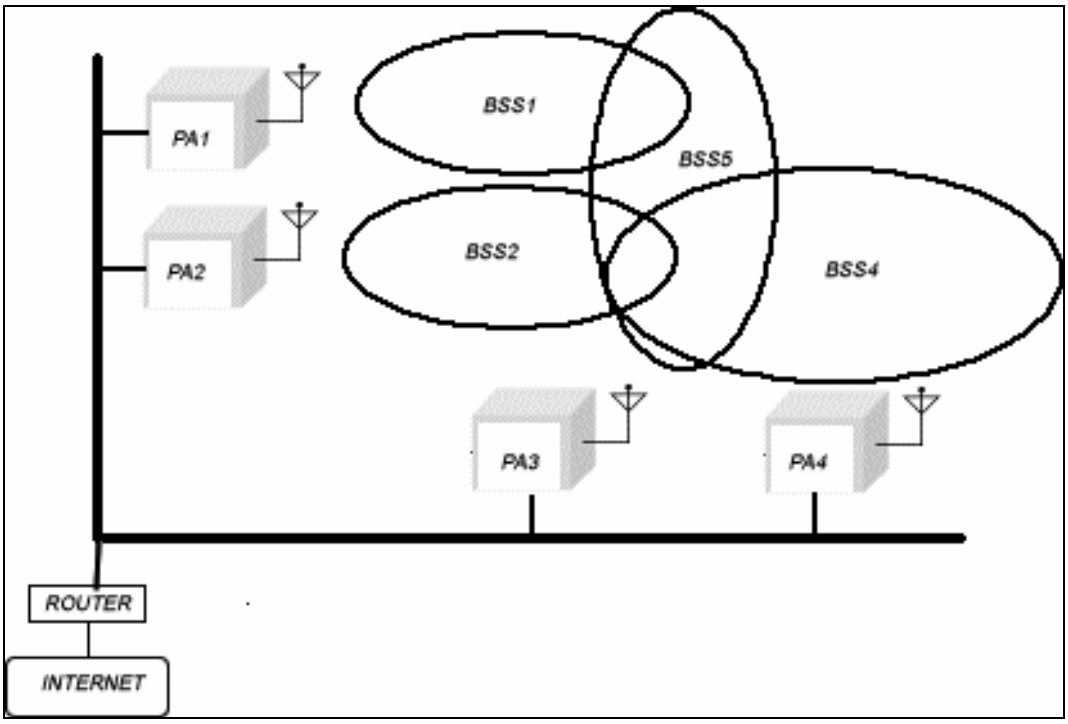

Figura 2.2.3. Esquema de un ESS [20].

Un sistema de distribución es el encargado de actualizar la ubicación física de las estaciones y atender correctamente los frames. La red Ethernet junto con los AP forman el sistema de distribución. El ESS entrega el soporte de movilidad a las estaciones (Roaming).

A partir de la norma 802.11 se han generado nuevas normas variantes de esta como son el 802.11a, 802.11b y el 802.11g [20].

2.2.1. Dispositivos inalámbricos.

Un dispositivo inalámbrico es aquel que es capaz de acceder a una red sin cables. Entre estos dispositivos encontramos los celulares, las pockets, las palms, las pdas, las laptops, bluetooth. Cada uno de estos dispositivos tienen características distintas por su funcionalidad pero todos pueden acceder a la red inalámbrica siempre y cuando cumplan con las características que se describen en la norma 802.11 u similar. Estos dispositivos según el esquema de la figura 2.2.1 conforman lo que llamamos estaciones.

2.2.1.1 PDA (PERSONAL DIGITAL ASSISTANT)

Un PDA es un dispositivo electrónico de tamaño pequeño que tiene muchas de las funcionalidades de una computadora normal. La gran aportación de las PDAs es que no se limitan a ciertos programas, sino que permiten cargar aplicaciones como las que tienen un ordenador común. A la PDAs también se les llama palmtops, computadoras de mano y computadoras de bolsillo. Un PDA tiene diferentes funcionalidades tales como la de teléfono móvil, fax, explorador de Internet, organizador personal, GPS, etc.

En la actualidad las PDAs pueden poseer teclado y/o reconocimiento de escritura, incluir reconocimiento de voz, entre otras funcionalidades. A las PDAs se les suele distinguir de acuerdo al sistema operativo que usan, a continuación se muestran algunas que existen en el mercado:

- Palm. Usan el sistema operativo Palm OS.
- Pocket PC. Usan el sistema operativo Windows Mobile.
- BlackBerry. Usan el sistema operativo para BalckBerry.

Capítulo III. KASUMI Y ALGORITMO DE CONFIDENCIALIDAD

3.1. KASUMI

El algoritmo KASUMI, también llamado A5/3, es una unidad de cifrado por bloques utilizada en algoritmos de confidencialidad (f8) e integridad (f9) para dispositivos móviles de tercera generación. Este fue diseñado por el grupo SAGE (Security Algorithm Group of experts). Kasumi ocupa un bloque de entrada y salida de 64 bits y una clave de 128 bits. Es una unidad de cifrado por bloque con 8 vueltas y tiene una estructura recursiva igual que misty [4].

