# Universidad Austral de Chile

Facultad de Ciencias de la Ingeniería

Escuela de Ingeniería Civil en Informática


# DISEÑO DE UNA PLATAFORMA DE COMUNICACIÓN MILITAR PARA TRANSMISIONES DE DATOS DIGITALES A TRAVÉS DE ONDAS DE RADIO FRECUENCIA

Tesis para optar al Título de:
Ingeniero Civil en Informática


Profesor Patrocinante:
Sr. Luis Hernán Vidal Vidal
Ingeniero Civil en Informática.
Licenciado en Ciencias de la Ingeniería

Profesor Co-Patrocinante:
Sr. German Loyola Núñez
Capitán de Ejército
Ejército de Chile

Profesor Patrocinante:
Sr. Jorge Morales Vilugron
Ingeniero Electrónico
Diplomado en Ingeniería, especialidad en Electricidad

MAURICIO ALEJANDRO DUARTE OJEDA



VALDIVIA - CHILE
2009

# Universidad Austral de Chile

Instituto de Informática

Valdivia, 31 de Marzo de 2008.

De : Luis Hernán Vidal Vidal.
A : Sr. Juan Pablo Salazar F.
Director de Escuela de Ingeniería Civil en Informática.
Ref. : Informa Calificación Trabajo de Titulación.

MOTIVO: Informar revisión y calificación del Proyecto de Título "Desarrollo de una plataforma de comunicación militar para transmisiones de datos digitales a través de ondas de radio que sea capaz de proporcionar un nivel seguridad y un sistema de monitoreo y gestión de los datos tanto transmitidos como recibidos.", presentado por el alumno MAURICIO ALEJANDRO DUARTE OJEDA, que refleja lo siguiente:

Se logró el objetivo planteado que permitió el desarrollo de una plataforma de comunicación militar para transmisiones de datos digitales a través de ondas de radio proporcionando un nivel seguridad y un sistema de monitoreo y gestión de los datos tanto transmitidos como recibidos. El proyecto se implemento y se expuso en un congreso Internacional de Telecomunicaciones (Senacitel 2006), siendo muy bien evaluado, lo que adicionalmente se presenta como un muy buen logro alcanzado.

El esfuerzo constante, junto con la dedicación al tema son aspectos destacables durante el proceso del trabajo realizado.

| | |
|---|---|
| Cumplimiento del objetivo propuesto. | 7,0 |
| Satisfacción de alguna necesidad. | 7,0 |
| Aplicación del método científico. | 7,0 |
| Interpretación de los datos y obtención de conclusiones. | 7,0 |
| Originalidad. | 7,0 |
| Aplicación de criterios de análisis y diseño. | 7,0 |
| Perspectivas del trabajo. | 7,0 |
| Coherencia y rigurosidad lógica. | 7,0 |
| Precisión del lenguaje técnico en la exposición, composición, redacción e ilustración. | 7,0 |
| **Evaluación Tesis.** | **7,0** |

Por todo lo anterior expuesto califico el trabajo de titulación del Sr. MAURICIO ALEJANDRO DUARTE OJEDA con nota 7,0 (siete coma cero).

Sin otro particular, se despide atentamente.

UNIVERSIDAD AUSTRAL DE CHILE
INSTITUTO DE INFORMATICA
FACULTAD DE CIENCIAS DE LA INGENIERIA

Ing. Luis Hernán Vidal Vidal.
Profesor Instituto de Informática.
Facultad de Ciencias de la Ingeniería.
Universidad Austral de Chile.

*Valdivia, 08 Abril del 2008*

*DEL: CAPITAN GERMAN LOYOLA NUÑEZ*

*AL : DIRECTOR ESCUELA INGENIERÍA CIVIL EN INFORMÁTICA*

*MOTIVO: INFORME TRABAJO DE TITULACIÓN*

---

*Por la presente quisiera expresar lo grato que es, ser participe del trabajo efectuado tanto de evaluación y patrocinio, no me cabe ninguna duda que la retroalimentación y los conocimientos compartidos han sido de gran provecho en forma conjunta, la cual a significado un gran aporte e integración*

*Junto con reiterar el compromiso de este oficial, aprovecho la oportunidad de felicitarlo, por la gran calidad profesional y humana que cuentan los docentes y alumnos egresados de esa facultad, la cual fue reflejada en una excelente experiencia durante estos dos años por parte del evaluado.*

*Nombre del Alumno: SR. MAURICIO ALEJANDRO DU ARTE OJEDA*

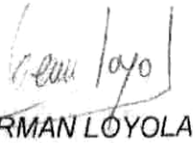
*Nombre Trabajo de Titulación: Diseño de una Plataforma de Comunicación Militar para Transmisiones de Datos Digitales a Través de Ondas de Radio Frecuencia.*

*Nota: 7.0 (SIETE)*

*FUNDAMENTO DE LA NOTA:*

*Cumplimiento del objetivo propuesto por la unidad regimentaría a entera satisfacción, apego estricto a los criterios y necesidades solicitadas, adecuándose o las terminologías y estructura militar, con gran innovación y obtención de resultados concretos.*

*Saluda a UD.*



GERMAN LOYOLA NUÑEZ
Capitán
Oficial de Ejército

VALDIVIA, 13 de Enero de 2009

De: JORGE ANTONIO MORALES VILUGRON
INFORMANTE

A: Juan Pablo Sal azar Fernández
Director
Escuela de Ingeniería Civil en Informática

Ref: Calificación proyecto de título

De mi consideración:

**Habiendo revisado el trabajo de titulación "Diseño de una Plataforma de Comunicación Militar para Transmisiones de Datos Digitales a Través de Ondas de Radio Frecuencia, presentado por el/la alumno(a) sr./sra./srta. MAURICIO ALEJANDRO DUARTE OJEDA, mi evaluación del mismo es la siguiente:**

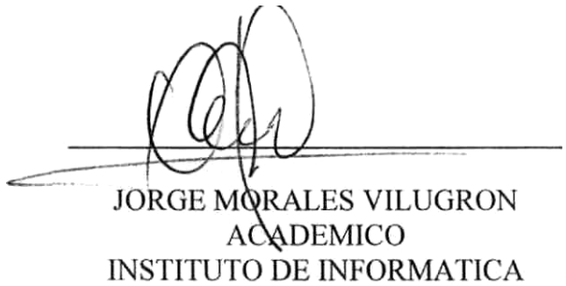
**Nota: 6,5** (Seis como cinco).

**Fundamento de la nota:**

Una tesis que en sus aspectos técnicos estaba lista hace dos años, se toma mucho tiempo en concluirla y por su alto componente de electrónica requirió ser revisada muchas veces.

| Aspecto | **Evaluación** |
|---|---|
| Cumplimiento de objetivos | 7.0 |
| Satisfacción de alguna necesidad | 6.5 |
| Aplicación del método científico | 6.5 |
| Interpretación de los datos y obtención de conclusiones | 6.5 |
| Originalidad | 6.5 |
| Aplicación de criterios de análisis y diseño | 6.5 |
| Perspectivas del trabajo | 6.0 |
| Coherencia y rigurosidad lógica | 6.5 |
| Precisión del lenguaje técnico | 6.5 |

Sin otro particular, saluda atentamente a usted,



JORGE MORALES VILUGRON
ACADEMICO
INSTITUTO DE INFORMATICA

Dedicada a mi madre Avelina, por todo su apoyo, comprensión y amor. A mi padre Oscar que siempre ha estado presente.

A mi hermana Vanessa, por su apoyo permanente y cariño.

# Contenidos

# Resumen

En este documento de tesis se presenta el análisis, diseño y construcción de un prototipo de una plataforma de comunicación para transmisiones de datos por ondas de radio frecuencia, orientado al ambiente militar, para ello se separó el proyecto en tres etapas: implementación de un MODEM para transmisiones por radio frecuencia, análisis de algoritmos criptográficos y finalmente implementación de un software para la gestión de la información.

Para las etapas que involucraron desarrollo de software se uso la herramienta de modelamiento UML [Dob+, 2006], los requerimientos fueron recopilados en una unidad de telecomunicaciones del Ejército de Chile, en base a las necesidades de esta rama de la institución y en forma especifica de esta unidad.

Se trabajo en forma permanente con el personal asignado por el Ejército de Chile, además la institución proporcionó los elementos técnicos, información y protocolos de seguridad involucrados. Finalmente se validaron los resultados obtenidos a través de múltiples pruebas y se paso a la etapa de producción dentro de la institución el software de gestión y criptografia.

# Summary

In this thesis paper presents the analysis, design and construction of a prototype of a communication platform for data transmissions by frequency radio waves, oriented military environment, is separating the project into three stages: implementation of a MODEM for transmission by radio frequency, implementation of a software for information management and finally analyses of cryptographic algorithms.

To stages involving software development will use the modelling tool UML [Dob+, 2006], the requirements were collected into a telecommunications unit of the Army of Chile, based on the requirements of this area of the institution and in form specified in this unit.

It constantly work with the personnel assigned by the Army of Chile, besides the institution provides the technical elements, information and security protocols involved in the project. Finally the results were validated through multiple tests and the institution move to the stage of the production management software and cryptography.

# 1. Introducción

La comunicación, es sin duda una de las herramientas más importantes para una institución privada o gubernamental, ya sea con o sin fines de lucro, producto de que los procesos de comunicación son indispensables en un mundo moderno y competitivo sin dejar de lado el rápido crecimiento de las tecnologías asociadas, esto nos lleva a una necesidad de estar constantemente analizando el tema de las comunicaciones internas de las organizaciones, en cuanto a su imperiosa necesidad de automatizar y optimizar sus procesos de comunicaciones, con el objetivo de sostener y mejorar su rapidez, confidencialidad y integridad de sus datos.

La confidencialidad es uno de los factores críticos de las transmisiones de datos, producto de que constantemente hay entes que tratan de interferir e interceptar información para beneficio propio o sólo para causar problemas a la institución, en segundo lugar se encuentra la integridad y la rapidez en la transferencia de datos, en cuanto a la integridad es necesario que los datos que se transmiten desde un sistema transmisor hacia un sistema receptor sean los mismos en ambos extremos, por otro lado la velocidad del medio de transmisión no es un tema menor, producto que generalmente un sistema de comunicación se evalúa en primera instancia por su velocidad y este factor cobra mas relevancia dependiendo del rubro de las instituciones, y de los servicios que se desea entregar a los usuarios.

El Ejército de Chile es una institución gubernamental, la cual esta inmersa en el área de la defensa nacional, a su vez esta institución esta conformada por unidades más pequeñas, ya sea del área de armas especificas o de servicios, entre las unidades en el área de armas se encuentran las Telecomunicaciones, que tienen básicamente la responsabilidad de mantener una comunicación constante entre todas las unidades de

la institución, ya sea en tiempos de paz y principalmente cuando se enfrenta un ambiente bélico.

“El valor de las comunicaciones toma un alto valor para la institución en un ambiente bélico, ya que de ello dependerá una ventaja consistente ante un enemigo potencial, producto de que para la toma de decisiones se necesita información, mientras más grande sea la diferencia entre decisión correcta y errónea, mayor será la importancia de contar con una buena información”. [Bit, 2002].

En la práctica, las circunstancias de una decisión mal tomada por falta de información pueden traer consecuencias catastróficas dentro de la institución, por esta razón constantemente se realizan los esfuerzos necesarios para mejorar la calidad y disponibilidad de la información que sus propios procesos generan. En el mundo de la defensa los 3 factores mencionados anteriormente toman una relevancia enorme, producto de que la información es de índole militar.

Otro factor a considerar es el medio de transmisión de la información, el cual puede ser alámbrico o inalámbrico, en el medio de transmisión alámbrico hoy en día existen una serie de soluciones que dependen básicamente de la velocidad, capacidad, entre otras. El medio de transmisión inalámbrico esta compuesto principalmente por ondas de radio frecuencias y aquí el factor distancia es una limitante en la elección de la banda base a utilizar.

Durante las últimas décadas la Informática y las Telecomunicaciones han provisto de las tecnologías necesarias para permitir a las instituciones mejorar sus procesos y tecnologías de comunicaciones, ya sea incorporando nuevos servicios o mejorando los ya existentes. Es así, como han aparecido una serie de protocolos y estándares tales como Radio Paquetes, IEEE 802.11 [Ohr+, 2003], y en cuanto a la seguridad han aparecido una serie de algoritmos criptográficos y herramientas capaces de dar solución a este problema.

Las tecnologías criptográficas se han aplicado mayoritariamente en ambientes alámbricos básicamente por instituciones que tienen dependencias geográficamente distantes, en las transmisiones inalámbricas, las mayorías de las soluciones han salido para estándares IEEE 802.11, principalmente para el estándar WIFI ó 802.11b, dentro de este se han aplicado mayoritariamente tecnologías VPNs y soluciones basadas en el estándar de seguridad 801.11x tanto para el tema de cifrado como para la autenticación de clientes, por otra parte en el tema de radio frecuencia no se han desarrollado muchas soluciones o son poco conocidas, lo más conocido es el uso de radio paquetes utilizado por empresas de telecomunicaciones y radioaficionados.

Creemos que el Ejército de Chile debe realizar los esfuerzos necesarios para mejorar sus procesos de comunicación, principalmente en unidades de Telecomunicaciones, para ello debe incorporar nuevas tecnologías de información y sistemas de información.

Nos proponemos demostrar que es posible aplicar tecnologías generalmente asociadas al ámbito de redes de datos alámbricas e inalámbricas en el área de transmisiones de radio frecuencia.

El objetivo del documento es presentar una metodología formal para la construcción de una plataforma de comunicación enfocado a instituciones de índole militar, las cuales debido al tipo de información que manejan requieren de una serie de políticas de seguridad.

Proponemos una metodología evolutiva, lo cual significa que la institución dependiendo de su desarrollo tenga la posibilidad de ir realizando mejoras ya sea con la finalidad de hacer pruebas o para subir sus estándares de seguridad.

## 1.1 Objetivos

### 1.1.1 Objetivos generales

- Desarrollo de una plataforma de comunicación militar para transmisiones de datos digitales a través de ondas de radio, que sea capaz de proporcionar un nivel seguridad y un sistema de monitoreo y gestión de los datos tanto transmitidos como recibidos.

### 1.1.2 Objetivos específicos

- Analizar el funcionamiento de las transmisiones de datos por ondas de radio frecuencia.
- Analizar las técnicas y herramientas criptográficas relacionadas con las transmisiones de datos a través de ondas de radio frecuencia.
- Diseñar y construir una plataforma que permita transmitir, recibir, gestionar y cifrar transmisiones de datos de índole militar por ondas de radio frecuencia.
- Evaluar la plataforma implementada y concluir si cumple con los estándares y requisitos exigidos por la institución patrocinadora del proyecto.

# Capitulo 1: Modelo de Desarrollo de un Modulador y Demodulador de Información para Transmisiones de Radio Frecuencia

# 1. Aspectos de las transmisiones de radio frecuencia

## 1.1 Características generales

Las transmisiones de radio frecuencia se identifican principalmente porque su medio no es guiado y es a través de ondas electromagnéticas, el medio puede ser el aire o el vacío, alcanzando una velocidad aproximada de 300.000 Km/s. Para enviar algún tipo de información a través de este medio, es necesario que dicha información pase por un proceso de conversión, es decir que las señales digitales se transformen en señales analógicas para que se puedan adaptar y enviar por el medio.

En todo sistema de transmisión por radio, debe existir un transmisor y una antena asociada al mismo, este emite entre su potencia de salida a la antena, la que genera una señal hacia el exterior. El proceso contrario se da cuando una antena receptora captura las señales y las deriva a un equipo capaz de extraer la información contenida en la misma, entre ambas antenas se propagan las señales electromagnéticas. Las ondas de radio tienen tres formas de propagarse. La primera es la denominada propagación por onda terrestre, la segunda es la propagación por línea recta o alcance visual, y la tercera es la propagación por onda espacial. El espectro de radiofrecuencias hace referencia a cómo está dividido todo el ancho de banda que se puede emplear para transmitir diversos tipos de señales. Existe una reglamentación que asignan determinadas frecuencias a determinados tipos de transmisión de información.

| Frecuencia | Denominación | Abr. | Longitud de onda |
|---|---|---|---|
| **3-30 kHz** | Frecuencia muy baja | VLF | 100.000-10.000 m |
| **30-300 kHz** | Frecuencia baja | LF | 10.000-1.000 m |
| **300-3.000 kHz** | Frecuencia media | MF | 1.000-100 m |
| **3-30 MHz** | Frecuencia alta (onda corta) | HF | 100-10 m |
| **30-300 MHz** | Frecuencia muy alta | VHF | 10-1 m |
| **300-3000 MHz** | Frecuencia ultraelevada | UHF | 1 m-10 cm |
| **3-30 GHz** | Frecuencia superelevada | SHF | 10-1 cm |
| **30-300 GHz** | Frecuencia extremadamente alta | EHF | 1 cm-1 mm |

**Tabla 1.1.1 Tabla de clasificación de frecuencias**

### 1.1.1 Espectro de una señal y su ancho de banda

Se dice que el espectro de la señal sinusoidal está constituido por un único punto, su frecuencia, con un valor asignado que es su amplitud.

Si en vez de tomar una señal sinusoidal consideramos la suma de n señales sinusoidales, cada una con una frecuencia $f_n$, su espectro estará formado por una función variable en la frecuencia con n puntos. Cada una de las tuplas ( $f_n, a_n$) se pueden definir como (frecuencia, amplitud) o (frecuencia, fase) [Opp+, 1994].

El espectro de una señal, es una doble función, una correspondiente a (frecuencia – amplitud) y la otra (frecuencia – fase). Esta función que representa el peso que cada frecuencia tiene en la formación de cada señal, tanto en amplitud, como en fase, se llama espectro de la señal**.**

Por tanto tenemos que cualquier señal tiene una doble representación:

- Señal en el dominio del tiempo.
- Señal en el dominio de la frecuencia, es decir, su espectro.

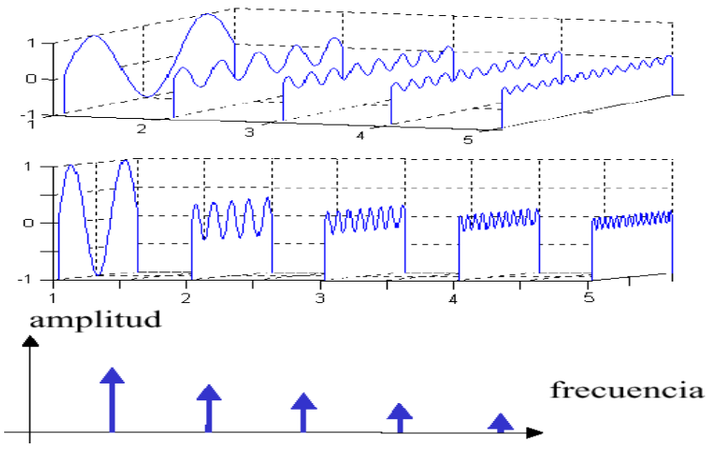



**Fig. 1.1.1 Representación de señales en el dominio del tiempo y la frecuencia**

Para lograr visualizar o representar una señal en el dominio de la frecuencia se utiliza una transformada de Fourier, la cual hace un cambio de espacio de representación del tiempo a la frecuencia dado por $f(\omega) = F\{x(t)\}$, donde

$F(\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt$ y $x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)e^{j\omega t}d\omega$ [Zoh+, 1995].

Al obtenerse la transformada de Fourier de una señal, esta ocupará una cierta región en el espacio de la frecuencia, también conocido como espectro. El ancho de esta región o banda estará en función directa de la cantidad de información de la señal en el tiempo. Por lo tanto el ancho de banda, es una medida de la capacidad de transmisión de información, en términos de la utilización de espectro.

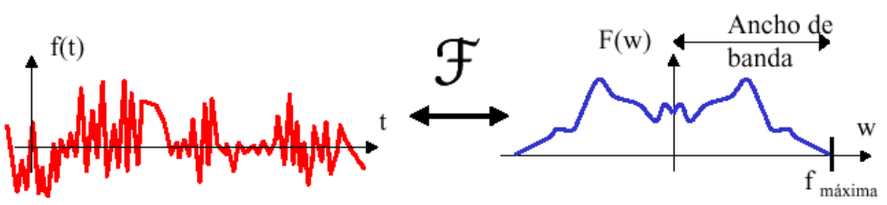



**Fig. 1.1.2 Representación del ancho de banda de una señal**

De modo análogo, se puede definir el ancho de banda de un canal como la diferencia entre las frecuencias máximas y mínimas que es capaz de transmitir. El

canal transmitirá todas aquellas señales cuyo espectro esté incluido dentro del ancho de banda del canal. Si una parte del espectro de la señal cae fuera del ancho de banda del canal, la transmisión será imposible o no será de fidelidad.

### 1.1.2 Problemas de la transmisión de señales en radio frecuencia

Hay una serie de factores que intervienen en el proceso de transmisión de señales y que deforman o alterán las mismas, estas contaminaciones o deformaciones pueden conducir a pérdidas de información, ya que los mensajes no llegan a sus destinos con integridad. Entre los efectos más negativos se encuentran la atenuación, distorsión, interferencia y el ruido [Pan+,1965]:

### 1.1.3 La información y la señal

Las señales son entidades de naturaleza diversas que se manifiestan como magnitudes físicas, electromagnéticas, mecánicas, luminosas, acústicas, etc. La información añade la interpretación de las señales de modo que signifiquen algo concreto y definido, tanto para el emisor como para el receptor [Tom+, 1996].

Para que la información pueda viajar en la señal debe codificarse, es decir, debe estructurarse de acuerdo con las reglas impuestas por un código. Este código es una normativa de interpretación en la que se han puesto de acuerdo los terminales de la comunicación: emisor y receptor.

Para que exista comunicación debe ser posible la interpretación de los datos recibidos, lo que hace necesario que el emisor y receptor se pongan de acuerdo en el código que utilizarán para expresar sus mensajes. A continuación se mencionarán los códigos mas utilizados:

- **Código BAUDOT.** Es el código utilizado en la red telegráfica, también recibe el nombre de CCITT Nº2 [Gos+,1981].
- **Código ASCII.** Es el más utilizado en la actualidad, para la representación de información alfanumérica (American Standard Code for Information Interchange) o CCITT Nº5 [Gos+,1981].
- **Código EBCDIC.** Es un código propuesto por IBM semejante al código ASCII (Extended Binary Coded Decimal Interchange Code), representa cada carácter con 8 bits [Gos+,1981].

## 1.2 Modulación

La información que maneja un computador es digital, es decir, está compuesta por un conjunto discreto de dos valores: 1 y 0. Sin embargo, por las limitaciones físicas de las líneas de transmisión no es posible enviar información digital a través de un circuito telefónico o mediante ondas electromagnéticas. Para poder utilizar las líneas de transmisión para el envío de información entre computadores digitales, es necesario un proceso de transformación de la información. Durante este proceso la información se adecua para ser transportada por el canal de comunicación, primero se transduce la señal si es necesario y posteriormente se modula.

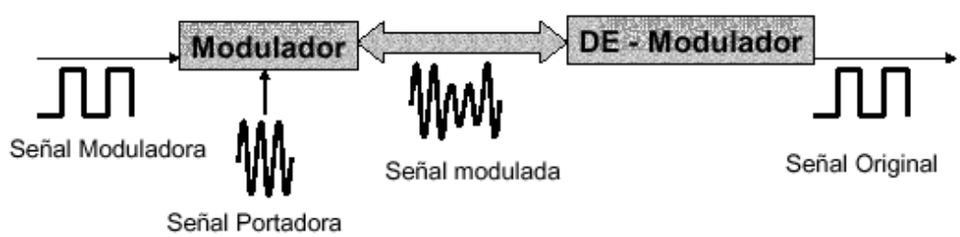



**Fig. 1.1.3 Proceso de modulación**

La señal moduladora es aquella que contiene la información emitida por un equipo. La señal Portadora puede ser analógica o digital, la portadora es la que lleva la

señal, no la información y es generada por un MODEM [Str+, 1990]. Explicado lo anterior se entiende por modulación como una técnica empleada para modificar una señal con la finalidad de posibilitar el transporte de información a través de un canal de comunicación y recuperar la señal en su forma original en la otra extremidad. Es decir, se trata de la modificación de una señal denominada portadora y al acoplarle las variaciones de otra señal denominada moduladora. Son posibles dos técnicas para la transmisión de datos: analógica y digital. Solamente la analógica realiza modulación, la digital, usa un recurso de codificación de pulsos.

| | Moduladora Analógica | Moduladora Digital |
|---|---|---|
| Portadora Analógica | AM, FM, PM | ASK, FSK, PSK |
| Portadora Digital | PAM, PDM, PPM, PCM, Delta. | Recodificaciones |

**Fig. 1.1.4 Clasificación de las modulaciones fundamentales**

La modulación es un proceso necesario para la comunicación a través de ondas electromagnéticas como también por enlaces telefónicos por las siguientes razones:

- Facilidad de Radiación, llevando a cabo una transformación en las señales no se requiere de antenas extremadamente grandes e imposibles de implementar.
- Reducción del Ruido y las Interferencias, es posible evitar los ruidos que se generan en los equipos de comunicación, pasando la información a otra banda de frecuencias donde los equipos no generen estos problemas.
- Asignaciones de Frecuencias, como las transmisiones son abiertas, es decir, libres, debe estar regulada la banda de frecuencias por la que pueden comunicarse un emisor y un receptor. Son organismos públicos quiénes conceden licencias para transmitir en determinadas bandas de frecuencias. La

modulación servirá para desplazar el espectro de cada mensaje a la banda de frecuencia asignada.

- Multicanalización, La mayor parte de los canales no son de utilización exclusiva por un emisor y un receptor, lo normal es que un canal sea compartido por varios emisores y receptores. Para que las informaciones no se superpongan hay que multicanalizar el canal, asignando a cada señal un rango de frecuencias. De este modo, por una misma línea de transmisión se pueden multiplexar varias señales, es decir, por un canal físico se crean varios canales lógicos. Básicamente existen dos formas de multiplexar un canal:
  - Multiplexación por División de Frecuencias (FDM). Se transmiten diferentes señales moduladas cada una sobre una portadora de distinta frecuencia, las portadoras están suficientemente separadas en el espectro de frecuencias como para evitar interferencias y diafonía que perjudiquen la comunicación [Str+, 1990].

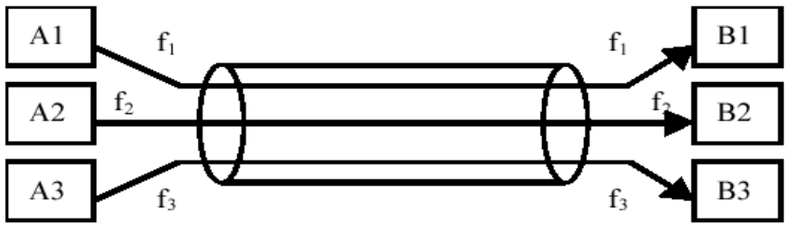



**Fig. 1.1.5 Multiplexación por división de frecuencias**

  - Multiplexación por División de Tiempo (TDM). Aquí las señales de entrada están desfasadas en el tiempo, de forma que en un instante solo se transmite información correspondiente a una única señal. Es decir, es como si el tiempo se dividiese en un conjunto de bloques sucesivos que se transmite de forma continuada y a cada bloque se le asigna a una señal [Str+, 1990].

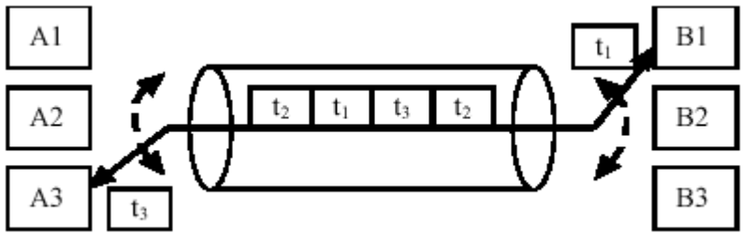



**Fig. 1.1.6 Multiplexación por división de tiempo**

### 1.2.1 Modulación con portadora analógica y moduladora digital

Las comunicaciones digitales abarcan un área extensa de técnicas de comunicaciones, incluyendo transmisión digital y radio digital. La transmisión digital es la transmisión de pulsos digitales, entre dos o más puntos de un sistema de comunicación. Los sistemas de transmisión digital requieren de un elemento físico, entre el transmisor y el receptor, como un par de cables metálicos, un cable coaxial, o un cable de fibra óptica. En un sistema de transmisión digital, la información de la fuente original puede ser en forma digital o analógica. Si está en forma analógica, tiene que convertirse a pulsos digitales, antes de la transmisión y convertirse de nuevo a la forma analógica en el extremo de recepción.

La radio digital es la transmisión de portadoras analógicas moduladas en forma digital, entre dos o más puntos de un sistema de comunicación. En los sistemas de radio digital, el medio de transmisión es el espacio libre o la atmósfera de la Tierra. En un sistema de radio digital, la señal de entrada modulada y la señal de salida demodulada, son pulsos digitales. Lo que distingue un sistema de radio digital de un sistema de radio AM, FM, o PM es que en un sistema de radio digital, las señales de modulación y demodulación son pulsos digitales, en lugar de formas de ondas analógicas. Sin embargo, la radio digital utiliza portadoras analógicas, al igual que los sistemas convencionales [Str+, 1990].

### 1.2.1.1 Modulación por desplazamiento de frecuencia (FSK)

Es similar a la modulación FM en transmisión analógica, el resultado de este tipo de modulación es una onda modulada cuya amplitud es la misma que la portadora, pero variando la frecuencia en función de los valores que toma la señal moduladora, es decir, a cada símbolo que se quiere transmitir se le asocia una frecuencia. Para que el receptor interprete bien el mensaje, tendrá que hacer un análisis de la frecuencia que le llega en cada momento en la señal portadora, es decir, la información del mensaje reside en la frecuencia. Normalmente esta técnica es usada para transmisión de datos a bajas velocidades.

El FSK binario es una forma de modulación angular de amplitud constante, donde la señal moduladora es un flujo de pulsos binarios que varía entre dos niveles de voltaje discreto, en lugar de una forma de onda analógica que cambia de manera continua. La expresión general para una señal FSK binaria esta dada por $\phi(t) = A_c \text{ sen} \left\{ \int_0^t [\omega_c + (\Delta\omega)x(t)]\, dt \right\}$ con:

$A_c$ = *Amplitud de la portadora no* mod*ulada*

$\omega_c$ = *Frecuencia portadora* $\left( \omega_c = \frac{\omega_2 + \omega_1}{2} \right)$

$x(t)$ = *Señal* mod*uladora digital binaria*

$\Delta\omega = \frac{\omega_2 - \omega_1}{2}$

De la ecuación puede verse que con el FSK binario la amplitud de la portadora $W_c$ se mantiene constante con la modulación. Sin embargo, la frecuencia en radianes de la portadora de salida $\omega_c$ cambia por una cantidad igual a $\pm \Delta\omega/2$. El cambio de frecuencia $\Delta\omega/2$ es proporcional a la amplitud y a la polaridad de la señal de entrada binaria, la señal de energia del digito binario esta dado por $E = \int_0^T A^2 \text{ sen}^2\, m\omega_c t\, dt = A^2 T/2$ [Str+, 1990].

#### 1.2.1.2 Modulación por desplazamiento de amplitud (ASK)

Es similar a la modulación AM en transmisión analógica. Consiste en transmitir una portadora cuando se envía un 1 y en no transmitir ninguna señal cuando se envía un 0, es decir la señal modulada en amplitud resultante consta de pulsos de radio frecuencia llamadas marcas, que representan unos binarios y espacios que representan ceros binarios.

#### 1.2.1.3 Modulación por desplazamiento de fase (PSK)

Transmitir por desplazamiento en fase es otra forma de modulación angular, modulación digital de amplitud constante. El PSK es similar a la modulación en fase convencional, excepto que con PSK la señal de entrada es una señal digital binaria y son posibles un número limitado de fases de salida.

#### 1.2.1.4 Modulación de amplitud en cuadratura (QAM)

Es una forma de modulación digital, en donde la información digital está contenida tanto en la amplitud como en la fase de la portadora transmitida.

#### 1.2.1.5 Modulación PSK diferencial (DPSK)

En este caso, la información se conduce por medio de las transiciones en la fase de la portadora, esta modulación se usa generalmente para resolver el problema de la sincronización, en que la información se codifica utilizando las diferencias entre bits en dos intervalos de bit sucesivos.

### 1.2.2 Comunicación en serie

La comunicación en paralelo no es aceptable en aquellas aplicaciones en que la distancia entre el transmisor y el receptor es grande debido a su gran sensibilidad al ruido. La alternativa es la comunicación serial, ésta es más antigua que la comunicación en paralelo y debido a su gran difusión existen diversas normas que han estandarizados este tipo de interfaz. Su transmisión se efectúa en forma secuencial en el tiempo, transmitiendo todos los bits de la palabra, uno tras otro por una única línea de datos, aunque pueden existir otras líneas de control. El circuito de acoplamiento de entrada/salida entre el bus de datos y las líneas de transmisión es normalmente una unidad programable, ésta efectúa la conversión serie-paralelo o paralelo-serie de los datos, siendo su modo de actuación totalmente similar al de los registros de desplazamiento.

Según sea el modo de transmisión, existen dos circuitos de acoplamiento universales para la transmisión. El transmisor/receptor universal síncrono (USRT – Universal Synchronous Receiver Transmiter) que se emplea para transmisiones síncronas. Para el caso de transmisiones asíncronas se emplea el transmisor/receptor universal asíncrono (UART – Universal Asynchronous Receiver Transmiter). Los circuitos pueden operar tanto en modo half-duplex como full-duplex, es decir, transmisión en ambos sentidos en forma alternada o simultánea respectivamente.

La sincronización de los bits en una transmisión en serie, se consigue utilizando en la recepción el mismo reloj de transmisión, el cual es enviado por una línea separada o codificado junto con la información (transmisión síncrona), o bién utilizando relojes independientes, pero con la misma frecuencia y fase (transmisión asíncrona). La frecuencia del reloj permite una codificación de los bits de información, teniendo los siguientes módelos: NRZ, NRZI, Manchester, etc. En los sistemas de comunicación en serie, se utilizan básicamente dos formas de transmisión para la sincronización:

- **Transmisión serie síncrona.** Se caracteriza porque la transmisión de información se realiza de una forma continua, bit a bit y sin ninguna separación entre caracteres. La sincronización de bits se consigue normalmente utilizando una señal externa de reloj o codificándola junto con la información. El transmisor envía generalmente por una línea independiente de las de datos su reloj, que es utilizado como reloj de recepción para la llegada de datos.

  Para la sincronización de carácter, se transmite al comienzo de cada bloque de datos unos caracteres de sincronismo que indican al receptor el instante en que se inicia la transmisión, lo que permite no utilizar nuevas líneas de control. La trama de información esta compuesta por un bloque de sincronismo seguido del bloque de datos y por ultimo un bloque para la detección de errores.

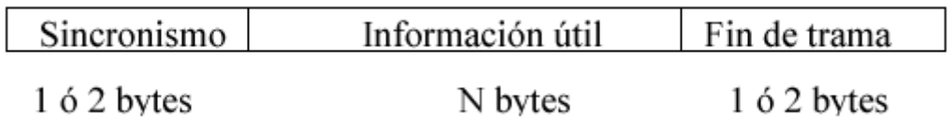



**Fig. 1.1.7 Trama serial síncrona**

Este tipo de comunicación es rápida, pero es necesaria una gran sincronización para evitar errores. Es útil para transmitir gran cantidad de información a largas distancias.

- **Transmisión serie asíncrona.** Los datos se envían en cualquier instante, debido a que no utilizan señales de sincronización es necesario que cada dato lleve unas marcas para indicar el comienzo y el fin del mismo. El control de la transmisión se efectúa por bits de arranque y de parada que enmarcan cada carácter transmitido y que son utilizados por el receptor para sincronizar su reloj con el del transmisor en cada carácter.

Mientras no se envíen datos por la línea, esta se mantiene al nivel lógico 1. Cuando se desea transmitir un carácter, se envía primero un bit de comienzo que pone a 0 lógico la línea durante el tiempo de un bit, de esta forma el otro dispositivo detecta que le van a enviar un dato. A continuación se envían todos los bits del carácter a transmitir con una velocidad marcada por el reloj de transmisión, la duración de cada bit de información es igual a la duración del bit de comienzo, por último se envía el bit de detección de errores por paridad si se utiliza y uno o dos bits de parada.

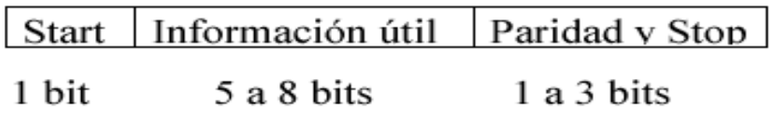



**Fig. 1.1.8 Trama serial asíncrona**

El bit o los bits de parada tiene como misión llevar la linea al estado lógico 1 para que el bit de comienzo del siguiente carácter provoque la transición hacia el estado lógico 0, lo cual permite el receptor sincronizar el siguiente carácter.

### 1.2.3 Comunicación en paralelo

Esta comunicación consiste en interconectar los sistemas digitales entre los que habrá transferencia de datos, mediante tantas líneas como número de bits tenga la longitud de la palabra de datos. Normalmente no es el propio microprocesador el que se encarga de hacer la transferencia de los datos, lo habitual es que exista un sistema intermedio entre el microprocesador y la línea, cuya función es iniciar, realizar y controlar el flujo de información a través de las líneas, evitando que el procesador haga esta tarea. La transmisión en paralelo suele estar limitada, en general, a dos tipos principales de aplicaciones:

- Transmisión de datos a través de los buses internos de los sistemas informáticos.
- Intercambio de información entre sistemas muy próximos entre si, y que requieren elevadas velocidades de transmisión.

Al sistema intermedio se le conoce como unidad de control de entrada/salida paralelo. La transferencia de información puede hacerse de dos formas distintas.

- **Transferencia síncrona.** También llamada incondicional, es la forma más simple y directa de realizar la transferencia. Es iniciada en cualquier instante y no se consulta al sistema periférico sobre su estado, como tampoco el periférico envía una señal de control. El mayor inconveniente que limita su uso, es que solamente es posible utilizar la transferencia síncrona con periféricos que estén siempre preparados para aceptar datos y que procesen la información en tiempos similares a los empleados por el origen.
- **Transferencia asíncrona.** También llamada condicional, es la más utilizada puesto que permite la comunicación entre equipos que presentan diferentes velocidades para el procesamiento de datos. Este modo de transferencia obliga a la sincronización (handshaking) de los interlocutores, cada extremo de la comunicación debe conocer el estado del otro, para ello junto con las líneas de datos se añaden líneas de control que permiten transmitir dicho estado de un extremo a otro.

# 2. Metodología de desarrollo de un MODEM

## 2.1 Objetivos

Analizadas las modulaciones fundamentales, las comunicaciones seriales y paralelas, se determinan los diferentes elementos que se necesitan para el diseño de un MODEM para el intercambio de información digital. Para llegar a un diseño formal de la solución se hará un análisis en base a los principales componentes que tendrá el MODEM.

## 2.2 Análisis de requerimientos

Los requerimientos fueron recopilados en una unidad táctica del Ejército de Chile, específicamente en el Regimiento de Telecomunicaciones Nº 4 “Membrillar” ubicado en Bueras s/n en la ciudad de Valdivia. Los requisitos fueron focalizados de acuerdo a las necesidades de esta unidad, específicamente limitada por los diferentes equipos de telecomunicaciones con que cuentan actualmente. La idea principal es que la solución sea compatible con la mayor cantidad de equipos posibles, en otras palabras, que el sistema de transmisión digital pueda operar en cualquier banda de frecuencias. A continuación se individualizarán los requerimientos mínimos que debe tener el MODEM.

### 2.2.1 Requerimiento 1

| | |
|---|---|
| Requerimiento | Transmisión Serial |
| Detalle | Los datos binarios que ingresarán al MODEM digital deben salir y entrar por el puerto RS232-D |
| Tipo | Primario |

### 2.2.2 Requerimiento 2

| | |
|---|---|
| Requerimiento | Intercambio de Información Digital |
| Detalle | Los datos que se transmitan y reciban deben ser de índole digital, en este caso mensajes y archivos |
| Tipo | Primario |

### 2.2.3 Requerimiento 3

| | |
|---|---|
| Requerimiento | Ancho de Banda |
| Detalle | Considerar el ancho de banda en el diseño para la transferencia de datos en las distintas gamas de frecuencia |
| Tipo | Secundario |

### 2.2.4 Requerimiento 4

| | |
|---|---|
| Requerimiento | Velocidad de Tansferencia |
| Detalle | Considerar este aspecto sobre todo en la transmisión de archivos |
| Tipo | Primario |

### 2.2.5 Requerimiento 5

| | |
|---|---|
| Requerimiento | Costo Asociado (Esfuerzo) |
| Detalle | Establecer el costo de fabricación del modem, considerando todos los aspectos involucrados en el diseño |
| Tipo | Secundario |

### 2.2.6 Requerimiento 6

| | |
|---|---|
| Requerimiento | Reusabilidad |
| Detalle | El modem debe ser fácilmente adaptable a diferentes tipos de radios, en lo posible solo cambiar las interfaces Radio-RS232-D |
| Tipo | Primario |

### 2.2.7 Requerimiento 7

| | |
|---|---|
| Requerimiento | Push to Talk automático |
| Detalle | Esta función debe ser ejecutado por algún software y el MODEM debe tener esta función implementada electrónicamente |
| Tipo | Primario |

## 2.3 Diseño

En esta parte, se definirán cada una de las soluciones adoptadas para cada requerimiento recopilado y analizado. La definición se hará considerando factores de reusabilidad y principalmente su costo asociado.