KASUMI es un cifrador Feistel que se compone por las sub-funciones FL, F0, FI, las cuales son usadas en conjunción con llaves asociadas KL, K0, KI, como se muestra en la figura 3.1.1.

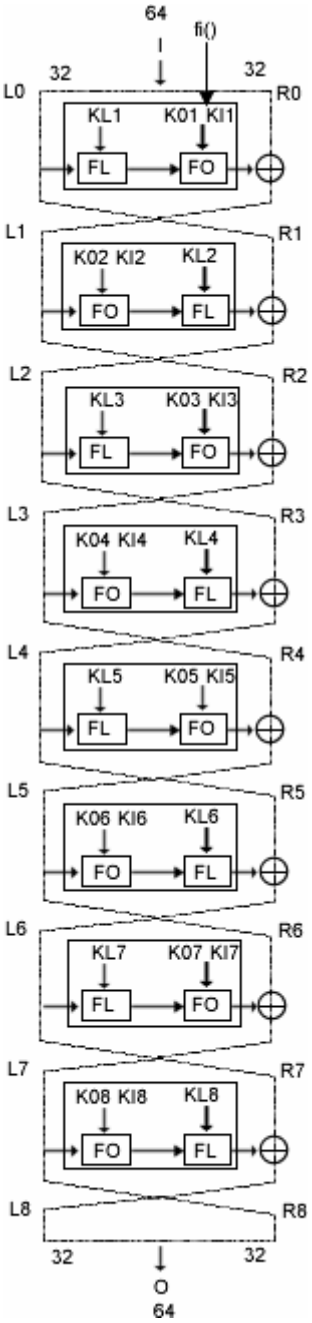

Figura. 3.1.1. Estructura Feistel del KASUMI.

A continuación se describe detalladamente el funcionamiento del KASUMI. A5/3 opera una entrada de datos I de 64 bits usando una llave K de 128 bits produciendo una salida O de 64 bits, como sigue:

La entrada *I* es dividida en dos cadenas de 32 bits L_0 y R_0 donde

$$I=L_0 \,||\, R_0 \qquad (3.1)$$

Para cada entero i con 1≤i≤8 definimos

$$R_i = L_{i-1} \oplus f_i (L_{i-1}, RK_i) \qquad (3.2)$$

Esto constituye los ciclos de la función KASUMI, donde f_i denota la función ciclo con L_{i-1} y la llave de ciclo RK_i como entradas. El resultado O es igual a una cadena de 64 bits (L8 || R8) resultado del último ciclo.

La función $f_i()$ toma una entrada i de 32 bits y retorna 32 bit de salida o bajo el control de la llave RK_i, donde las llaves de ciclo comprenden a KL_i, $K0_i$, KI_i. La función en si es construida de dos funciones FL y F0 con subllave KL_i usada en FL y subllave KO_i y KI_i. usadas en FO.

La función tiene dos diferentes formas dependiendo en si el ciclo es par o impar. Para ciclos 1, 3, 5, 7 fi () se define así:

$$fi\ (I,RK_i) = FO(\ FL(\ I, KL_i), KO_i, KI_i\ ) \qquad (3.3)$$

Y para ciclos 2, 4, 6, 8, se define como:

$$fi\ (I,K_i) = FL(\ FO(\ I, KO_i, KI_i\ ), KL_i) \qquad (3.4)$$

La función FL toma como entrada un dato i de 32 bits y una sub-llave KL_i de 32 bits, como se muestra en la Fig. 3.1.2. La sub-llave es dividida en 2 sub-llaves de 16 bits, $KL_{i,}1$ y KL_i,2, donde:

$$KL_i = KL_{i,1} \,||\, KL_{i,2}. \qquad (3.5)$$

El dato de entrada i es dividido en 2 datos de 16 bits, L y R donde

$$i=L \,||\, R. \qquad (3.6)$$

A partir de esto definimos:

$$R´=R \oplus ROL\,(\,L \cap KL_{i,1}\,) \qquad (3.7)$$

$$L´=L \oplus ROL\,(\,R \cup KL_{i,2}\,) \qquad (3.8)$$

La salida de 32 bits estará formada por:

$$o = \quad R` \,||\, L`$$

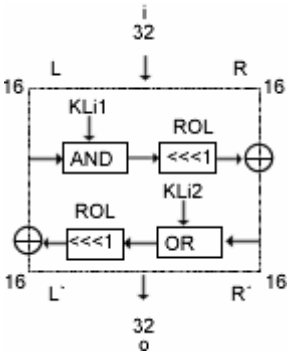

Figura. 3.1.2. Funcion FL.

La función F0 toma un dato de entrada i de 32 bits y usa dos sub-llaves de 48 bits cada una, KO_i y KI_i, como se muestra en la Fig. 3.1.3.

El dato i es dividido en dos partes de 16 bits, L0 y R0 donde

$$i= L0 \,||\, R0. \qquad (3.9)$$

Las sub-llaves son divididas en 3 sub-llaves de 16 bits donde:

$$KOi = KOi,1 \,||\, KOi,2 \,||\, KOi,3 \qquad (3.10)$$

$$KIi = KIi,1 \,||\, KIi,2 \,||\, KIi,3. \qquad (3.11)$$

Para cada entero i con 1≤i≤8 definimos:

$$Rj = FI(Lj\text{-}1 \oplus KOi,j \,,\, KIi,j\,) \oplus Rj\text{-}1 \qquad (3.12)$$

$$Lj = Rj\text{-}1 \qquad (3.13)$$

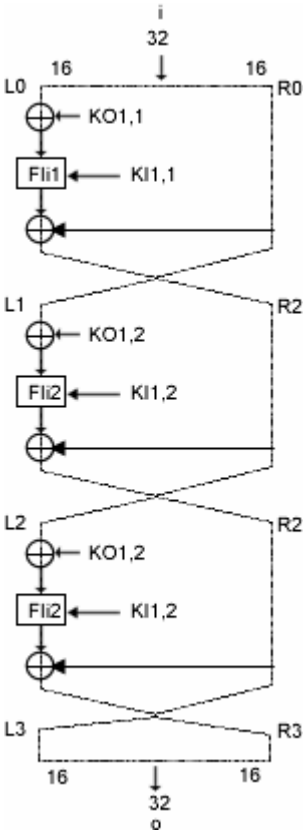

Figura. 3.1.3. Función FO.

De esto se regresa un valor “o” de 32 bits descrita por

$$o = L3 \,||\, R3 \qquad (3.14)$$

La función FI, como lo muestra la Fig. 3.1.4. toma un dato de 16 bits i y una sub-llave KIi,j como entrada. La entrada i es dividida dentro de dos componentes desiguales, una parte de 9 bits L0 y otra de 7 bits R0, donde

$$i = L0 \,||\, R0 \qquad (3.15)$$

Similarmente la llave $KI_{i,j}$ es dividida en una llave $KI_{i,j,1}$ de 9 bits y una llave $KI_{i,j,2}$ donde

$$KIi,j = KI_{i,j,1} \,||\, KI_{i,j,2} \qquad (3.16)$$

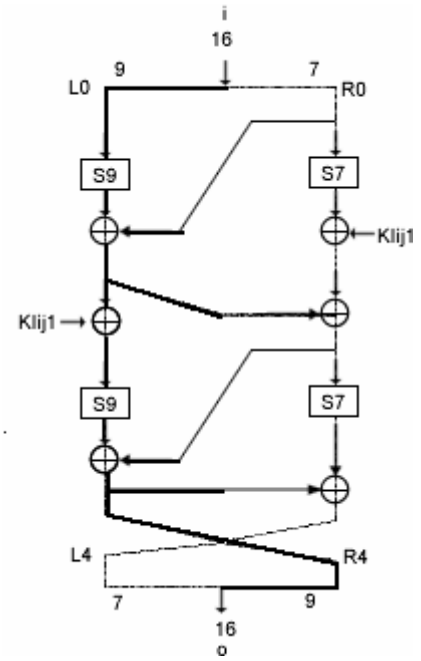

Figura. 3.1.4. Función FI.

La función usa dos funciones S, S7 que toma 7 bits de entrada y saca 7 bits, y S9 que toma 9 bits de entrada y saca 9 bits.

También se usan dos funciones ZE(x) y TR(x). ZE convierte valores de 7 bits a valores de 9 bits añadiendo ceros. TR convierte valores de 9 bits en valores de 7 bits quitando ceros.

A partir de lo anterior definimos

$$L_1 = R_0 \quad (3.17)$$

$$R_1 = S9\,[L_0] \oplus ZE\,(R_0) \quad (3.18)$$

$$L_2 = R_1 \oplus KI_{i,j,2} \quad (3.19)$$

$$R_2 = S7\,[L_1] \oplus TR\,(R_1) \oplus KI_{i,j,1} \quad (3.20)$$

$$L_3 = R_2 \quad (3.21)$$

$$R_3 = S9\,[L_2] \oplus ZE\,(R_2) \quad (3.22)$$

$$L_4 = S7\,[L_3] \oplus TR\,(R_3) \quad (3.23)$$

$$R_4 = R_3 \quad (3.24)$$

La función retorna un valor o de 16 bits definido como:

$$o = (L_4 \,||\, R_4) \quad (3.25)$$

En cuanto a las funciones S7 y S9, toman como entrada un dato x de 7 o 9bits respectivamente, teniendo como salidas un dato de la misma longitud. Por tanto tenemos:

$$x = x8 \,||\, x7 \,||\, x6 \,||\, x5 \,||\, x4 \,||\, x3 \,||\, x2 \,||\, x1 \,||\, x0 \quad (3.26)$$

$$y = y8 \,||\, y7 \,||\, y6 \,||\, y5 \,||\, y4 \,||\, y3 \,||\, y2 \,||\, y1 \,||\, y0 \quad (3.27)$$

Donde los bits x8, y8 y x7, y7 solo aplican para S9. x0 y y0 son los bits menos significativos. El procesamiento en estas funciones están basadas en una lógica combinacional simple que podemos encontrar descrita detalladamente en el articulo “Specification of the 3GPP Confidentiality and Integrity Algorithms” [2].