### 2.3.1 MODEM FSK externo

El MODEM proporcionará una comunicación serial y establecerá un canal de comunicaciones half-duplex. Para efectos de diseño este MODEM deberá comportase como una TNC (Terminal Node Controller), sin embargo el aspecto económico es un factor relevante para este proyecto, por lo cual se tendrá un MODEM con funcionalidades mínimas, quedarán pendientes el uso del protocolo de radio paquetes y se utilizarán circuitos integrados de bajo costo. Este MODEM establecerá un uso automatizado del PTT (Push To Talk), la interfaz serial será del tipo RS232-D. Este tipo de conexión no necesita una validación de usuarios, como tampoco ningún servidor adicional para algún servicio. Se realizará una modulación FSK binaria, ya que es la más adecuada para poder aplicarla en UHF, VHF y HF, además es más eficiente que otras en canales de transmisión ruidosos, pero con inferiores velocidades. En cuanto a las velocidades están determinadas dependiendo del tipo de comunicación que se tiene, es decir, la velocidad de transmisión del lado de la radio es de 300 bits/seg (HF), de 1200 bits/seg o superior (VHF-UHF). En cuanto a las velocidades de modulación será de 1200 bits/seg, para ello se ajustaron las frecuencias de señalización (marca y espacio) en 1200 y 2200 Hertz respectivamente.

Para la selección de los moduladores y demoduladores digitales se considero fuertemente el aspecto económico, siendo este punto una limitante a la hora de desarrollar un prototipo.

### 2.3.1.1 Modulador y demodulador

Los paquetes de información sólo tendrán el formato de una transmisión serial asíncrona, ya que los circuitos integrados que se utilizaran no trabajan con el protocolo de radio paquetes, éstos son de la empresa EXAR modelos XR2000 [Exa+, 2006], los cuales realizarán las funciones de modulación y demodulación FSK binaria.

**Modulador XR2206.** Es un circuito integrado generador de funciones monolíticas, capaz de producir formas de onda seno, cuadradas, triángulos y rampas de alta calidad, además de ondas de pulsos de alta estabilidad y precisión. Es ideal para comunicaciones, instrumentación y aplicaciones que requieren generar tonos senoidales, modulación AM o FM, Modulación digital FSK [Exa+, 2006].

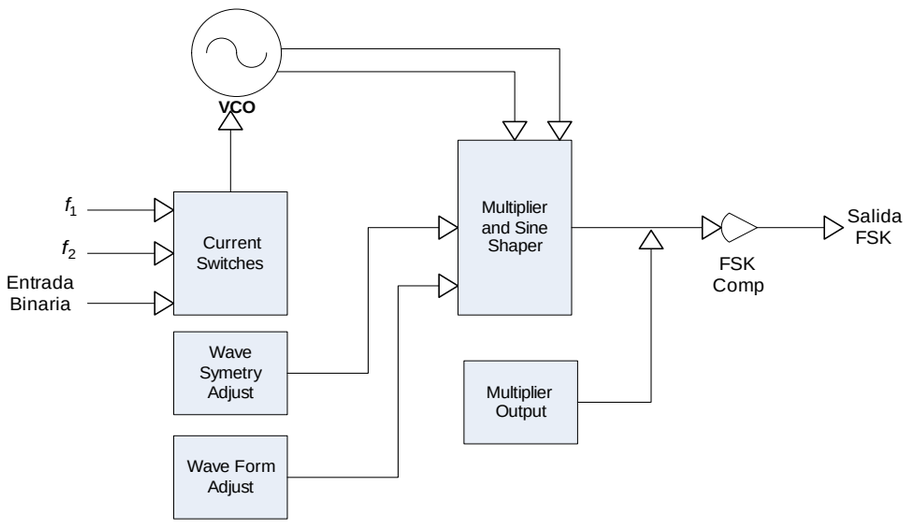



**Fig. 1.2.1 Diagrama funcional para modulación FSK**

**Demodulador XR2211.** Es un circuito integrado PLL (Phase-Locked Loop) diseñado para comunicaciones de datos, especialmente para diseño de MODEM FSK con detección de portadora. Tiene compatibilidad lógica con DTL/TTL/ECL [Exa+, 2006].

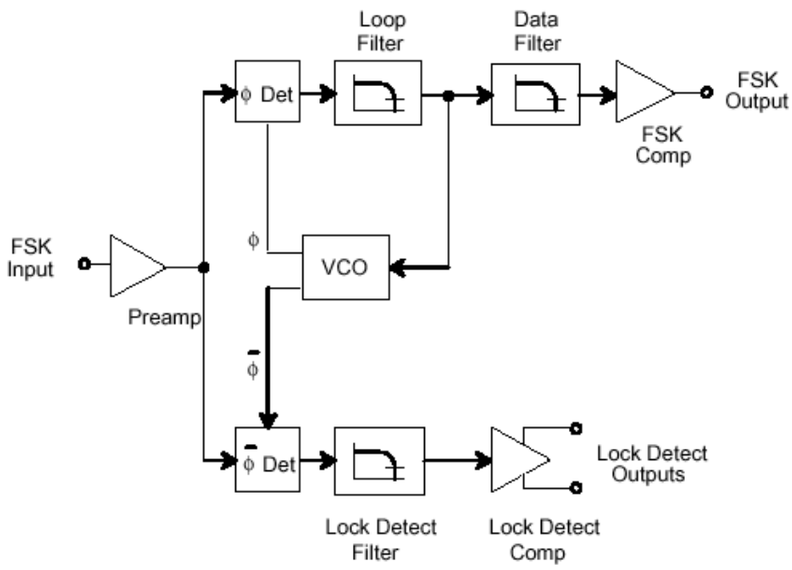



**Fig. 1.2.2 Diagrama funcional para demodulación FSK**

### 2.3.1.2 PTT (Push To Talk)

Producto de que el MODEM manejara la función de PTT para darle un uso más transparente y automatizado a la plataforma se utilizará un circuito para lograr hacer esta función, de esta forma se pretende la menor intervención de un usuario en la comunicación, como también dar la impresión de que se tiene un canal full duplex. Para controlar este proceso se utilizará una línea de datos del RS232-D. Se debe recordar que un circuito específico para este trabajo no es aplicable a toda la gama de radios, básicamente el diseño depende de la radio que se quiera utilizar. En cuanto a la linea de datos del puerto RS232-D que controla el PTT se debe activar sólo cuando haya datos para enviar, cuando esté inactiva la linea de datos el modem estará en modalidad de recepción.

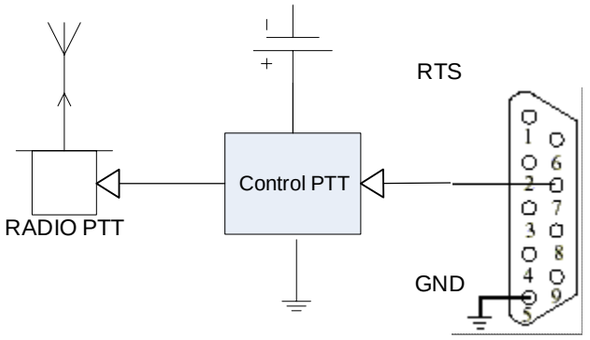
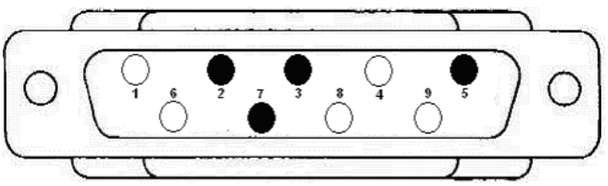



**Fig. 1.2.3 Diagrama funcional de pulsador de PTT**

### 2.3.1.3 Interfaz serial

Como la comunicación será serial se utilizará el puerto RS232-D del computador, para ello es necesario confeccionar un RS232-D Hembra con los pines que se necesitarán para transmitir, recibir y pulsar el PTT, donde cada linea irá conectada al MODEM.



| Pin DB9 | Nombre | Función |
|---|---|---|
| **3** | TXD | TRANSMISION DE DATOS (SALIDA) |
| **2** | RXD | RECEPCION DE DATOS (ENTRADA) |
| **7** | RTS | PETICION DE ENVIO (SALIDA) |
| **8** | CTS | DISPUESTO PARA ENVIAR (ENTRADA) |
| **6** | DSR | DISPOSITIVO DE DATOS LISTO (ENTRADA) |
| **5** | COMUN | COMUN (REFERENCIA) |
| **1** | DCD | DETECCION DE PORTADORA DE DATOS (ENTRADA) |
| **4** | DTR | TERMINAL DE DATOS LISTO (SALIDA) |
| **9** | RI | INDICADOR DE LLAMADA (ENTRADA) |

**Fig. 1.2.4 Nomenclatura y esquema RS232-D para Rx y Tx**

### 2.3.1.4 Conversor RS232-D a TTL

Es necesario cambiar los niveles TTL a los del estándar RS232-D cuando se tiene una recepción, por otro lado se deben cambiar los niveles del estándar RS232-D a TTL cuando se hace una transmisión.

| Voltaje | Lógico | Control | Terminología |
|---|---|---|---|
| **+3[v] a +25[v]** | 0 | Activo | Espacio |
| **-3[v] a – 25[v]** | 1 | Inactivo | Marca |

**Tabla 1.2.1 Equivalentes TTL de los voltajes del RS232**

Para este proceso se utilizará un circuito integrado, se eligió un circuito de la familia MAX200, específicamente el MAX232.

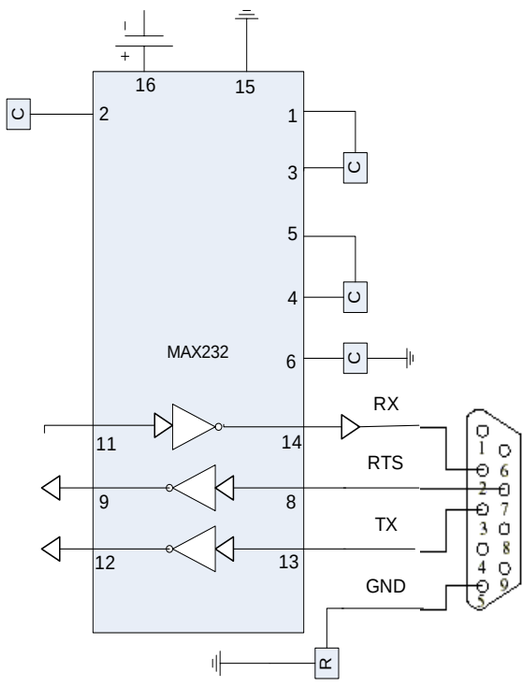



**Fig. 1.2.5 Diagrama funcional del conversor TTL**

## 2.4 Implementación

Finalmente se tuvo que integrar cada dispositivo, para la confección de un MODEM externo FSK que envié información digital por medio de los equipos de telecomunicación militares análogicos. El diseño final contempla todos los factores mencionados anteriormente que son necesarios para satisfacer los requisitos de la institución. El MODEM esta ajustado para realizar transmisiones a 1200 bits/seg.

La línea Rx del MODEM irá a la línea de datos del RS232-D RXD, la línea de Tx del MODEM irá a la línea de datos del RS232-D TXD, el PTT del MODEM irá a la línea de datos del RS232-D RTS y por último la línea de tierra del MODEM irá a la línea de datos del RS232-D GROUND.

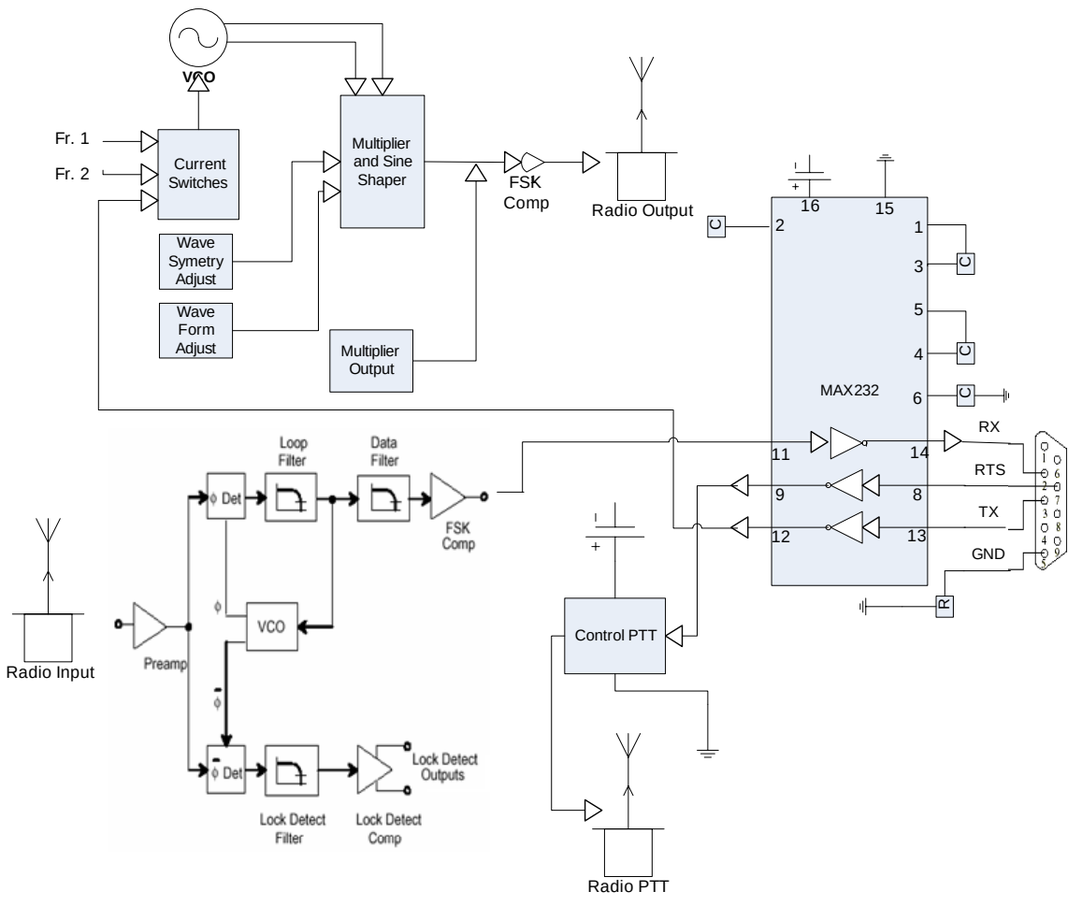



**Fig. 1.2.6 Diagrama funcional MODEM FSK para 1200 bits/seg**

Para confeccionar el MODEM se necesito de una serie de equipos electrónicos tales como: osciloscopio digital, osciloscopio análogo, generador de señales, fuente de poder de 12V y 5V, un multimetro y un multitester digital. En la tesis se hizó uso de equipos pertenecientes al Instituto de Electrónica de la Universidad Austral de Chile. A continuación se detallan los esquemas de diseño para el modulador y demodulador FSK.

### 2.4.1 Esquema del modulador FSK

Se detallará el circuito base y las principales ecuaciones para su diseño considerando las especificaciones del fabricante [Exa+, 2006].

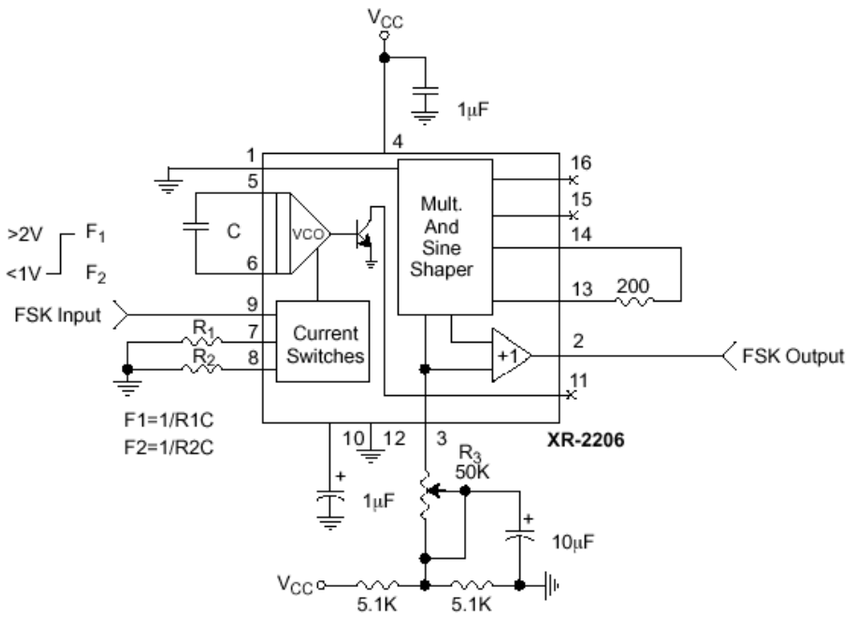



**Fig. 1.2.7 Esquema modulador XR2206**

- Frecuencias de señalización

$$f_1 = \frac{1}{R_1 C}, f_2 = \frac{1}{R_2 C}$$

R1, R2 y C deben tener valores tal que las frecuencias de señalización den como resultado 1200 y 2200 Hz respectivamente.

- Frecuencia de operación

  $f_0 = \frac{1}{RC}[Hz]$

  R y C están definidos dependiendo de la frecuencia de señalización que se este utilizando al modular, dando como resultado la frecuencia portadora no modulada que sale del VCO.

- Para la multiplicación de salida, donde la señal es ajustada en intensidad se tendrá un potenciometro de 50 K conectado al PIN 3 del XR2206.

- El índice de modulación esta dado por $m = \frac{|f_2 - f_1|}{f_b} = \frac{|2200 - 1200|}{1200} = 0.83$ y como se tendrá una detección coherente el factor $2\Delta fT$ debe estar entre $0.5 < 2\Delta fT < 1$ , con ello se obtiene una ventaja en S/N y minimizar el ancho de banda, donde T es la duración del símbolo en segundos, $2\Delta fT$ equivale al índice de modulación en este caso.

### 2.4.2 Esquema del demodulador FSK

Se detallará el circuito base y las principales ecuaciones para su diseño considerando las especificaciones del fabricante [Exa+, 2006].

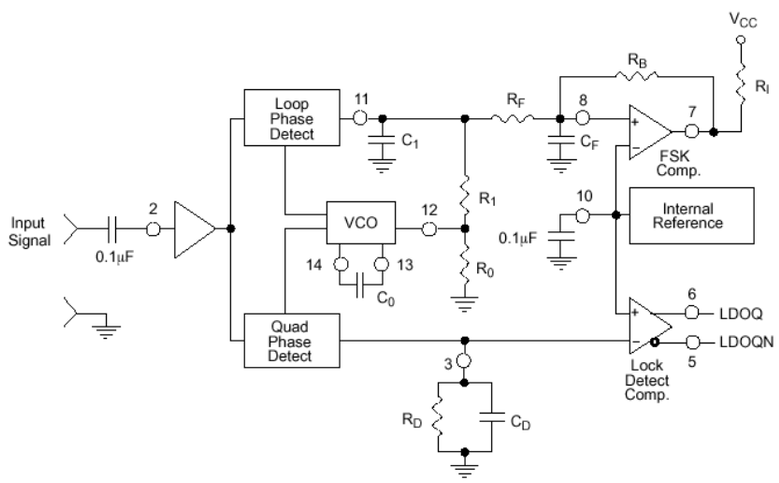



**Fig. 1.2.8 Esquema demodulador XR2211**

- Frecuencia central del VCO

$$f_0 = \frac{1}{R_0 C_0}$$

Ro y Co deben tener valores tal que la frecuencia central del VCO debe dar como resultado 1700 Hz que es la frecuencia de la portadora no modulada, esta frecuencia es también la frecuencia central del PLL y realimenta la detección de fase donde la frecuencia de entrada se ajustará hasta igualar la frecuencia del PLL [Dav+, 2001].

- Voltaje de referencia interno

$$V_{REF} = \left(\frac{V_{CC}}{2}\right) - 650\,mV\ [Volts]$$

Vcc es el voltaje que viene de la fuente de alimentación.

- Constante de tiempo $\tau$ para el filtro pasabajas de amarre

$$\tau = C_1 R_{pp}\ [s]\ , R_{pp} = \left(\frac{R_1 R_F}{R_1 + R_F}\right)$$

La señal entregada por la detección de fase debe pasar por este filtro de amarre para eliminar los armónicos superiores y entregar el voltaje para el VCO.

- Constante de tiempo $\tau_F$ para el filtro de datos

$$\tau_F = \left(\frac{R_B R_F}{R_B + R_F} C_F\right)[s]$$

- Como $2\Delta fT$ <1 el ancho de banda sera como mínimo $2\Delta f = 1000\,[Hz]$, donde las frecuencias de señalización están dadas por $f_0 \pm 2\Delta f$ y fo es la frecuencia de la portadora no modulada, lo cual da un ancho de banda de 2000 Hz.

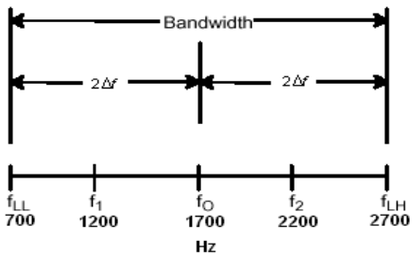



**Fig. 1.2.9 Ancho de banda**

En cuanto a la densidad espectral, se concentrará en la frecuencia central y en las frecuencias de señalización producto que $0.5 < 2\Delta fT < 1$.

## 2.5 Evaluación de costos

El costo de diseño del MODEM se confecciono de acuerdo a sus principales componentes, por otro lado el costo específico de cada uno de ellos esta fundado sobre un promedio entre los de mejor calidad y mayor número de cualidades y los de menor calidad, dentro del ámbito de una modulación FSK.

| | | |
|---|---|---|
| Detalle de Componentes | | |
| | Modulador | $ 50.000 |
| | Demodulador | $ 50.000 |
| | Conversor TTL | $ 40.000 |
| | Interfaces | $ 50.000 |
| | Otros | $ 60.000 |
| Diseño | | |
| | Implementación | $ 200.000 |
| | Equipamiento de Pruebas | $ 50.000 |
| Tiempo Estimado (Meses) | | 1 |
| Costo MODEM | | $ 500.000 |

**Fig. 1.2.10 Cálculo de esfuerzo del MODEM FSK**

# Capitulo 2: Modelo de Desarrollo de un Sistema de Cifrado de Información

# 1. Criptografía

La criptografía ha sido usada a través de los años para enviar mensajes confidenciales, cuyo propósito es que sólo las personas autorizadas puedan entender el mensaje. Desde sus inicios la criptografía llegó a ser una herramienta muy usada en el ambiente militar, por ejemplo: en la segunda guerra mundial tuvo un papel determinante, una de las máquinas de cifrado que tuvo gran popularidad se llamó ENIGMA. Al terminar la guerra las agencias de seguridad de las grandes potencias invirtieron muchos recursos para su investigación. La criptografía actual se inicia en la segunda mitad de la década de los años 70. No es hasta la invención del sistema de cifrado conocido como DES (Data Encryption Standard) en 1976 que se da a conocer mas ampliamente [ANS+, 1983], principalmente en el mundo industrial y comercial. Posteriormente con el sistema RSA (Rivest, Shamir, Adleman) en 1978 [RSA+, 1998], se abre el comienzo de la criptografía en un gran rango de aplicaciones: en transmisiones militares, en transacciones financieras, en comunicación de satélite, en redes de computadoras, en líneas telefónicas, en transmisiones de televisión.

Para diseñar una estrategia de seguridad, depende en general de la actividad que se desarrolle. Sin embargo, se pueden considerar tres pasos fundamentales: crear una política global de seguridad, realizar un análisis de riesgos y aplicar las medidas correspondientes [Foc+, 1995].

- **Política global de seguridad.** Aquí se establece el estatus de la información para la empresa o la organización, debe de contener un objetivo general, la importancia de la tecnología de la información para la empresa, el periodo de tiempo de validez de la política, los recursos con que se cuenta, objetivos específicos de la empresa.
- **Análisis de riesgos.** Consiste en listar todo tipo de riesgos a los cuales esta expuesta la información y cuáles son las consecuencias, los posibles atacantes

entre persona, empresas y dependencias de inteligencia, las posibles amenazas, etc.

- **Medidas de seguridad.** Esta parte la podemos plantear como la terminación de toda la estructura de seguridad de la información. Una vez planteada una política de seguridad, o sea decir cuanto vale la información (en un análisis de riesgo), decir que tanto se pierde si le ocurre algo a la información o que tanto se gana si está protegida, debemos de establecer las medidas para que cumpliendo con la política de seguridad las pérdidas sean las menores posibles.

Por otro lado los principales problemas de seguridad que resuelve la criptografía son: la privacidad, la integridad, la autenticación y el no rechazo [Foc+, 1995].

- **La privacidad.** Se refiere a que la información sólo pueda ser leída por personas autorizadas.
- **La integridad**. Se refiere a que la información no pueda ser alterada en el transcurso de ser enviada.
- **La autenticidad**. Se refiere a que se pueda confirmar que el mensaje recibido haya sido mandado por quien dice lo mando o que el mensaje recibido es el que se esperaba.
- **El no rechazo.** se refiere a que no se pueda negar la autoría de un mensaje enviado.

Utilizando una herramienta criptografía conjuntamente con otras técnicas, como el buen manejo de las claves y la legislación adecuada resuelven satisfactoriamente los problemas planteados anteriormente. En cuanto al intercambio de información por instituciones de defensa, estos aspectos toman un alto nivel de prioridad ya sea en tiempos de paz o en caso de guerra.

Un aspecto importante de mencionar es que los sistemas y algoritmos criptográficos actualmente operan o están hechos para intercambio de información en Internet, Intranet, como también en redes inalámbricas de datos. Sin embargo, en el área de las comunicaciones por radio frecuencia no son muy conocidas, pero esto ha ido cambiando con la modulación digital de datos. La criptografía se divide en dos grandes ramas, la criptografía de clave privada o simétrica y la criptografía de clave pública o asimétrica.

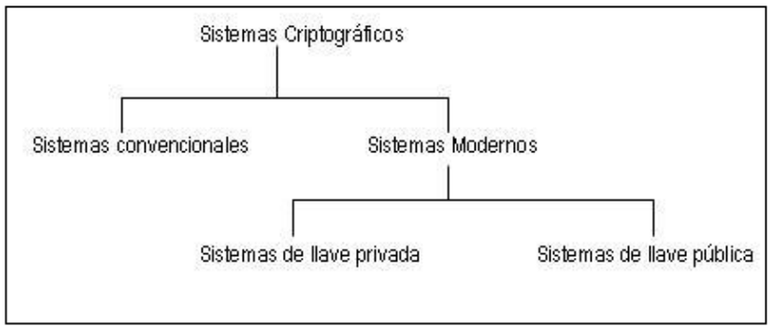



**Fig. 2.1.1 Clasificación de sistemas criptográficos**

El enfoque de análisis que se dará a este tema en la tesis, será básicamente investigar los algoritmos y herramientas criptográficas que sean factibles de aplicar a la modulación digital para transmisiones de radio frecuencia en base a la metodología propuesta.

## 1.1 Definiciones generales

### 1.1.1 Permutación

Son funciones que a menudo son usadas en varios algoritmos criptográficos. Se define S como un conjunto finito de elementos y se define permutación como P sobre S y es una biyección de S sobre si mismo $P : S \rightarrow S$.

### 1.1.2 Cifrado por sustitución

Se tiene un alfabeto A de q símbolos, y un conjunto M de todas las cadenas de largo t sobre A, se tiene un conjunto K que son todas las permutaciones sobre A. Con ello de define una Encriptación $E_e$ para cada e $\in$ K como $E_e(m) = (e(m_1)\, e(m_2) \ldots e(m_t)) = (c_1\, c_2 \ldots c_t) = c$, donde $m = (m_1\, m_2 \ldots m_t)$ $\in$M, en otras palabras para cada símbolo de una t-tupla se remplaza por otro símbolo de A según alguna permutación fija e. Para desencriptar c se calcula la permutación inversa $d = e^{-1}$. El cifrado puede ser por sustitución Monoalfabetica y Polialfabetica.

### 1.1.3 Cifrado por sustitución monoalfabetica

Se tiene que para todo símbolo a $\in$ A, asociado a un conjunto H(a) de caracteres de t símbolos, con la restricción de que los conjuntos H(a) sean parejas distintas. Un cifrado por sustitución Monoalfabetica reemplaza cada símbolo a de un bloque de mensaje de texto plano con una selección aleatoria de caracteres de H(a). Para desencriptar un carácter c de t símbolos, primero debe determinar que a $\in$ A y que c $\in$ H(a). La clave de cifrado consiste en elementos de H(a).

### 1.1.4 Cifrado por sustitución polialfabetica

Se tiene un bloque de cifrado de largo t sobre un alfabeto A, considerando las siguientes propiedades:

1. El espacio de claves K consiste de todos los conjuntos ordenados de t permutaciones $(p_1, p_2, \ldots, p_t)$, donde cada permutación $p_i$ es definida sobre el conjunto A.

2. Encriptación de un mensaje $m = (m_1 m_2 ... m_t)$ mediante las claves $c = (p_1, p_2, ..., p_t)$ esta determinado por $E_c(m) = (p_1(m_1) p_2(m_2) ... p_t(m_t))$.
3. La clave de desencriptación asociada con $(p_1, p_2, ..., p_t)$ es $d = (p_1^{-1}, p_2^{-1}, ..., p_t^{-1})$.

### 1.1.5 Cifrado por transposición

Se tiene un bloque de largo t y un conjunto K de todas las permutaciones sobre el conjunto $\{1,2,...,t\}$, para cada e $\in$ K se define la función de encriptación $E_e(m) = (m_e(1)\, m_e(2) \ldots m_e(t))$ donde $m = (m_1\, m_2 \ldots m_t) \in$ M. El conjunto de todas las transformaciones es llamada cifrado por transposición simple. La clave de desencriptación corresponde a e y es la permutación inversa $d = e^{-1}$.

### 1.1.6 Generación de datos aleatorios y seudoaleatorios

*Generación Aleatoria:* Requieren naturalmente de la existencia de una fuente de carácter aleatorio. Para la mayoría de las aplicaciones criptográficas el generador no debe ser sujeto de observación o manipulación por algún atacante. Los generadores de datos aleatorios basados en fuentes aleatorias naturales están sujetos a influencias por factores externos y también a un mal funcionamiento. Los generadores se pueden diseñar en base a dispositivos de hardware o softwares [Van+, 1996].

*Generación Seudoaleatoria:* Una función de dirección única f puede ser utilizada para generar secuencias de datos seudoaleatorios, para ello se debe seleccionar primero aleatoriamente una semilla, entonces al aplicar la función para la secuencia de valores s, s+1,… la secuencia de salida es f(s), f(s+1),… Dependiendo de las propiedades de

la función usada, puede ser necesario reservar unos pocos bits de los valores de salida de f(s+i) a fin de evitar posibles correlaciones entre valores sucesivos. Los generadores de datos seudoaleatorios son los más utilizados, sus algoritmos son muy similares, sus variaciones dependen básicamente del algoritmo de encriptación y entre algunos algoritmos están [Van+, 1996]:

- Generador ANSI X9.17. Utilizado por DES
- Generador FIPS 186. Utilizado por DSA con claves privadas y para mensajes secretos, también usado por SHA-1 y DES.
- Generador RSA
- Generador Micali-Schnorr
- Generador Blum-Blum-Shub

### 1.1.7 Encriptación de cascada

Es una concatenación de L>=2 bloques de cifrado llamados *etapas*, cada uno con claves independientes. Texto plano es la entrada de la primera etapa y su salida es la entrada de la siguiente etapa y la salida de la etapa L es la salida en cascada de texto cifrado. En el caso más sencillo todas las etapas del cifrado en cascada tienen claves de k bits y las etapas de entrada y salida son todas de cantidades de n bits [Van+, 1996].

### 1.1.8 Encriptación múltiple

Es similar a la encriptación en cascada, pero las claves de las etapas no requieren que sean independientes y la etapa de cifrado puede ser cualquier bloque cifrador E o la correspondiente función de desencriptación $D = E^{-1}$ . Dos casos importantes de encriptación múltiple son la encriptación doble y triple. Donde la

encriptación doble se define como $E(x) = E_{K2}(E_{K1}(x))$, donde $E_K$ es un bloque cifrador E con clave K. Por otro lado la encriptación Triple se define como $E(x) = E^3_{K3}(E^2_{K2}(E^1_{K1}(x)))$, donde $E^j_K$ representa a $E_K$ o $D_K = E^{-1}_K$. El caso $E(x) = E_{K3}(D_{K2}(E_{K1}(x)))$ es llamado encriptación triple E-D-E y el sub caso con K1=K2 es a menudo llamado encriptación triple de dos claves [Van+, 1996].

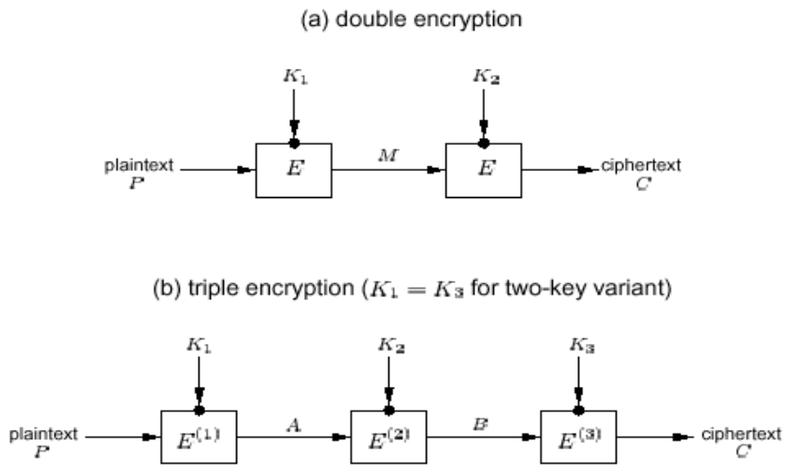



**Fig. 2.1.2 Encriptación múltiple**

### 1.1.9 Red de sustitución-permutación (SP)

Es un producto cifrado compuesto de un número de estados cada uno involucrando sustituciones y permutaciones. Un producto cifrado combina dos o más transformaciones de un modo determinado pretendiendo que el cifrado resultante es más seguro que usar componentes individuales.

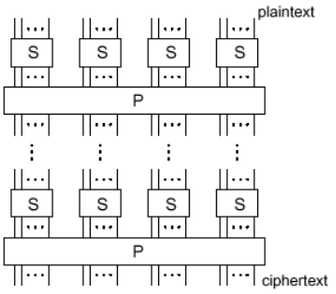



**Fig. 2.1.3 Red de sustitución-permutación**

### 1.1.10 Bloque de cifrado iterativo

Es un bloque de cifrado que involucra la repetición secuencial de una función interna llamada función de ciclo, entre sus parámetros incluye el número de ciclos r, el bloque de tamaño en bit n, y k de tamaño en bit de la entrada de la clave K, desde lo cual r subclaves $K_i$ (ciclo de claves) son deducidas.

## 1.2 Análisis de sistemas convencionales

Los sistemas convencionales fueron los primeros pasos que se dieron en criptografía satisfaciendo las necesidades de su época. Estos básicamente se pueden clasificar como algoritmos monoalfabéticos y polialfabéticos producto de la estacionalidad de la claves de cifrado.

| Algoritmo | Clasificación | Detalle |
|---|---|---|
| **Sustitución de cambio trivial (cesar)** | Monoalfabético | Es un cifrado de sustitución simple con una permutación obligada c para un cambio de alfabeto a través de k caracteres, para algunos casos k es fijo |
| **Sustitución affine** | Monoalfabético | Funciona sobre un alfabeto de 26 letras y esta definido por $e_K(x) = ax + b \mod 26$ donde $0 \le a, b \le 25$. La clave es (a,b) y el texto cifrado esta definido por $c = e_K(x)$ |
| **Sustitución playfair** | Monoalfabético | Funciona en base a un diagrama de sustitución y puede ser definido por arreglos de caracteres de un alfabeto de 25 letras (I y J son iguales) en una matriz M de 5x5, los caracteres de texto plano son ajustados por pares |
| **Sustitución hill** | Monoalfabético | Funciona en base a un diagrama de sustitución, un n diagrama puede ser definido usando una matriz invertible $A = a_{ij}$ de nxn, así también la clave, el mapa y el texto plano $m_1 ... m_n$ serán de n caracteres |
| **Transposición simple** | Monoalfabético | Esta determinado por un cifrado con un periodo fijo t, la encriptación involucra un agrupamiento de texto plano en bloques de t caracteres, y aplicando para cada bloque una única permutación c sobre todos los caracteres del bloque |
| **Sustitución de vigenére** | Polialfabético | Se usan t cambios de cifrado, definiendo t valores de cambio para $k_i$, especificando una de s substituciones |

| | | |
|---|---|---|
| | | cambio para $k_i$, especificando una de s substituciones (Monoalfabético), en general cada sustitución es diferente y es equivalente a usar t alfabetos |
| **Sustitución polialfabético (cilindro de jefferson)** | Polialfabético | Es un cilindro de 6 pulgadas de largo formado por 36 discos con una barra insertada a través de los ejes del cilindro para hacerlo girar, la periferia de cada disco es dividida en 26 partes, sobre cada disco son escritas letras de la A-Z en orden aleatorio |
| **Sustitución polialfabético (rotores)** | Polialfabético | Consiste de un número de rotores, cada uno implementando una substitución Monoalfabetica fija diferente, mapeando un carácter en su cara de entrada obteniendo una salida en su cara de salida |

**Tabla 2.1.1 Clasificación de algoritmos convencionales**

## 1.3 Análisis de sistemas modernos

En los sistemas modernos se habla de esquemas de encriptación, los cuales están determinados por el conjunto formado por $\{P,C,K,E,D\}$, donde cada subconjunto se define como.

$$\left\{\begin{array}{l} P = Conjunto\ \ de\ \ textos\ \ claros \\ C = Conjunto\ \ de\ \ textos\ \ cifrados \\ K = Conjunto\ \ de\ \ claves\ \ de\ \ encriptación \\ E = Familia\ \ de\ \ funciones\ \ de\ \ encriptación \\ D = Familia\ \ de\ \ funciones\ \ de\ \ desencriptación \end{array}\right\}$$

Las funciones de encriptación actúan como $E_k : P \to C, \forall\ k \in K$ y las de desencriptación como $D_k : C \to P, \forall\ k \in K$ , para que un esquema de encriptación se considere como tal, debe cumplir la propiedad $\forall\ e \in K, \exists\ d \in K\ /\ D_k(E_e(p)) = p, \forall\ p \in P$ , donde un esquema de encriptación se denomina simétrico cuando d y e son iguales, por otro lado cuando d y e son distintos se denomina asimétrico. Por otro lado se tiene los protocolos criptográficos que son algoritmos distribuidos, definidos por una secuencia de pasos

precisos especificando las acciones requeridas de dos o más entidades para lograr un objetivo específico de seguridad.

### 1.3.1 Propiedades

- En un Esquema de Encriptación se denota $P_p(p)$ a la probabilidad de que un texto claro p aparezca en P, de forma similar la probabilidad de una clave se denota por $P_k(k)$. Por lo tanto la probabilidad de que un texto claro p aparezca cifrado por la clave k es $P(p,k) = P_p(p) \bullet P_k(k)$ , la función P define una distribución de probabilidad en el espacio producto P X K. Entonces un esquema de encriptación será seguro perfecto si $P(p/c) = P(p) \quad \forall p \in P, \forall c \in C$. Es decir, que la probabilidad de un texto cifrado y la probabilidad de un texto claro hayan sido cifrados son independientes.
- Teorema de Shannon, Sea $|C| = |K| \quad y \quad P(p) > 0, \forall p \in P$ , el esquema de encriptación será seguro perfecto si y solo si la distribución de probabilidad en el espacio de claves es uniforme y si para cualquier texto claro p y texto cifrado c, existe una clave única con $E_k(p) = c$.
- Entropía. Con esta propiedad conseguimos optimizar la longitud de la codificación generada, es decir, es posible obtener un número medio de bits necesarios para cifrar cada uno de los estados posibles de una variable aleatoria. Si en un esquema de encriptación cada estado representa un símbolo a cifrar para transmitir, la entropía nos da los bits/símbolo necesarios para cifrar. Matemáticamente se define la entropía como una suma ponderada de la cantidad de información de todos los posibles estados de una variable aleatoria V

y se expresa como $H(V) = -\sum_{i=1}^{n} P(X_i)\log_2[P(X_i)] = \sum_{i=1}^{n} P(X_i)\log_2\left[\frac{1}{P(X_i)}\right]$ ,

donde $X_i$ representa los estados de V.

- $0 \le H(V) \le \log_2 N$ , donde $\log_2 N$ se da cuando $P(X_i) = \frac{1}{N}\ \forall i$
- $H(V) = 0 \Leftrightarrow \exists\, i / P(X_i) = 1 \quad y \quad P(X_j) = 0\ \forall i \neq j$
- $H(X_1, X_2, ..., X_n) = H(X_1, X_2, ..., X_n, X_{n+1})\ si\ P(X_{n+1}) = 0$

- Shannon propuso que la cantidad de información entre dos variables se representa por $I(X,Y) = H(Y) - H(Y / X)$ , esto significa que la cantidad de información que nos aporta el hecho de conocer X al medir la incertidumbre de Y es igual a la disminución de la entropía que produce este conocimiento.
- Un esquema de encriptación será ideal de shannon si:
  - La cantidad de información que nos aporta conocer el mensaje cifrado C sobre la entropía del texto en claro es 0, es decir $I(C,M) = 0$ , con $H(M) = H(M / C)$ y donde M es el texto en claro.
  - La cardinalidad del espacio de claves debe ser como mínimo igual a la del espacio de mensajes. Es decir tener claves tan largas como mensaje.
  - Que a partir de K claves se generen seudo claves más largas K'.
  - Que el sistema de claves sea aleatoria y continua de un solo uso.
- El índice absoluto de un lenguaje se define como $R = \log_2(M)$ , que es la cantidad máxima de información que puede ser codificada por cada carácter.
- Los lenguajes naturales son redundantes, la entropía de un mensaje de longitud N se aleja bastante de la máxima teórica. Por ello la entropía real de un mensaje se define como Índice de un Lenguaje y esta expresado por

$r_k = H_k(M)/K$ para mensajes de largo K, donde $H_k(M)$ es la entropía de todos los posibles mensajes de largo K.

- La redundancia de un mensaje es un aspecto importante, ya que mientras la entropía sea menor para un mensaje mayor será la redundancia y será más fácil quebrar el esquema de encriptación. La redundancia se define como $D = R - r$, donde r es la entropía. Y el índice de redundancia como $I = D/R$. Para lograr ocultar la redundancia se pueden utilizar dos formas:
  - Confusión. Trata de ocultar la relación entre m y c a través de k (m: mensaje en claro, c: mensaje cifrado y k: clave de cifrado). Se logra con la substitución, es decir cambiar cada ocurrencia del texto en claro por otra en el cifrado.
  - Difusión. Diluye la redundancia del texto en claro, repartiéndola a lo largo de todo el texto cifrado. Se logra con la transposición, es decir cambiar de posición elementos individuales del texto en claro.
- Básicamente los esquemas de encriptación no están libres de ataques, estos ataques pueden ser sobre los mismos esquemas o sobre los protocolos de encriptación, entre los más importantes se pueden mencionar.
  - Fuerza Bruta. Consiste en probar con todas las claves posibles, dentro de este ámbito se puede hacer una búsqueda lógica (buscar en diccionarios aplicado a password) o aleatoria, en este último generalmente se considera una probabilidad de acierto del 50%.
  - Error de Diseño. Consiste en encontrar debilidades en el diseño del algoritmo de cifrado.
  - Aproximaciones. Consiste en encontrar aproximaciones matemáticas al algoritmo.

### 1.3.2 Análisis de 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 clave, tanto para encriptar como para desencriptar. La criptografía simétrica ha sido la más usada en toda la historia, a podido ser implementada en diferente dispositivos, manuales, mecánicos, eléctricos, en definitiva son algoritmos que en su mayoría pueden ser programables en un computador. La idea general es aplicar diferentes funciones al mensaje que se quiere cifrar, de tal modo que solo conociendo una clave pueda aplicarse de forma inversa para poder así descifrar el mensaje. Existe una clasificación de este tipo de criptografía en tres familias, la criptografía simétrica por bloques (**block cipher**), la criptografía simétrica por flujo (**stream cipher**) y la criptografía simétrica de resumen (**hash functions**). Pero haciendo ligeras modificaciones es posible pasar desde un cifrado por bloque a un cifrado por flujo o por de resumen. Aunque no existe un tipo de diseño estándar, quizá el más popular es el de Fiestel, que consiste esencialmente en aplicar un número finito de iteraciones de determinada forma, que finalmente da como resultado el mensaje cifrado [Van+, 1996].

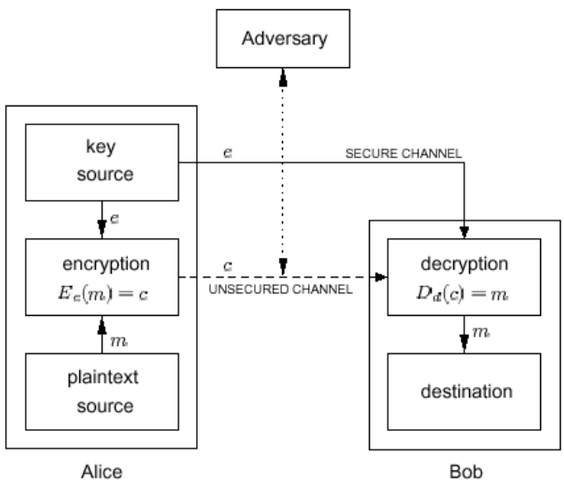



**Fig. 2.1.4 Proceso de encriptación simétrico**

### 1.3.2.1 Cifrado por bloques (block cipher)

Es un esquema de encriptación que divide un mensaje de texto plano a ser transmitido en cadenas (bloques) de largo fijo t sobre un alfabeto A y finalmente encripta ese bloque. Este tipo de cifrado puede ser cifrado por sustitución y cifrado por transposición.

#### 1.3.2.1.1 Modos de operación del cifrado por bloques

Un cifrado por bloque encripta texto plano en bloques de n bits de tamaño fijo (usualmente de 64). Para mensajes que exceden los n bits, una aproximación simple es dividir el mensaje en bloques de n bits y encriptarlos en forma separada. Para ello existen cuatro modos de operación ECB, CBC, CFB y OFB.

- ECB (Electronic CodeBook). Dado que los bloques de texto cifrado son independientes, sustituciones maliciosas de bloques ECB no afectan la desencriptación de bloques adyacentes. Además un cifrado por bloques no oculta los patrones de los datos, es decir que bloques de texto cifrado idénticos implican bloques de texto plano idénticos. Por esta razón no es recomendable usar el modo ECB para mensajes que exceden el tamaño del bloque, se utiliza en mensajes cortos de menos de 64 bits.

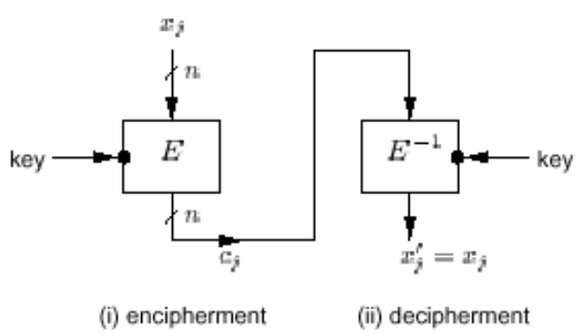



**Fig. 2.1.5 Modo ECB**

- CBC (Cipher-Block Chaining). A pesar que esta modalidad al desencriptar recupera la información proveniente de bloques de texto cifrado con errores, modificaciones para un bloque de texto plano durante la encriptación altera todas las subsequencias de bloques cifrados. Esto daña la funcionalidad de los modos de asociación a causa de que las aplicaciones requieren acceso de lectura y escritura aleatoria para encriptar la data, generalmente se utiliza en mensajes largos.

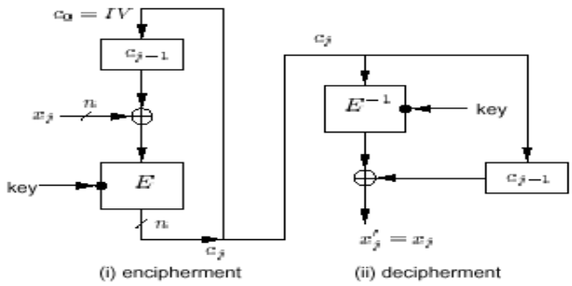



**Fig. 2.1.6 Modo CBC**

- CFB (Cipher Feedback). Puesto que la función de encriptación E es usada para encriptación y desencriptación CFB. Esta modalidad no debe ser usada si el cifrado por bloque E es un algoritmo de clave pública, en otro caso seria bién usado. Además esta modalidad tiene una variante ISO, su uso es para cifrar bit por bit o byte por byte.

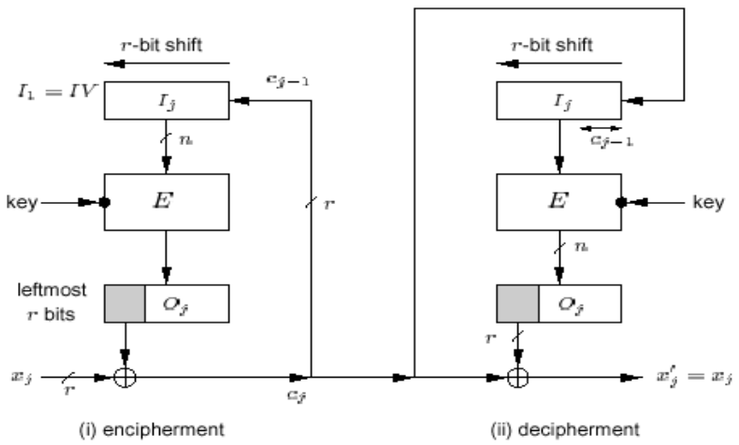



**Fig. 2.1.7 Modo CFB**

- OFB (Output Feedback). Puede ser usado donde todo error de propagación puede ser evitado. Es similar a CFB y permite encriptación de varios tamaños de bloque, pero difiere en que un bloque de salida de la función de encriptación E sirve como realimentación. Existen 2 versiones de OFB, la versión ISO 10116 que requiere una alimentación de n bits, esta es más segura, por otro lado esta la primera versión FIPS que permite r<n bits de realimentación. Su uso es para cifrar bit por bit o byte por byte y evita propagación de errores.

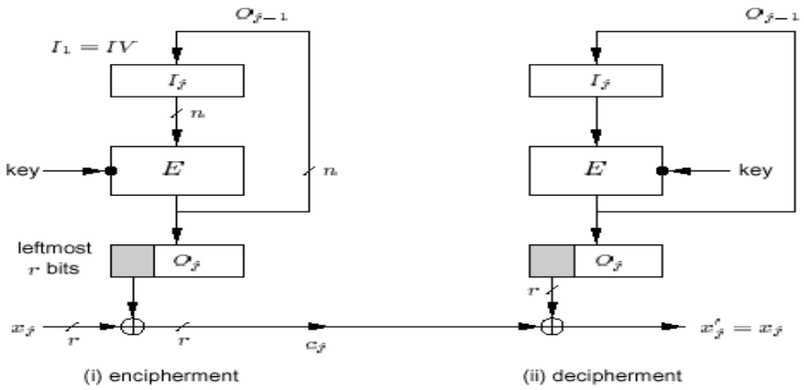



**Fig. 2.1.8 Modo OFB ISO 10116**

### 1.3.2.1.2 Algoritmo DES (Data Encryption Standard)

Es un cifrado de Fiestel que procesa bloques de texto plano de 64 bits, produciendo bloques de texto cifrado de 64 bits. El tamaño efectivo de la clave secreta K es 56 bits, más precisamente, la clave K de entrada es especificada como una clave de 64 bits, de los cuales 8 bits pueden ser usados como bits de paridad (bits 8, 16, …, 64). Los $2^{56}$ implementan (como máximo) $2^{56}$ claves de los $2^{64}!$ posibles biyecciones sobre bloques de 64 bits. Una creencia ampliamente adoptada es que los bits de paridad fueron introducidos para reducir la efectividad del tamaño de la clave desde 64 a 56 bits, para intencionalmente reducir el costo de la búsqueda exhaustiva de claves por un factor de 256 [ANS+, 1983].

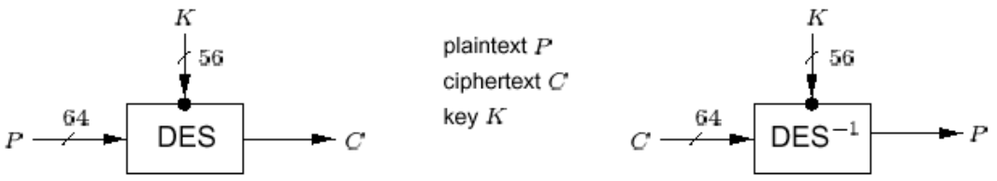



**Fig. 2.1.9 Entrada y salida DES**

El proceso de encriptación conforma 16 estados o ciclos, desde la entrada de la clave K, 16 subclaves $K_i$ de 48 bits son generadas, una para cada ciclo. Dentro de cada ciclo, se selecciona cuidadosamente 8 $S_i$ (S-cajas) fijas de 6 a 4 bits para el mapeado de sustitución. Los 64 bits de texto plano son divididos en dos partes $L_0$ y $R_0$ de 32 bits, cada ciclo es funcionalmente equivalente, tomando entradas $L_{i-1}$ y $R_{i-1}$ de 32 bits desde el ciclo anterior y produciendo salidas $L_i$ y $R_i$ de 32 bits, generalizando para $1 \le i \le 16$.

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$
$$f(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i))$$

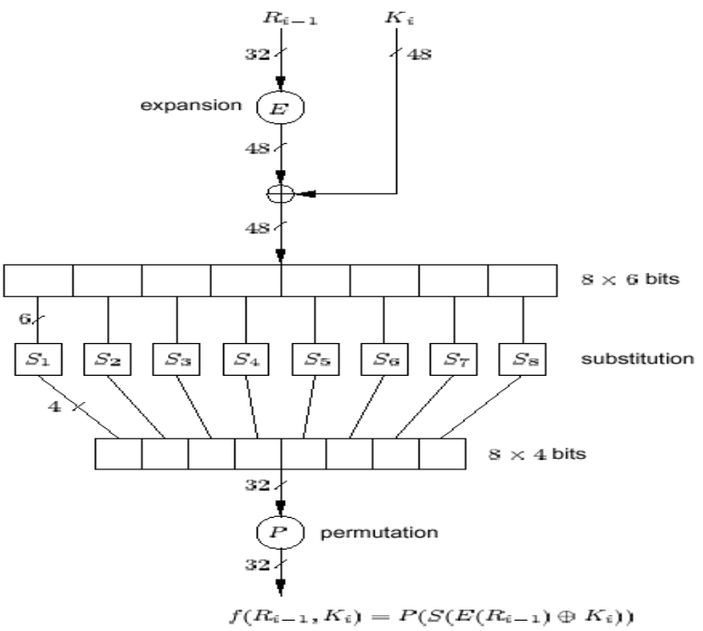



**Fig. 2.1.10 Función interna f**

Donde E es una expansión de una permutación fija mapeando $R_{i-1}$ de 32 a 48 bits (todos los bits son usados una vez, algunos son usados dos veces). P es otra permutación fija sobre 32 bits, un bit inicial de permutación (IP) precede el primer ciclo, siguiendo hasta el último ciclo, la parte izquierda y derecha son intercambiadas, finalmente la cadena resultante es un bit permutado por el inverso de IP. La desencriptación involucra la misma clave y algoritmo, pero con subclaves aplicadas en orden inverso para los ciclos internos.

*Algoritmo DES*

Entrada:

- Texto plano $m_1 \ldots m_{64}$
- Claves $K = k_1 \ldots k_{64}$ de 64 bits (incluye bits de paridad)

Salida:

- son bloques de texto cifrado $C = c_1 \ldots c_{64}$ de 64 bits

Encriptación:

- Programa de claves, 16 claves $K_i$ de 48 bits desde K
- $(L_0, R_0) \leftarrow IP(m_1 m_2 \ldots m_{64})$, usar tabla IP. Divide el resultado en izquierda y derecha es decir, $L_0 = m_{58} m_{50} \ldots m_8$ y $R_0 = m_{57} m_{49} \ldots m_7$ de 32 bits.
- *for* $i = 1$ *to* 16
    1. $L_i = R_{i-1}$
    2. $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$
    3. Calculando $f(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i))$ como sigue:

a. Expandir $R_{i-1} = r_1 r_2 ... r_{32}$ de 32 a 48 bits usando E de la tabla.

   $T \leftarrow E(R_{i-1})$ , así $T = r_{32} r_1 r_2 ... r_{32} r_1$

b. $T' \leftarrow T \oplus k_i$ , representa $T'$ como 8 cadenas de caracteres de 6 bit $(B_1,...,B_8) = T'$

c. $T'' \leftarrow (S_1(B_1), S_2(B_2),..., S_8(B_8))$ , aquí $S_i(B_i)$ mapea $B_i = b_1 b_2 ... b_6$ para los 4 bit de entrada en la fila r y columna c de $S_i$ en la tabla S-cajas, donde $r = 2 \bullet b_1 + b_6$ y $b_2 b_3 b_4 b_5$ es la representación en base 2 de $0 \leq c \leq 15$.

d. $T''' \leftarrow P(T'')$, usar P de la tabla P' para permutar los 32 bits de $T'' = t_1 t_2 ... t_{32}$.

- $b_1 b_2 ... b_{64} \leftarrow (R_{16}, L_{16})$, intercambia bloques finales $R_{16}, L_{16}$.
- $C \leftarrow IP^{-1}(b_1 b_2 ... b_{64})$, transponer usando $IP^{-1}$ de la tabla.

Desencriptación:

- Consiste del algoritmo de encriptación con la misma clave, pero invirtiendo el programa de claves usando el orden $K_{16}, K_{15}, ..., K_1$. El efecto de $IP^{-1}$ es cancelado por $IP$ en la desencriptación, partiendo por $(R_{16}, L_{16})$, considerar imputación del ciclo 1 para esta entrada. La operación sobre el producto de la izquierda, antes $L_0 \oplus f(R_0, K_1)$, ahora $R_{16} \oplus f(L_{16}, K_{16})$ obteniendo los valores desde $L_{16} = R_{15}$ y $R_{16} = L_{15} \oplus f(R_{15}, K_{16})$, esto es igual a $L_{15} \oplus f(R_{15}, K_{16}) \oplus f(R_{15}, K_{16}) = L_{15}$. Así la desencriptación del

ciclo 1 produce $(R_{15}, L_{15})$. Notar que la cancelación de cada ciclo es independiente de la definición de f y del valor específico de $K_i$, el intercambio de las partes combinadas con el proceso XOR es invertido por la segunda aplicación. La diferencia de los 15 ciclos son cancelados así mismo una por una en orden inverso a la aplicación, debido a el programa inverso de claves.

| IP | | | | | | | |
|---|---|---|---|---|---|---|---|
| 58 | 50 | 42 | 34 | 26 | 18 | 10 | 2 |
| 60 | 52 | 44 | 36 | 28 | 20 | 12 | 4 |
| 62 | 54 | 46 | 38 | 30 | 22 | 14 | 6 |
| 64 | 56 | 48 | 40 | 32 | 24 | 16 | 8 |
| 57 | 49 | 41 | 33 | 25 | 17 | 9 | 1 |
| 59 | 51 | 43 | 35 | 27 | 19 | 11 | 3 |
| 61 | 53 | 45 | 37 | 29 | 21 | 13 | 5 |
| 63 | 55 | 47 | 39 | 31 | 23 | 15 | 7 |

| $E$ | | | | | |
|---|---|---|---|---|---|
| 32 | 1 | 2 | 3 | 4 | 5 |
| 4 | 5 | 6 | 7 | 8 | 9 |
| 8 | 9 | 10 | 11 | 12 | 13 |
| 12 | 13 | 14 | 15 | 16 | 17 |
| 16 | 17 | 18 | 19 | 20 | 21 |
| 20 | 21 | 22 | 23 | 24 | 25 |
| 24 | 25 | 26 | 27 | 28 | 29 |
| 28 | 29 | 30 | 31 | 32 | 1 |

| $IP^{-1}$ | | | | | | | |
|---|---|---|---|---|---|---|---|
| 40 | 8 | 48 | 16 | 56 | 24 | 64 | 32 |
| 39 | 7 | 47 | 15 | 55 | 23 | 63 | 31 |
| 38 | 6 | 46 | 14 | 54 | 22 | 62 | 30 |
| 37 | 5 | 45 | 13 | 53 | 21 | 61 | 29 |
| 36 | 4 | 44 | 12 | 52 | 20 | 60 | 28 |
| 35 | 3 | 43 | 11 | 51 | 19 | 59 | 27 |
| 34 | 2 | 42 | 10 | 50 | 18 | 58 | 26 |
| 33 | 1 | 41 | 9 | 49 | 17 | 57 | 25 |

| $P$ | | | |
|---|---|---|---|
| 16 | 7 | 20 | 21 |
| 29 | 12 | 28 | 17 |
| 1 | 15 | 23 | 26 |
| 5 | 18 | 31 | 10 |
| 2 | 8 | 24 | 14 |
| 32 | 27 | 3 | 9 |
| 19 | 13 | 30 | 6 |
| 22 | 11 | 4 | 25 |

**Tabla 2.1.2 Tablas de permutación inicial y inversa, tablas de funciones por ciclo de expansión y permutación**

| row | column number | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9] | [10] | [11] | [12] | [13] | [14] | [15] |
| $S_1$ | | | | | | | | | | | | | | | | |
| [0] | 14 | 4 | 13 | 1 | 2 | 15 | 11 | 8 | 3 | 10 | 6 | 12 | 5 | 9 | 0 | 7 |
| [1] | 0 | 15 | 7 | 4 | 14 | 2 | 13 | 1 | 10 | 6 | 12 | 11 | 9 | 5 | 3 | 8 |
| [2] | 4 | 1 | 14 | 8 | 13 | 6 | 2 | 11 | 15 | 12 | 9 | 7 | 3 | 10 | 5 | 0 |
| [3] | 15 | 12 | 8 | 2 | 4 | 9 | 1 | 7 | 5 | 11 | 3 | 14 | 10 | 0 | 6 | 13 |
| $S_2$ | | | | | | | | | | | | | | | | |
| [0] | 15 | 1 | 8 | 14 | 6 | 11 | 3 | 4 | 9 | 7 | 2 | 13 | 12 | 0 | 5 | 10 |
| [1] | 3 | 13 | 4 | 7 | 15 | 2 | 8 | 14 | 12 | 0 | 1 | 10 | 6 | 9 | 11 | 5 |
| [2] | 0 | 14 | 7 | 11 | 10 | 4 | 13 | 1 | 5 | 8 | 12 | 6 | 9 | 3 | 2 | 15 |
| [3] | 13 | 8 | 10 | 1 | 3 | 15 | 4 | 2 | 11 | 6 | 7 | 12 | 0 | 5 | 14 | 9 |
| $S_3$ | | | | | | | | | | | | | | | | |
| [0] | 10 | 0 | 9 | 14 | 6 | 3 | 15 | 5 | 1 | 13 | 12 | 7 | 11 | 4 | 2 | 8 |
| [1] | 13 | 7 | 0 | 9 | 3 | 4 | 6 | 10 | 2 | 8 | 5 | 14 | 12 | 11 | 15 | 1 |
| [2] | 13 | 6 | 4 | 9 | 8 | 15 | 3 | 0 | 11 | 1 | 2 | 12 | 5 | 10 | 14 | 7 |
| [3] | 1 | 10 | 13 | 0 | 6 | 9 | 8 | 7 | 4 | 15 | 14 | 3 | 11 | 5 | 2 | 12 |
| $S_4$ | | | | | | | | | | | | | | | | |
| [0] | 7 | 13 | 14 | 3 | 0 | 6 | 9 | 10 | 1 | 2 | 8 | 5 | 11 | 12 | 4 | 15 |
| [1] | 13 | 8 | 11 | 5 | 6 | 15 | 0 | 3 | 4 | 7 | 2 | 12 | 1 | 10 | 14 | 9 |
| [2] | 10 | 6 | 9 | 0 | 12 | 11 | 7 | 13 | 15 | 1 | 3 | 14 | 5 | 2 | 8 | 4 |
| [3] | 3 | 15 | 0 | 6 | 10 | 1 | 13 | 8 | 9 | 4 | 5 | 11 | 12 | 7 | 2 | 14 |
| $S_5$ | | | | | | | | | | | | | | | | |
| [0] | 2 | 12 | 4 | 1 | 7 | 10 | 11 | 6 | 8 | 5 | 3 | 15 | 13 | 0 | 14 | 9 |
| [1] | 14 | 11 | 2 | 12 | 4 | 7 | 13 | 1 | 5 | 0 | 15 | 10 | 3 | 9 | 8 | 6 |
| [2] | 4 | 2 | 1 | 11 | 10 | 13 | 7 | 8 | 15 | 9 | 12 | 5 | 6 | 3 | 0 | 14 |
| [3] | 11 | 8 | 12 | 7 | 1 | 14 | 2 | 13 | 6 | 15 | 0 | 9 | 10 | 4 | 5 | 3 |
| $S_6$ | | | | | | | | | | | | | | | | |
| [0] | 12 | 1 | 10 | 15 | 9 | 2 | 6 | 8 | 0 | 13 | 3 | 4 | 14 | 7 | 5 | 11 |
| [1] | 10 | 15 | 4 | 2 | 7 | 12 | 9 | 5 | 6 | 1 | 13 | 14 | 0 | 11 | 3 | 8 |
| [2] | 9 | 14 | 15 | 5 | 2 | 8 | 12 | 3 | 7 | 0 | 4 | 10 | 1 | 13 | 11 | 6 |
| [3] | 4 | 3 | 2 | 12 | 9 | 5 | 15 | 10 | 11 | 14 | 1 | 7 | 6 | 0 | 8 | 13 |
| $S_7$ | | | | | | | | | | | | | | | | |
| [0] | 4 | 11 | 2 | 14 | 15 | 0 | 8 | 13 | 3 | 12 | 9 | 7 | 5 | 10 | 6 | 1 |
| [1] | 13 | 0 | 11 | 7 | 4 | 9 | 1 | 10 | 14 | 3 | 5 | 12 | 2 | 15 | 8 | 6 |
| [2] | 1 | 4 | 11 | 13 | 12 | 3 | 7 | 14 | 10 | 15 | 6 | 8 | 0 | 5 | 9 | 2 |
| [3] | 6 | 11 | 13 | 8 | 1 | 4 | 10 | 7 | 9 | 5 | 0 | 15 | 14 | 2 | 3 | 12 |
| $S_8$ | | | | | | | | | | | | | | | | |
| [0] | 13 | 2 | 8 | 4 | 6 | 15 | 11 | 1 | 10 | 9 | 3 | 14 | 5 | 0 | 12 | 7 |
| [1] | 1 | 15 | 13 | 8 | 10 | 3 | 7 | 4 | 12 | 5 | 6 | 11 | 0 | 14 | 9 | 2 |
| [2] | 7 | 11 | 4 | 1 | 9 | 12 | 14 | 2 | 0 | 6 | 10 | 13 | 15 | 3 | 5 | 8 |
| [3] | 2 | 1 | 14 | 7 | 4 | 10 | 8 | 13 | 15 | 12 | 9 | 0 | 3 | 5 | 6 | 11 |

**Tabla 2.1.3 Tabla DES S-cajas**

*Algoritmo Generador de Claves DES*

Entrada:

- Claves $K = k_1 \ldots k_{64}$ de 64 bits (incluyendo 8 bits de paridad)

Salida:

- 16 claves $K_i$ de 48 bits, $1 \leq i \leq 16$

Proceso:

- Definir $v_i$ con $1 \leq i \leq 16$ como sigue.

  1. $v_i = 1$ para $i \in \{1,2,9,16\}$
  2. $v_i = 2$ en otro caso

  Estos valores son dirigidos a la izquierda para rotaciones circulares bajo 28 bits.

- $T \leftarrow PC1(K)$, representa T como la división $(C_0, D_0)$ de 28 bits, usar la tabla PC1 para seleccionar bits desde K, con $C_0 = k_{57}k_{49}...k_{36}$ y $D_0 = k_{63}k_{55}...k_4$.
- *for* $i = 1$ *to* 16
    1. $C_i \leftarrow (C_{i-1} \circlearrowleft v_i)$
    2. $D_i \leftarrow (D_{i-1} \circlearrowleft v_i)$
    3. $K_i \leftarrow PC2(C_i, D_i)$, usar la tabla PC2 para seleccionar 48 bits de la concatenación $K_i = b_1 b_2 ... b_{56}$ de $C_i$ y $K_i = b_{14} b_{17} ... b_{32}$ de $D_i$, $\circlearrowleft$ denota un cambio circular izquierdo.

Subclaves $K_1,...,K_{16}$ pueden ser generadas por este algoritmo, en el caso de la desencriptación, estas son generadas y usadas en orden inverso, notar que después de que $K_{16}$ es generada, los valores originales de los registros C y D de 28 bits son restaurados. Consecuentemente y debido al cambio de los valores se puede modificar el algoritmo para generar subclaves en orden $K_{16},...,K_1$, remplazando los cambios en la izquierda por rotaciones de cambios en la derecha, cambiando el valor de cambio $v_1$ a 0.

| PC1 | | | | | | |
|---|---|---|---|---|---|---|
| 57 | 49 | 41 | 33 | 25 | 17 | 9 |
| 1 | 58 | 50 | 42 | 34 | 26 | 18 |
| 10 | 2 | 59 | 51 | 43 | 35 | 27 |
| 19 | 11 | 3 | 60 | 52 | 44 | 36 |
| above for $C_i$; below for $D_i$ | | | | | | |
| 63 | 55 | 47 | 39 | 31 | 23 | 15 |
| 7 | 62 | 54 | 46 | 38 | 30 | 22 |
| 14 | 6 | 61 | 53 | 45 | 37 | 29 |
| 21 | 13 | 5 | 28 | 20 | 12 | 4 |

| PC2 | | | | | |
|---|---|---|---|---|---|
| 14 | 17 | 11 | 24 | 1 | 5 |
| 3 | 28 | 15 | 6 | 21 | 10 |
| 23 | 19 | 12 | 4 | 26 | 8 |
| 16 | 7 | 27 | 20 | 13 | 2 |
| 41 | 52 | 31 | 37 | 47 | 55 |
| 30 | 40 | 51 | 45 | 33 | 48 |
| 44 | 49 | 39 | 56 | 34 | 53 |
| 46 | 42 | 50 | 36 | 29 | 32 |

**Tabla 2.1.4 Tabla de selección de bit para el programa de claves**

#### 1.3.2.1.3 Otros algoritmos para cifrado por bloques

A continuación se resumen los más importantes algoritmos simétricos para cifrado por bloques que se analizaron desde el punto de vista de los sistemas informáticos actuales.

| Algoritmo | Detalle |
|---|---|
| **Cifrado de fiestel** | Es un cifrado iterativo mapeado en un texto plano $(L_0, R_0)$ de 2t bit, para bloques $L_0$ y $R_0$ de t bit, para un texto cifrado $(L_r, R_r)$, a través de r ciclos donde $r \geq 1$ |
| **FEAL (Fast Data Encipherment)** | FEAL-N mapea 64 bits de texto plano para bloques de texto cifrado de 64 bits sobre una clave secreta de 64 bits, esto son N-ciclos del cifrado de Fiestel muy similar a DES |
| **IDEA (International Data Encryption)** | Encripta texto plano de 64 bit para bloques de texto cifrado de 64 bit usando una clave de entrada de 128 bit, basado en parte sobre la generalización de la estructura de Fiestel y consiste de 8 ciclos de calculo idéntico seguido de una transformación en la salida |
| **SAFER (Secure And Fast Encryption Routine)** | Es un cifrado iterativo con bloques de texto plano de 64 bit y bloques de texto cifrado de 64 bit, con una clave de cifrado de 64 bit, a lo que se denomina SAFER K-64. Consiste en r ciclos idénticos seguido por una transformación en la salida, la recomendación original fue de 6 ciclos |
| **RC5** | Es un algoritmo con una arquitectura orientada a la palabra para palabras de largo variables, con w=16, 32 o 64 bit. El número de ciclos r y la clave b con largo en byte, la cual igual puede ser variable. Esto es conocido más ampliamente como RC5-w, RC5-w/r y RC5-w/r/b |

**Tabla 2.1.5 Clasificación de algoritmos para cifrado por bloques**

### 1.3.2.2 Cifrado por flujo (stream cipher)

Se define K como el espacio de claves para un conjunto de transformaciones de encriptación y una secuencia de símbolos $e_1 e_2 \ldots e_i \in K$ es llamada flujo de claves. Se tiene un alfabeto A de q símbolos y $E_e$ que es un cifrado de sustitución simple con bloques de largo 1 donde e $\in$ K. Además $m_1 m_2 \ldots$ son cadenas de texto plano y $e_1 e_2 \ldots$ son claves de K. Un cifrado por flujo toma cadenas de texto plano y produce

cadenas de texto cifrado $c_1\, c_2 \ldots$ donde $c_i = E_{ei}(m_i)$. Si $d_i$ denota el inverso de $e_i$ entonces $D_{di}(c_i) = m_i$ desencripta la cadena de texto cifrado. Este cifrado aplica transformaciones de encriptación simple de acuerdo con las claves de flujo estipuladas, estas claves podrían generarse aleatoriamente o con un algoritmo que las genere a partir de claves más pequeñas llamadas seed (semilla) [Van+, 1996].

El cifrado por flujo procesa bloques muchos más pequeños incluso bit individuales y la función de encriptación puede variar dependiendo como el texto plano es procesado. Así el cifrado por flujo es indicado cuando se tiene memoria, algunas veces estas son llamadas cifrado de estados, dado que la encriptación no solo depende de la clave y del texto plano, ya que también del estado del flujo.

#### 1.3.2.2.1 Clasificaciones

##### 1.3.2.2.1.1 Una sola pasada (One-Time Pad)

Es cuando se tiene $m_1, m_2, m_3, \ldots$ dígitos de texto plano, $k_1, k_2, k_3, \ldots$ dígitos de claves (flujo de claves), $c_1, c_2, c_3, \ldots$ dígitos de texto cifrado y una función XOR ($\oplus$) que trabaja en el nivel de bit y es una adición módulo 2. Se define la función de encriptación como $c_i = m_i \oplus k_i$ para $i = 1,2,3,\ldots$ y la función de desencriptación se define como $m_i = c_i \oplus k_i$, el flujo de claves es generado independientemente y en forma aleatoria, un algoritmo de este tipo se le considera incondicionalmente seguro, es decir se puede demostrar teóricamente que es irrompible. Un inconveniente de este cifrado es que la clave puede ser larga como el texto plano, lo que incrementa la dificultad de la distribución y administración de las claves. Esto ha motivado el diseño de cifrados por flujo donde el flujo de claves es generado seudo aleatoriamente desde

una clave secreta más pequeña, con la intención de que el flujo de claves aparezca aleatorio para un adversario definido.

#### 1.3.2.2.1.2 Cifrado por flujo síncrono

Es cuando el flujo de claves es generado independientemente del mensaje de texto plano y del texto cifrado. El proceso de encriptación puede ser descrito por las siguientes ecuaciones [Van+, 1996].

$$\sigma_{i+1} = f(\sigma_i, k)$$
$$z_i = g(\sigma_i, k)$$
$$c_i = h(z_i, m_i)$$

Donde $\sigma_0$ es el estado inicial y puede ser determinado desde la clave k, f es la función del siguiente estado, g es la función que produce el flujo de claves $z_i$, y h es la función de salida que combina el flujo de claves y texto plano $m_i$ para producir texto cifrado $c_i$. El modo OFB de un cifrado por bloque es un ejemplo de un cifrado por flujo síncrono.

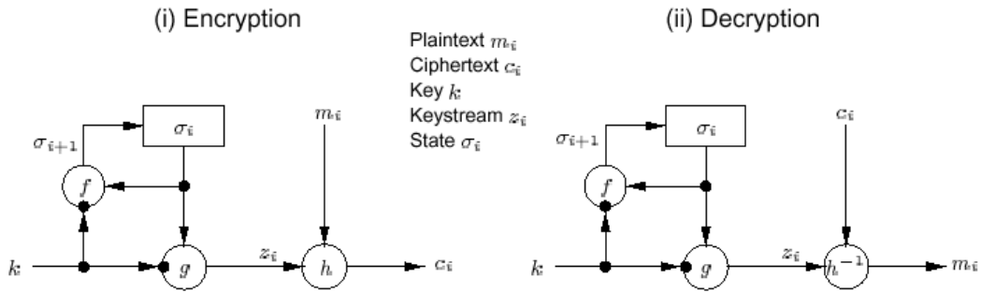



**Fig. 2.1.11 Modelo general de un cifrado por flujo sincronizado**

La mayor parte de los cifrados por flujo que se han propuesto a la fecha en la literatura se les conoce como cifrados por flujo aditivos. Estos son un cifrado por flujo sincronizado en que el flujo de claves, texto plano y el texto cifrado son dígitos binarios

y la función de salida h es la función XOR, en cuanto al generador del flujo de claves esta compuesto de la función f del siguiente estado y la función g.

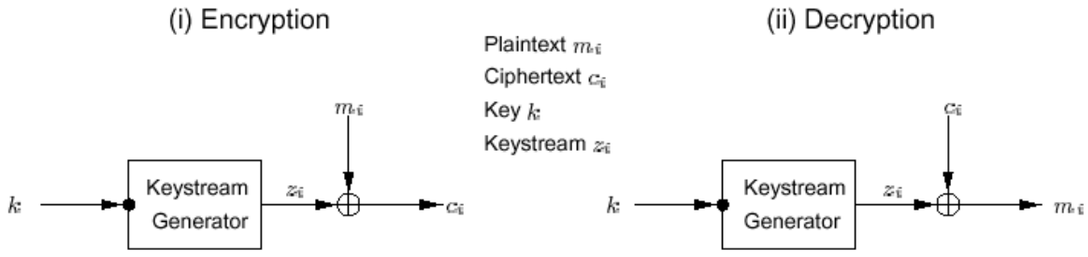



**Fig. 2.1.12 Modelo general de un cifrado por flujo aditivo binario**

**Propiedades:**

1. Requerimientos de Sincronización, el emisor y el receptor deben estar sincronizados usando la misma clave y operando la misma posición (estado) con esa clave para dejar un márgen correcto de desencriptación. Si el sincronismo es perdido debido a que dígitos de texto cifrado fueron insertados o borrados durante la transformación, entonces la desencriptación falla y solo puede ser restaurada a través de técnicas adicionales para resincronización. Entre estas técnicas tenemos reinicialización, colocando marcas especiales a intervalos regulares en el texto cifrado o si el texto plano contiene bastante redundancia.
2. Sin error de Propagación, un digito de texto cifrado que es modificado (pero no borrado) durante la transmisión no afecta la desencriptación de otros dígitos de texto cifrado.
3. Ataques activos, como una consecuencia de la propiedad 1, la inserción , borrado o reemplazo de dígitos de texto cifrado por un atacante activo causa la perdida inmediata de la sincronización y por lo tanto es posible ser detectado por un descriptor. Por otro lado como consecuencia de la propiedad 2, un atacante activo es posible que este capacitado para hacer cambios a dígitos de texto

cifrado específicos conociendo exactamente como afectan esos cambios sobre el texto plano.

#### 1.3.2.2.1.3 Cifrado por flujo asíncrono

Es cuando el flujo de claves es generado como una función de la clave y del número fijo de dígitos anteriores de texto cifrado. El proceso de encriptación se puede definir por las siguientes ecuaciones [Van+, 1996].

$$\sigma_i = (c_{i-t}, c_{i-t+1}, ..., c_{i-1})$$
$$z_i = g(\sigma_i, k)$$
$$c_i = h(z_i, m_i)$$

Donde $\sigma_0 = (c_{-t}, c_{-t+1}, ..., c_{-1})$ es el estado inicial (no secreto), k es la clave y g es la función que produce el flujo de claves $z_i$ y h es la función de salida que combina el flujo de claves y texto plano $m_i$ para producir texto cifrado $c_i$. El uso más común de este cifrado, en este momento esta basado sobre un cifrado por bloques en modo realimentación de 1 bit.