Una parte importante es la definición de las llaves a partir de la cadena de entrada. La llave de entrada K de 128 bits es dividida en 8 valores de 16 bits que van de K1 hasta K8 donde

$$K = K1 \,||\, K2 \,||\, K3 \,||\ldots||\, K8. \qquad (3.28)$$

Un segundo arreglo de subllaves, K_j' es derivado de K_j por la aplicación de lo siguiente:

Para cada entero j con 1≤i≤8 definimos

$$Kj = Kj \oplus Cj \qquad (3.29)$$

Donde Cj es el valor de las constantes definida en la tabla 3.1.1.

Las llaves de ciclo son entonces derivadas de Kj y Kj´ [1].

La implementación del código KASUMI en c++ viene descrita en la referencia [2].

C1	0x0123
C2	0x4567
C3	0x89AB
C4	0xCDEF
C5	0xFEDC
C6	0xBA98
C7	0x7654
C8	0x3210

Tabla 3.1.1. Constantes.

3.2 ALGORITMO DE CONFIDENCIALIDAD

Dentro de los sistemas 3GPP hay dos algoritmos estándares: El algoritmo de confidencialidad (f8) mostrado en la Fig. 3.2.1 y el algoritmo de integridad f9. Cada uno de estos algoritmos está basado en el algoritmo KASUMI que fue explicado en la sección 3.1. Más para fines de este trabajo solo se implemento el algoritmo f8 que es un cifrador de flujo que es usado para cifrar y descifrar bloques de datos bajo una llave de confidencialidad CK. El bloque de datos puede estar entre 1 a 5114 bits de longitud. Si es mayor existe un overflow en el algoritmo f8. El algoritmo KASUMI es usado en un modo de output-feedback y genera flujos de salida en múltiplos de 64 bits.

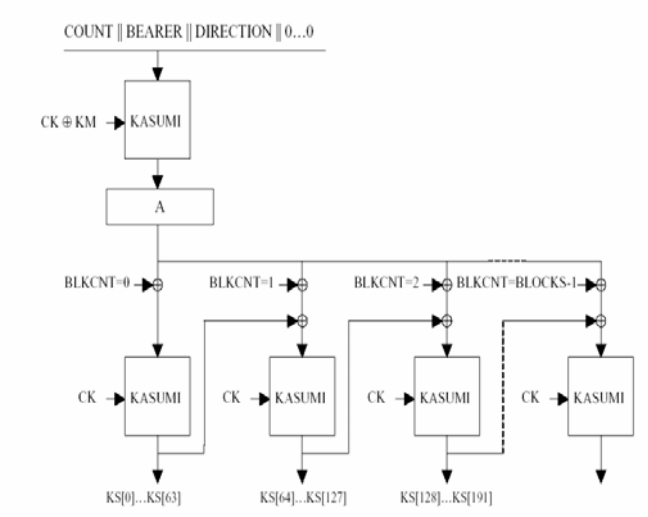

Figura 3.2.1. Algoritmo de confidencialidad f8 [3].

Capítulo VI. APLICACIÓN Y RESULTADOS.

La aplicación constituye la parte principal de este trabajo, pues a través de ella se ilustra el funcionamiento del algoritmo de cifrado KASUMI usado por el algoritmo f8 descritos en la parte 3.1 y 3.2.

Para la implementación de este sistema se utilizó en cuanto al hardware:

- Dos pocket`s.
- Una máquina Pentium III a 500MHz.
- Un punto de acceso.

Con respecto al software se uso

- visual embebed C++ 3.0.

La aplicación consiste en comunicar dos pocket usando un servidor que comunica a ambos, trasladando los mensajes de una pocket a otra.

La arquitectura esencial de la aplicación se ilustra en la Figura 4.1.

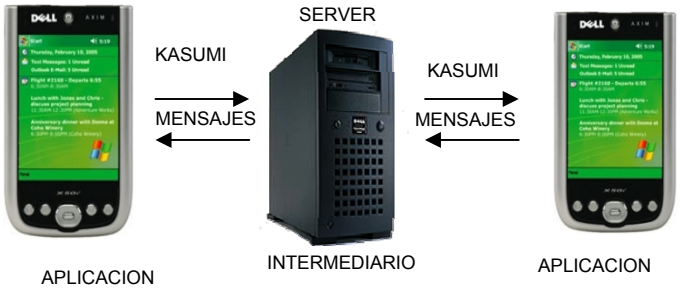

Figura. 4.1. Arquitectura de la aplicación.