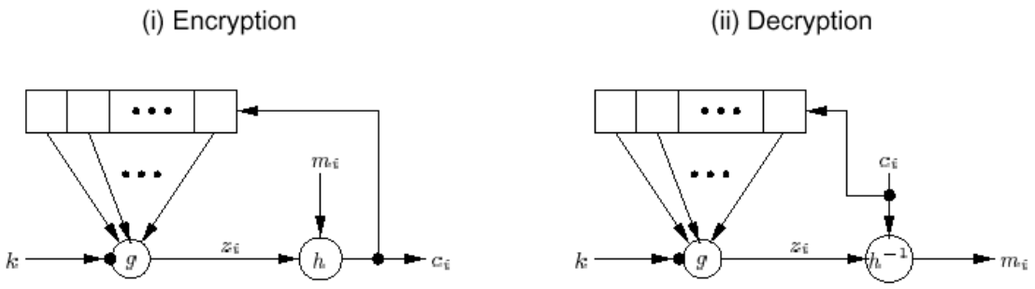



**Fig. 2.1.13 Modelo general de un cifrado por flujo asíncrono**

**Propiedades:**

1. Auto sincronismo es posible si dígitos de texto cifrado son borrados o insertados, porque la desencriptación depende sólo del mapeado de un número fijo de caracteres precedentes de texto cifrado. Tales cifrados son capaces de

restablecer la correcta desencriptación automática después de la pérdida de la sincronización, con solo un número fijo de caracteres de texto plano inrrecobrables.

2. Error de propagación limitado, suponer que el estado de un cifrado por flujo asíncrono depende de t dígitos anteriores de texto cifrado. Si un único digito de texto cifrado es modificado durante la transmisión, entonces la desencriptación subsiguiente al límite de t de dígitos de texto cifrado puede ser incorrecta.
3. Ataques activos, la propiedad 2 implica que alguna modificación de dígitos de texto cifrado por un atacante activo cause que varios otros dígitos de texto cifrado puedan ser desencriptados incorrectamente, en consecuencia mejora la probabilidad de ser detectado por un descriptor. Como consecuencia de la propiedad 1, este algoritmo es más difícil de detectar inserciones, borrados o reemplazos de dígitos de texto cifrado por un atacante activo.
4. Estadística de difusión de texto plano, ya que cada digito de texto plano influye completamente al texto cifrado, las propiedades estadísticas del texto plano son propagadas a través del texto cifrado. Por lo tanto, un cifrado por flujo asíncrono puede ser más resistente que un cifrado por flujo síncrono contra ataques basados en redundancia de texto plano.

#### 1.3.2.2.1.4 Conmutación de registros realimentados lineales (LFSRs)

Los LFSRs son usados en muchos de los generadores de flujo de claves que han sido propuestos en la literatura, entre las principales razones se tiene.

1. LFSRs son adecuados para una implementación en Hardware.
2. Pueden producir secuencias de grandes periodos.
3. Pueden producir secuencias con buenas propiedades estadísticas.

4. A causa de su estructura, pueden ser fácilmente analizados usando técnicas algebraicas.

Un LFSR de largo L consiste de L estados (o elementos de retardo) numerados desde $0,1,\ldots,L-1$, cada uno capaz de almacenar un bit y considerando una entrada y una salida, y un reloj que controla el movimiento de los datos. Durante cada unidad de tiempo las siguientes operaciones son ejecutadas.

1. El contenido de salida de la etapa 0 forma parte de la salida de la secuencia.
2. El contenido de la etapa i es desplazado a la etapa i-1 para cada i, con $1 \le i \le L-1$.
3. El nuevo contenido de la etapa L-1 es el bit de realimentación $S_j$, el cual es calculado por una suma conjunta módulo 2 de los contenidos anteriores de un subconjunto fijo de estados $0,1,\ldots,L-1$.

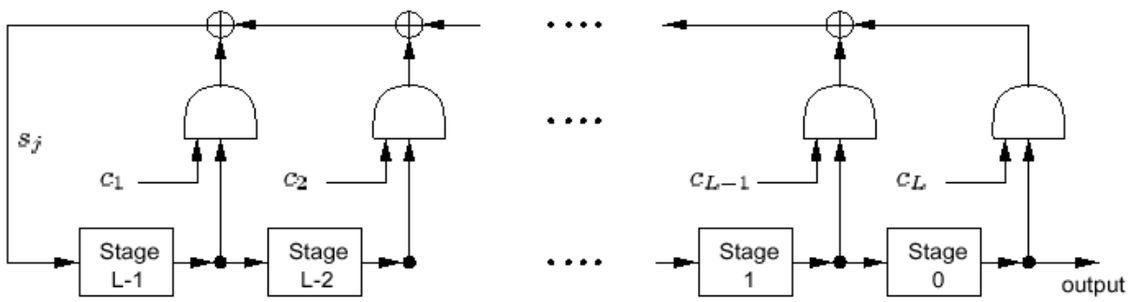



**Fig. 2.1.14 LFSR de largo L, donde cada $c_i$ es un 0 o 1, y el semicírculo es una puerta AND, el $s_j$ de realimentación es una suma módulo 2 de los contenidos de las etapas con $c_{L-i}=1$.**

### 1.3.2.2.1.5 Conmutación de registros realimentados no lineales

Un FSR de largo L consiste de L estados (o elementos de retardo) numerados desde $0,1,\ldots,L-1$ donde cada uno es capaz de almacenar un bit y considerando una

entrada y una salida, y un reloj que controla el movimiento de los datos. Durante cada unidad de tiempo las siguientes operaciones son ejecutadas.

1. El contenido de salida de la etapa 0 forma parte de la salida de la secuencia.
2. El contenido de la etapa i es desplazado a la etapa i-1 para cada i, con $1 \leq i \leq L-1$.
3. El nuevo contenido de la etapa L-1 es el bit de realimentación $s_j = f(s_{j-1}, s_{j-2}, \ldots, s_{j-L})$, donde la función de realimentación f es una función booleana y $s_{j-i}$ es el contenido previo de la etapa L-i con $1 \leq i \leq L$.

Si el contenido de la etapa inicial i es $s_i \in \{0,1\}$, para cada $1 \leq i \leq L-1$, entonces $[s_{L-1}, \ldots, s_1, s_0]$ es llamado el estado inicial de FSR. Si la función de realimentación f es una función lineal entonces el FSR es un LFSR, en otro caso es un FSR no lineal.

### 1.3.2.2.2 Algoritmos para cifrado por flujo

A continuación se resumen los dos más importantes algoritmos simétricos para cifrado por flujo, este cifrado generalmente se implementa en base a componentes eléctricos.

| Algoritmo | Tipo | Detalle |
|---|---|---|
| **Vernam** | One-time pad | Esta definido sobre un alfabeto $A = \{0,1\}$, un mensaje binario $m_1 m_2 \ldots m_t$ es operado sobre cadenas de claves binarias $k_1 k_2 \ldots k_t$ del mismo largo para producir cadenas de texto cifrado un cifrado $c_1 c_2 \ldots c_t$ donde la función de encriptación queda definida como $c_i = m_i \oplus k_i$ para $1 \leq i \leq t$ y la función de desencriptación se define como $m_i = c_i \oplus k_i$ |
| **SEAL (Optimized Encryption for Software)** | Aditivo binario | Fue propuesto en 1993, este algoritmo todavía no ha sido admitido por la mayoría de la comunidad criptográfica, pero sin embargo fue diseñado especialmente para una implementación eficiente de software, en particular para microprocesadores de 32 bit |

**Tabla 2.1.6 Clasificación de algoritmos para cifrado por flujo**

### 1.3.2.3 Cifrado por funciones hash (hash functions)

Una función Hash es una eficiente función computable que mapea cadenas binarias de largos arbitrarios a cadenas binarias, con algunas de largo fijo llamadas valores Hash. Para una función Hash con salidas de valores Hash de n bits, la probabilidad de que aleatoriamente escojan una cadena mapeada para un valor Hash particular de n bits es $2^{-n}$. La idea básica de un valor Hash es que sirva como una representación compacta de una cadena de entrada. Para usos criptográficos una función Hash h es elegida típicamente, tal que esta es no sea factible computacionalmente para encontrar dos entradas distintas con un valor hash común, se habla de colisión si para dos entradas x y z se tiene $h(x) = h(z)$. Los usos más comunes de funciones de este tipo en criptografía son en firma digital y para integración de datos. Una función Hash es una función h, la cual tiene como mínimo 2 propiedades.

1. Compresión, h mapea una entrada x arbitraria finita de largo en bit para una salida $h(x)$ fija de largo en bit n.
2. Facilidad de cálculo, un h determinado y una entrada x donde $h(x)$ es de fácil cálculo.

Como se definió una función Hash implica una función Hash sin clave, sin embargo para usos actuales una clasificación con objetivos más orientados es necesaria fuera de la existente (sin clave v/s con clave), se analizaran las dos más importantes.

1. Código de Detección de modificación (MDC)

   También se le conoce como código de detección de manipulación, y menos comúnmente como código de integridad de mensajes (MIC), el objetivo final es resolver el problema de la integridad de la información, al mensaje se le aplica un MDC y se manda junto con el propio mensaje, al recibirlo el receptor

aplica la función Hash al mensaje y comprueba que sea igual al Hash que se envió antes, la cadena que se envía es del tipo (M, h(M)). Las MDC son una subclase de funciones Hash sin clave, dentro de esta categoría tenemos principalmente.

- Funciones Hash de un solo camino (OWHF), encontrar una entrada la cual este dividida para una predescripción de valores Hash es bastante difícil.
- Funciones Hash resistentes a coalición (CRHF), encontrar cualquiera de 2 entradas considerando el mismo valor Hash es difícil.

2. Código de autenticación de mensajes (MAC)

   Las MAC sirven para autentificar el origen de los mensajes así como también su integridad, la función Hash que se aplica es h(M, K), donde M es el mensaje y K una clave privada. Las MAC tienen dos parámetros funcionalmente distintos, un mensaje de entrada y una clave secreta, estas son una subclase de funciones Hash con clave.

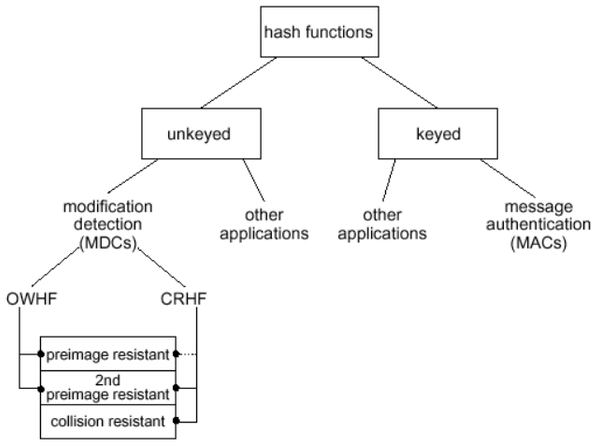



**Fig. 2.1.15 Clasificación simplificada de criptografía de funciones hash y aplicaciones**

Las primitivas Hash deben cumplir las siguientes propiedades.

1. Resistencia a la preimagen, significa que dada cualquier imagen, es computacionalmente imposible encontrar un mensaje x tal que $h(x) = y$, otra forma como se conoce esta propiedad es que h sea de un solo sentido.
2. Resistencia a la 2ª preimagen, significa que dado x, es computacionalmente imposible encontrar una $x'$ tal que $h(x) = h(x')$, otra forma de conocer de conocer esta propiedad es que h sea resistente a una coalisión suave.
3. Resistencia a coalisión, significa que es computacionalmente imposible encontrar dos diferentes mensajes $x$ , $x'$ tal que $h(x) = h(x')$, esta propiedad también se conoce como resistencia a una coalisión fuerte.

Las funciones de un solo camino (OWHF) cumplen la primera y segunda propiedades descritas, en cambio las funciones resistentes a coalisión (CRHF) cumplen las tres propiedades.

#### 1.3.2.3.1 Modelo general para funciones hash iterativas

Más funciones Hash sin clave h son diseñadas como procesos iterativos cuyas entradas son Hash de largos arbitrarios para procesamientos sucesivos de bloques de tamaño fijo en la entrada [Van+, 1996].

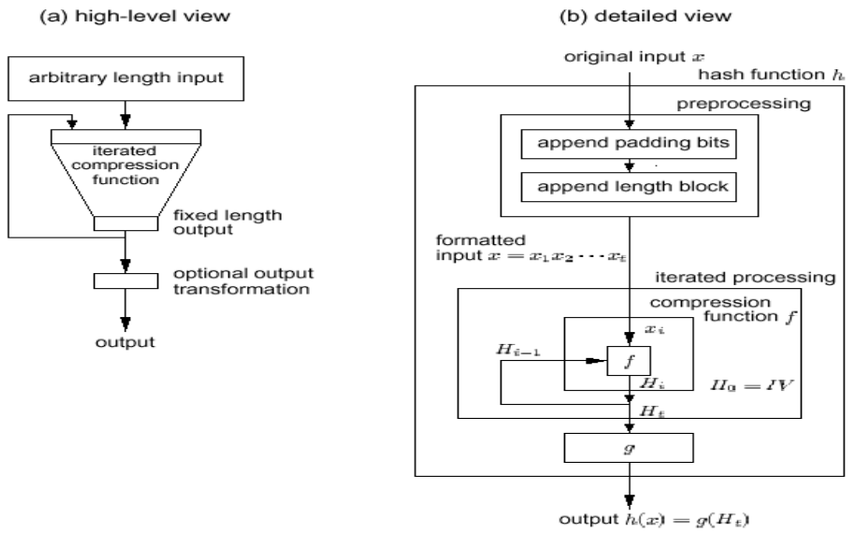



**Fig. 2.1.16 Modelo general para funciones hash iterativas**

Una entrada Hash x arbitraria de largo finito es dividida en bloques $x_i$ de r bit de largo fijo, este preprocesamiento típico involucra necesariamente agregar bit extras para lograr el largo en bit completo, lo cual es un múltiplo de un bloque de largo r y a menudo incluye un bloque (o bloque parcial) indicando el largo en bit de la entrada no procesada. Cada bloque $x_i$ entonces sirvió de entrada para una función Hash interna f de tamaño fijo denominada la función de compresión de h, la cual calcula un resultado intermedio de largo en bit n, para algunas n es fijo, este resultado es la entrada para el bloque $x_i$, $H_i$ denota el resultado parcial después de la etapa i, el proceso general para una función Hash iterativa con una entrada $x = x_1 x_2 \ldots x_t$ puede modelarse como:

$$H_0 = IV$$
$$H_i = f(H_{i-1}, x_i), 1 \le i \le t$$
$$h(x) = g(H_t)$$

Donde $H_{i-1}$ sirve como variable de cambio de n bit entre la etapa $i-1$ y la etapa i y $H_0$ es un valor predefinido al comienzo o valor de iniciación (IV). Una transformación de salida opcional g es usada en el paso final para mapear la variable cambiante de n bit.

### 1.3.2.3.2 Funciones hash sin claves (MDC)

#### 1.3.2.3.2.1 Funciones hash basadas sobre cifrado por bloques

Un cifrado por bloques $(n,r)$ se define como una función invertible desde textos planos de n bit para textos cifrados de n bit usando una clave de r bit. La construcción de funciones Hash para cifrados por bloques se divide en valores Hash de largo simple (n bit) y de largo doble (2n bit), donde este valor se relaciona con el tamaño del bloque cifrado de salida. Se analizarán las funciones más importantes de esta categoría, a continuación se muestra una clasificación [Van+, 1996].

| Algoritmo | Aplicación | Detalle |
|---|---|---|
| **Matyas Meyer Oseas** | Sobre cifrado por bloques | Es un cifrado para bloques de n bit con clave de k bit y un mensaje de entrada de n bit |
| **Davies Meyer** | Sobre cifrado por bloques | Es un cifrado para bloques de n bit con clave de k bit y un mensaje de entrada de n bit |
| **Miyaguchi Preneel** | Sobre cifrado por bloques | Es un cifrado para bloques de n bit con clave de k bit y un mensaje de entrada de n bit |
| **MDC-2** | Sobre cifrado por bloques | Es un cifrado para bloques de 64 bit con clave de 56 bit y un mensaje de entrada de 128 bit, en general la construcción suele hacerse en base al cifrado por bloques DES |
| **MDC-4** | Sobre cifrado por bloques | Es un cifrado para bloques de 64 bit con clave de 56 bit y un mensaje de entrada de 128 bit, e s construido usando la función de compresión de MDC-2 |

**Tabla 2.1.7 Funciones hash basadas sobre cifrado por bloques**

#### 1.3.2.3.2.2 Funciones hash optimizadas basadas sobre MD4 (Message Digest)

Mejorar la seguridad de MD4 dio como resultado la mejora MD5 en poco tiempo después, otras variantes que siguieron fueron SHA-1 (Secure Hash Algorithm), la función Hash RIPEMD y sus variantes fortalecidas RIPEMD-128 y RIPEMD-160.

| Algoritmo | Aplicación | Detalle |
|---|---|---|
| **MD4** | Sobre MD4 | Fue diseñado específicamente para implementación en softwares para computadores de 32 bit |
| **MD5** | Sobre MD4 | Fue diseñado como un fortalecimiento de MD4, producto de las coalisiones encontradas en MD4 |
| **Sha-1** | Sobre MD4 | Esta basado sobre MD4, y fue propuesto por U.S. NIST (National Institute for Standards and Technology) para la seguridad de aplicaciones del gobierno federal de Estados Unidos |
| **Ripemd-160** | Sobre MD4 | La función de compresión global de RIPEMD-160 mapea entradas de 21 palabras (5 palabras de variables de cambio más 16 palabras de bloques de mensajes, con palabras de 32 bit) para salidas de 5 palabras |

**Tabla 2.1.8 Funciones hash basadas sobre MD4**

### 1.3.2.3.2.3 Funciones hash basadas sobre aritmética modular

La idea basica es construir una función Hash iterativa usando mod M aritmético, como la función de compresión básica.

| Algoritmo | Aplicación | Detalle |
|---|---|---|
| **MASH-1 (Modular Arithmetic Secure Hash)** | Sobre aritmética modular | Este habia sido propuesto por la incorporación de un borrador de un estándar ISO/IEC. MASH-1 involucra el uso de un parecido módulo M RSA, cuyo largo en bit afecta la seguridad, M podria ser dificil de factorizar y para un M de factorización desconocida la seguridad esta basada en parte sobre la dificultad de estraer las raices de los módulos |

**Tabla 2.1.9 Funciones hash sobre aritmética modular**

### 1.3.2.3.3 Funciones hash con claves (MAC)

El propósito específico de estas funciones es la autenticación del mensaje, muchas iteraciones MAC pueden ser descritas como iteraciones de funciones Hash. En estos casos, la clave MAC generalmente parte de la transformación de salida g, ésta también puede ser la entrada para la función de compresión en la primera iteración y puede estar involucrada en todas las etapas de la función de compresión f.

#### 1.3.2.3.3.1 MAC basadas sobre cifrados por bloque

En esta categoría sólo se estudio el algoritmo más importante ya que los demás no tienen mucha aplicabilidad o no han sido estándarizados.

| Algoritmo | Aplicación | Detalle |
|---|---|---|
| **CBC-MAC** | Sobre cifrado por bloques | Este algoritmo hace uso de los encadenamientos de bloques de cifrado, cuando DES es usado como un cifrado por bloque E con n=64 y la clave MAC es una clave DES de 56 bit |

**Tabla 2.1.10 Funciones hash sobre cifrado por bloques**

#### 1.3.2.3.3.2 Construcciones MAC desde MDC

Una sugerencia común es la construcción de un algoritmo MAC desde un algoritmo MDC, simplemente incluyendo una clave secreta k como parte de la entrada MDC.

#### 1.3.2.3.3.3 MAC personalizadas

Se analizaran dos algoritmos diseñados especificamente con el propósito de autenticar mensajes, estos son MAA y MD5-MAC.

| Algoritmo | Aplicación | Detalle |
|---|---|---|
| **MAA (Message** | Sobre MAC | Este algoritmo data de 1983, su loop principal consiste |

| | | |
|---|---|---|
| **Authenticator Algorithm)** | | de dos flujos de calculos paralelos interdependientes, los mensajes son procesados en bloques de 4 bytes usando variables de cambio de 8 bytes |
| **MD5-MAC** | Sobre MAC | Representa una aproximación más conservadora para construir una MAC desde un MDC es para definir que la función de compresión MAC dependa de si misma sobre k |

**Tabla 2.1.11 Funciones hash sobre MAC**

### 1.3.2.3.3.4 MAC para cifrados por flujo

Suministrar autenticación de datos de origen y garantizar la integridad de los datos para cifrados por flujo, es particularmente importante porque la manipulación de bit en la adición de cifrados por flujo puede resultar directamente en modificaciones impredecibles en el texto plano. Mientras funciones hash iterativas procesan bloques de datos de un mensaje en un determinado tiempo, MAC se puede diseñar para procesar mensajes usando cifrados por flujo ambos de 1 bit o 1 símbolo (bloque) en un tiempo determinado, estos pueden ser implementados usando LFSR debido a razones de eficiencia.

| Algoritmo | Aplicación | Detalle |
|---|---|---|
| **CRC** | Sobre MAC | Una familia Hash H (b, m) es una colección de funciones Hash mapeando mensajes de b bit para valores Hash de m bit, una familia de funciones Hash es $\xi$ balanceada si para todos los mensajes $B \neq 0$ y todos los valores Hash c de m bit se tiene $prob_h(h(B) = c) \leq \xi$ donde la probabilidad son en general funciones $h \in H$ seleccionadas aleatoriamente |

**Tabla 2.1.12 Funciones hash sobre MAC**

### 1.3.3 Análisis de algoritmos asimétricos

La criptografía asimétrica es por definición aquella que utiliza dos claves diferentes para cada usuario, una para cifrar que se le llama clave pública y otra para descifrar que es la clave privada. El nacimiento de la criptografía asimétrica se dio al estar buscando un modo más práctico de intercambiar las llaves simétricas. Diffie y Hellman, proponen una forma para hacer esto, sin embargo no fue hasta que el popular método de Rivest Shamir y Adleman RSA que fue publicado en 1978 [RSA+, 1998], cuando toma forma la criptografía asimétrica, su funcionamiento esta basado en la imposibilidad computacional de factorizar números enteros grandes. Actualmente la criptografía asimétrica es muy usada; sus dos principales aplicaciones son el intercambio de claves privadas y la firma digital. En la actualidad la criptografía asimétrica o de clave pública se divide en tres familias según el problema matemático en el cual basan su seguridad, por otra parte la encriptación de clave pública es sustancialmente más lenta que la encriptación de clave simétrica La primera familia es la que basa su seguridad en el Problema de Factorización Entera PFE. La segunda familia es la que basa su seguridad en el Problema del Logaritmo Discreto PLD. Y la tercera familia es la que basa su seguridad en el Problema del Logaritmo Discreto Elíptico PLDE.

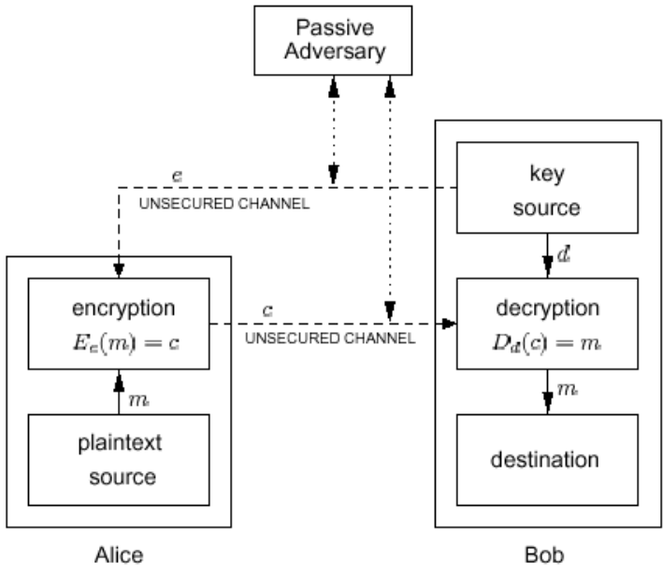



**Fig. 2.1.17 Proceso de encriptación asimétrico**

### 1.3.3.1 Distribución de claves públicas

Los esquemas de encriptación de clave pública que se analizarán en esta tesis asumen que existe un medio por el cual enviar un mensaje a un receptor deseado obteniendo una copia autorizada de la clave pública de éste. En la ausencia de este medio el esquema de encriptación es susceptible a un ataque de personificación. En la práctica existen muchas técnicas por las cuales claves públicas autorizadas puedan ser distribuidas, incluyendo intercambio de claves sobre un canal confiable, usando un archivo público confiable, usando un servidor on-line confiable, usando un servidor off-line y certificados [Odl+, 1993].

### 1.3.3.2 Algoritmos de clave pública

Para efectos de estudio se analizaron los algoritmos más eficientes y usados en los sistemas informáticos modernos, los cuales se clasifican dependiendo de la aleatoriedad de sus procesos.

.

| Algoritmo | Tipo | Detalle |
|---|---|---|
| **RSA** | Determinístico | **Este criptosistema debe su nombre a sus creadores R. Rivest, A. Shamir y L. Adleman y corresponde al criptosistema más ampliamente usado, puede ser usado para proveer confidencialidad y firma digital** |
| **RABIN** | Determinístico | **Este constituye el primer ejemplo de un esquema de encriptación de clave pública probablemente seguro, el problema confronta a un adversario pasivo que recupera texto plano de un determinado texto cifrado y computacionalmente es equivalente a factorizar** |
| **ElGamal** | Determinístico | **Este esquema puede ser visto como una clave Diffie-Hellman de acuerdo con el modo de transferir la clave, su seguridad esta basada sobre la inflexibilidad del problema del logaritmo discreto y del problema de Diffie-Hellman** |
| **McEliece** | Determinístico | **Este esquema esta basado sobre codigos de corrección de errores, la idea inicial de este esquema es que para la primera selección de un código particular para lo cual un eficiente algoritmo de decodificación es conocido para esconder el código como un código lineal general** |
| **Knapsack Merkle-Hellman** | Determinístico | **Este esquema intenta ocultar fácilmente la solución del problema de una suma de subconjunto, llamado problema de una suma de subconjunto super creciente por medio de múltiplicación modular y una permutación** |
| **Knapsack Chor-Rivest** | Determinístico | **Este esquema no usa una forma de multiplicación modular simple para ocultar el problema de suma de subconjunto** |
| **Goldwasser-Micali** | No Determinístico | **Este esquema esta .basado sobre la inflexibilidad del problema del residuo cuadrático** |
| Blue-Goldwasser | **No Determinístico** | **Este esquema es el más eficiente dentro de los esquemas probabilisticos y es comparable con el esquema de encriptación RSA y esta .basado sobre la inflexibilidad del problema de factorización entera** |

**Tabla 2.1.13 Algoritmos de encriptación de clave pública**

### 1.3.4 Análisis de otras herramientas criptográficas

Sin duda de que existen una serie de métodos y herramientas criptograficas adicionales a las ya estudiadas, sin embargo las dos más grandes metodologías son: Redes Privadas Virtuales y el Estándar IEEE 802.1X [Ohr+, 2003] para redes de datos cableadas e inalámbricas, cabe mencionar que solo las Redes Privadas Virtuales es factible de implementar en un sistema de transmisión digital por radio frecuencia, su limitación esta dada en general por los requerimientos de equipos, infraestructura y principalmente una serie de softwares adicionales. En cuanto al estándar IEEE 802.1X no se detallara ya que es básicamente utilizado para la autenticación de clientes en redes de datos, para ello provee una serie de protocolos para el manejo de contraseñas, en cuanto al cifrado este opera mayormente con WEP y WPA en redes de datos inalámbricas [Ohr+, 2003].

# 2. Metodología de desarrollo de un sistema de cifrado de información

## 2.1 Objetivos

Analizados los algoritmos criptográficos, se determinará el algoritmo para el cifrado de la información a nivel de software, ya que el MODEM solo se encargará de transformar la data para la adaptación al medio electromagnético. Para llegar a un diseño formal del sistema de cifrado se hará un modelamiento de software utilizando UML [Dob+, 2006].

## 2.2 Análisis de requerimientos

Los requerimientos fueron recopilados en una unidad táctica del Ejército de Chile, específicamente en el Regimiento de Telecomunicaciones Nº 4 “Membrillar” ubicado en Bueras s/n en la ciudad de Valdivia. Se consideró el aspecto de no afectar la forma de trabajo que actualmente esta vigente. A continuación se individualizarán los requerimientos deducidos después de un análisis con el personal a cargo del área de telecomunicaciones.

### 2.2.1 Requerimiento 1

| | |
|---|---|
| Requerimiento | Algoritmo de cifrado simétrico |
| Detalle | Se debe contar con la misma clave privada en ambos lados de la comunicación para cifrar y descifrar los datos |
| Tipo | Primario |

### 2.2.2 Requerimiento 2

| | |
|---|---|
| Requerimiento | Algoritmo generador de claves |
| Detalle | Se deben generar las claves a partir de una semilla de tamaño inferior a la clave de cifrado |
| Tipo | Primario |

### 2.2.3 Requerimiento 3

| | |
|---|---|
| Requerimiento | Periodo de vigencia de claves |
| Detalle | El tiempo que permanecerá activa una clave semilla quedara a criterio de la institución |
| Tipo | Secundario |

### 2.2.4 Requerimiento 4

| | |
|---|---|
| Requerimiento | Tamaño de las claves de cifrado |
| Detalle | Las claves de cifrado deben ser de un largo de 128 bits |
| Tipo | Primario |

## 2.3 Diseño

Se definió cada una de las soluciones adoptadas para cada requisito recopilado y analizado. Se utilizará un cifrado por flujo producto de que es el más adecuado para el tipo de información y la forma en que se enviarán los datos, producto que el cifrado por bloques y funciones hash usan tamaños de bloques fijos y en nuestro caso se pueden generar mensajes que no alcanzen el tamaño requerido. La encriptación se hará sobre una codificación ASCII con un alfabeto de 256 carácteres, en cuanto al algoritmo específico se utilizará una variante del algoritmo DES para cifrado por flujo [Van+, 1996]. La longitud de las claves de cifrado será de 128 bit, estás se obtendrán a partir de una clave base definida según el periodo establecido (semilla), para su generación se utilizara un algoritmo seudoaleatorio y se establecerá la política de que cada unidad desplegada tendrá una clave asignada y se complementará con la unidad con que se establezca comunicación, siendo esta una condición suficiente de seguridad, por

ejemplo si se tiene 3 unidades desplegadas se tendrá un conjunto de claves con una cardinalidad de 6, es decir toda la red de radios estará utilizando 6 claves distintas de cifrado. El período de vigencia de las claves se dejará al criterio de la institución pudiéndose establecer claves diarias, semanales, mensuales y anuales.

### 2.3.1 Diagrama de casos de uso

En esta parte solo se analizarán los casos de uso involucrados directamente en los procesos de encriptación y desencriptación de datos. Los casos de uso específico son.

- Administrar Claves
- Encriptar Datos
- Desencriptar Datos
- Generar Conjunto de Claves

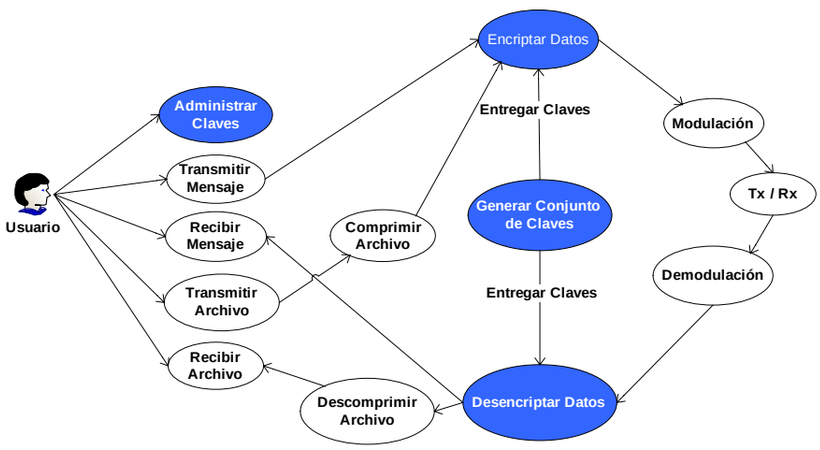



**Fig. 2.2.1 Diagrama de casos de uso del sistema de cifrado**

### 2.3.2 Diagrama de componentes

Se mostrarán las dependencias entre las componentes de software involucrados en el sistema de cifrado, el uso de librerías externas o paquetes va a depender del lenguaje de desarrollo que se utilize.

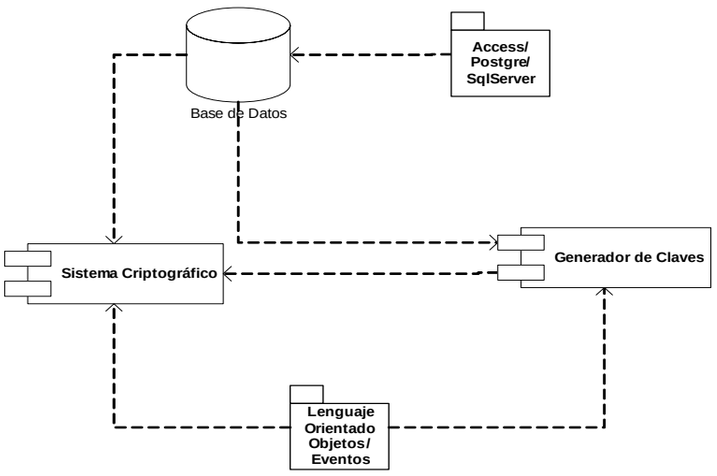



**Fig. 2.2.2 Diagrama de componentes del sistema de cifrado**

### 2.3.3 Escenarios

Se detallarán los escenarios de cada uno de los casos de uso descritos

#### 2.3.3.1 Caso de uso: administrar claves

| | |
|---|---|
| Actores | Usuario del sistema – Sistema |
| Propósito | Ingresar y eliminar claves para la generación del conjunto de claves de cifrado |
| Precondición | -Sistema en línea<br>-Que un comandante de unidad necesite ingresar o eliminar claves |
| Tipo | Primario |

Escenario 1:

El usuario del sistema podrá agregar claves de cifrado estableciendo el período de vigencia de dichas claves (diaria, semanal, mensual y anual), para ello el sistema validará que en ese período no existan claves y que el usuario tenga los privilegios para realizar el proceso.

| Acción de Actores | Respuesta del Sistema |
|---|---|
| 1.- Solicitud de ingreso de claves<br><br>3.- Ingreso del período y la clave<br><br><br><br><br>6.- Ingresa contraseña | <br>2.- Verifica si esta en línea<br><br>4.- Verifica que no existan claves en el período<br>5.- Si no hay claves en el período, solicita la contraseña del comandante de la unidad<br><br>7.- Verifica contraseña<br>8.- Si la contraseña es válida, almacena |

Escenario 2:

El usuario del sistema podrá eliminar claves de cifrado, para ello el sistema validará que tenga los privilegios para realizar el proceso.

| Acción de Actores | Respuesta del Sistema |
|---|---|
| 1.- Solicitud de eliminación de claves<br><br><br><br>4.- Ingresa contraseña | <br>2.- Verifica si esta en línea<br>3.- Solicita la contraseña del comandante de la unidad<br><br>5.- Verifica contraseña<br>6.- Si la contraseña es válida, elimina |

.

### 2.3.3.2 Caso de uso: generar conjunto de claves

| | |
|---|---|
| Actores | Sistema – Sistema |
| Propósito | Generación del conjunto de claves de cifrado |
| Precondición | -Sistema en línea<br>-Existencia de un proceso de encriptación ó desencriptación |
| Tipo | Primario |

Escenario 1:

Existe un proceso de encriptación o desencriptación, para lo cual cada proceso necesita el conjunto de claves de cifrado de 128 bit, para ello el sistema generará el conjunto requerido utilizando como semilla la clave fijada para el período.

| Acción de Actores | Respuesta del Sistema |
|---|---|
| 1.- Solicita conjunto de claves de cifrado de 128 bit | 2.- Verifica la existencia de una clave para el período<br>3.- Si hay claves en el período, genera el conjunto de claves de 128 bit |

### 2.3.3.3 Caso de uso: encriptar datos

| | |
|---|---|
| Actores | Sistema – Sistema |
| Propósito | Encriptar datos, ya sea mensajes cortos o archivos |
| Precondición | -Sistema en línea<br>-Existe un archivo comprimido o un mensaje para encriptación |
| Tipo | Primario |

Escenario 1:

Existe un proceso de envio de datos desde una unidad a otra, para lo cual el sistema necesita de un proceso de encriptación.

| Acción de Actores | Respuesta del Sistema |
|---|---|
| 1.- Solicita encriptación de datos a enviar | 2.- Encripta<br>3.- Envia datos al proceso de modulación |

### 2.3.3.4 Caso de uso: desencriptar datos

| | |
|---|---|
| Actores | Sistema – Sistema |
| Propósito | Desencriptar datos, ya sea mensajes o archivos |
| Precondición | -Sistema en línea<br>-Existe un archivo comprimido o un mensaje para desencriptación proveniente del proceso de demodulación |
| Tipo | Primario |

Escenario 1:

Existe un proceso de recepción de datos desde una unidad a otra, para lo cual el sistema necesita de un proceso de desencriptación.

| Acción de Actores | Respuesta del Sistema |
|---|---|
| 1.- Solicita desencriptación de datos recibidos | 2.- Desencripta<br>3.- En caso de mensajes envía datos a consola de mensajes<br>4.- En caso de archivos envía a proceso de descompresión |

## 2.4 Implementación

Producto de que la implementación del sistema de cifrado se hará en base a un software no orientado a objetos, esto implica no definir las clases que pueda tener el sistema y se considerá suficiente definir los algoritmos en seudocódigo indicando solo sus principales operaciones, se detallarán los algoritmos de cifrado y el generador de claves indicando las entradas y salidas de cada uno de ellos. En cuanto al algoritmo es una variante del algoritmo simétrico DES para cifrado por flujo y se representarán algunas de las operaciones que se realizan.

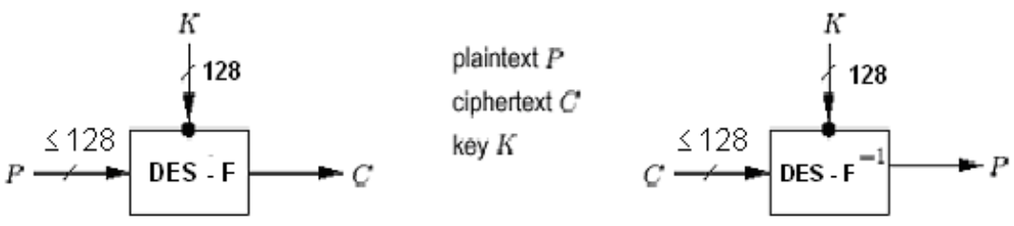



**Fig. 2.2.3 Esquema del cifrado simétrico utilizado**

*Algoritmo Generador de Claves para Military_V1*

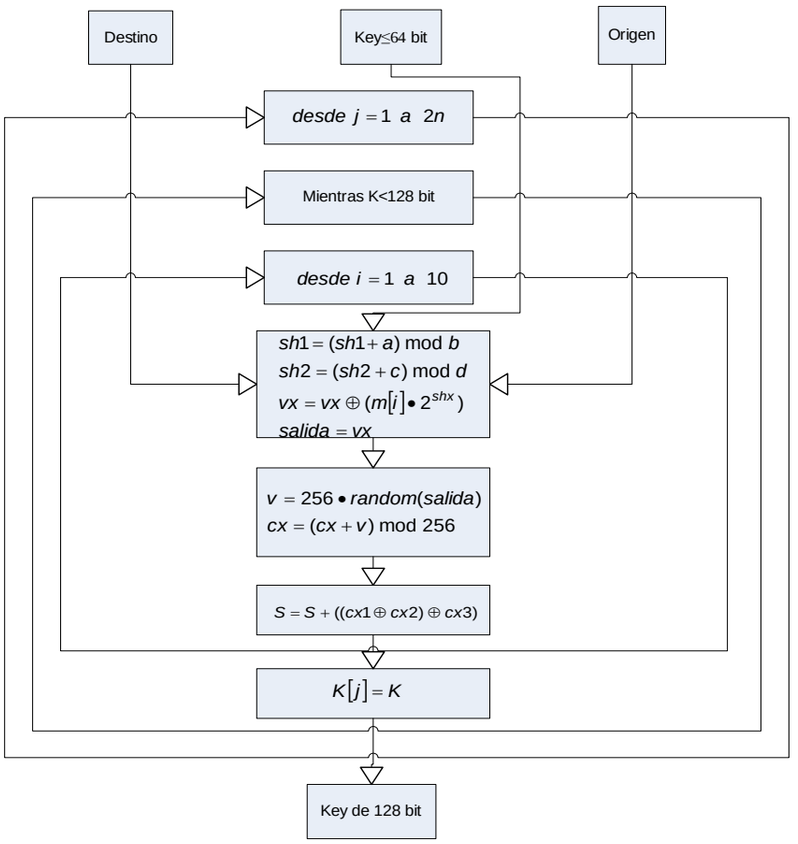
Entrada:

- Clave $K$ predefinida del período con $K \leq 64$ bit.
- $A$ Nombre unidad origen.
- $B$ Nombre unidad destino.

Salida:

- Conjunto de claves de 128 bit $[S_1,...,S_{2n}]$, donde n=1.

Proceso:




*Algoritmo Military_V1*

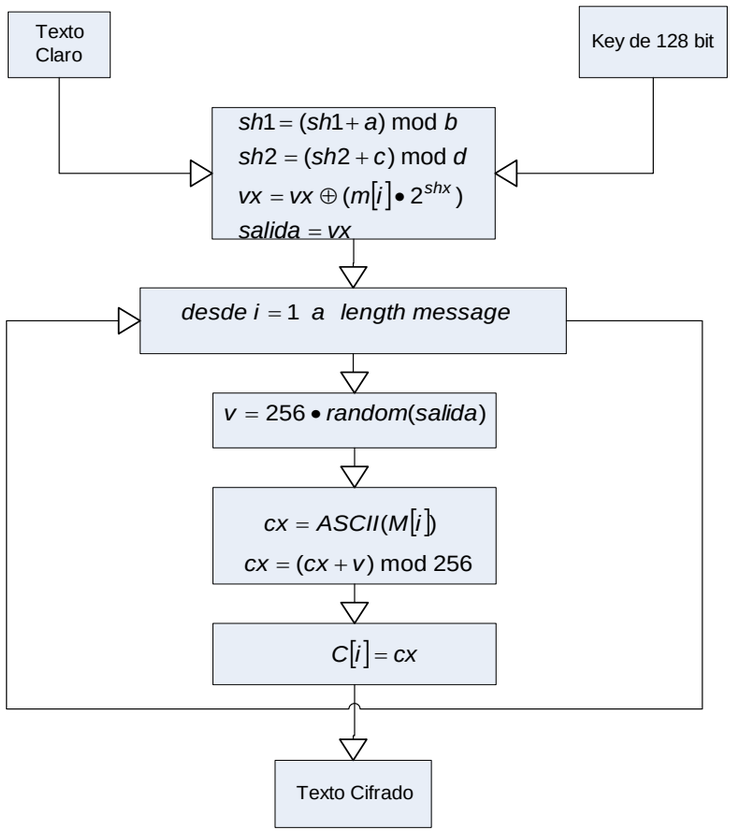
Entrada:

- Mensaje de texto plano M de largo t.
- Conjunto de claves de 128 bit $[S_1,...,S_{2n}]$.

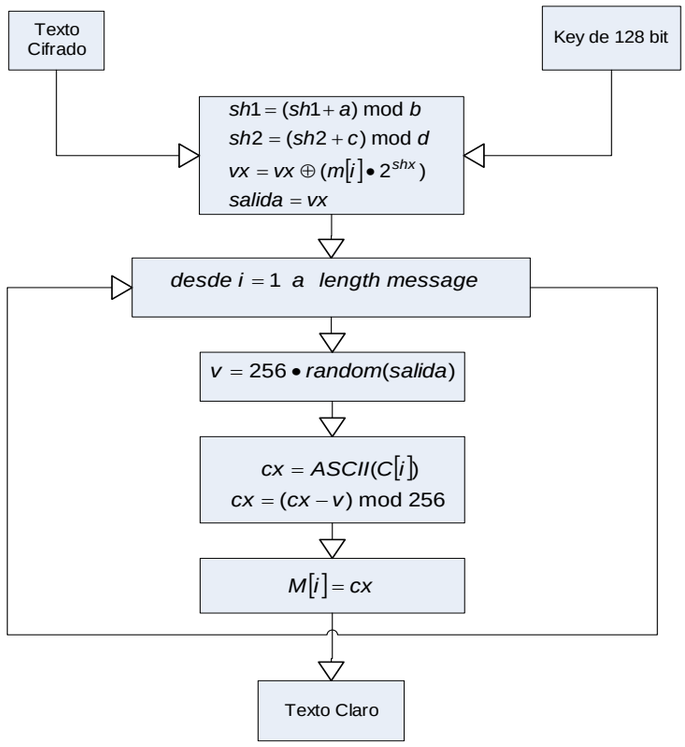
Salida:

- Texto cifrado $[C_1,C_2,...,C_t]$.o texto claro $[M_1,M_2,...,M_t]$.

Encriptación:




Desencriptación:




La función $sh1 = (sh1 + a) \bmod b$, $sh2 = (sh2 + c) \bmod d$, $vx = vx \oplus (m[i] \bullet 2^{shx})$, $salida = vx$ retorna un entero grande que sirve de semilla numérica para generar un número aleatorio, repitiendo los cálculos con dos valores de cambio distintos, donde $m[i]$ es un digito de la cadena de entrada. La adición modular tiene una complejidad $O(\lg n)$ (orden del tiempo de operación por bit) y esta definida por

$$(a+b) \bmod n = \begin{Bmatrix} a+b, si\ a+b < n \\ a+b-n, si\ a+b \geq n \end{Bmatrix}.$$

| Operation | | Bit complexity |
|---|---|---|
| Modular addition | $(a+b) \bmod n$ | $O(\lg n)$ |
| Modular subtraction | $(a-b) \bmod n$ | $O(\lg n)$ |
| Modular multiplication | $(a \cdot b) \bmod n$ | $O((\lg n)^2)$ |
| Modular inversion | $a^{-1} \bmod n$ | $O((\lg n)^2)$ |
| Modular exponentiation | $a^k \bmod n, k < n$ | $O((\lg n)^3)$ |

**Fig. 2.2.4 Complejidad por bit de operaciones en Zn**

## 2.5 Evaluación de costos

Producto que el sistema de cifrado necesariamente debe formar parte de un software que sea capaz de transferir y gestionar datos de índole militar y para tener una mejor aproximación, es que la evaluación de costos se hará tomando la aplicación en su forma global, dado esto se detallará dicho análisis en el capitulo 3 de esta tesis.

# Capitulo 3: Modelo de Desarrollo de un Software de Comunicación y Gestión de Mensajes Oficiales

# 1. Metodología de desarrollo de un software de comunicación y gestión de mensajes oficiales

## 1.1 Objetivos

Diseñar un software de comunicación militar que permita transferencia de mensajes cortos y archivos en cualquier formato, además de gestionar y confeccionar mensajes oficiales de la institución. Para llegar a un diseño formal del software se hará un modelamiento de software utilizando UML [Dob+, 2006].

## 1.2 Análisis de requerimientos

Los requerimientos fueron recopilados en una unidad táctica del Ejército de Chile, específicamente en el Regimiento de Telecomunicaciones Nº 4 “Membrillar” ubicado en Bueras s/n en la ciudad de Valdivia. Los requisitos fueron recopilados considerando que no signifique un cambio drástico a la forma de trabajo que actualmente esta vigente. A continuación se individualizarán los requerimientos deducidos después de un análisis con el personal a cargo del área de telecomunicaciones.

### 1.2.1 Requerimiento 1

| | |
|---|---|
| Requerimiento | Transferencia de mensajes cortos |
| Detalle | Debe existir un módulo que permita comunicación de mensajes cortos, que tenga una similitud a una consola de Chat |
| Tipo | Primario |

### 1.2.2 Requerimiento 2

| | |
|---|---|
| Requerimiento | Transferencia de archivos |
| Detalle | Debe existir un módulo que permita comunicación de archivos sin importar su formato |
| Tipo | Primario |

### 1.2.3 Requerimiento 3

| | |
|---|---|
| Requerimiento | Compresión de datos |
| Detalle | Debe existir un proceso de compresión de datos, específicamente para la transmisión de archivos |
| Tipo | Secundario |

### 1.2.4 Requerimiento 4

| | |
|---|---|
| Requerimiento | Manejo de usuarios y privilegios |
| Detalle | Debe existir acceso restringido al software y manejo de privilegios en el tratamiento de la información |
| Tipo | Secundario |

### 1.2.5 Requerimiento 5

| | |
|---|---|
| Requerimiento | Manejo de unidades |
| Detalle | Debe existir un manejo de unidades, es decir se debe poder crear unidades y administrar el personal perteneciente a ellas |
| Tipo | Secundario |

### 1.2.6 Requerimiento 6

| | |
|---|---|
| Requerimiento | Sistema de auto destrucción |
| Detalle | Debe contar con un proceso de inhabilitación permanente del software en caso se vea comprometida la integridad física de una unidad |
| Tipo | Secundario |

### 1.2.7 Requerimiento 7

| | |
|---|---|
| Requerimiento | Almacenamiento de datos |
| Detalle | Debe contar con un mecanismo de almacenamiento de la información generada por la transmisión, recepción y gestión de mensajes oficiales |
| Tipo | Primario |

### 1.2.8 Requerimiento 8

| | |
|---|---|
| Requerimiento | Encriptación y desencriptación de datos |
| Detalle | Debe contar con un proceso de cifrado de datos para asegurar la integridad y confidencialidad de la información |
| Tipo | Primario |

### 1.2.9 Requerimiento 9

| | |
|---|---|
| Requerimiento | Gestión de mensajes oficiales |
| Detalle | Debe contar con un módulo de gestión de mensajes oficiales que permita creación, almacenamiento y visualización |
| Tipo | Primario |

### 1.2.10 Requerimiento 10

| | |
|---|---|
| Requerimiento | Impresión de informes |
| Detalle | Debe ser factible obtener una serie de informes relacionados con la institución, tanto de la comunicación por radio frecuencia como de la gestión de mensajes oficiales |
| Tipo | Secundario |

### 1.2.11 Requerimiento 11

| | |
|---|---|
| Requerimiento | Determinación de posición (GPS) |
| Detalle | Debe existir un proceso que permita obtener la localización de una unidad y comunicar esta posición a las demás unidades desplegadas |
| Tipo | Secundario |

## 1.3 Diseño

Se definió cada una de las soluciones adoptadas para cada requisito recopilado y analizado. El software de comunicación trabajará sobre una base de datos, la cual almacenará todo el tráfico desde y hacia cada nodo de la red, además gestionará el manejo de mensajes oficiales de la institución, el modelo de datos debe soportar ambos casos mencionados. Proveerá la capacidad de gestionar las claves, las unidades que estarán en terreno y los usuarios del sistema. Se dispondrá de un mecanismo de autodestrucción en el caso de que den de baja a una unidad, este mecanismo funciona presionando un botón inhabilitando permanentemente la aplicación, borrando la base de datos y los archivos de instalación. Tendrá un módulo de transmisión y recepción de datos, es decir habra una consola de mensajes y una consola de archivos, en dichas

consolas se podrán tener todos los informes solicitados y además se podrá eliminar tráfico dependiendo si el usuario tiene los privilegios suficientes. Se tendrá un proceso de cifrado tanto para mensajes como para archivos, esto significa encriptar datos en el caso de transmisiones y desencriptar en recepciones, el algoritmo de cifrado será simétrico y realizará un cifrado por flujo empleando para ello una clave de 128 bit. Existirá un proceso de compresión por software de datos para el caso de tráfico de archivos, esto significa comprimir en el caso de transmisiones y descomprimir en recepciones, la compresión de datos será utilizando el formato zip. Se dispondrá de un proceso de localización mediante un GPS conectado al computador, el software extraerá la posición desde el GPS y la transmitirá a las demás unidades de la red, el GPS estará conectado a un puerto serial (RS 232-D). Tendrá un módulo de confección y gestión de mensajes oficiales de la institución, esto involucra crear, almacenar y actualizar mensajes oficiales, además de imprimirlos y generar otros informes específicos.

### 1.3.1 Diagrama de casos de uso

En esta parte se analizarán los casos de uso deducidos de acuerdo al análisis de requerimientos, que formarán la parte principal del software. Sin embargo, no se detallarán los casos de uso involucrados en el proceso de cifrado de datos ya que fueron modelados en el capitulo 2 de esta tesis.

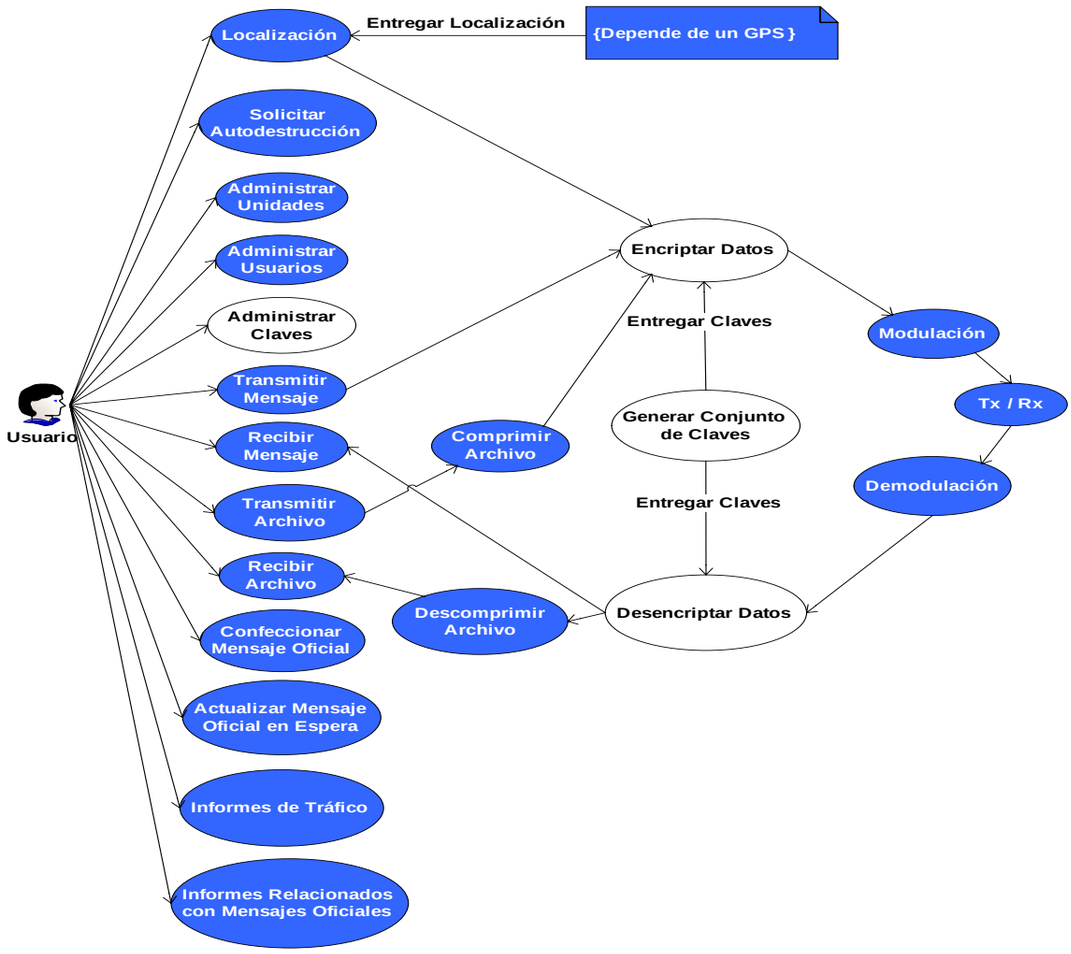



**Fig. 3.1.1 Diagrama de casos de uso del software de comunicaciones**

### 1.3.2 Diagrama de componentes

En las dependencias entre las componentes de softwares se considero el sistema de cifrado y algunos componentes de hardware que interactuarán con el software de comunicaciones.

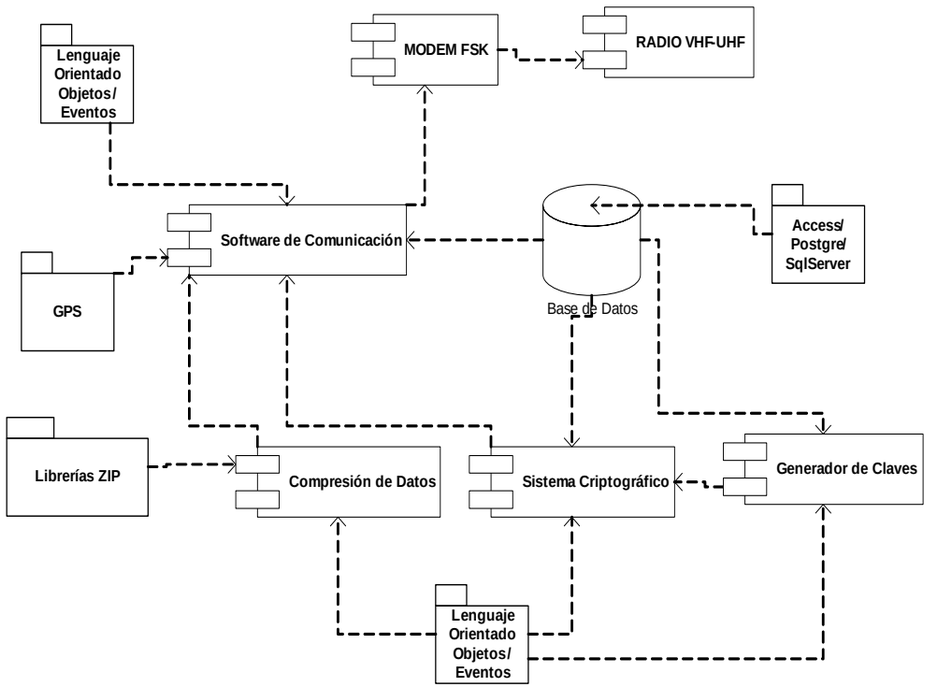



**Fig. 3.1.2 Diagrama de componentes del software de comunicaciones**

### 1.3.3 Escenarios

Se detallarán los escenarios de cada uno de los casos de uso descritos

#### 1.3.3.1 Caso de uso: administrar unidades

| | |
|---|---|
| Actores | Usuario del sistema – Sistema |
| Propósito | Ingresar y eliminar unidades que existiran el en sistema |
| Precondición | -Sistema en línea<br>-Que un comandante de unidad necesite ingresar o eliminar unidades |
| Tipo | Primario |

Escenario 1:

El usuario del sistema podrá agregar unidades, este usuario debe ser necesariamente un comandante de alguna unidad.

| Acción de Actores | Respuesta del Sistema |
|---|---|
| 1.- Solicitud de ingreso de unidades<br><br>3.- Ingreso de datos requeridos | <br>2.- Verifica si esta en línea<br><br>4.- Verifica que no existan errores en los datos<br>5.- Si no hay errores almacena |

Escenario 2:

El usuario del sistema podrá eliminar unidades, para ello el sistema validará que tenga los privilegios para realizar el proceso.

| Acción de Actores | Respuesta del Sistema |
|---|---|
| 1.- Solicitud de eliminación de unidad<br><br><br><br>4.- Ingresa contraseña | <br>2.- Verifica si esta en línea<br>3.- Solicita la contraseña del comandante de la unidad<br><br>5.- Verifica contraseña<br>6.- Si la contraseña es válida, elimina |

.

### 1.3.3.2 Caso de uso: administrar usuarios

| | |
|---|---|
| Actores | Usuario del sistema – Sistema |
| Propósito | Ingresar y eliminar usuarios que pertenecerán a una unidad |
| Precondición | -Sistema en línea<br>-Que un comandante de unidad necesite ingresar o eliminar usuarios |
| Tipo | Secundario |

Escenario 1:

El usuario del sistema podrá agregar al personal que pertenecerá a una unidad específica, para ello el sistema validará que tenga los privilegios para realizar el proceso.

| Acción de Actores | Respuesta del Sistema |
|---|---|
| 1.- Solicitud de ingreso de usuarios<br><br>3.- Ingreso de datos requeridos | <br>2.- Verifica si esta en línea<br><br>4.- Verifica que no existan errores en los datos |

| | |
|---|---|
| 6.- Ingresa contraseña | 5.- Solicita la contraseña del comandante de la unidad<br><br>7.- Si la contraseña es válida, agrega |

Escenario 2:

El usuario del sistema podrá eliminar a personal de una unidad, para ello el sistema validará que tenga los privilegios para realizar el proceso.

| Acción de Actores | Respuesta del Sistema |
|---|---|
| 1.- Solicitud de eliminación de usuarios<br><br>4.- Ingresa contraseña | <br><br>2.- Verifica si esta en línea<br>3.- Solicita la contraseña del comandante de la unidad<br><br>5.- Verifica contraseña<br>6.- Si la contraseña es válida, elimina |

.

**1.3.3.3 Caso de uso: solicitar autodestrucción**

| | |
|---|---|
| Actores | Usuario del sistema – Sistema |
| Propósito | Destruir datos y funcionalidad del software |
| Precondición | -Sistema en línea<br>-Que un usuario necesite dar de baja al software |
| Tipo | Primario |

Escenario 1:

El usuario del sistema podrá inhabilitar en forma permanente al software, generalmente por motivos de seguridad.

| Acción de Actores | Respuesta del Sistema |
|---|---|
| 1.- Solicitud de autodestrucción<br><br>4.- Confirma | <br>2.- Verifica si esta en línea<br>3.- Solicita confirmación<br><br>5.- Inhabilita |

### 1.3.3.4 Caso de uso: confeccionar mensaje oficial

| | |
|---|---|
| Actores | Usuario del sistema – Sistema |
| Propósito | Confeccionar mensaje oficial |
| Precondición | -Sistema en línea<br>-Que un usuario necesite confeccionar un mensaje oficial |
| Tipo | Secundario |

Escenario 1:

El usuario del sistema necesite confeccionar un mensaje oficial, el sistema sólo validará que sea un usuario registrado.

| Acción de Actores | Respuesta del Sistema |
|---|---|
| 1.- Solicitar crear mensaje oficial | |
| | 2.- Verifica si esta en línea<br>3.- Solicita identificación |
| 4.- Ingresa nombre de usuario y contraseña | |
| | 5.- Verifica datos de login<br>6.- Si los datos son correctos carga módulo |
| 7.- Ingresa mensaje<br>8.- Especifica si es para transmitir o es recibido<br>9.- Indica cifrado especial | |
| | 10.- Cifra, de acuerdo a la clave elegida |
| 11.- Ingresa datos adicionales, y solicita almacenamiento | |
| | 12.- Pide confirmación |
| 13.- Confirma | |
| | 14.- Almacena en mensajes de espera |
| 15.- Solicita Impresión de mensaje | |
| | 16.- Imprime mensaje |

### 1.3.3.5 Caso de uso: actualizar mensaje oficial en espera

| | |
|---|---|
| Actores | Usuario del sistema – Sistema |
| Propósito | Actualizar mensaje oficial en espera |
| Precondición | -Sistema en línea<br>-Que un usuario necesite actualizar un mensaje oficial |
| Tipo | Secundario |

Escenario 1:

El usuario del sistema necesite actualizar un mensaje oficial agregando datos que faltaban para su almacenamiento definitivo, el sistema sólo validará que sea un usuario registrado.

| Acción de Actores | Respuesta del Sistema |
|---|---|
| 1.- Solicitar crear mensaje oficial | |
| | 2.- Verifica si esta en línea<br>3.- Solicita identificación |
| 4.- Ingresa nombre de usuario y contraseña | |
| | 5.- Verifica datos de login<br>6.- Si los datos son correctos carga módulo |
| 7.- Solicita mensajes en espera | |
| | 8.- Muestra mensajes en espera |
| 9.- Selecciona mensaje a actualizar | |
| | 10.- Pide ingreso de datos faltantes |
| 11.- Ingresa datos faltantes | |
| | 12.- Verifica datos y almacena en forma permanente el mensaje |

**1.3.3.6 Caso de uso: imprimir informes relacionados con mensajes oficiales**

| | |
|---|---|
| Actores | Usuario del sistema – Sistema |
| Propósito | Impresión de informes adicionales relacionados con mensajes oficiales |
| Precondición | -Sistema en línea<br>-Que un usuario necesite imprimir algun informe relacionado con un mensaje oficial |
| Tipo | Secundario |

Escenario 1:

El usuario del sistema necesite imprimir un informe relacionado con los mensajes oficiales, el sistema sólo validará que sea un usuario registrado.

| Acción de Actores | Respuesta del Sistema |
|---|---|
| 1.- Solicitar crear mensaje oficial | |
| | 2.- Verifica si esta en línea<br>3.- Solicita identificación |
| 4.- Ingresa nombre de usuario y | |

| | |
|---|---|
| contraseña<br><br><br><br>7.- Selecciona el informe que desea | <br><br>5.- Verifica datos de login<br>6.- Si los datos son correctos carga módulo<br><br>8.- Imprime el informe |

### 1.3.3.7 Caso de uso: transmitir un mensaje

| | |
|---|---|
| Actores | Usuario del sistema – Sistema |
| Propósito | Transmitir un mensaje |
| Precondición | -Sistema en línea<br>-Que un usuario necesite enviar un mensaje |
| Tipo | Primario |

Escenario 1:

El usuario del sistema necesite enviar un mensaje a otra unidad, el sistema sólo validará que sea un usuario registrado.

| Acción de Actores | Respuesta del Sistema |
|---|---|
| 1.- Solicitar ingreso a consola de mensajes y archivos | |
| | 2.- Verifica si esta en línea<br>3.- Solicita identificación |
| 4.- Ingresa nombre de usuario y contraseña | |
| | 5.- Verifica datos de login<br>6.- Si los datos son correctos carga módulo |
| 7.- Selecciona el puerto de comunicación y sus parametros<br>8.- Selecciona la unidad de destino<br>9.- Escribe mensaje y confirma | |
| | 10.- Envia mensaje a proceso de encriptación |

### 1.3.3.8 Caso de uso: recibir un mensaje

| | |
|---|---|
| Actores | Usuario del sistema – Sistema |
| Propósito | Recibir un mensaje |
| Precondición | -Sistema en línea<br>-Que un usuario necesite recibir un mensaje |
| Tipo | Primario |

Escenario 1:

El usuario del sistema necesite recibir un mensaje de otra unidad, el sistema sólo validará que sea un usuario registrado.

| Acción de Actores | Respuesta del Sistema |
|---|---|
| 1.- Solicitar ingreso a consola de mensajes y archivos | |
| | 2.- Verifica si esta en línea<br>3.- Solicita identificación |
| 4.- Ingresa nombre de usuario y contraseña | |
| | 5.- Verifica datos de login<br>6.- Si los datos son correctos carga módulo |
| 7.- Selecciona el puerto de comunicación y sus parametros | |
| | 8.- Muestra mensaje que viene del proceso de desencriptación |

### 1.3.3.9 Caso de uso: transmitir un archivo

| | |
|---|---|
| Actores | Usuario del sistema – Sistema |
| Propósito | Transmitir un archivo |
| Precondición | -Sistema en línea<br>-Que un usuario necesite transmitir un archivo |
| Tipo | Primario |

Escenario 1:

El usuario del sistema necesite transmitir un archivo a otra unidad, el sistema sólo validará que sea un usuario registrado.

| Acción de Actores | Respuesta del Sistema |
|---|---|
| 1.- Solicitar ingreso a consola de mensajes y archivos | |
| | 2.- Verifica si esta en línea<br>3.- Solicita identificación |
| 4.- Ingresa nombre de usuario y contraseña | |
| | 5.- Verifica datos de login<br>6.- Si los datos son correctos carga módulo |
| 7.- Selecciona el puerto de comunicación y sus parametros<br>8.- Selecciona el archivo y confirma | |

| | |
|---|---|
| | 9.- Envia el archivo al proceso de compresión |

### 1.3.3.10 Caso de uso: recibir un archivo

| | |
|---|---|
| Actores | Usuario del sistema – Sistema |
| Propósito | Recibir un archivo |
| Precondición | -Sistema en línea<br>-Que exista un archivo en el proceso de descompresión |
| Tipo | Primario |

Escenario 1:

El usuario del sistema necesite recibir un archivo de otra unidad, el sistema sólo validará que sea un usuario registrado.

| Acción de Actores | Respuesta del Sistema |
|---|---|
| 1.- Solicitar ingreso a consola de mensajes y archivos | |
| | 2.- Verifica si esta en línea<br>3.- Solicita identificación |
| 4.- Ingresa nombre de usuario y contraseña | |
| | 5.- Verifica datos de login<br>6.- Si los datos son correctos carga módulo |
| 7.- Selecciona el puerto de comunicación y sus parametros | |
| | 8.- Muestra datos del archivo que viene del proceso de descompresión |

### 1.3.3.11 Caso de uso: comprimir un archivo

| | |
|---|---|
| Actores | Sistema – Sistema |
| Propósito | Comprimir un archivo |
| Precondición | -Sistema en línea<br>-Que exista un archivo seleccionado para envio |
| Tipo | Primario |

Escenario 1:

El sistema necesite comprimir un archivo que se encuentra seleccionado para encriptación y el posterior envio a otra unidad.

| Acción de Actores | Respuesta del Sistema |
|---|---|
| 1.- Archivo seleccionado para envio | 2.- Verifica nombre, formato y tamaño<br>3.- Comprime<br>4.- Envia archivo comprimido al proceso de encriptación |

### 1.3.3.12 Caso de uso: descomprimir un archivo

| | |
|---|---|
| Actores | Sistema – Sistema |
| Propósito | Descomprimir un archivo |
| Precondición | -Sistema en línea<br>-Que exista un archivo en el proceso de desencriptación |
| Tipo | Primario |

Escenario 1:

El sistema necesite descomprimir un archivo que haya sido previamente recibido y desencriptado proveniente desde otra unidad.

| Acción de Actores | Respuesta del Sistema |
|---|---|
| 1.- Archivo proveniente del proceso de desencriptación | 2.- Descomprime<br>3.- Envia el archivo a la consola de archivos |

### 1.3.3.13 Caso de uso: modulación

| | |
|---|---|
| Actores | Sistema – MODEM |
| Propósito | Enviar datos por radio frecuencia |
| Precondición | -Sistema en línea<br>-MODEM conectado<br>-Radio conectada<br>-Que exista un archivo o un mensaje en el proceso de encriptación |
| Tipo | Primario |

Escenario 1:

El sistema necesite enviar datos al MODEM para la transmisión.

| Acción de Actores | Respuesta del MODEM |
|---|---|
| 1.- Archivo o mensaje proveniente del proceso de encriptación | 2.- Aplica modulación FSK<br>3.- Envia datos modulados a la radio militar |

### 1.3.3.14 Caso de uso: demodulación

| | |
|---|---|
| Actores | MODEM – Sistema |
| Propósito | Recibir datos por radio frecuencia |
| Precondición | -Sistema en línea<br>-MODEM conectado<br>-Radio conectada<br>-Que exista un archivo o un mensaje en el MODEM |
| Tipo | Primario |

Escenario 1:

El sistema necesite recibir datos del MODEM de una recepción.

| Acción de Actores | Respuesta del Sistema |
|---|---|
| 1.- Recibe datos modulados de la radio militar<br>2.- Demodula archivo o mensaje recibido | 3.- Envia archivo o mensaje al proceso de desencriptación |

### 1.3.3.15 Caso de uso: localización

| | |
|---|---|
| Actores | Usuario del sistema – Sistema |
| Propósito | Entregar localización por medio de GPS |
| Precondición | -Sistema en línea<br>-GPS conectado<br>-Que se necesite transmitir la posición de una unidad |
| Tipo | Secundario |

Escenario 1:

El usuario del sistema necesite transmitir su posición actual a las demás unidades desplegadas, el sistema sólo validará que sea un usuario registrado.

| Acción de Actores | Respuesta del Sistema |
|---|---|
| 1.- Solicitar ingreso a consola de mensajes y archivos | |
| | 2.- Verifica si esta en línea<br>3.- Solicita identificación |
| 4.- Ingresa nombre de usuario y contraseña | |
| | 5.- Verifica datos de login<br>6.- Si los datos son correctos carga módulo |
| 7.- Selecciona el puerto de comunicación y sus parametros | |
| | 8.- Esperando datos para enviar al proceso de encriptación o compresión |
| 9.- Conectar GPS al puerto RS-232<br>10.- Seleccionar puerto al cual esta conectado el GPS | |
| | 11.- Verifica si existe recepción satélital<br>12.- Si existe recepción satélital envia posición al proceso de encriptación |

### 1.3.3.16 Caso de uso: imprimir informes de tráfico

| | |
|---|---|
| Actores | Usuario del sistema – Sistema |
| Propósito | Impresión de informes relacionados con el tráfico de mensajes y archivos |
| Precondición | -Sistema en línea<br>-Que un usuario necesite imprimir algun informe relacionado con el tráfico de datos |
| Tipo | Secundario |

Escenario 1:

El usuario del sistema necesite imprimir un informe relacionado con el tráfico de mensajes y archivos que han sido transmitidos o recibidos, el sistema sólo validará que sea un usuario registrado.

| Acción de Actores | Respuesta del Sistema |
|---|---|
| 1.- Solicitar ingreso a consola de mensajes y archivos | |
| | 2.- Verifica si esta en línea<br>3.- Solicita identificación |
| 4.- Ingresa nombre de usuario y contraseña | |
| | 5.- Verifica datos de login<br>6.- Si los datos son correctos carga módulo |
| 7.- Selecciona el informe que desea | |
| | 8.- Imprime el informe |

### 1.3.4 Modelo relacional

Se detalla el modelo relacional de datos obtenido, tanto para transmisión y recepción de datos como para la gestión de mensajes oficiales.

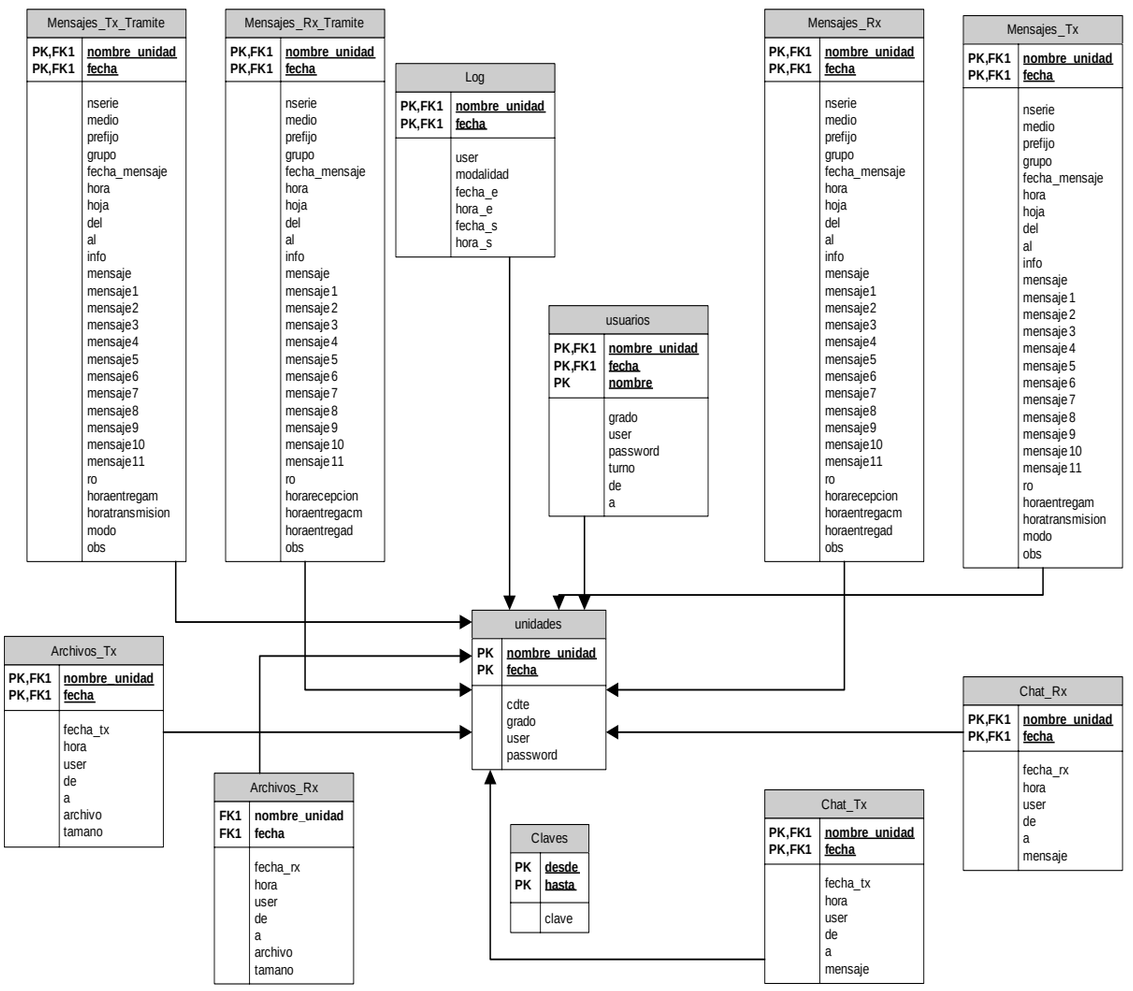



**Fig. 3.1.3 Modelo de datos del software de comunicaciones**

## 1.4 Implementación

Producto del tamaño de la aplicación sólo se mostrarán los diagramas de las principales operaciones que el software realiza.

### 1.4.1 Transmitir o recibir un mensaje

Para transmitir un mensaje, este pasa por un proceso de encriptación posteriormente se envia a un proceso de modulación y finalmente se transmite. En el caso de recepción, el mensaje es enviado a un proceso de demodulación para luego pasar a un proceso de desencriptación y finalmente ser visualizado en la interfaz.

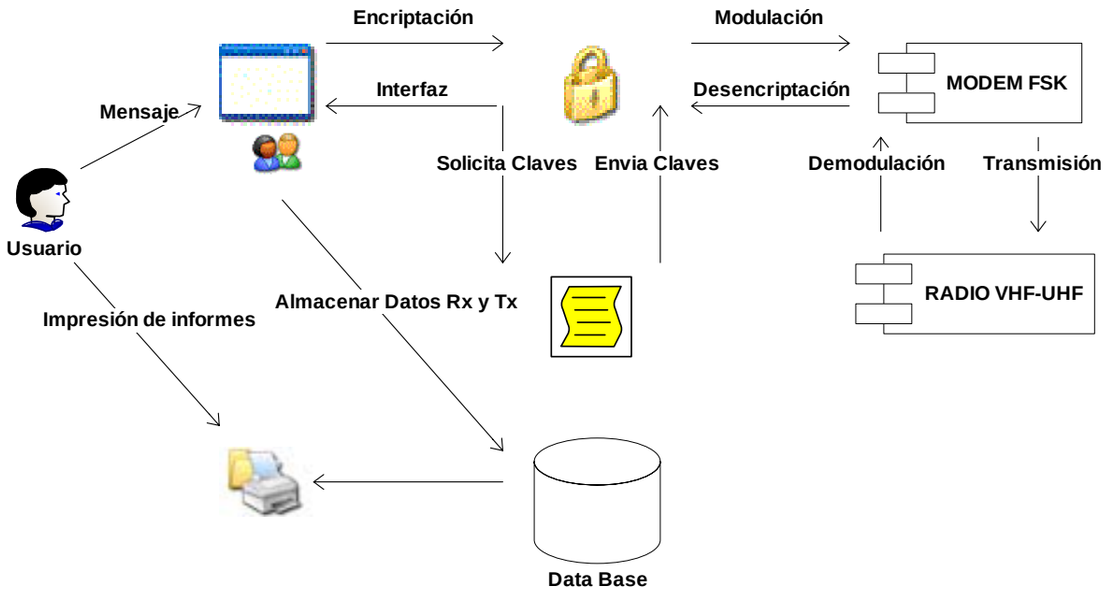



**Fig. 3.1.4 Transmisión y recepción de un mensaje**

### 1.4.2 Transmitir o recibir un archivo

Para transmitir un archivo, este pasa por un proceso de compresión y luego es enviado a encriptación, posteriormente se envia a un proceso de modulación y finalmente se transmite. En el caso de recepción, el archivo es enviado a un proceso

de demodulación para luego pasar a un proceso de desencriptación y descompresión y finalmente ser visualizado en la interfaz.

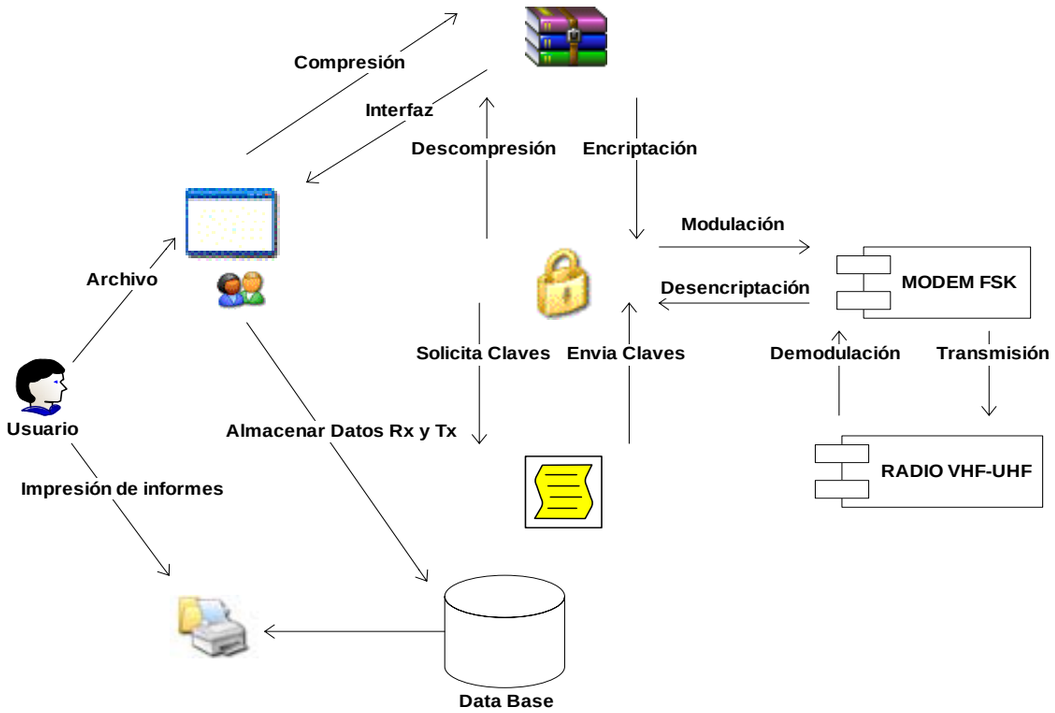



**Fig. 3.1.5 Transmisión y recepción de un archivo**

### 1.4.3 Generar un mensaje oficial

Para generar un mensaje este es ingresado completamente en una interfaz, luego se encripta utilizando un algoritmo simétrico que opera sobre un alfabeto de 26 letras y 10 números, posteriormente se ingresan los datos propios del mensaje oficial y luego queda almacenado en un registro temporal hasta tener los datos definitivos de su proceso de transmisión o recepción, para finalmente pasar a un registro permanente y formar parte de una serie de informes que se generan en el proceso de su confección.