La aplicación para las pocket´s que permite escribir mensajes fue realizada en C++. Esta misma aplicación es la que encripta los mensajes usando el algoritmo f8 descrito en la parte 3.2 que hace uso del algoritmo A5/3. Al encriptar los mensajes trasmitidos logramos evitar que aunque sean interceptados estos mensajes no podrán ser legibles.

El servidor corre una aplicación que escucha mediante un socket las peticiones de comunicación de las pocket. El contenido de los mensajes que pasan por el servidor está cifrado, el servidor sólo sirve como intermediario.

La clave confidencial se declara como constante dentro del programa debido a la falta de un protocolo de intercambio de llaves. Esta llave es necesaria tanto para el cifrado como para el descifrado.

4.1. Detalles de Implementación

En la implementación del algoritmo Kasumi, se crearon diversas funciones para el adecuado funcionamiento del envió de mensajes encriptados. La primera función importante con la cual se inicia el proceso es la que manda el mensaje y la clave para encriptar, esta

función se muestra en la figura 4.1.1. Esta función es la más importante pues es la que llama a las demás funciones para encriptar el mensaje enviado. El mensaje que se envía a través de esta función se tiene que convertir al formato ANSI, pues es el que reconocía la aplicación que estaba en el servidor con el sistema operativo WINDOWS XP, que sirve de intermediario entra las pockets. Esta aplicación del servidor al enviar el mensaje hacia las pockets tenia que convertir el mensaje al formato UNICODE para que se pudiera visualizar por en las pockets.

```
void CKasumDlg::OnButonMio(char @mensaje)
{
    UpdateData(TRUE);
    CString m_numero_aux = m_minumero;
    CString m_texto_aux = m_texto;
        u8 clave[16];
        int count = 0xFA556B26;
        int bearer = 0x03;
        int dir = 1;
        u8 data[15];
        int length = 120;

       @data = @mensaje;

        clave[0] = 'a';
        clave[1] = 'b';
        clave[2] = 'r';
        clave[3] = 'a';
        clave[4] = 'c';
        clave[5] = 'a';
        clave[6] = 'd';
        clave[7] = 'a';
        clave[8] = 'b';
        clave[9] = 'r';
        clave[10] = 'a';
        clave[11] = 's';
        clave[12] = 'e';
        clave[13] = 's';
        clave[14] = 'a';
        clave[15] = 'm';

        data[0] = (unsigned char) m_numero_aux.GetAt(0);
        data[1] = (unsigned char) m_numero_aux.GetAt(1);
        data[2] = (unsigned char) m_numero_aux.GetAt(2);
        data[3] = (unsigned char) m_numero_aux.GetAt(3);
        data[4] = (unsigned char) m_numero_aux.GetAt(4);
        data[5] = (unsigned char) m_numero_aux.GetAt(5);
        data[6] = (unsigned char) m_numero_aux.GetAt(6);
        data[7] = (unsigned char) m_numero_aux.GetAt(7);
        data[8] = (unsigned char) m_numero_aux.GetAt(8);
        data[9] = (unsigned char) m_numero_aux.GetAt(9);
        data[10] = (unsigned char) m_numero_aux.GetAt(10);
        data[11] = (unsigned char) m_numero_aux.GetAt(11);
        data[12] = (unsigned char) m_numero_aux.GetAt(12);
        data[13] = (unsigned char) m_numero_aux.GetAt(13);
        data[14] = (unsigned char) m_numero_aux.GetAt(14);

        f8(clave,count,bearer,dir,data,length);
        m_numero_aux = (CString)data;
        f8(clave,count,bearer,dir,data,length);
    m_texto_aux = (CString)data;
        m_minumero = m_numero_aux;
        m_texto = m_texto_aux;
    UpdateData(FALSE);                      // de programa a dialogo
}
```

Figura 4.1.1. Función de envió de mensajes.

En la función de la figura 4.1.1se muestra que se manda a llamar a la función f8, esta función es la que implementa el algoritmo de confidencialidad explicada en la sección 3.2, y que se muestra en la figura 4.1.2.

```
void f8( u8 *key, int count, int bearer, int dir, u8 *data, int length )
{
     REGISTER64 A;         /* the modifier         */
     REGISTER64 temp;      /* The working register */
     int i, n;
     u8 ModKey[16];        /* Modified key      */
     u16 blkcnt;           /* The block counter */
     /* Start by building our global modifier */
     temp.b32[0]   = temp.b32[1]  = 0;
     A.b32[0]      = A.b32[1]     = 0;
     /* initialise register in an endian correct manner*/
     A.b8[0] =   (u8) (count>>24);
     A.b8[1] =   (u8) (count>>16);
     A.b8[2] =   (u8) (count>>8);
     A.b8[3] =   (u8) (count);
     A.b8[4] =   (u8) (bearer<<3);
     A.b8[4] |=  (u8) (dir<<2);
     /* Construct the modified key and then "kasumi" A */
     for( n=0; n<16; ++n )
         ModKey[n] = (u8)(key[n] ^ 0x55);
     KeySchedule( ModKey );
     Kasumi( A.b8 ); /* First encryption to create modifier */
     /* Final initialisation steps */
     blkcnt = 0;
     KeySchedule( key );
     /* Now run the block cipher */
     while( length > 0 )
     {
         /* First we calculate the next 64-bits of keystream */
         /* XOR in A and BLKCNT to last value */
         temp.b32[0] ^= A.b32[0];
         temp.b32[1] ^= A.b32[1];
         temp.b8[7] ^= blkcnt;
         /* KASUMI it to produce the next block of keystream */
         Kasumi( temp.b8 );
         /* Set <n> to the number of bytes of input data   *
          * we have to modify. (=8 if length <= 64)        */
         if( length >= 64 )
              n = 8;
         else
              n = (length+7)/8;
         /* XOR the keystream with the input data stream */
         for( i=0; i<n; ++i )
              *data++ ^= temp.b8[i];
         length -= 64;     /* done another 64 bits */
         ++blkcnt;         /* increment BLKCNT     */
     }
}
```

Figura 4.1.2. Función que implementa el algoritmo de confidencialidad.

La función f8 hace uso de la función KeySchedule la cual sirve para hacer un tratamiento especial a la clave utilizada en el proceso de encriptación, el código de esta función se muestra en la figura 4.1.3. Además la función f8 hace un llamado al algoritmo de encriptación

KASUMI, cuyo código se muestra en la figura 4.1.4, junto con las funciones FI, FO, FL, explicadas en la sección 3.1.

```
void KeySchedule( u8 *k )
{


       static u16 C[] = {
         0x0123,0x4567,0x89AB,0xCDEF, 0xFEDC,0xBA98,0x7654,0x3210 };
     u16 key[8], Kprime[8];
     WORD1 *k16;
     int n;

/*Start by ensuring the subkeys are endian correct on a 16-bit basis*/
     k16 = (WORD1 *)k;
     for( n=0; n<8; ++n )
         key[n] = (u16)((k16[n].b8[0]<<8) + (k16[n].b8[1]));
     /* Now build the K'[] keys */
     for( n=0; n<8; ++n )
         Kprime[n] = (u16)(key[n] ^ C[n]);
     /* Finally construct the various sub keys */
     for( n=0; n<8; ++n )
     {
         KLi1[n] = ROL16(key[n],1);
         KLi2[n] = Kprime[(n+2)&0x7];
         KOi1[n] = ROL16(key[(n+1)&0x7],5);
         KOi2[n] = ROL16(key[(n+5)&0x7],8);
         KOi3[n] = ROL16(key[(n+6)&0x7],13);
         KIi1[n] = Kprime[(n+4)&0x7];
         KIi2[n] = Kprime[(n+3)&0x7];
         KIi3[n] = Kprime[(n+7)&0x7];
     }
}
```