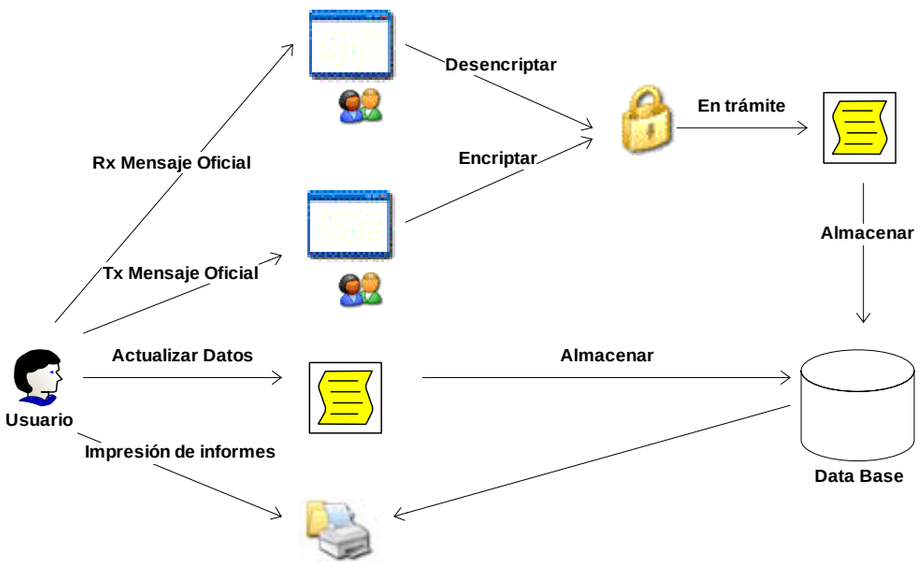



**Fig. 3.1.6 Generación de un mensaje oficial**

## 1.5 Evaluación de costos

Se utilizaran herramientas de ingenieria de software para la estimación del esfuerzo del proyecto, específicamente la metodología basada en puntos de función [Coc+, 2001].

### 1.5.1 Análisis de puntos de función

Es una técnica desarrollada por Allan Albrecht, la organización ISO/IEC ha definido un estándar de medida del tamaño funcional, llamado “ISO/IEC 14143-1:1998” y esta orientado a hacer una estimación del esfuerzo requerido para construir un sistema a partir de la fase de requerimientos [Coc+, 2001]. Una forma de estimar el esfuerzo es asumir un cierto nivel de productividad en puntos de función por persona, en un período de tiempo dado.

Los puntos de función son la unidad de medida de la cuantificación que se realiza a partir de un diseño lógico de la funcionalidad provista al usuario por un

determinado software. La funcionalidad es clasificada en entradas, salidas y consultas definiendo para cada una de ellas tres niveles de complejidad: simple, media y compleja. Estos niveles de complejidad se seleccionan de acuerdo al número de archivos lógicos que la funcionalidad accede para realizar su función, también se mide, utilizando la misma fórmula pero con distintos pesos, la creación de las tablas del modelo de datos y las interfaces que contenga el sistema.

### 1.5.2 Software de estimación de esfuerzo

El proceso para medir el esfuerzo de construcción de un proyecto de software, ha generado resultados muy asertivos, eliminando casi en su totalidad el método perceptivo para la evaluación de los trabajos involucrados. Los parámetros de ajustes son de acuerdo al modelo de estimación por medio de conceptos teóricos utilizando COCOMO II (Constructive Cost Model) [Mor+, 2000].

### 1.5.3 Estimación de esfuerzo

Para obtener la estimación del software, se contemplo la metodología propuesta para el sistema de cifrado y para el software de comunicación y gestión de mensajes oficiales, que corresponden a la parte 2 y 3 de la tesis respectivamente. Producto de que en el momento de hacer la estimación ya se tenia un prototipo del software de comunicación la estimación del esfuerzo se baso en un diseño post arquitectura.

| | | Baja | Media | Alta | Total |
|---|---|---|---|---|---|
| Entradas | | 0 | 8 | 0 | 32 |
| Salidas | | 4 | 4 | 0 | 36 |
| Ficheros Lógicos Internos | | 5 | 6 | 0 | 95 |
| Ficheros Lógicos Externos | | 2 | 0 | 2 | 30 |
| Consultas | | 7 | 10 | 0 | 61 |
| Puntos de Función sin ajustar | | | | | 254 |
| Número de instrucciones por punto de función (Visual Basic) | | | | | 32 |
| KSLOC (Puntos de Función * Nº de instrucciones) | | | | | 8,128 |
| A | | | | | 2,45 |
| SFj | PREC | | | Alta | 2,48 |
| | FLEX | | | Nominal | 3,04 |
| | RESL | | | Nominal | 4,24 |
| | TEAM | | | Alta | 2,19 |
| | EPML | | | Alta | 3,12 |
| B | B=0,91+0,01*(SUMA SFj) | | | | 1,0607 |
| Esfuerzo Nominal sin ajustar (Meses-Personas) | MM=A*(KSLOC)^B | | | | 22,61443749 |
| Tiempo de desarrollo (Meses) | Tdev=2,5*(MM)^0,38 | | | | 8,177237446 |
| Número de Personas | Nº Personas=MM/Tdev | | | | 2,765535138 |
| Factores de Ajuste Post Arquitectura | RELY | | | Alta | 1,1 |
| | DATA | | | Alta | 1,14 |
| | DOCU | | | Nominal | 1 |
| | CPLX | | | Alta | 1,17 |
| | RUSE | | | Nominal | 1 |
| | TIME | | | Nominal | 1 |
| | STOR | | | Nominal | 1 |
| | PVOL | | | Nominal | 1 |
| | ACAP | | | Alta | 0,85 |
| | AEXP | | | Alta | 0,88 |
| | PCAP | | | Alta | 0,88 |
| | PEXP | | | Alta | 0,91 |
| | LTEX | | | Alta | 0,91 |
| | PCON | | | Alta | 0,9 |
| | TOOL | | | Alta | 0,9 |
| | SITE | | | Nominal | 1 |
| | SCED | | | Nominal | 1 |
| Factor de Ajuste | | | | | 0,647791838 |
| MM Final | MMF=MM*Factor | | | | 14,64944803 |
| Costo medio mensual de cada recurso | | | | | 600000 |
| Costo del Proyecto | | | | | $ 8.789.669 |

**Tabla 3.1.1 Cálculo de esfuerzo del software de comunicación**

## 1.6 Integración

Esta parte contempla la integración del sistema de cifrado al software de comunicación y gestión, como también la utilización del MODEM propuesto, que en su conjunto conforman la plataforma de comunicación propuesta en esta tesis.

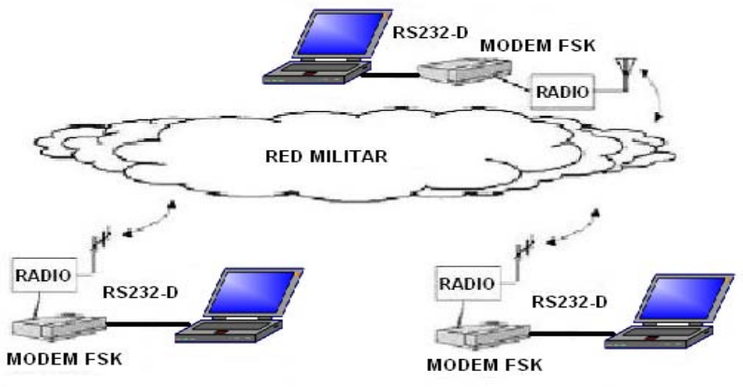



**Fig. 3.1.7 Arquitectura de la plataforma de comunicación**

El MODEM de comunicación toma las tramas seriales encriptadas que vienen del software de comunicación y las modula en base a dos portadoras, en cuanto a los datos antes de confeccionar las tramas se les agrega unos identificadores para distinguir las tramas que pertenecen a un mensaje o a un archivo y además considerán a que unidad van dirigidas. Por otra parte la interfaz MODEM-radio toma 3 aspectos importantes: entrada de datos, salida de datos y PTT (Push To Talk).

**Fig. 3.1.8 Esquema de nodo de la plataforma de comunicación**

En cuanto a la aplicabilidad que tiene la plataforma dentro del área de telecomunicaciones del Ejército de Chile, conforme a su red institucional debe funcionar en los enlaces HF/VHF/UHF que corresponden generalmente a unidades de enlaces desplegadas en terreno.

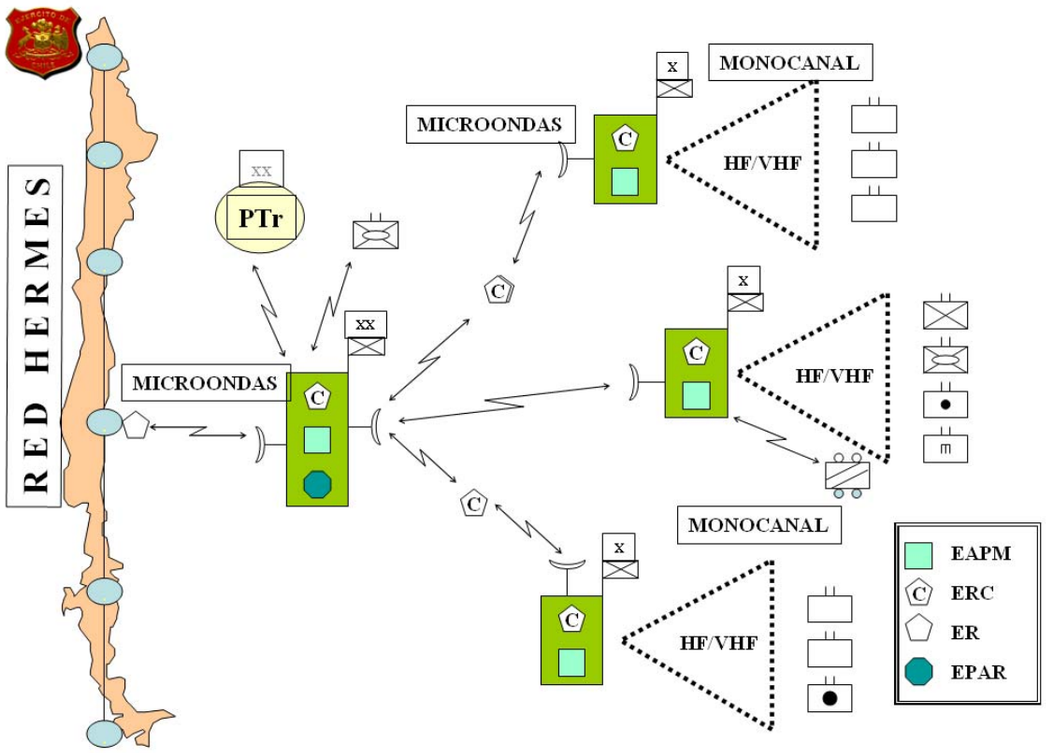



**Fig. 3.1.9 Red institucional de telecomunicaciones del Ejército de Chile**

# Capitulo 4: Evaluación de Resultados

# 1. Resultados obtenidos

El software de comunicación permite y necesita definir tres parámetros fundamentales para el correcto funcionamiento de cada uno de los módulos que contiene. Primero se deben definir las unidades que existirán junto con sus comandantes y posteriormente los integrantes de dichas unidades que a su vez serán usuarios del software. Por otro lado es necesario definir la clave de cifrado del período que interactuará de forma diferente en cada uno de los módulos, es decir para el módulo de mensajes oficiales servirá para hacer un cifrado polialfabético y en el módulo de envio de datos servirá de semilla para el algoritmo generador de claves de cifrado de 128 bit.

El primer módulo del software de comunicación realizado en la tesis abarcó el tema de gestión de mensajes oficiales, dicho módulo esta siendo utilizado en las unidades de telecomunicaciones del Ejército de Chile, específicamente se encuentra en un período de prueba de dos meses para su aprobación definitiva. Este módulo usa un sistema de cifrado polialfabético sobre un alfabeto de 26 letras y 10 digitos, producto que posteriormente es transmitido por fonía utilizando un código internacional de comunicación.

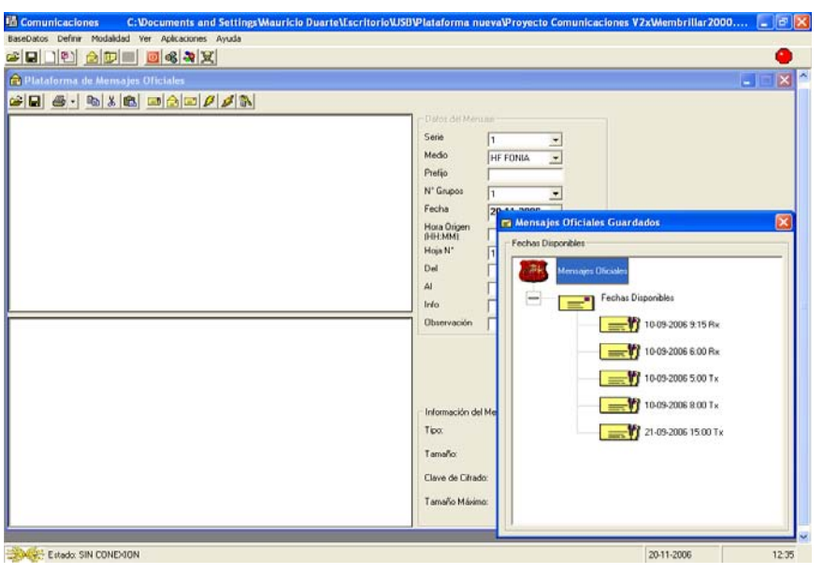



**Fig. 4.1.1 Pantalla principal módulo de gestión de mensajes oficiales.**

El módulo de envio de datos tiene dos consolas, la consola de mensajes y archivos, además cuenta con un proceso de compresión automática y todo el módulo esta sobre el sistema de cifrado de 128 bit.

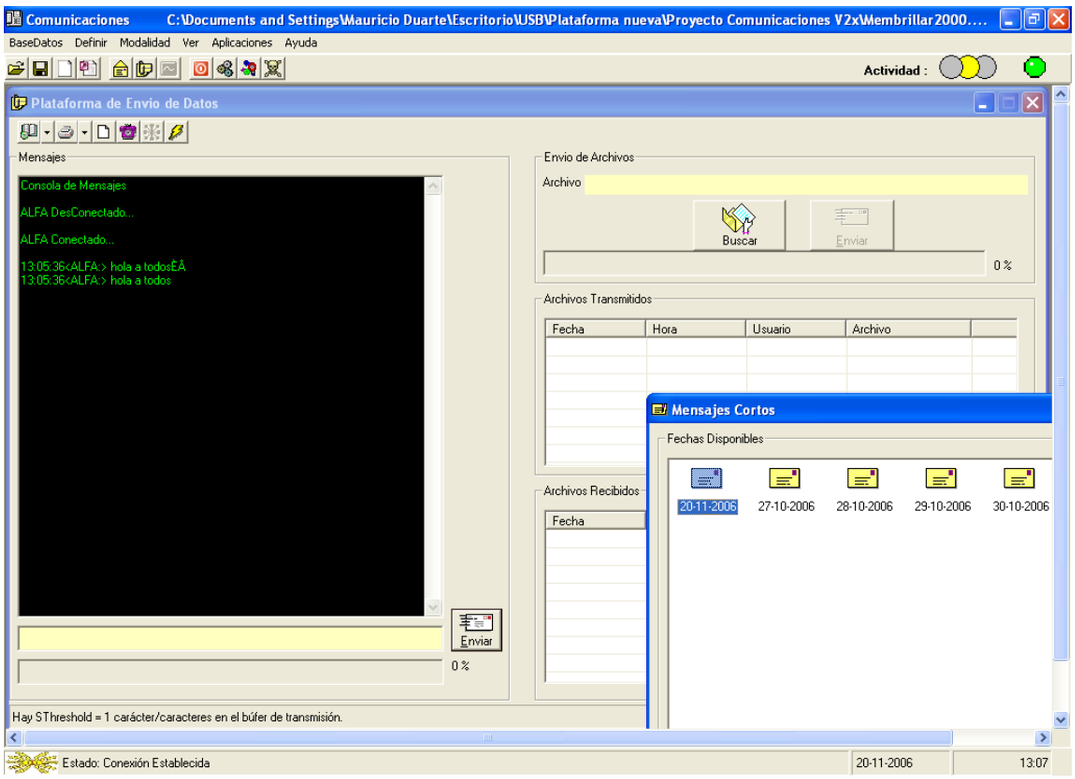



**Fig. 4.1.2 Pantalla principal módulo de envio de datos.**

Por otra parte toda la plataforma cuenta con un proceso de destrucción, en caso de que alguna unidad es dada de baja, este proceso elimina librerías de compresión, librerías de seguridad, registros de Windows y principalmente la base de datos que contiene el tráfico y las claves de cifrado, con todo ello el software de comunicación queda completamente inhabilitado funcionalmente, además cuenta con un sistema de inhabilitación de todas sus características en caso logren cargar nuevamente el software.

La compresión de datos por software, sin duda es uno de los factores observados más importantes, producto que nuestra velocidad de modulación es de

1200 bits/segundos sumado a ello la limitación de velocidad de los equipos de radio frecuencia, lo que contribuye a una baja velocidad de transferencia.

| Formato | Tamaño (Bytes) | Compresión (Bytes) | % Compresión |
|---|---|---|---|
| Excel | 547.326 | 156.134 | 71,5 |
| Word | 898.048 | 281.354 | 68,7 |
| Texto | 352.002 | 38.394 | 89,1 |
| Jpg | 450.260 | 449.840 | 0,1 |
| Gif | 17.493 | 17.441 | 0,3 |
| Pdf | 802.576 | 474.898 | 40,8 |
| Bmp | 3.631.158 | 48.251 | 98,7 |

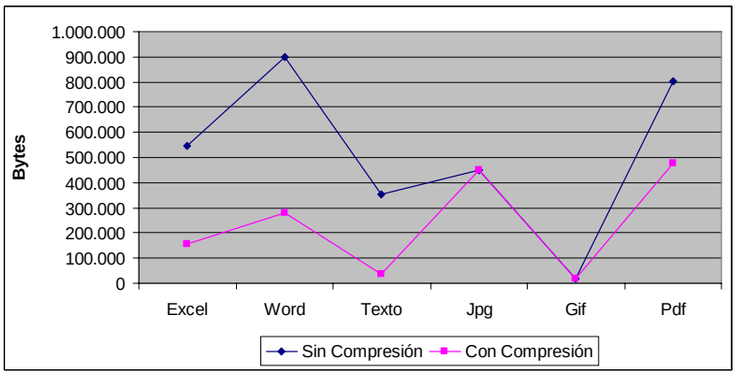



**Gráfico 4.1.1 Comparación de tipos de archivos en el proceso de compresión automática.**

El software de comunicación cuenta con un pequeño módulo de localización, este funciona en base a un GPS conectado a un puerto serial, el software extrae la información de localización y envia los datos a las unidades con las cuales se esta comunicando en ese momento.

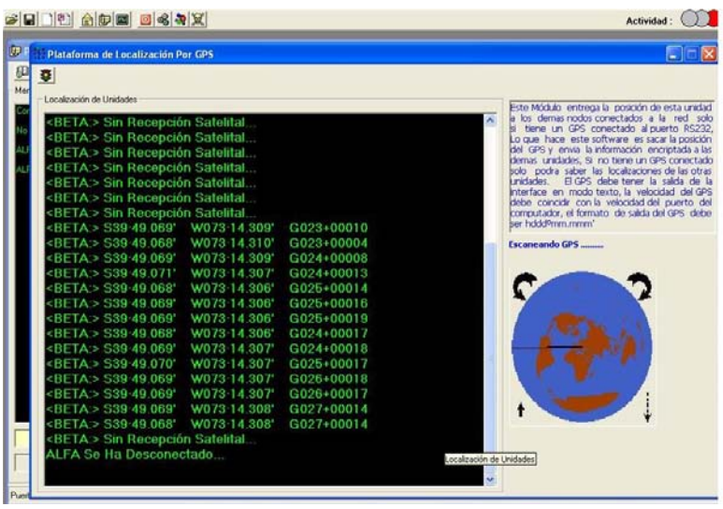



**Fig. 4.1.3 Pantalla de localización por GPS**

El MODEM de comunicación se implementó sobre una protoboard en una etapa de pruebas para evaluar y corregir el proceso de demodulación y sincronización, producto de que es bastante más complejo de implementar que el modulador.

Una vez depurado el proceso de demodulación se confeccionó el circuito en una placa impresa, donde se eliminan una serie de deficiencias de presición producto del uso de una protoboard.

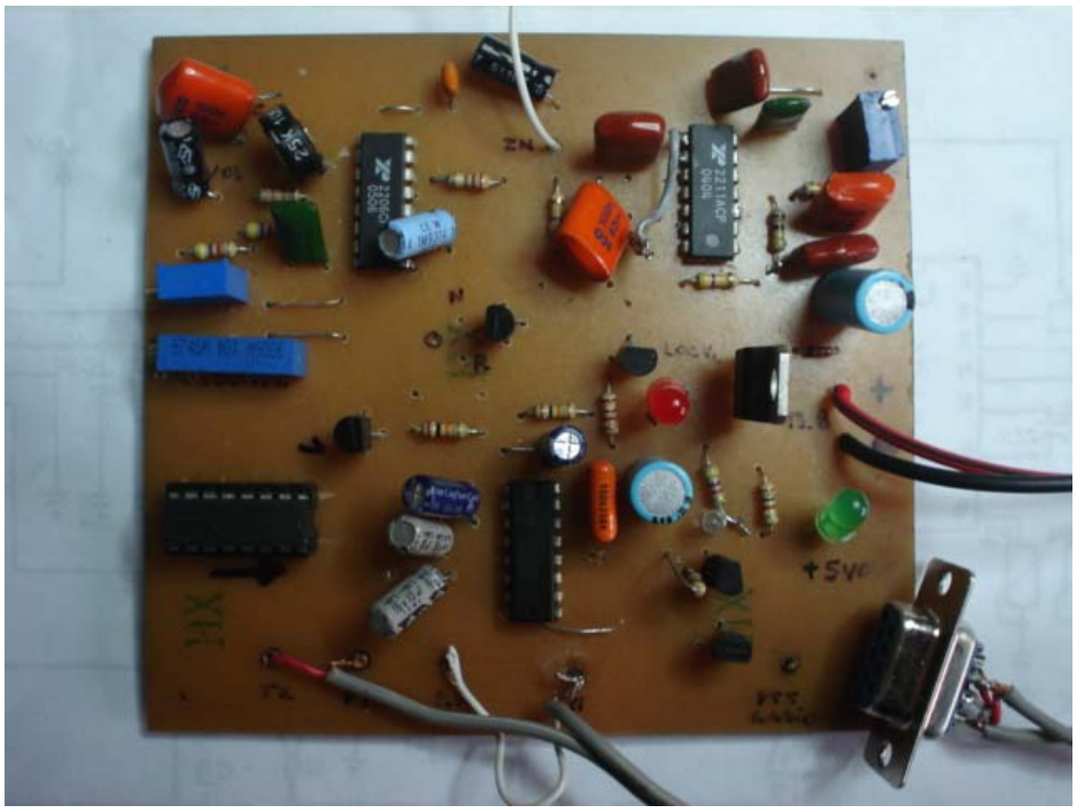



**Fig. 4.1.4 Modem FSK externo diseñado y fabricado para el proyecto de tesis**

# 2. Conclusiones

Se analizó el funcionamiento de las transmisiones de datos por radio frecuencia, específicamente la modulación digital y en base a las normas exigidas por la institución patrocinante se definió la modulación digital a utilizar y sus características de funcionamiento.

Se analizaron los algorítmos y herramientas criptográficas factibles de aplicar a una modulación digital para la transferencia de datos por radio frecuencia, se sumó a esto las politicas de seguridad de la institución patrocinante además de herramientas de desarrollo de software, que en su conjunto permitió modelar los procesos necesarios para el algoritmo que se utilizó.

En base al análisis realizado se diseño un software para la gestión de mensajes oficiales, además de un módulo para el envió/recepción de mensajes y archivos a través de transmisiones por ondas de radio frecuencia, para ello se construyó un MODEM eléctronico utilizando una modulación FSK binaria y se diseño un sistema criptográfico, para todos los procesos de desarrollo de software se utilizaron metodologías propias del área, tanto para la descripción y modelado de los procesos.

Se evalúo la plataforma implementada tanto en su funcionamiento como chequeando que cumpla cada uno de los requisitos recopilados en la institución patrocinadora del proyecto, considerando que se desarrollo cada uno de los sistemas propuestos como prototipos y en forma integrada, además se probarón los sistemas por separado y con el equipo correspondiente.

Se definio la plataforma utilizando la metodología UML para el desarrollo del software de comunicación y el sistema de cifrado, se especificaron los esquemas de diseño del MODEM electrónico y sus ecuaciones principales. En todos los casos es posible implementar estos sistemas y dispositivos, en base al estudio realizado, estableciendo a priori que este documento no constituye un manual pero puede ser considerado como una guía.

Uno de los aspectos más importantes fue el hecho de aplicar tecnologías de informática y telecomunicaciones para obtener la plataforma funcional, además de entender en forma práctica el proceso de modulación digital de datos. Además se logró establecer un mecanismo de gestión flexible y transparente para los usuarios, esto es producto de que se pueden visualizar, borrar y obtener fácilmente todo el tráfico de la red dependiéndo de los privilegios que tenga cada usuario.

Se observó una considerable ventaja al ocupar los equipos de radio frecuencias para transferir información digital, además de la voz que es la forma tradicional de comunicación de los equipos de radio de la institución patrocinadora, agregando además la transparencia para el usuario ya que no necesita de sólidos conocimientos para hacer un uso correcto de la plataforma.

La escalabilidad de la plataforma involucra establecer con el tiempo una nueva modulación para alcanzar una mayor velocidad y un mejor uso del espectro electromagnético, también mejorar su seguridad estableciendo un nuevo algorítmo criptográfico ya que su diseño fue contemplado de tal forma que sea relativamente sencillo establecer un cambio de este tipo.

La modulación se ve afectada cuando la energía de las radios disminuye, provocando errores en las transmisiones, ésta limitante ha sido superada por una modulación que se va adaptando de acuerdo a la energia de los equipos de comunicación.

Es importante recalcar el aporte académico entregado por medio de la presentación del proyecto en el congreso internacional SENACITEL 2006, donde se expuso con personal del Ejército de Chile y se mostró su aplicabilidad y la importancia del desarrollo para la institución en el aréa de telecomunicaciones.

# 3. Glosario

**Baudio.** Es un concepto físico, y son las veces por segundo que puede modificarse la onda electromagnética para transmitir la información.

**BER.** Bit Error Rate, se usa para medir la tasa de errores del medio y se define BER=bits erróneos / bits transmitidos.

**GPS.** Global Position System, es un sistema de navegación satélital dependiente de la fuerza aérea de EEUU que permite obtener la ubicación en cualquier punto de la tierra, además de la velocidad y el tiempo.

**MODEM.** Modulador-Demodulador, abreviatura usada para referirse a un dispositivo capaz de cambiar cierta información adaptándola para transmitirla en un determinado medio y reconstruirla al llegar al receptor.

**S/N.** Relación de potencia de portadora a ruido.

**Aritmética modular.** Son las operaciones de suma o producto que se llevan a cabo sobre los números enteros módulo algún entero *n*. Es decir el resultado de una suma o un producto es el residuo de la división entre *n*.

**Problema del logaritmo discreto.** Es el problema de encontrar el número de veces que hay que multiplicar un número conocido, para obtener como resultado otro también conocido, por ejemplo dado el 1024 y el 2, ¿cuántas veces hay que multiplicar el 2 para obtener 1024? La respuesta es 10 y se dice que 10 es el logaritmo de 1024 base 2.

**UML.** Lenguaje de Modelamiento Unificado, permite definir un proyecto de ingenieria de software, es una metodología orientada a la programación orientada a objetos. [Dob+, 2006].

**Mensaje oficial.** Es un mensaje con cierto formato que usa el Ejército de Chile en situaciones conflictivas, llamándose comunicado a un mensaje que no cumpla con el formato oficial.

**PLL.** Phase Locked Loops, los lazos de seguimiento de fase son dispositivos realimentados, este se encuentra enganchado cuando ante una variación dentro de unos márgenes de la frecuencia de entrada su frecuencia de salida evolucionará hasta igualarla.

**VCO.** Oscilador controlado por tensión, es un dispositivo electrónico que usa amplificación, realimentación y circuitos resonantes que da a su salida una señal eléctrica de frecuencia proporcional a la tensión de entrada.

**Detector de fase.** Es un dispositivo electrónico que proporciona a su salida una señal proporcional a la diferencia de fase de las dos señales de entrada.

# 4. Referencias


| | |
|---|---|
| [Van+, 1996] | A Meneses, P. Van Oorschot, S. Vanstone: “Handbook of applied cryptography”, 1996 |
| [Dob+, 2006] | Brian Dobing and Jeffrey Parsons: “How UML is used”, Communications of the ACM, Mayo 2006. |
| [Foc+, 1995] | R. Focardi: “A classification of security properties”, Journal of computer security, 1995 |
| [RSA+, 1998] | RSA Data Security, “RSA Laboratories”, 1998 |
| [ANS+, 1983] | ANSI X3.92, “American National Standard-Data Encryption Algorithm-Modes of Operation”, American National Standard Institute 1983 |
| [Odl+, 1993] | A. M. Odlyzko, “Public Key Cryptography”, AT&T Bell Laboratories, 1993 |
| [Ohr+,2003] | Frank Ohrtman, “WI-FI Handbook”, Ed. McGraw-Hill., 2003 |
| [Tom+,1996] | W. Tomasi, “Sistemas de Comunicación Electrónicas”, Segunda Edición, 1996 |
| [Str+,1990] | F. Stremler, “Introducción a los sistemas de comunicación”, Addison-Wesley Iberoamericana, 1990 |
| [Mor+,2000] | Ana Moreno, “COCOMO II”, Universidad Politécnica de Madrid, 2000 |
| [Riv+,1995] | R.L. Rivest, The RC5 encryption algorithm, Fast software Encryption LNCS 1008, pp 86-96, 1995 |
| [Dav+,2001] | W.Alan. Davis, Radio Frecuency Circuit Design, A Wiley-Interscience Publication, 2001 |

| | |
|---|---|
| [Zoh+,1995] | Zoher.Z. Karu, Signals and Systems, Zizi press Cambrigde MA, 1995 |
| [Opp+,1994] | Oppenheim Alan, Señales y Sistemas, Prentice Hall Hispanoamericana, 1994 |
| [Pan+,1965] | Panter Philip, Modulation and Spectral Analysis, Mc Graw Hill, 1965 |
| [Gos+,1981] | Gosling P., Códigos para ordenadores y microprocesadores, Paraninfo, 1981 |
| [Den+,1982] | Denning Dorothy, Cryptography and data security, Addison Wesley, 1982 |
| [Bit, 02] | Bitam: “Acerca de Business Intelligence”. http://www.bitam.com/spanish/AcercaDeBI.html |
| [Coc+,2001] | COCOMO II: una familia de modelos de estimación, Universidad Simón Bolivar, Ingeniería de Software, 2001, http://www.ldc.usb.ve/~teruel/ci4713/clases2001 |
| [Vpn,2002] | VPN: Redes Privadas Virtuales, http://www.vpnlabs.org |
| [Exa+, 2006] | EXAR: Circuitos Integrados, http://www.exar.com |

# Capitulo 5: Anexos

# 1. Modulación digital

## 1.1 Problemas de la transmisión de señales en radio frecuencia

- **Atenuación.** Es un efecto producido por el debilitamiento de la señal debido a la resistencia eléctrica que presentan tanto el canal como los demás elementos que intervienen en la transmisión. Este debilitamiento se manifiesta en un descenso de la amplitud de la señal transmitida.

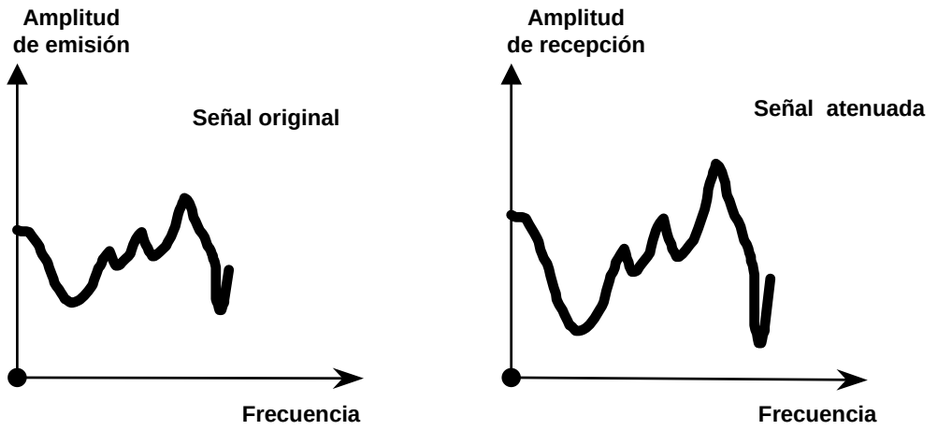



**Fig. 5.1.1 Representación gráfica de la atenuación**

- **Distorsión.** Es la deformación de la señal producida normalmente porque el canal se comporta de modo distinto en cada frecuencia. Es producto de una falta de linealidad.

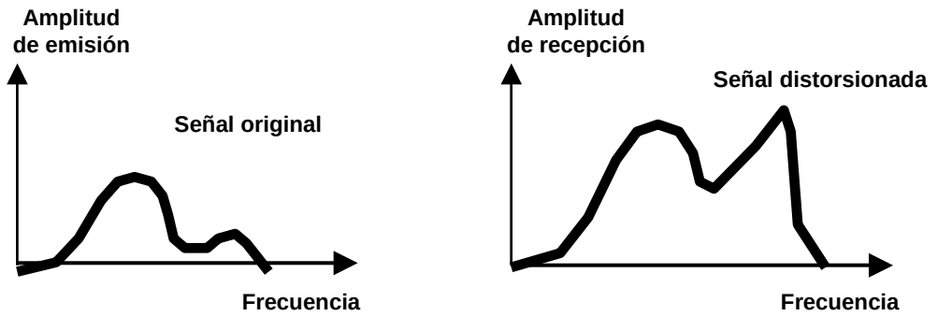



**Fig. 5.1.2 Representación gráfica de la distorsión**

- **Interferencia.** Es la adición de una señal conocida y no deseada a la señal que se transmite. Se produce, cuando dos estaciones emisores emiten en la misma frecuencia produciéndose la superposición de ambos mensajes.

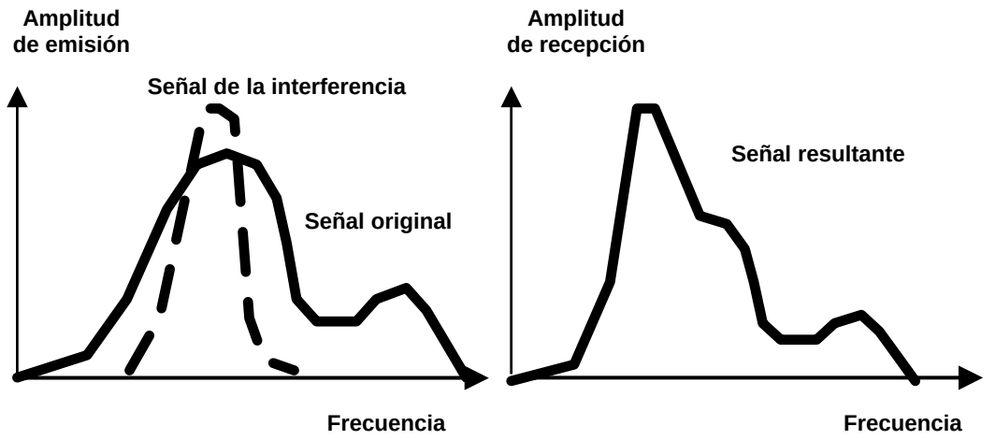



**Fig. 5.1.3 Representación gráfica de la interferencia**

- **Ruido.** Es la suma de múltiples interferencias, posiblemente de origen desconocido y de naturaleza aleatoria. Los propios componentes físicos de cualquier canal o dispositivo de transmisión generan ruido eléctrico, en ocasiones, el ruido es selectivo y se puede aislar, en otros casos el ruido se encuentra muy extendido en toda la gama de frecuencias y su neutralización se hace difícil.

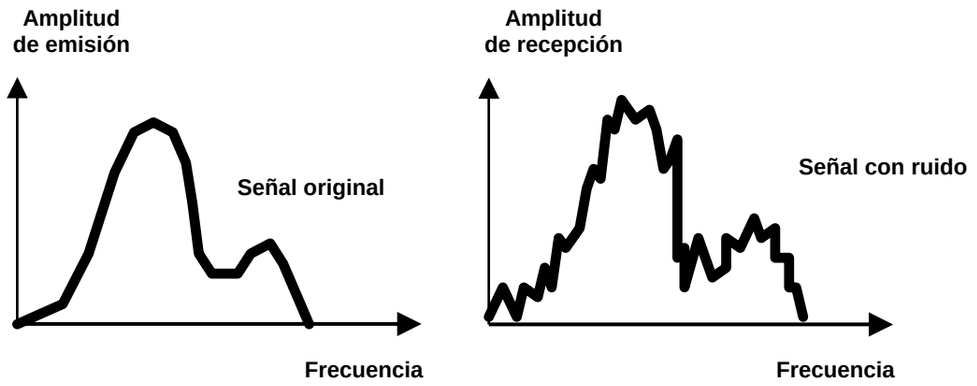



**Fig. 5.1.4 Representación gráfica del ruido**

## 1.2 Modulación por desplazamiento de frecuencia (FSK)

Es similar a la modulación FM en transmisión analógica, el resultado de este tipo de modulación es una onda modulada cuya amplitud es la misma que la portadora pero variando la frecuencia en función de los valores que toma la señal moduladora, es decir, a cada símbolo que se quiere transmitir se le asocia una frecuencia. Para que el receptor interprete bien el mensaje tendrá que hacer un análisis de la frecuencia que le llega en cada momento en la señal portadora, es decir, la información del mensaje reside en la frecuencia. Normalmente esta técnica es usada para transmisión de datos a bajas velocidades.

El FSK binario es una forma de modulación angular de amplitud constante, donde la señal moduladora es un flujo de pulsos binarios que varía, entre dos niveles de voltaje discreto, en lugar de una forma de onda analógica que cambia de manera continua. La expresión general para una señal FSK binaria esta dada por

$$\phi(t) = A_c \text{ sen} \left\{ \int_0^t \left[ \omega_c + (\Delta\omega) x(t) \right] dt \right\}$$ con:

$A_c$ = *Amplitud de la portadora no* modulada

$\omega_c$ = *Frecuencia portadora* $\left(\omega_c = \frac{\omega_2 + \omega_1}{2}\right)$

$x(t)$ = *Señal* modulada *digital binaria*

$\Delta\omega = \frac{\omega_2 - \omega_1}{2}$

De la ecuación puede verse que con el FSK binario, la amplitud de la portadora $W_c$ se mantiene constante con la modulación. Sin embargo, la frecuencia en radianes de la portadora de salida $\omega_c$ cambia por una cantidad igual a $\pm \Delta\omega/2$. El cambio de frecuencia $\Delta\omega/2$ es proporcional a la amplitud y a la polaridad de la señal de entrada binaria, la señal de energia del digito binario esta dado por $E = \int_0^T A^2 \operatorname{sen}^2 m\omega_c t \, dt = A^2 T / 2$ [Str+, 1990].

- **Transmisor FSK.** La salida de un modulador de FSK binario, es una función escalón en el dominio del tiempo, conforme cambia la señal de entrada binaria de 0 lógico a 1 lógico, y viceversa, la salida del FSK se desplaza entre dos frecuencias: una frecuencia de marca ó de 1 lógico y una frecuencia de espacio ó de 0 lógico. Con el FSK binario, hay un cambio en la frecuencia de salida, cada vez que la condición lógica de la señal de entrada binaria cambia.