Figura 4.1.3. Función KeySchedule.

```
void Kasumi( u8 *data )
{
     u32 left, right, temp;
     DWORD1 *d;
     int n;

 /* Start by getting the data into two 32-bit words (endian corect) */
    d = (DWORD1*)data;

     left = (d[0].b8[0]<<24)+(d[0].b8[1]<<16)+(d[0].b8[2]<<8)+(d[0].b8[3]);
     right = (d[1].b8[0]<<24)+(d[1].b8[1]<<16)+(d[1].b8[2]<<8)+(d[1].b8[3]);
     n = 0;
     do{
             temp = FL( left, n    );
         temp = FO( temp, n++ );
         right ^= temp;
         temp = FO( right, n    );
         temp = FL( temp,    n++ );
         left ^= temp;
     }while( n<=7 );
     /* return the correct endian result */
     d[0].b8[0] = (u8)(left>>24);     d[1].b8[0] = (u8)(right>>24);
     d[0].b8[1] = (u8)(left>>16);     d[1].b8[1] = (u8)(right>>16);
     d[0].b8[2] = (u8)(left>>8);      d[1].b8[2] = (u8)(right>>8);
     d[0].b8[3] = (u8)(left);         d[1].b8[3] = (u8)(right);

}

static u16 FI( u16 in, u16 subkey )
{
     u16 nine, seven;
```

```
      static u16 S7[] = {
         54, 50, 62, 56, 22, 34, 94, 96, 38, 6, 63, 93, 2, 18,123, 33,
         55,113, 39,114, 21, 67, 65, 12, 47, 73, 46, 27, 25,111,124, 81,
         53, 9,121, 79, 52, 60, 58, 48,101,127, 40,120,104, 70, 71, 43,
         20,122, 72, 61, 23,109, 13,100, 77, 1, 16, 7, 82, 10,105, 98,
         117,116, 76, 11, 89,106, 0,125,118, 99, 86, 69, 30, 57,126, 87,
         112, 51, 17, 5, 95, 14, 90, 84, 91, 8, 35,103, 32, 97, 28, 66,
         102, 31, 26, 45, 75, 4, 85, 92, 37, 74, 80, 49, 68, 29,115, 44,
         64,107,108, 24,110, 83, 36, 78, 42, 19, 15, 41, 88,119, 59, 3};
      static u16 S9[] = {
        167,239,161,379,391,334, 9,338, 38,226, 48,358,452,385, 90,397,
        183,253,147,331,415,340, 51,362,306,500,262, 82,216,159,356,177,
        175,241,489, 37,206, 17, 0,333, 44,254,378, 58,143,220, 81,400,
        95, 3,315,245, 54,235,218,405,472,264,172,494,371,290,399, 76,
        165,197,395,121,257,480,423,212,240, 28,462,176,406,507,288,223,
        501,407,249,265, 89,186,221,428,164, 74,440,196,458,421,350,163,
        232,158,134,354, 13,250,491,142,191, 69,193,425,152,227,366,135,
        344,300,276,242,437,320,113,278, 11,243, 87,317, 36, 93,496, 27,
        487,446,482, 41, 68,156,457,131,326,403,339, 20, 39,115,442,124,
        475,384,508, 53,112,170,479,151,126,169, 73,268,279,321,168,364,
        363,292, 46,499,393,327,324, 24,456,267,157,460,488,426,309,229,
        439,506,208,271,349,401,434,236, 16,209,359, 52, 56,120,199,277,
        465,416,252,287,246, 6, 83,305,420,345,153,502, 65, 61,244,282,
        173,222,418, 67,386,368,261,101,476,291,195,430, 49, 79,166,330,
        280,383,373,128,382,408,155,495,367,388,274,107,459,417, 62,454,
        132,225,203,316,234, 14,301, 91,503,286,424,211,347,307,140,374,
        35,103,125,427, 19,214,453,146,498,314,444,230,256,329,198,285,
        50,116, 78,410, 10,205,510,171,231, 45,139,467, 29, 86,505, 32,
        72, 26,342,150,313,490,431,238,411,325,149,473, 40,119,174,355,
        185,233,389, 71,448,273,372, 55,110,178,322, 12,469,392,369,190,
        1,109,375,137,181, 88, 75,308,260,484, 98,272,370,275,412,111,
        336,318, 4,504,492,259,304, 77,337,435, 21,357,303,332,483, 18,
        47, 85, 25,497,474,289,100,269,296,478,270,106, 31,104,433, 84,
        414,486,394, 96, 99,154,511,148,413,361,409,255,162,215,302,201,
        266,351,343,144,441,365,108,298,251, 34,182,509,138,210,335,133,
        311,352,328,141,396,346,123,319,450,281,429,228,443,481, 92,404,
        485,422,248,297, 23,213,130,466, 22,217,283, 70,294,360,419,127,
        312,377, 7,468,194, 2,117,295,463,258,224,447,247,187, 80,398,
        284,353,105,390,299,471,470,184, 57,200,348, 63,204,188, 33,451,
        97, 30,310,219, 94,160,129,493, 64,179,263,102,189,207,114,402,
        438,477,387,122,192, 42,381, 5,145,118,180,449,293,323,136,380,
        43, 66, 60,455,341,445,202,432, 8,237, 15,376,436,464, 59,461};

      /* The sixteen bit input is split into two unequal halves,  *
       * nine bits and seven bits - as is the subkey              */

      nine = (u16)(in>>7);
      seven = (u16)(in&0x7F);

    /* Now run the various operations */
    nine = (u16)(S9[nine] ^ seven);
    seven = (u16)(S7[seven] ^ (nine & 0x7F));
    seven ^= (subkey>>9);
    nine ^= (subkey&0x1FF);
    nine = (u16)(S9[nine] ^ seven);
    seven = (u16)(S7[seven] ^ (nine & 0x7F));
    in = (u16)((seven<<9) + nine);
    return( in );
}

/*---------------------------------------------------------------------
    * FO()
    *      The FO() function.
    *      Transforms a 32-bit value. Uses <index> to identify the
    *      appropriate subkeys to use.
    *---------------------------------------------------------------------*/
static u32 FO( u32 in, int index )
{
      u16 left, right;
      /* Split the input into two 16-bit words */
      left = (u16)(in>>16);
      right = (u16) in;
      /* Now apply the same basic transformation three times        */
      left ^= KOi1[index];
```

```
      left = FI( left, KIi1[index] );
      left ^= right;
      right ^= KOi2[index];
      right = FI( right, KIi2[index] );
      right ^= left;
      left ^= KOi3[index];
      left = FI( left, KIi3[index] );
      left ^= right;
      in = (right<<16)+left;
      return( in );
}

/*---------------------------------------------------------------------
  * FL()
  *      The FL() function.
  *      Transforms a 32-bit value. Uses <index> to identify the
  *      appropriate subkeys to use.
  *---------------------------------------------------------------------*/
static u32 FL( u32 in, int index )
{
      u16 l, r, a, b;
      /* split out the left and right halves */
      l = (u16)(in>>16);
      r = (u16)(in);
      /* do the FL() operations            */
      a = (u16) (l & KLi1[index]);
      r ^= ROL16(a,1);
      b = (u16)(r | KLi2[index]);
      l ^= ROL16(b,1);
      /* put the two halves back together */
      in = (l<<16) + r;
      return( in );
}
```

Figura 4.1.4. Implementación del Algoritmo KASUMI, con las funciones FO, FI Y FL.

En la implementación de la aplicación el principal problema que se tuvo fue que WINDOWS XP utiliza el formato ANSI en el uso de los caracteres y WINDOWS CE utiliza el formato UNICODE. Este problema provocaba que el cifrado mandado por la pocket hacia el servidor no se descifrara bien por la aplicación del servidor, y viceversa, aunque los datos si llegaran a su destino.

También otra dificultad fue idear una manera de pasar los mensajes a través del servidor y realizar una interfaz que permitiera la comunicación entre las pockets.

Capítulo V. Conclusiones

El trabajo permitió conocer a profundidad la arquitectura del algoritmo KASUMI, que es utilizado comúnmente en las aplicaciones de tercera generación como mecanismo de confidencialidad e integridad.

Se logró implementar en una aplicación inalámbrica el algoritmo de confidencialidad F8 que usa como principal rutina el A5/3.

La implementación inalámbrica con las pda´s y una computadora pentium nos permitió darnos cuenta de que existen aplicaciones que necesitan conversión de datos para su buen funcionamiento debido al formato ANSI usado en WINDOWS para PC y el formato UNICODE usado en WINDOWS CE.

La implementación en aplicaciones se realiza mediante los algoritmos f8 y f9, que tienen como función principal el algoritmo KASUMI. La implementación del algoritmo KASUMI por si sola es laboriosa.

BIBLIOGRAFIA

[1] C. Balderas, “An Efficient Hardware Implementation of the KASUMI Block Ciphier for Third Generation Cellular Networks”.

[2] Specification of the 3GPP Confidentiality and Integrity Algorithms. Documento 2: KASUMI Specification.

[3] Specification of the 3GPP Confidentiality and Integrity Algorithms.

[4] Matsui M. Block encryption algorithm MISTY. In Fast Software Encryption, 4th International Workshop, FSE '97, LNCS 1267, pages 64–74. Springer-Verlag, 1997.

[5] Schneier B., Applied Cryptography. Protocols, Algorithms, and Source Code in C. Wiley, USA, segunda edición, 1996.

[6] Zimmermann. An Introduction to Cryptography.

[7] Schneier B., Applied Cryptography. Protocols, Algorithms, and Source Code in C. Wiley, USA, segunda edición, 1996.

[8] Lucena L., Criptografía y Seguridad en Computadores. Tercera Edición, Versión 1.00, 2001. Libro electrónico. Disponible en: http://wwwdi.ujaen.es/~mlucena/

[9] Artech House Publishers.User's Guide To Cryptography And Standards (Artech House Computer Security)

[10] Menezes J., Oorschot V. and Vanstone A. Handbook of Applied Cryptography. CRC Press,New York, quinta edición, 2001.

[11] http://www.revista.unam.mx/vol.7/num7/art55/jul_art55.pdf

[12] Jhonson S. Cryptography For Developers. Tom St Denis. 2007.

[13] Data Encryption Standar (DES). Federal Information Processing Standards Publication 46-3 (FIPS PUB 46-3). October 25, 1999. Disponible en: http://csrc.nist.gov/publications/fips/fips46-3/fips463.pdf

[14] Cormen H., Leiserson E., Rivest L., and Stein C.. Introduction to Algorithms, Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0262032937. Section 31.7: The RSA public-key cryptosystem, pp.881–887.

[15] Preneel B. “On Fibonacci keystream generators”, editor, Fast Software Encryption. Second International Workshop (LNCS 1008), 346–352, Springer-Verlag, 1995.

[16] Menezes J.,Oorschot P. y Vanstone S.. Handbook of Applied Cryptography. CRC Press, New York, quinta edición, 2001.

[17] Washington C., Introduction to Cryptography with Coding Theory. Prentice Hall, New Jersey, 2002.

[18] López C. Sistema de Seguridad para Intercambio de Datos en Dispositivos Móviles. Departamento de Ingeniería Eléctrica-Sección de Computación. Abril 2006

[19] http://www.cordobawireless.net/portal/descargas/Wireless_intro.pdf

[20] Gast M. 802.11 Wireles Networks The Definitive Guide. Editorial O'Reilly. ISBN 596001835.

[21] http://www.inf.utfsm.cl/~jcanas/ramos/Redes/Apuntes/Capitulo5

b.pdf

[22] Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197(FIPS PUB197). November 26, 2001. Disponible en: http:// csrc.nist.gov/publications/fips/fips197/fips197.pdf

[23] Petrovic S., Fúster A. Criptoanálisis del algoritmo A5/2 para telefonía móvil. Primer Congreso Iberoamericano de Seguridad Informática CIBSI'02, Morelia, México. 2002.

[24] Diffie W. and M. Hellman M., New directions in cryptography. IEEE Transaction on Information Theory,IT-22(6), pp. 644–654, 1976.