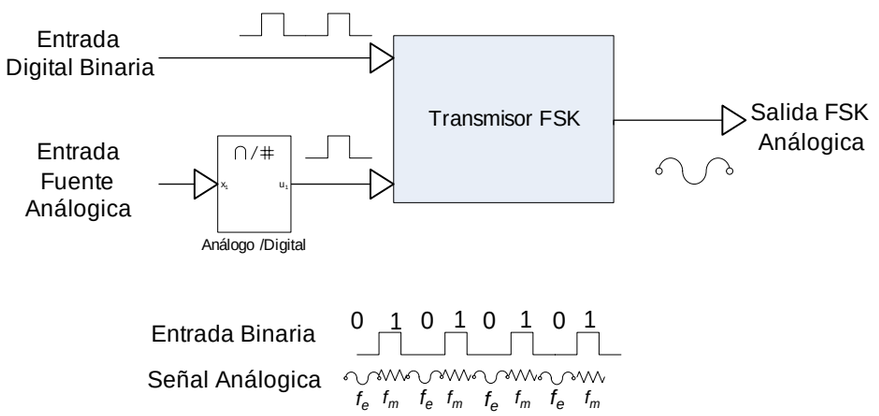



**Fig. 5.1.5 Transmisor FSK, $f_m$ frecuencia de marca, $f_s$ frecuencia de espacio**

- **Receptor FSK.** Las dos señales recibidas ahora son diferentes, por lo que se utilizan dos filtros acoplados, uno para cada señal, por ende la detección es síncrona. Existen dos formas de detectar una señal FSK, coherentes y no coherentes, en el caso de una detección coherente se hace por medio de lazos de amarre de fase (PLL) [Dav+, 2001], el intervalo de frecuencia de cada lazo está restringido y el filtro pasabajos es lo bastante angosto para que los osciladores controlados por tensión (VCO) no cambien de manera apreciable su frecuencia durante una pausa, el espaciamiento de frecuencia debe ser de al menos $2\Delta fT \geq 1$, donde $2\Delta f$ es la diferencia de las dos frecuencias utilizadas y T es la duración del símbolo [Str+, 1990]. Otro método posible es la utilización de un discriminador para convertir las variaciones en frecuencia a variaciones de amplitud, seguidas de un detector de envolvente, este método tiene un rendimiento ligeramente mas pobre. Por lo regular, la frecuencia natural del PLL se hace igual a la frecuencia central del modulador de FSK $\omega_c = \frac{\omega_2 + \omega_1}{2}$. Debido a que sólo hay dos frecuencias de entrada (marca y espacio), también hay sólo dos voltajes de error de salida. Uno representa un 1 lógico y el otro un 0 lógico. En consecuencia, la salida es una representación de dos niveles (binaria) de la entrada de FSK.

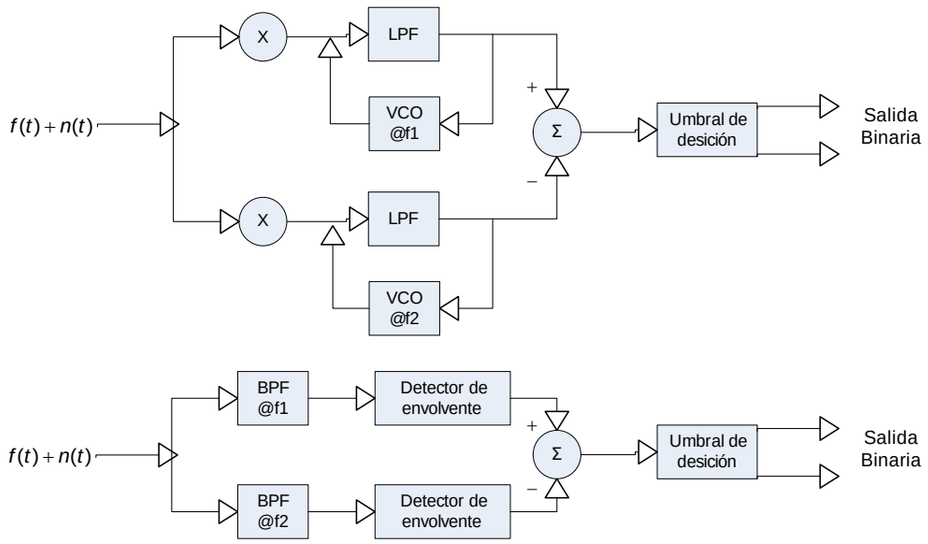



**Fig. 5.1.6 Diagrama de detección coherente y no coherente**

- **Desplazamiento mínimo de FSK.** La transmisión de desplazamiento mínimo de FSK (MFSK), es una forma de transmitir desplazando la frecuencia de fase continua (CPFSK). En esencia, el MFSK es un FSK binario, excepto que las frecuencias de marca y espacio están sincronizadas con la razón de bit de entrada binario. Con MFSK, las frecuencias de marca y espacio están seleccionadas, de tal forma que están separadas de la frecuencia central, por exactamente, un múltiplo impar de la mitad de la razón de bit [$f_m$ y $f_s = n(f_b / 2)$, con n = entero impar]. Esto asegura que haya una transición de fase fluida, en la señal de salida analógica, cuando cambia de una frecuencia de marca a una frecuencia de espacio, o viceversa.

- **Ancho de banda de FSK.** Depende de la separación de frecuencia utilizada, donde la sepración óptima está dada por $\frac{T}{2}\left[1-\frac{\text{sen}(2\Delta\omega T)}{(2\Delta\omega T)}+\frac{\text{sen}(2\omega_c T)}{(2\omega_c T)}\right]$ con $2\Delta\omega = \omega_2 - \omega_1$ y $2\omega_c = \omega_2 + \omega_1$, además esta relacionada directamente con la tasa de información para frecuencias bajas, el mínimo o máximo ancho de banda ocurre cuando tanto la desviación de frecuencia y la frecuencia modulante están en sus

valores máximos. La desviación de frecuencia es constante y, siempre, en su valor máximo. $f_a$ es igual a la frecuencia fundamental de entrada binaria que bajo la condición del peor caso es igual a la mitad de la razón de bit ($f_b$). En un FSK binario el índice de modulación, por lo general, se mantiene bajo 1, produciendo así un espectro de salida de FM de banda relativamente angosta. Debido a que el FSK binario es una forma de modulación en frecuencia de banda angosta, el mínimo ancho de banda depende del índice de modulación. Por lo tanto se tiene que:

$$m = \frac{\Delta f}{f_a} = \frac{\left|\frac{f_m - f_s}{2}\right|}{\frac{f_b}{2}} = \frac{|f_m - f_s|}{f_b}$$

$m$ = Índice de Modulación

$\Delta f$ = Desviación de Frecuencia (Hertz)

$f_a$ = Frecuencia Moduladora (Hertz)

$f_b$ = Razón de bits de entrada

$f_b$ / **2** = Frecuencia fundamental de la señal de entrada binaria

Sin embargo, en la modulación digital ninguna de estás conclusiones es necesariamente válida, por lo que se debe ser cauto al aproximar el ancho de banda requerido para FSK, para $2\Delta fT > 1$ se puede aproximar utilizando la regla de carson con $f_m = 1/T$ . Sin embargo para $2\Delta fT < 1$ el ancho de banda siempre sera mayor que $2\Delta f$ y puede ser menor que el ancho de banda bilateral de la señal moduladora. En cuanto a la densidad espectral de la señal FSK, para valores bajos de $2\Delta fT$ la densidad espectral tiene un solo pico centrado en la frecuencia portadora, conforme aumenta $2\Delta fT$ el pico central disminuye y empiezan a aparecer picos cercas de las frecuencias de desviación $f_c \pm \Delta f$ , para

valores aun mayores de $2\Delta fT$ tiende a dos grupos espectrales identificables por separado centrados en $f_c \pm \Delta f$ [Str+, 1990].

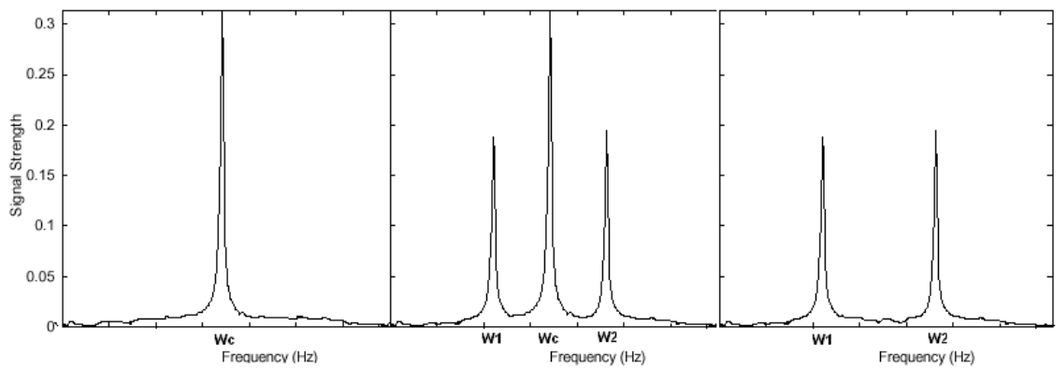



**Fig. 5.1.7 Densidad espectral de una señal FSK**

## 1.3 Modulación por desplazamiento de amplitud (ASK)

Es similar a la modulación AM en transmisión analógica. Consiste en transmitir una portadora cuando se envía un 1 y en no transmitir ninguna señal cuando se envía un 0, es decir la señal modulada en amplitud resultante consta de pulsos de radio frecuencia llamadas marcas que representan unos binarios y espacios que representan ceros binarios. La señal ASK para un pulso (un uno binario) se puede escribir como $\phi(t) = \begin{Bmatrix} A \text{ sen } \omega_c t & 0 < 1 \leq T \\ 0 & \textit{en otro caso} \end{Bmatrix}$, donde T es la duración del símbolo. La señal de energia del digito binario esta dado por $E = \int_0^T A^2 \text{ sen}^2 \omega_c t \, dt = A^2 T / 2$. El receptor debe tomar una desición en t=T basado en las dos posibilidades $y(t) = n_0(T)$ y $y(T) = E + n_0(T)$ para unos y ceros con igual probabilidad en la fuente y ruido con una función de densidad de probabilidad simétrica, el umbral de desición óptimo se fija en $E/2$. En el caso de una detección síncrona el receptor debe tener un filtro acoplado sincronizado en frecuencia y fase para un oscilador a la frecuencia de señalización, o utilizar una detección de envolvente que es más simple de construir, la densidad espectral en ASK se concentra en Wc [Str+, 1990].

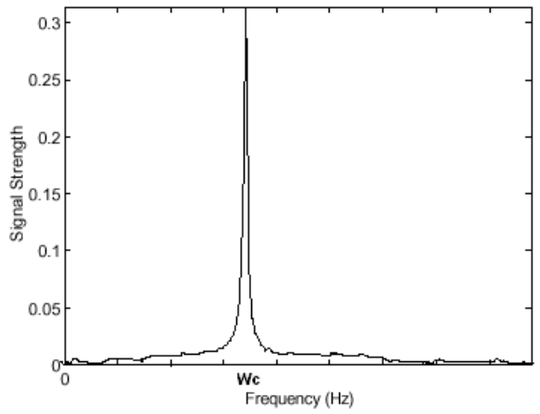



**Fig. 5.1.8 Densidad espectral de una señal ASK**

Es una modulación ineficiente ya que esta susceptible a cambios abruptos y al ruido eléctrico, generalmente usada sobre fibras ópticas y su ancho de banda esta directamente relacionado con la tasa de bits.

## 1.4 Modulación por desplazamiento de fase (PSK)

Transmitir por desplazamiento en fase (PSK) es otra forma de modulación angular, modulación digital de amplitud constante. El PSK es similar a la modulación en fase convencional, excepto que con PSK la señal de entrada es una señal digital binaria y son posibles un número limitado de fases de salida.

- **Desplazamiento de fase binaria (BPSK).** Con la transmisión por desplazamiento de fase binaria (BPSK), son posibles dos fases de salida para una sola frecuencia de portadora. Una fase de salida representa un 1 lógico y la otra un 0 lógico, conforme la señal digital de entrada cambia de estado, la fase de la portadora de salida se desplaza entre dos ángulos que están 180° fuera de fase. La representación de la BPSK esta dada por $\phi(t) = A\cos[\omega_c t + \Delta\theta\, p(t)]$ con $\Delta\theta$ como la desviación de fase pico y p(t) es una función de conmutación con $\pm 1$ estados posibles [Str+, 1990]. El BPSK es una forma de modulación de onda cuadrada de portadora suprimida de una señal de onda continua. Los datos de

entrada al circuito de producto deben ser del tipo bipolar (+1,-1) para polarizar en forma alternada los diodos, el demodulador se fundamenta en el mismo esquema de funcionamiento pero la complejidad es superior debido a que se requiere una referencia de fase para poder reconocer la modulación de 0° y 180° de fase.

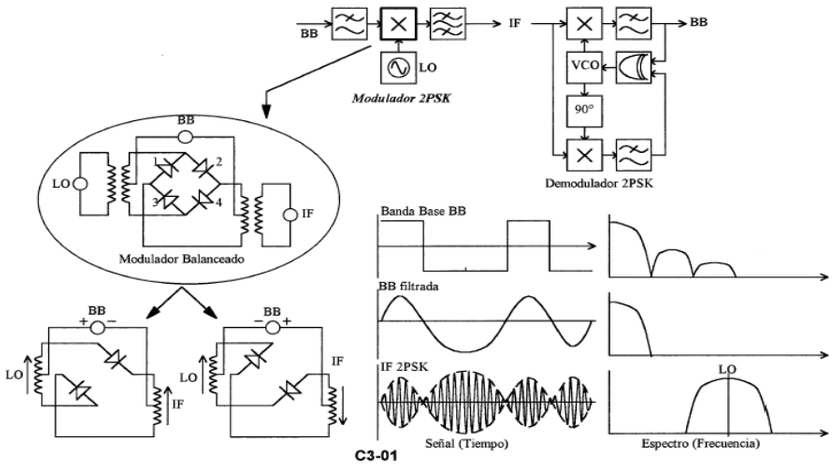



**Fig. 5.1.9 Modulador y demodulador BPSK**

- **Transmisor BPSK.** El modulador balanceado actúa como un conmutador para invertir la fase. Dependiendo de la condición lógica de la entrada digital, la portadora se transfiere a la salida, ya sea en fase o 180° fuera de fase, con el oscilador de la portadora de referencia.

- **Receptor BPSK.** El circuito de recuperación de portadora coherente detecta y regenera una señal de portadora que es coherente, tanto en frecuencia como en fase, con la portadora del transmisor original. El modulador balanceado es un detector de producto; la salida es el producto de las dos entradas (la señal de BPSK y la portadora recuperada). El filtro pasa bajos (LPF) separa los datos binarios recuperados de la señal demodulada compleja. Se puede utilizar un PLL para demodular BPSK si está presente una componente portadora suficiente o un lazo de amarre de fase controlado por cristal. Otra forma de recuperación de portadora

es el lazo de costas, este utiliza detectores tanto en fase como en cuadratura para mantener un VCO centrado en la frecuencia de portadora suprimida.

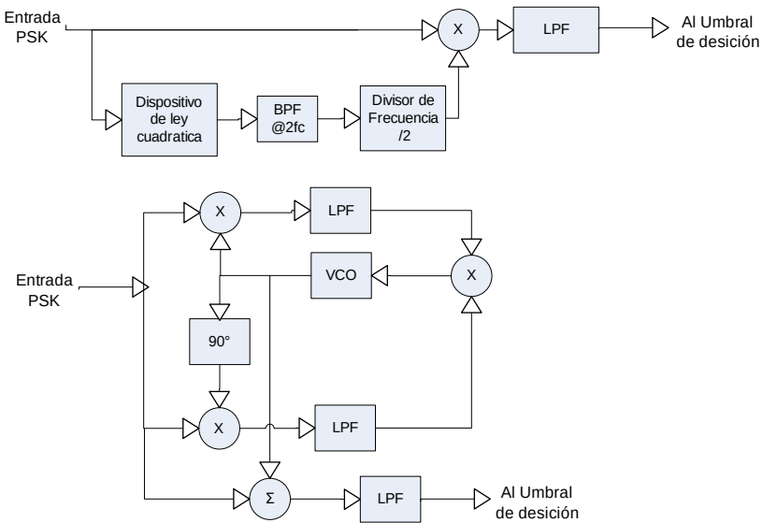



**Fig. 5.1.10 Detección PSK y lazo de costas**

- **Ancho de banda de BPSK.** La razón de cambio de salida, es igual a la razón de cambio de entrada, y el ancho de banda de salida más amplio ocurre cuando los datos binarios de entrada son una secuencia alterativa 1/0. La frecuencia fundamental ($f_a$) de una secuencia alterativa de bits 1/0 es igual a la mitad de la razón de bit ($f_b/2$). Por lo tanto la fase de salida de un modulador de BPSK es:

$$F_{salida} = f_{fsmb}\ x\ f_{pnm} = (\operatorname{sen}\omega_a t)x(\operatorname{sen}\omega_c t)$$

$$F_{salida} = \frac{1}{2}\cos(\omega_c - \omega_a) - \frac{1}{2}\cos(\omega_c + \omega_a)$$

$f_{fsmb}$ = Frecuencia fundamental de la señal moduladora binaria

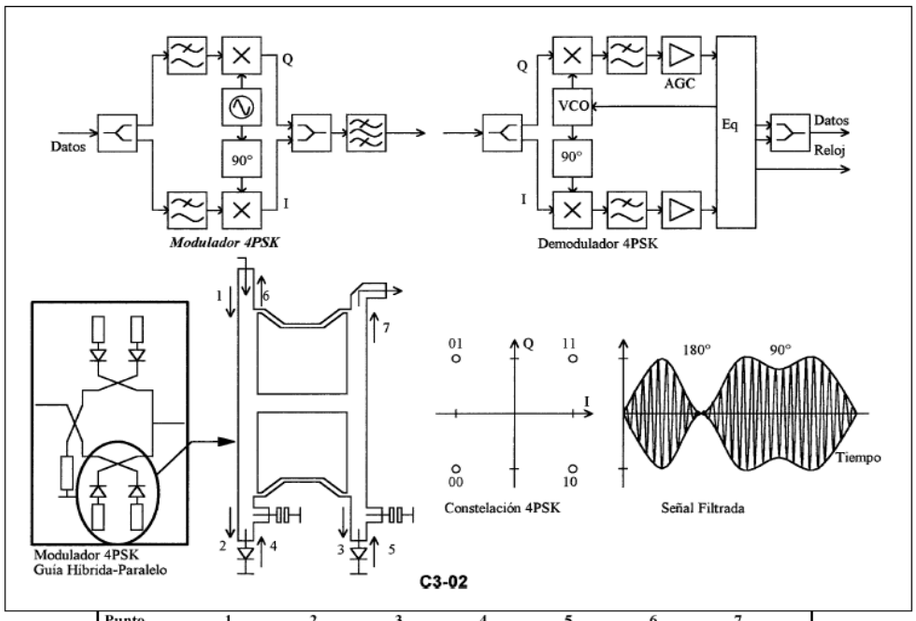
$f_{pnm}$ = Frecuencia portadora no modulada

En consecuencia, el mínimo ancho de banda de Nyquist de doble lado ($f_N$) es $2\pi f_N = (\omega_c + \omega_a) - (\omega_c - \omega_a) = 2\omega_a$ y como $f_a = f_b/2$, se tiene

$f_N = \frac{2\omega_a}{2\pi} = 2f_a = f_b$. El índice de modulación esta dado por $m = \cos \Delta\theta$ y la densidad espectral se centra alrededor de $\omega_c$ y tiene una forma representada por $(\text{sen } x/x)^2$.

- **Desplazamiento de fase cuaternaria (QPSK).** Con QPSK son posibles cuatro fases de salida, para una sola frecuencia de la portadora. Debido a que hay cuatro fases de salida diferentes, tiene que haber cuatro condiciones de entrada diferentes. Ya que la entrada digital a un modulador de QPSK es una señal binaria (base 2), para producir cuatro condiciones diferentes de entrada, se necesita más de un bits de entrada. Con 2 bits, hay cuatro posibles condiciones: 00, 01, 10 y 11. En consecuencia, con QPSK, los datos de entrada binarios se combinan en grupos de 2 bits llamados dibits, cada código dibit genera una de las cuatro fases de entrada posibles.

  La modulación de 4 estados de fase QPSK resulta tener una mejor eficiencia espectral (relación entre la velocidad de información en b/s y el ancho de banda necesario en Hz). En otras palabras se requiere menor ancho de banda para transmitir la misma información debido a que cada nivel de fase lleva 2 bits de información [Str+, 1990].




| Punto | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| BB=0 | V ↑ | V/√2↑ | V/√2→ | V/√2↑ | V/√2→ | 0 | V→ |
| BB=1 | V ↑ | V/√2↑ | V/√2→ | V/√2↓ | V/√2← | 0 | V← |

**Fig. 5.1.11 Modulador y demodulador QPSK**

- **Transmisor QPSK.** Dos bits (un dibit) se introducen al derivador de bits, después que ambos bits han sido introducidos, en forma serial, salen simultáneamente en forma paralela. Un bit se dirige al canal I (en fase) y el otro al canal Q (en cuadratura), el bit I modula una portadora que está en fase con el oscilador de referencia, y el bit Q modula una portadora que está 90° fuera de fase o en cuadratura con la portadora de referencia. Una vez que un dibit ha sido derivado en los canales I y Q, la operación es igual que en el modulador de BPSK. En esencia, un modulador de QPSK son dos moduladores BPSK, combinados en paralelo. La distribución de bits para cada fase se realiza mediante la codificación cíclica (Gray) de tal forma que entre una fase y las adyacentes a 90° solo se tiene el cambio de un bit.

- **Receptor QPSK.** El derivador de potencia dirige la señal QPSK de entrada a los detectores de producto, I y Q, y al circuito de recuperación

de la portadora. El circuito de recuperación de la portadora reproduce la señal original del modulador de la portadora de transmisión. La portadora recuperada tiene que ser coherente, en frecuencia y fase, con la portadora de referencia transmisora. La señal QPSK se demodula en los detectores de producto, I y Q, que generan los bits de datos, I y Q, originales. Las salidas de los detectores de productos alimentan al circuito para combinar bits, donde se convierten de canales de datos, I y Q, paralelos a un solo flujo de datos de salida binarios.

- **Ancho de banda de QPSK.** Esta directamente relacionada con la tasa de bits, ya que los datos de entrada se dividen en dos canales, la tasa de bits en el canal I, o en el canal Q, es igual a la mitad de la tasa de datos de entrada ($f_b/2$). En consecuencia, la frecuencia fundamental, más alta, presente en la entrada de datos al modulador balanceado, I o Q, es igual a un cuarto de la tasa de datos de entrada (la mitad de $f_b/2$). Como resultado, la salida de los moduladores balanceados, I y Q, requiere de un mínimo ancho de banda de Nyquist de doble lado, igual a la mitad de la tasa de bits que están entrando. Por lo tanto se tiene $f_N = 2(f_b/4) = f_b/2$ , con QPSK, se realiza una compresión de ancho de banda (el ancho de banda mínimo es menor a la tasa de bits que están entrando).

- **Desplazamiento de 8 fases (8-PSK).** Un modulador de 8-PSK, hay ocho posibles fases de salida, para codificar ocho fases diferentes, los bits que están entrando se consideran en grupos de 3 bits, llamados tribits ($2^3 = 8$).

- **Desplazamiento de 16 fases (16-PSK).** Un modulador de 16-PSK actúa en datos que están entrando en grupos de 4 bits ($2^4$ = 16), llamados quadbits (bits en cuadratura). La fase de salida no cambia, hasta que 4 bits han sido introducidos al modulador. Por tanto, la razón de cambio de salida y el mínimo ancho de banda son iguales a un cuarto de la tasa de bits que están entrando ($f_b/4$).

Generalmente con el propósito de obtener una eficiencia espectral mayor se recurre a métodos de modulación de mayor número de fases, aumentando con ello la complejidad de los procesos de modulación y demodulación.

## 1.5 Modulación de amplitud en cuadratura (QAM)

Es una forma de modulación digital en donde la información digital está contenida, tanto en la amplitud como en la fase de la portadora trasmitida.

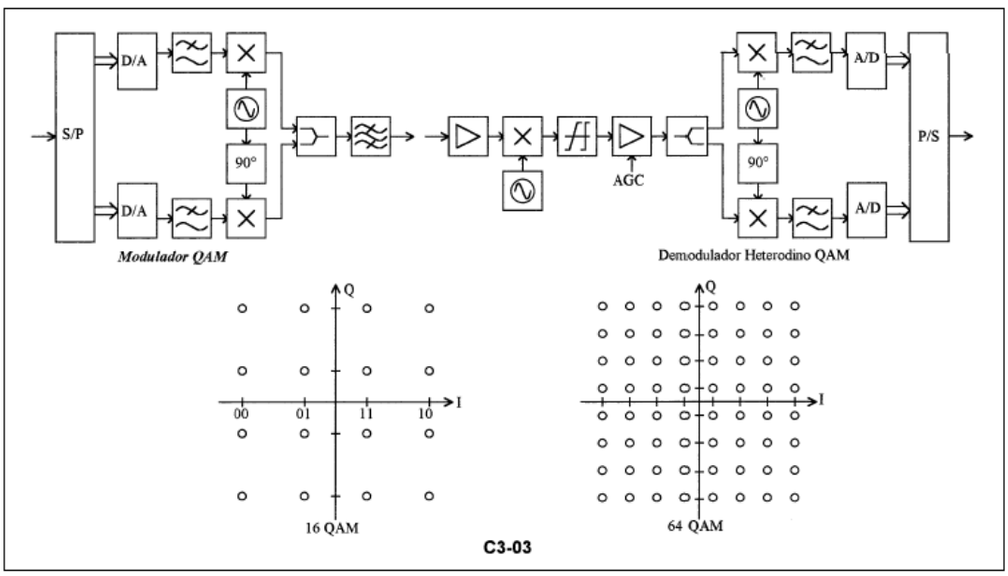



**Fig. 5.1.12 Modulador y demodulador QAM**

- **QAM de 8 fases.** La señal de salida de un modulador de 8-QAM no es una señal de amplitud constante.
    - **Transmisor 8-QAM.** La única diferencia, entre el transmisor de 8-QAM y el transmisor de 8-PSK es la omisión del inversor entre el canal C y el modulador de producto Q.

    - **Receptor 8-QAM.** Un receptor de 8-QAM es casi idéntico al receptor de 8-PSK. Las diferencias son los niveles PAM, en la salida de los detectores de producto, y las señales binarias a la salida de los convertidores análogo a digital, debido a que hay dos amplitudes de transmisión posibles. Con 8-QAM las señales de salida binarias del conversor analógico a digital del canal I, son los bits I y C, y las señales de salida binarias del conversor analógico a digital del canal Q, son los bits Q y C.

- **QAM de 16 y 64 fases.** 16-QAM Actúa sobre los datos de entrada en grupos de cuatro ($2^4$ = l6). Como en 8-QAM, tanto la fase y la amplitud de la portadora transmisora son variados. La virtud de 16-QAM frente a 16-PSK es que las fases se encuentran más separadas una de otra con lo cual admiten una amplitud de ruido mayor.

. Un Transmisor 16-QAM se puede implementar de dos formas: Mediante un codificador apropiado se disponen de 4 trenes de datos en paralelo y se agrupan de a dos para obtener dos señales analógicas con 4 estados de amplitud cada una (-3,-1,+3,+1). A continuación se efectúa la modulación en cuadratura convencional del tipo 4-PSK. La otra es mediante 2 moduladores del tipo 4-PSK se generan 4 estados de fase en cada uno, uno de ellos se lo afecta con una atenuación de 6 dB antes de la suma. La modulación 16-QAM resulta

ser una modulación 4-PSK por cuadrante, donde los cuadrantes se obtienen mediante otra modulación 4-PSK [Tom+, 1996].

Un Transmisor 64-QAM es una extensión del concepto anterior con 6 trenes de datos en paralelo en lugar de 4. Se administran 2 señales analógicas de 8 niveles de amplitud moduladas en cuadratura o se utilizan 3 moduladores 4-PSK con relación de atenuación de 6 y 12 dB. La distribución de códigos a cada fase se realiza siguiendo una codificación cíclica; de tal forma que un error de fase introduce en las fases más cercanas solo un error de bit [Tom+, 1996].

- **Ancho de banda de 8-QAM, 16-QAM y 64-QAM.** Esta directamente relacionada con la tasa de bits en su entrada. En 8-QAM la tasa de bits es un tercio de la tasa binaria de entrada al igual que con 8-PSK. Por tanto, el mínimo ancho de banda requerido para 8-QAM es $f_b/3$, al igual que en el 8-PSK.

  En 16-QAM los datos de entrada se dividen en 4 canales, la tasa de bits en cada canal es igual a un cuarto de la tasa binaria de entrada ($f_b/4$). En 64-QAM los datos de entrada se dividen en 6 canales, la tasa de bits en cada canal es igual a un sexto de la tasa binaria de entrada ($f_b/6$).

## 1.6 Modulación PSK diferencial (DPSK)

En este caso la información se conduce por medio de las transiciones en la fase de la portadora, como un error de desición en el bit presente induce otro error en el siguiente, el desempeño de la DPSK es inferior a la modulación PSK coherente, esta modulación se usa generalmente para resolver el problema de la sincronización en que la información se codifica utilizando las diferencias entre bits en dos intervalos de bit sucesivos. Se genera una secuencia binaria diferencial a partir del mensaje binario de

entrada en el transmisor, esta secuencia tiene un dígito de entrada extra que es arbitrario, los dígitos siguientes en la codificación diferencial se determinan por la regla de que no existe cambio en el estado de salida si esta presente un uno, en caso de un cero existe un cambio de estado de salida [Str+, 1990].

- **Transmisor DPSK.** Un bit de información entrante usará la XNOR con el bit anterior, antes de entrar al modulador de BPSK (modulador balanceado). Para el primer bit de datos, no hay un bit anterior con el cual comparar. Por tanto, se asume un bit de referencia inicial.

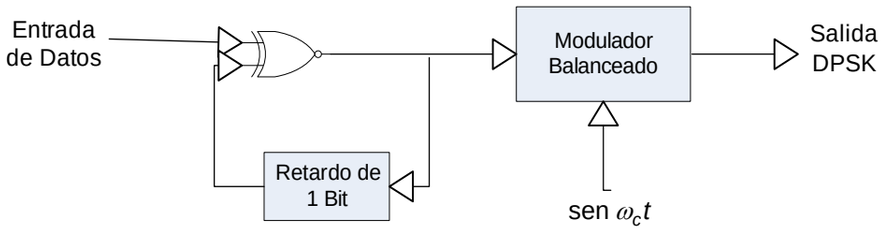



**Fig. 5.1.13 Diagrama de un transmisor DPSK**

- **Receptor DPSK.** La fase del dígito previo sirve como señal de referencia, la señal recibida se retarda por un tiempo de bit, luego si las fases son iguales se genera un 1 lógico (voltaje +), si son diferentes se genera un 0 lógico (voltaje –). Si se supone incorrectamente la fase de referencia, sólo el primer bit demodulado está en error. Una desventaja de la modulación DPSK es que la velocidad de señalización se fija mediante el retardo utilizado, además como la determinación de un bit se hace con base en la señal recibida en dos intervalos de bit sucesivos, lo que induce a que los errores de bit se presenten por pares.

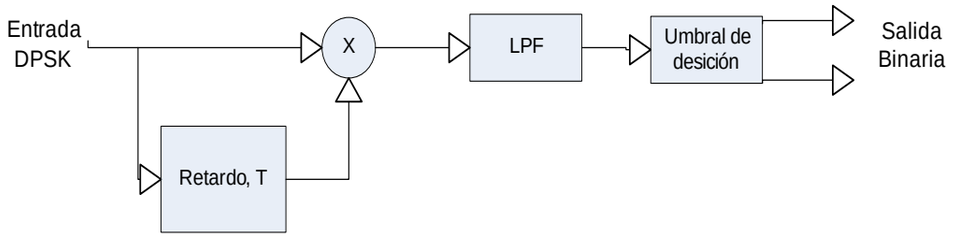



**Fig. 5.1.14 Diagrama de un receptor DPSK**

## 1.7 Características de la modulación digital

Con el propósito de visualizar algunas características que son fundamentales a la hora de elegir algún tipo de modulación digital se define:

- **Rendimiento de error de PSK.** El rendimiento de error de bit para los distintos sistemas de modulación digital multifase está directamente relacionado con la distancia entre puntos en un diagrama de espacio de estado de la señal. La expresión general para la probabilidad de error del bit de un sistema PSK es [Str+, 1990]:

$$P(e) = \frac{1}{\log_2 M} f_{ERROR}(x)$$

$$x = \operatorname{sen}\frac{\pi}{M} * \sqrt{\log_2 M}\sqrt{\frac{E_b}{N_0}}$$

$$\frac{E_b}{N_0} = \frac{C}{N} x \frac{B}{f_b}$$

Donde:

M = es el número de fases

$\frac{E_b}{N_0}$ = es la relación de densidad de potencia de energía por bit a ruido

$\frac{C}{N}$ = es la relación de potencia de portadora a ruido

$\frac{B}{f_b}$ = es la relación del ancho de banda de ruido a la tasa de bits

- **Rendimiento de error de QAM.** Para modulaciones con más de 4 fases, la modulación QAM funcionará mejor que la PSK. Esto se debe a que la distancia, entre dos puntos de señalización en un sistema de PSK, es más pequeña que la distancia entre puntos en un sistema QAM comparable. La expresión general para la probabilidad de error del bit de un sistema QAM es [Str+, 1990]:

$$P(e) = \frac{1}{\log_2 L}\left(\frac{L-1}{L}\right) f_{ERROR}(x)$$

$$x = \sqrt{\frac{\log_2 L}{L-1}}\sqrt{\frac{E_b}{N_0}}$$

Donde:

L = es el número de niveles en cada eje

- **Rendimiento de error de FSK.** La probabilidad de error para los sistemas FSK se evalúa en forma diferente a los PSK y QAM. Hay en esencia sólo dos tipos de sistemas FSK: no coherente (asíncronos) y coherentes (síncronos), con FSK no coherente, el transmisor y el receptor no están sincronizados en frecuencia o fase. Con FSK coherente, las señales de referencia del receptor local están cerradas, en frecuencia y en fase, con las señales transmitidas [Str+, 1990]. La probabilidad de error para FSK no coherente es:

$$P(e) = \frac{1}{2}\exp\left(-\frac{E_b}{2N_0}\right)$$

Y para FSK coherente es:

$$P(e) = f_{ERROR}\left(\sqrt{\frac{E_b}{N_0}}\right)$$

- **Filtrado del canal.** Una particularidad del espectro en la transmisión digital es que en un instante de tiempo todo el espectro transmitido le corresponde al mismo bit (canal de información). En cambio en la transmisión analógica el espectro en cada instante lleva información de cada uno de los canales que componen la multiplexación en frecuencia FDM.

  El espectro de la señal digital antes del modulador es recortado mediante un filtro pasabajos; luego del modulador se filtra mediante un filtro pasabanda. El espectro de la señal de banda base o de frecuencia intermedia consiste en una envolvente del tipo sinc f (sen f/f). El número de armónicas contenidas por la envolvente depende de la periodicidad de la señal. La separación entre ellas corresponde a la inversa del período expresado en segundos.

  Cuando se limita la banda del canal el espectro transmitido se ve truncado y el pulso rectangular se extiende en el tiempo. Una señal rectangular en el tiempo tiene asociado un espectro infinito en la frecuencia; en cambio, un espectro limitado en frecuencia tiene asociada una señal no limitada en el tiempo. Se produce la interferencia intersímbolo ISI producto de la superposición de las "colas" de un pulso sobre adyacentes. La ISI se anula cuando la frecuencia de corte W del filtro es igual a la mitad de la velocidad de transmisión expresada en Hz.

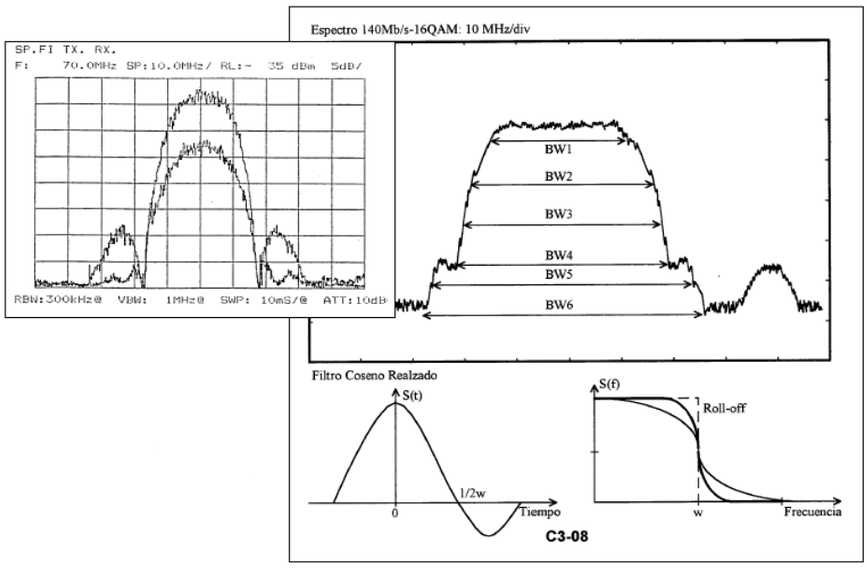



**Fig. 5.1.15 Espectro de modulación digital**

El coeficiente *Roll off* es un parámetro de diseño del filtro. Cuando el Roll off tiende a cero se acorta la banda y se tiende al filtrado ideal. El valor máximo posible es uno. Generalmente se indica el valor del Roll off como un número (típico 0,2 a 0,7) o un porcentaje (20 a 70%). Los filtros en banda base se realizan mediante filtros digitales y en frecuencia intermedia mediante filtros L-C o de Onda Acústica Superficial SAW.

- **Ancho de banda.** Existen varios criterios para la definición del ancho de banda que ocupa un canal.

| Tipo | Sigla | Detalle |
|---|---|---|
| **Ruido** | BW1 | **El ancho de banda equivalente de ruido se trata de un espectro rectángular de ruido con igual valor de potencia que el espectro digital completo** |
| **3 dB** | BW2 | **El ancho de banda a mitad de potencia donde se tiene una atenuación de 3 dB respecto del valor máximo en el centro del espectro** |
| **-50 dB** | BW3 | **El ancho de banda con densidad de potencia delimitada consiste en declarar un umbral entre 35 y 50 dB respecto del máximo en la portadora por debajo del cual se encuentra la densidad de potencia** |

| | | |
|---|---|---|
| **Nyquist** | BW4 | **Corresponde a la frecuencia de corte del filtrado ideal W=Vtx/2** |
| **Nulo** | BW5 | **El ancho de banda al primer punto de anulación del espectro corresponde al primer lóbulo coincidente con 1/T (T es el tiempo de duración del pulso)** |
| 99% | **BW6** | **El ancho de banda que contiene la mayoría de la potencia es, por ejemplo, el 99% de la potencia total. La FCC de USA adopta este criterio** |

**Tabla 5.1.1 Definiciones alternativas de ancho de banda**

De acuerdo con el ancho de banda previsto por el ITU-R en las distintas gamas de frecuencias se requieren los siguientes métodos de modulación.

| Ancho de banda | Sistema de transmisión posible |
|---|---|
| **40 MHz** | **34 Mb/s-4 PSK** |
| **60 MHz (±28 MHz)** | **140 Mb/s-64 QAM y 155 Mb/s-128 TCM** |
| 80 MHz (±40 MHz) | **140 Mb/s-16 QAM;155 Mb/s-64 TCM y 2x155 Mb/s-512 TCM** |

**Tabla 5.1.2 Clasificación de modulaciones de acuerdo al BW**

A continuación se muestra la relación entre baudio y ancho de banda de las distintas formas de FSK, PSK y QAM.

| Modulación | Codificación | BW (Hz) | Baudio | Eficiencia BW (bps/Hz) |
|---|---|---|---|---|
| **FSK** | Bit | $f_b$ | $f_b$ | 1 |
| **BPSK** | Bit | $f_b$ | $f_b$ | 1 |
| **QPSK** | Dibit | $f_b / 2$ | $f_b / 2$ | 2 |
| **8-PSK** | Tribit | $f_b / 3$ | $f_b / 3$ | 3 |
| **8-QAM** | Tribit | $f_b / 3$ | $f_b / 3$ | 3 |
| **16-PSK** | Quadbit | $f_b / 4$ | $f_b / 4$ | 4 |
| **16-QAM** | Quadbit | $f_b / 4$ | $f_b / 4$ | 4 |

**Tabla 5.1.3 Resumen ancho de banda y eficiencia espectral**

**con $f_b$ = Razón de bits de entrada**

- **Eficiencia espectral (Ee).** es el cociente entre la velocidad de transmisión Vtx en b/s y el ancho de banda ocupado en Hz. Como el ancho de banda mínimo teórico es el de Nyquist (las dos bandas laterales hasta Vtx/2 reducido por la modulación multinivel) y se expresa mediante Vtx/K, la Ee es un número independiente de la velocidad de transmisión y solo asociado al método de modulación. El factor K corresponde al número de bits transmitidos en un símbolo. La eficiencia espectral teórica para los métodos de modulación es igual al número de bits por símbolo transmitido. El valor práctico es inferior debido a que la banda ocupada también es superior; el filtrado no es ideal.

- **Relación BER v/s C/N.** En la medida que el número de fases se incrementa la tasa de error BER aumenta con el mismo nivel de ruido. Para un mismo método de modulación en la medida que la relación portadora a ruido C/N disminuye la BER se incrementa. Se denomina Back off a la diferencia entre la potencia de saturación y la potencia de emisión, este valor debe ser suficientemente alto como para no eliminar la modulación de amplitud superpuesta a la de fase. El valor del Back off se incrementa con el número de fases: 2 dB para 4PSK; 6 dB para 16QAM y 8 dB para 64QAM.

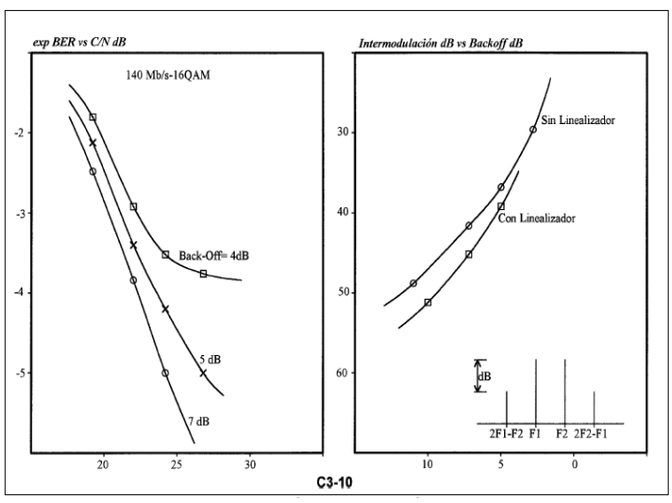



**Fig. 5.1.16 Comparación grafica de rendimientos**

Otra comparación útil es analizar el rendimiento de las modulaciones en relación a la cantidad de información transmitida.

| Modulación | Relación C/N (dB) | Relación $E_b/N_0$ (dB) |
|---|---|---|
| **BPSK** | 10.6 | 10.6 |
| **QPSK** | 13.6 | 10.6 |
| **4-QAM** | 13.6 | 10.6 |
| **8-QAM** | 17.6 | 10.6 |
| **8-PSK** | 18.5 | 14 |
| **16-PSK** | 24.3 | 18.3 |
| **16-QAM** | 20.5 | 14.5 |
| **32-QAM** | 24.4 | 17.4 |
| **64-QAM** | 26.6 | 18.8 |

**Tabla 5.1.4 Comparación de rendimientos (BER = $10^{-6}$)**

Se suele pensar que para obtener una eficiencia espectral mayor se recurre a métodos de modulación de mayor número de fases. Debido a las prestaciones de tasa de error BER en función de la relación portadora a ruido C/N no es conveniente continuar incrementando el número de fases PSK.

# 2. Criptografía

## 2.1 Análisis de sistemas convencionales

### 2.1.1 Algoritmo de sustitución monoalfabético de cesar o de cambio trivial

Este algoritmo consiste en un cifrado de sustitución simple con una permutación obligada c para un cambio de alfabeto a través de k caracteres, para algunos casos k es fijo. Más precisamente se define $e(m_i)$ como un carácter donde $m_i$ es mapeado por e, $m = m_1 m_2 m_3 \ldots$ como un mensaje de texto plano, A = {A, B, …, Z}, y $m_i \in A$. Si $|A| = s$ y $m_i$ esta asociado con un valor entero i con $0 \le i \le s-1$ entonces $c_i = e(m_i) = m_i + k \mod s$. El mapeado de desencriptación esta definido por $d(c_i) = c_i - k \mod s$. Se le conoce como Cesar ya que este utilizaba el algoritmo con una clave K = 3. Este algoritmo es susceptible a un análisis de frecuencia con dicho proceso es posible averiguar la clave fácilmente [Van+, 1996].

### 2.1.2 Algoritmo por sustitución monoalfabético affine

Este algoritmo funciona sobre un alfabeto de 26 letras y esta definido por $e_k(x) = ax + b \mod 26$ donde $0 \le a, b \le 25$. La clave es (a,b) y el texto cifrado esta definido por $c = e_K(x)$ y es desencriptado usando $d_k(c) = (c-b)a^{-1} \mod 26$, con una condición suficiente y necesaria para la invertibilidad, esta es que $gcd(a,26) = 1$. Este algoritmo es susceptible a un análisis de frecuencia con dicho proceso es posible averiguar la clave fácilmente [Van+, 1996].

### 2.1.3 Algoritmo por sustitución monoalfabético playfair

Este algoritmo funciona en base a un diagrama de sustitución y puede ser definido por arreglos de caracteres de un alfabeto de 25 letras (I y J son iguales) en una matriz M de 5x5, los caracteres de texto plano son ajustados por pares. El par $(p_1, p_2)$ es reemplazado por el diagrama $(c_3, c_4)$ y así sucesivamente. Si $p_1$ y $p_2$ están en distintas filas y columnas, estos definen el limite de una submatriz (posiblemente dentro de M), con la permanencia en los limites de c3 y c4; c3 es definido como el carácter en la misma columna que p1. Si $p_1$ y $p_2$ tienen fila en común, c3 es definido como el carácter inmediatamente a la derecha de p1 y c4 definido por el carácter inmediatamente a la derecha de p2 (la primera columna es vista como el comienzo a la derecha del final). Si $p_1$ y $p_2$ están en la misma columna, los caracteres que están inmediatamente abajo son c3 y c4. Si $p_1 = p_2$ es poco frecuente, pero debe insertarse una carácter de texto plano entre ellos y el texto plano es reagrupado. La clave esta definida perpendicularmente en la matriz y es de 5x5, una ayuda mnemónica puede ser usada para recordar más fácilmente el ángulo recto que define la clave. Este algoritmo no es susceptible a un análisis de frecuencia por letra, pero criptoanálisis basados en diagrama de frecuencias pueden romperlo [Van+, 1996].

### 2.1.4 Algoritmo por sustitución monoalfabético hill

Este algoritmo funciona en base a un diagrama de sustitución, un n diagrama puede ser definido usando una matriz invertible $A = a_{ij}$ de nxn, así también la clave, el mapa y el texto plano $m_1 ... m_n$ serán de n caracteres. Para un diagrama n de cifrado de texto tenemos $c_i = \sum_{j=1}^{n} a_{ij} m_j \ , i = 1, ..., n$. Para desencriptar se debe usar $A^{-1}$.

Este algoritmo no es susceptible a un análisis de frecuencia por letra, pero criptoanálisis basados en diagrama de frecuencias pueden romperlo [Van+, 1996].

### 2.1.5 Algoritmo de transposición simple

Este algoritmo esta determinado por un cifrado con un periodo fijo t, la encriptación involucra un agrupamiento de texto plano en bloques de t caracteres, y aplicando para cada bloque una única permutación c sobre todos los caracteres del bloque. Más precisamente, el texto cifrado correspondiente a un bloque de texto plano $m = m_1 \ldots m_t$ es $c = E_c(m) = m_{c(1)} \ldots m_{c(t)}$. La clave de encriptación es c, con t definido implícitamente, el espacio de claves tiene cardinalidad **t!** para un determinado valor de t. La desencriptación implica el uso de una permutación d invertida de c. Este algoritmo es susceptible a un análisis de frecuencia con dicho proceso es posible averiguar la clave fácilmente.

### 2.1.6 Algoritmo por sustitución polialfabético de vigenère

Existen diferentes variantes de este algoritmo, así que se definirá solo un vigenére simple de periodo t sobre un alfabeto de s caracteres, involucra claves $k_1 k_2 \ldots k_t$ de t caracteres. El mapeado de texto plano $m = m_1 m_2 \ldots$ para texto cifrado $c = c_1 c_2 \ldots$ es definido sobre caracteres individuales por $c_i = m_i + k_i \mod s$ donde el subíndice i en $k_i$ es tomado del módulo t (la clave es re-usada). Se usan t cambios de cifrado, definiendo t valores de cambio para $k_i$, especificando una de s substituciones (Monoalfabético); $k_i$ es usada sobre caracteres en las posiciones $i, i+s, i+2s, \ldots$. En general cada sustitución es diferente y es equivalente a usar t alfabetos [Van+, 1996].

La principal característica del cifrado de Vigenère es que una misma letra se codifica con símbolos distintos, lo que imposibilita un análisis de frecuencias. Pero sin embargo cuando se introdujo un análisis probabilístico, específicamente un método llamado índice de coincidencias y consiste en la probabilidad de que sacadas dos letras al azar de un texto sea la misma, a partir de un texto cifrado se calcula $I_t = \frac{1}{n(n-1)} \sum_{i=1}^{26} n_i(n_i - 1)$ , donde n es el número de caracteres en el texto y $n_i$ el número de apariciones de la letra número i. Si se sabe el lenguaje de implementación dicho valor deberá coincidir con el teórico dado por $I = \sum_{i=1}^{26} p_i^{\,2}$ , donde $p_i$ es la probabilidad de aparición de cada letra, calculada a partir de la tabla de frecuencias del lenguaje.

### 2.1.7 Algoritmo por sustitución polialfabético – cilindro de jefferson

Este cilindro es posible adaptarlo en un software y consiste en un cilindro de 6 pulgadas de largo formado por 36 discos con una barra insertada a través de los ejes del cilindro para hacerlo girar, la periferia de cada disco es dividida en 26 partes, sobre cada disco son escritas letras de la A-Z en orden aleatorio. Mensajes de texto plano son encriptados en bloques de 36 caracteres, una barra de referencia es colocada a lo largo del cilindro, cada uno de las 36 ruedas gira individualmente para traer el carácter apropiado a la posición a lo largo de la línea de referencia, donde las otras 25 posiciones paralelas referencian y definen el texto cifrado. Para desencriptar hay que rotar cada uno de los 36 discos para obtener caracteres a lo largo de la línea de referencia fija que hace juego con el texto cifrado. Las otras 25 posiciones de referencia son examinadas para la reorganización del texto plano, para que el mensaje original sea reconocible las partes deben estar previamente de acuerdo sobre los

índices de 1 a 25 especificando el offset entre líneas de texto plano y texto cifrado [Van+, 1996].

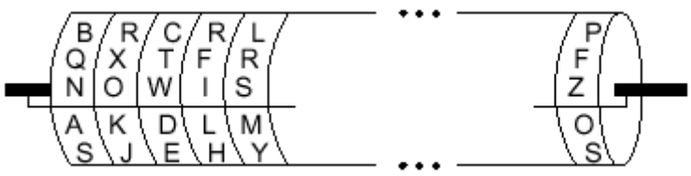



**Fig. 5.2.1 Cilindro de jefferson**

### 2.1.8 Algoritmo por sustitución polialfabético – rotores

Consiste de un número de rotores, cada uno implementando una substitución Monoalfabetica fija diferente, mapeando un carácter en su cara de entrada obteniendo una salida en su cara de salida. Un carácter de texto plano de entrada para el primer rotor genera una salida que es la entrada para el segundo rotor y así sucesivamente hasta que salga el carácter cifrado del último rotor. Dado que los rotores tienen posiciones fijas el conjunto de rotores implementa una sustitución Monoalfabetica, lo cual es una composición de sustituciones definida por cada rotor.

Cada rotor $R_i$ efectúa una sustitución Monoalfabetica $f_i$, $R_i$ puede rotar en $t_i$ posiciones ($t_i$ = 26). Con el offset j puesto a una referencia fija, $R_i$ mapea una entrada a en $f_i(a-j)+j$ , donde ambos la entrada en $f_i$ y la salida final son reducidas a módulo 26. La clave de cifrado es definida por sustituciones monoalfabeticas determinadas por la posición inicial de los rotores y una rotación fija. Reordenando los rotores se puede obtener una variabilidad adicional [Van+, 1996]. Un ejemplo clásico de una maquina que usa rotores es la inventada por el alemán Scherbius llamada Enigma.

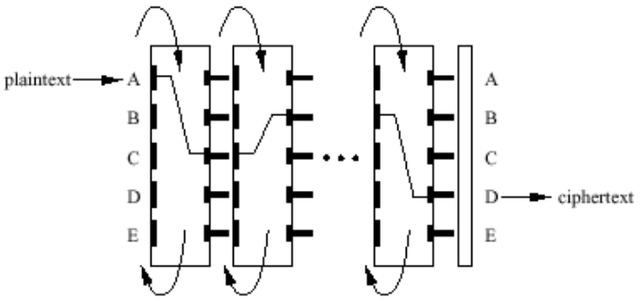



**Fig. 5.2.2 Máquina basada en rotores**

# 2.2 Análisis de algoritmos simétricos

### 2.2.1 Cifrado por bloques (block cipher)

#### 2.2.1.1 Modos de operación del cifrado por bloques

- ECB (Electronic CodeBook).

  *Algoritmo ECB*

  Entrada:

  - claves K de k bits
  - bloques de texto plano $x_1,\ldots,x_t$ de n bits

  Salida:

  - bloques de texto cifrado $c_1,\ldots,c_t$ , entonces.

  Encriptación:

  - *for* $1 \le j \le t$

    1. $c_j \leftarrow E_K(X_j)$

  Desencriptación:

  - *for* $1 \le j \le t$

    1. $x_j \leftarrow E_K^{-1}(c_j)$

Propiedades:

1. Bloques de texto plano idénticos dan como resultado bloques de texto cifrado idéntico.
2. Asociación de dependencias: los bloques son encriptados independientemente de los demás bloques, reordenando bloques de texto cifrado resulta una correspondencia a reordenar bloques de texto plano.
3. Error de propagación: uno o más bits erróneos dentro de un bloque de texto cifrado $c_j$ afectan la desencriptación de solo ese bloque.

- CBC (Cipher-Block Chaining).

*Algoritmo CBC*

Entrada:

- claves K de k bits
- un vector de iniciación de n bits (IV)
- bloques de texto plano $x_1,\ldots,x_t$ de n bits

Salida:

- son bloques de texto cifrado $c_1,\ldots,c_t$ , entonces.

Encriptación:

- $c_0 \leftarrow IV$
- $for\ 1 \leq j \leq t$

    1. $c_j \leftarrow E_K(c_{j-1} \oplus x_j)$

Desencriptación:

- $c_0 \leftarrow IV$
- $for\ 1 \leq j \leq t$

1. $x_j \leftarrow c_{j-1} \oplus E_K^{-1}(c_j)$

Propiedades:

1. Textos planos idénticos: resultan en idénticos bloques de texto cifrado, cuando el mismo texto plano es encriptado bajo la misma clave y IV. Cambiar el IV, la clave o el primer bloque de texto plano resulta en bloques de texto cifrado diferentes.
2. Asociación de dependencias: el mecanismo de asociación origina texto cifrado $c_j$ para depender de $x_j$ y todas precedidas de bloques de texto plano. Consecuentemente reacomodando el orden de los bloques de texto cifrado afectan la desencriptación.
3. Error de propagación: un único error en un bit en un bloque de texto cifrado $c_j$ afecta la desencriptación de los bloques $c_j$ y $c_{j+1}$.
4. Error de recuperación: este modo esta sincronizado consigo mismo o tiene auto claves en la detección de que si un error ocurre en un bloque $c_j$ pero no en $c_{j+1}$, $c_{j+2}$ es correctamente desencriptado para $x_{j+2}$.

- CFB (Cipher Feedback).

*Algoritmo CFB*

Entrada:

- claves K de k bits
- un vector de iniciación de n bits (IV)
- bloques de texto plano $x_1,...,x_u$ $(1 \leq r \leq n)$ de r bits

Salida:

- son bloques de texto cifrado $c_1,\ldots,c_u$ de r bits, entonces.

Encriptación:

- $I_1 \leftarrow IV$ ($I_j$ es un valor de entrada en un cambio de registro)
- *for* $1 \leq j \leq u$
    1. $O_j \leftarrow E_K(I_j)$ (Calcula el bloque cifrado de salida)
    2. $t_j \leftarrow$ El r bits mas a la izquierda de $O_j$ (asume el más a la izquierda si esta identificado con un bit 1)
    3. $c_j \leftarrow x_j \oplus t_j$ (transmite el bit r del bloque de texto cifrado $c_j$)
    4. $I_{j+1} \leftarrow 2^r \bullet I_j + c_j \bmod 2^n$ (cambia $c_j$ al final de la derecha en un cambio de registro)

Desencriptación:

- $I_1 \leftarrow IV$
- *for* $1 \leq j \leq u$, bajo recibir $c_j$
    1. $x_j \leftarrow c_j \oplus t_j$ , donde $t_j$, $O_j$ y $I_j$ están calculadas como arriba.

Propiedades:

1. Textos Planos idénticos: como a través de la encriptación CBC cambiando el IV resulta el mismo texto plano de entrada a ser encriptado en una salida diferente. No es necesario que el IV sea secreto.
2. Asociación de dependencias: similar a la encriptación CBC, el mecanismo de asociación causa bloques de texto cifrado $c_j$ para

depender de ambos $x_j$ y precediendo bloques de texto plano, consecuentemente reordenando bloques de texto cifrado afecta la desencriptación. Una desencriptación adecuada de un bloque de texto cifrado correcto requiere la precedencia de [n/r] bloques de texto cifrado para que sea correcto.

3. Error de propagación: uno o más bit erróneos en algún r bit de un bloque de texto cifrado $c_j$ afecta la desencriptación de el y de los [n/r] bloques de texto cifrado siguientes.
4. Error de recuperación: esta sincronizado consigo mismo similar a CBC, pero requiere de [n/r] bloques de texto cifrado para recuperar.
5. Rendimiento: para r<n, el rendimiento esta reducido al factor de n/r, en eso cada ejecución de E rinde solo r bits de texto cifrado de salida.

- OFB (Output Feedback).

  *Algoritmo de ISO 10116 (con realimentación completa)*

  Entrada:

  - claves K de k bits
  - un vector de iniciación de n bits (IV)
  - bloques de texto plano $x_1,\ldots,x_u$ $(1 \le r \le n)$ de r bits

  Salida:

  - son bloques de texto cifrado $c_1,\ldots,c_u$ de r bits, entonces.

  Encriptación:

  - $I_1 \leftarrow IV$
  - $for\ 1 \le j \le u$ , determinado bloques de texto plano $x_j$

1. $O_j \leftarrow E_K(I_j)$ (Calcula el bloque cifrado de salida)
2. $t_j \leftarrow$ El r bits mas a la izquierda de $O_j$ (asume el más a la izquierda si esta identificado con un bit 1)
3. $c_j \leftarrow x_j \oplus t_j$ (transmite el bit r del bloque de texto cifrado $c_j$)
4. $I_{j+1} \leftarrow O_j$ (Actualiza el bloque cifrado de entrada para el siguiente bloque)

Desencriptación:

- $I_1 \leftarrow IV$
- *for* $1 \leq j \leq u$, bajo recibir $c_j$
    1. $x_j \leftarrow c_j \oplus t_j$ , donde $t_j$, $O_j$ y $I_j$ están calculadas como arriba.

*Algoritmo FIPS (con realimentación de r bits)*

Entrada:

- claves K de k bits
- un vector de iniciación de n bits (IV)
- bloques de texto plano $x_1,\ldots,x_u$ $(1 \leq r \leq n)$ de r bits

Salida:

- son bloques de texto cifrado $c_1,\ldots,c_u$ de r bits, entonces.

Encriptación:

- $I_1 \leftarrow IV$
- *for* $1 \leq j \leq u$ , determinado bloques de texto plano $x_j$

1. $O_j \leftarrow E_K(I_j)$ (Calcula el bloque cifrado de salida)
2. $t_j \leftarrow$ El r bits mas a la izquierda de $O_j$ (asume el más a la izquierda si esta identificado con un bit 1)
3. $c_j \leftarrow x_j \oplus t_j$ (transmite el bit r del bloque de texto cifrado $c_j$)
4. $I_{j+1} \leftarrow 2^r \bullet I_j + t_j \bmod 2^n$ (cambia la salida $t_j$ al final de la derecha en un cambio de registro)

Desencriptación:

- $I_1 \leftarrow IV$
- *for* $1 \leq j \leq u$, bajo recibir $c_j$
    1. $x_j \leftarrow c_j \oplus t_j$ , donde $t_j$, $O_j$ y $I_j$ están calculadas como arriba.

Propiedades:

1. Textos Planos idénticos: como a través de CBC y CFB cambiando el IV resulta el mismo texto plano de entrada a ser encriptado en una salida diferente.
2. Asociación de dependencias: el flujo de claves es independiente del texto plano.
3. Error de propagación: uno o más bit erróneos en algún bloque de texto cifrado $c_j$ afecta la desencriptación solo ese bloque.
4. Error de recuperación: recupera errores de bits desde textos cifrados, pero no puede estar sincronizado consigo mismo después de la

pérdida de bits cifrados, lo cual destruye la alineación del flujo de claves para desencriptar.

5. Rendimiento: para r<n el rendimiento puede verse disminuido. Sin embargo en todos los casos, dado que el conjunto de claves es independiente del texto plano y del texto cifrado, esto puede ser calculado antes.

### 2.2.1.2 Cifrado de fiestel

Es un cifrado iterativo mapeado en un texto plano $(L_0, R_0)$ de 2t bit, para bloques $L_0$ y $R_0$ de t bit, para un texto cifrado $(L_r, R_r)$, a través de r ciclos donde $r \geq 1$. Para $1 \leq i \leq r$, el ciclo $i$ $(L_{i-1}, R_{i-1}) \overset{K_i}{\rightarrow} (L_i, R_i)$ es trazado como sigue: $L_i = R_{i-1}, R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$, donde cada subclave $K_i$ es derivada de la clave de cifrado $K$. Generalmente en un cifrado de Fiestel, $r \geq 3$ a menudo es igual. Específicamente la estructura de Fiestel ordena la salida de texto cifrado como $(R_r, L_r)$ antes que $(L_r, R_r)$, los bloques son intercambiados desde su orden usual después del último ciclo. En consecuencia el desencriptado es conseguido usando los mismos procesos de r ciclos pero con subclaves usadas en orden inverso $K_r$ a $K_1$. La función $f$ del cifrado de Fiestel puede ser producto del cifrado, aunque $f$ necesita de si misma, no es invertible para permitir la inversión del cifrado de Fiestel [Van+, 1996].

### 2.2.1.3 Algoritmo DES (Data Encryption Standard)

Es un cifrado de Fiestel que procesa bloques de texto plano de 64 bits, produciendo bloques de texto cifrado de 64 bits. El tamaño efectivo de la clave secreta

K es 56 bits, más precisamente, la clave K de entrada es especificada como una clave de 64 bits, de los cuales 8 bits pueden ser usados como bits de paridad (bits 8, 16, …, 64) [ANS+, 1983].

### 2.2.1.4 Algoritmo FEAL (Fast Data Encipherment)

Es una familia de algoritmos en la cual juega un rol crítico el desarrollo y refinamiento de varios avances técnicos de criptoanálisis, incluyendo criptoanálisis lineal y diferencial. FEAL-N mapea 64 bits de texto plano para bloques de texto cifrado de 64 bits sobre una clave secreta de 64 bits, esto son N-ciclos del cifrado de Fiestel muy similar a DES, pero con una lejana simpleza de la función f y aumentada para los estados iniciales y finales con un XOR para ambas partes de datos, así como también un XOR directo de las subclaves sobre ambas partes de datos. FEAL fue diseñado por su velocidad y simpleza, especialmente para software sobre microprocesadores de 8 bit, en este caso detallaremos el FEAL mas utilizado que es el de 8 ciclos, la función $f(A,Y)$ mapea un par de entrada de 32X16 bits para una salida de 32 bits. Dentro de la función f dos sustituciones de datos orientadas al byte $S_0$ y $S_1$ (S-cajas) son usadas dos veces, cada una mapea un par de 8 bit de entrada para una salida de 8 bit [Van+, 1996].

| | $U \leftarrow f(A,Y)$ | $U \leftarrow f_K(A,B)$ |
|---|---|---|
| $t_1 =$ | $(A_0 \oplus A_1) \oplus Y_0$ | $A_0 \oplus A_1$ |
| $t_2 =$ | $(A_2 \oplus A_3) \oplus Y_1$ | $A_2 \oplus A_3$ |
| $U_1 =$ | $S_1(t_1, t_2)$ | $S_1(t_1, t_2 \oplus B_0)$ |
| $U_2 =$ | $S_0(t_2, U_1)$ | $S_0(t_2, U_1 \oplus B_1)$ |
| $U_0 =$ | $S_0(A_0, U_1)$ | $S_0(A_0, U_1 \oplus B_2)$ |
| $U_3 =$ | $S_1(A_3, U_2)$ | $S_1(A_3, U_2 \oplus B_3)$ |

**Tabla 5.2.1 Salida $U = (U_0, U_1, U_2, U_3)$ , donde $A_i, B_i, Y_i, t_i, U_i$ son variables de 8 bit para las funciones FEAL f y $f_k$**

$S_0$ y $S_1$ agregan un bit $d \in \{0,1\}$ para los argumentos x e y de 8 bit, ignorar el traslado del bit superior y rotar a la izquierda el resultado en 2 bit (ROT2) $S_d(x,y) = ROT2(x + y + d \mod 256)$. El programa de claves usa la función $f_k(A,B)$ similar a la función f mapeando dos entradas de 32 bit para una salida de 32 bit. Como las operaciones de rotación de 2 bit y el XOR son ambas lineales, solamente la operación elemental no lineal en FEAL es suma módulo 256.

*Algoritmo FEAL-8*

Entrada:

- Texto plano $M = m_1 ... m_{64}$ de 64 bit
- Claves $K = k_1 ... k_{64}$ de 64 bits

Salida:

- son bloques de texto cifrado $C = c_1 ... c_{64}$ de 64 bits

Encriptación:

- Programa de claves, 16 subclaves $K_i$ de 16 bits desde K
- $M_L = m_1 ... m_{32}$
- $M_R = m_{33} ... m_{64}$
- $(L_0, R_0) \leftarrow (M_L, M_R) \oplus ((K_8, K_9), (K_{10}, K_{11}))$, XOR con subclaves iniciales.
- $R_0 \leftarrow R_0 \oplus L_0$
- *for* $i = 1$ *to* 8
    1. $L_i \leftarrow R_{i-1}$

2. $R_i \leftarrow L_{i-1} \oplus f(R_{i-1}, K_{i-1})$, usa tabla para $f(A,Y)$ con $A = R_{i-1} = (A_0, A_1, A_2, A_3)$ y $Y = K_{i-1} = (Y_0, Y_1)$.

- $L_8 \leftarrow L_8 \oplus R_8$
- $(R_8, L_8) \leftarrow (R_8, L_8) \oplus ((K_{12}, K_{13}), (K_{14}, K_{15}))$, XOR con subclaves finales.
- $C \leftarrow (R_8, L_8)$, el orden del bloque final es intercambiado.

Desencriptación:

- La desencriptación puede ser lograda usando el mismo algoritmo que se usa para encriptar, con la misma clave K y el texto cifrado $C = (R_8, L_8)$ como el texto plano de entrada M, pero con el programa de claves inverso. Más específicamente, subclaves $((K_{12}, K_{13}), (K_{14}, K_{15}))$ son usadas para el XOR inicial y $((K_8, K_9), (K_{10}, K_{11}))$ para el XOR final, y los ciclos de claves son usadas desde $K_7$ a $K_0$, es decir en reverso. Esto es directamente análogo a la desencriptación de DES.

*Algoritmo Generador de Claves FEAL-8*

Entrada:

- Claves $K = k_1 ... k_{64}$ de 64 bits

Salida:

- Clave extendida de 256 bit (subclaves $K_i$ de 16 bit, $0 \le i \le 15$)

Proceso:

- $U^{(-2)} \leftarrow 0$
- $U^{(-1)} \leftarrow k_1 ... k_{32}$

- $U^{(0)} \leftarrow k_{33}...k_{64}$
- $U \overset{def}{=} (U_0, U_1, U_2, U_3)$ para $U_i$ de 8 bit
- *for* $i = 1$ *to* $8$, calcular $K_0,...,K_{15}$

    1. $U \leftarrow f_K(U^{(i-2)}, U^{(i-1)} \oplus U^{(i-3)})$, usar $f_K$ de la tabla, donde A y B denotan vectores de 4 bytes $(A_0, A_1, A_2, A_3), (B_0, B_1, B_2, B_3)$ .
    2. $K_{2i-2} = (U_0, U_1)$
    3. $K_{2i-1} = (U_2, U_3)$
    4. $U^{(i)} \leftarrow U$

**FEAL-N.** Es una generalización del algoritmo y usando una clave de 64 bit puede ser generalizado para N-ciclos, donde $N = 2^x$ , es recomendado x=3, es decir FEAL-8. FEAL-N usa N+8 subclaves de 16 bit $K_0,...,K_{N-1}$, respectivamente en el ciclo i se tiene $K_N,...,K_{N+3}$ para el XOR inicial y $K_{N+4},...,K_{N+7}$ para el XOR final. El algoritmo del programa de claves es directamente generalizado para calcular claves desde $K_0$ a $K_{N+7}$, como i va desde 1 a $(N/2)+4$.

**FEAL-NX.** Es un FEAL-N extendido para usar una clave de 128 bit, para lo cual se debe alterar el programa de claves. La clave es cortada en dos partes $(K_L, K_R)$ de 64 bit, $K_R$ es particionada en dos partes $(K_{R1}, K_{R2})$ de 32 bit. Los ciclos están limitados por $1 \le i \le (N/2)+4$, se define $Q_i = K_{R1} \oplus K_{R2}$ para $i \equiv 1 \mod 3$, $Q_i = K_{R1}$ para $i \equiv 2 \mod 3$ y $Q_i = K_{R2}$ para $i \equiv 0 \mod 3$. El segundo argumento $(U^{(i-1)} \oplus U^{(i-3)})$ para $f_K$ en el paso 1 del algoritmo es remplazado por

$(U^{(i-1)} \oplus U^{(i-3)} \oplus Q_i)$. Para $K_R = 0$ FEAL-NX es igual a FEAL-N con $K_L$ como la clave K de 64 bit de FEAL-N.

### 2.2.1.5 Algoritmo IDEA (International Data Encryption)

Este algoritmo encripta texto plano de 64 bit para bloques de texto cifrado de 64 bit usando una clave de entrada de 128 bit, basado en parte sobre la generalización de la estructura de Fiestel y consiste de 8 ciclos de calculo idéntico seguido de una transformación en la salida. El ciclo r usa 6 subclaves $K_i^{(r)}$ de 16 bit, con $1 \le i \le 6$, para transformar una entrada X de 64 bit en una salida de 4 bloques de 16 bit, lo cual es la entrada para el siguiente ciclo. La salida del ciclo 8 es la salida de la transformación, empleando 4 subclaves adicionales $K_i^9$ con $1 \le i \le 4$ para producir el texto cifrado final $Y = (Y_1, Y_2, Y_3, Y_4)$, todas las subclaves son derivadas desde K.

Un diseño dominante del concepto de IDEA es la mezcla de operaciones de tres diferentes grupos algebraicos de $2^n$ elementos. El correspondiente grupo de operaciones sobre los sub bloques a y b de largo en bit de n=16 son trabajadas con un XOR en el ámbito del bits $(a \oplus b)$, sumando $\bmod 2^n$ quedando $(a+b)\ AND\ 0xFFFF$, denotado por a$\boxplus$b, y la multiplicación (modificada) por $\bmod 2^n + 1$ con $0 \in Z_{2^n}$, asociado con $2^n \in Z_{2^n+1}$, denotado por a$\odot$b [Van+, 1996].

**Operación a$\odot$b.** Es una multiplicación modificada, específicamente es una multiplicación módulo $2^{16}+1$ donde a y b son enteros de 16 bit sin signo, con $0 \in Z_{2^{16}}$ esta asociado con $2^{16} \in Z^*_{2^{16}+1}$, es decir, si a=0 ó b=0 remplazar esto por $2^{16}$ (lo cual es $\equiv -1 \bmod 2^{16}+1$) previo a la multiplicación modular y si el resultado es

$2^{16}$ remplazar esto por 0. Así $\odot$ mapea dos entradas de 16 bit para una salida de 16 bit. El seudo código para $\odot$ es el siguiente.

- Para un entero sin signo c de 32 bit
- *if* $(a \equiv 0)$

  $r \leftarrow (0x10001 - b)$ (desde $2^{16} b \equiv -b$)

  *else if* $(b \equiv 0)$

  $r \leftarrow (0x10001 - a)$ (de similar razonamiento)

  *else*

  $c \leftarrow ab$

  $r \leftarrow ((c \ AND \ 0xFFFF) - (c >> 16))$

  *if* $(r < 0)$

  $r \leftarrow (0x10001 + r)$
- Con un valor de retorno $(r \ AND \ 0xFFFF)$ en los 3 casos

**Operación** $ab \mod 2^n + 1$**.** Esta multiplicación puede ser eficientemente implementada de la siguiente forma. Para $0 \le a, b \le 2^{16}$ dejar $c = ab = c_O \bullet 2^{32} + c_H \bullet 2^{16} + c_L$ donde $c_O \in \{0,1\}$ y $0 \le c_L, c_H < 2^{16}$. Para calcular $c' = c \mod (2^{16} + 1)$ primero hay que obtener $c_L$ y $c_H$ por una multiplicación estándar. Para $a = b = 2^{16}$ notar que $c_O = 1, c_L = c_H = 0$ y $c' = (-1)(-1) = 1$ desde que $2^{16} \equiv -1 \mod (2^{16} + 1)$, en otro caso $c_O = 0$. Consecuentemente $c' = c_L - c_H + c_O$ si $c_L \ge c_H$ mientras que $c' = c_L - c_H + (2^{16} + 1)$ si $c_L < c_H$ (desde entonces $-2^{16} < c_L - c_H < 0$).

*Algoritmo IDEA*

Entrada:

- Texto plano $M = m_1 ... m_{64}$ de 64 bit
- Claves $K = k_1 ... k_{128}$ de 128 bits

Salida:

- Bloques de texto cifrado $Y = (Y_1, Y_2, Y_3, Y_4)$ de 64 bits

Encriptación:

- Programa de claves, subclaves $K_1^{(r)}, ..., K_6^{(r)}$ de 16 bits para ciclos entre $1 \le r \le 8$, y $K_1^{(9)}, ..., K_4^{(9)}$ para la transformación de la salida.
- $(X_1, X_2, X_3, X_4) \leftarrow (m_1 ... m_{16}, m_{17} ... m_{32}, m_{33} ... m_{48}, m_{49} ... m_{64})$ dond e $X_i$ es un deposito de datos de 16 bit
- *for* $r = 1$ *to* $8$

1. $X_1 \leftarrow X_1 \odot X_1^r$
2. $X_4 \leftarrow X_4 \odot X_4^r$
3. $X_2 \leftarrow X_2 \boxplus X_2^r$
4. $X_3 \leftarrow X_3 \boxplus X_3^r$
5. $t_0 \leftarrow X_5^r \odot (X_1 \oplus X_3)$
6. $t_1 \leftarrow X_6^r \odot (t_0 \boxplus (X_2 \oplus X_4))$
7. $t_2 \leftarrow t_0 \boxplus t_1$
8. $X_1 \leftarrow X_1 \oplus t_1$
9. $X_4 \leftarrow X_4 \oplus t_2$
10. $a \leftarrow X_2 \oplus t_2$

11. $X_2 \leftarrow X_3 \oplus t_1$

12. $X_3 \leftarrow a$

- Transformación de salida $Y_1 \leftarrow X_1 \odot K_1^{(9)}$
- $Y_4 \leftarrow X_4 \odot K_4^{(9)}$
- $Y_2 \leftarrow X_3 \boxplus K_2^{(9)}$
- $Y_3 \leftarrow X_2 \boxplus K_3^{(9)}$

Desencriptación:

- Se logra utilizando el mismo algoritmo de encriptación con el texto cifrado Y provisto como entrada M, y la misma clave de encriptación K, pero con cambios para el programa de claves. Primero usar K para obtener todas las subclaves $K_i^{(r)}$, después de esto calcular las subclaves $K_i^{'(r)}$ por la tabla, entonces usar $K_i^{'(r)}$ en reemplazo de $K_i^{(r)}$ en el algoritmo. En la tabla $-K_i$ denota la adición inversa $(\text{mod } 2^{16})$ de $K_i$, el entero $u = (2^{16} - K_i)$ $AND$ $0xFFFF$ con $0 \le u \le 2^{16} - 1$. $K_i^{-1}$ denota el inverso multiplicativo $(\text{mod } 2^{16} + 1)$ de $K_i$, también en $\{0,1,...,2^{16} - 1\}$ derivable por el algoritmo euclediano extendido, con entradas $a \ge b \ge 0$ retorna enteros x e y tal que $ax + by = gcd(a,b)$. Usando $a = 2^{16} + 1$ y $b = K_i$, el gcd es siempre 1 (excepto para $K_i = 0$, tratado separadamente) y así $K_i^{-1} = y$ o $2^{16} + 1 + y$ si $y < 0$. Con $K_i = 0$ esta entrada es mapeada para $2^{16}$ (donde la inversa es

definida por $K_i \odot K_i^{-1} = 1$) y $(2^{16})^{-1} = 2^{16}$ es entonces definida para obtener $K_i^{-1} = 0$.

| round $r$ | $K'^{(r)}_1$ | $K'^{(r)}_2$ | $K'^{(r)}_3$ | $K'^{(r)}_4$ | $K'^{(r)}_5$ | $K'^{(r)}_6$ |
|---|---|---|---|---|---|---|
| $r = 1$ | $(K_1^{(10-r)})^{-1}$ | $-K_2^{(10-r)}$ | $-K_3^{(10-r)}$ | $(K_4^{(10-r)})^{-1}$ | $K_5^{(9-r)}$ | $K_6^{(9-r)}$ |
| $2 \le r \le 8$ | $(K_1^{(10-r)})^{-1}$ | $-K_3^{(10-r)}$ | $-K_2^{(10-r)}$ | $(K_4^{(10-r)})^{-1}$ | $K_5^{(9-r)}$ | $K_6^{(9-r)}$ |
| $r = 9$ | $(K_1^{(10-r)})^{-1}$ | $-K_2^{(10-r)}$ | $-K_3^{(10-r)}$ | $(K_4^{(10-r)})^{-1}$ | — | — |

**Tabla 5.2.2 Subclaves de desencriptación $K_i^{'(r)}$ derivadas de las subclaves de encriptación $K_i^{(r)}$**

*Algoritmo Generador de Claves IDEA*

Entrada:

- Claves $K = k_1 ... k_{128}$ de 128 bits

Salida:

- 52 sub bloques $K_i^{(r)}$ de claves de 16 bit, para 8 ciclos r y la transformación de salida.

Proceso:

- Ordenar las subclaves $K_1^{(1)} ... K_6^{(1)}, K_1^{(2)} ... K_6^{(2)}, ..., K_1^{(8)} ... K_6^{(8)}, K_1^{(9)} ... K_4^{(9)}$
- Partir K en 8 bloques de 16 bit, asignar estos directamente a las primeras 8 subclaves.
- Hacer lo siguiente hasta que las 52 subclaves sean asignadas, cambios cíclicos a la izquierda de K en 25 bit, partiendo el resultado en 8 bloques y asignándolos a las 8 subclaves siguientes.

### 2.2.1.6 Algoritmo SAFER (Secure And Fast Encryption Routine)

Es un cifrado por bloques iterativo con bloques de texto plano de 64 bit y bloques de texto cifrado de 64 bit, con una clave de cifrado de 64 bit, a lo que se denomina SAFER K-64. Consiste en r ciclos idénticos seguido por una transformación en la salida, la recomendación original fue de 6 ciclos seguida por una recomendación que adopto una ligera modificación en el programa de claves y para usar 8 ciclos (de un máximo de r=10). Ambos programas de claves expandían la clave externa de 64 bit en $2r+1$ subclaves cada una de 64 bit (dos para cada ciclo mas uno para la transformación de salida). SAFER se basa enteramente de operaciones simples de bytes, a parte de las rotaciones de bytes en el programa de claves, esto es adecuado para microprocesadores con pequeños tamaños de palabras [Van+, 1996].

Las operaciones XOR y adiciones (módulo 256) son intercambiadas en la subsiguiente etapa adición-XOR, las S-cajas son substituciones invertibles byte a byte usando una biyección fija de 8 bit. Una transformación lineal f usada en capas lineales de 3 niveles especialmente construidas para difusión rápida. La introducción de claves aditivas predispone la eliminación de claves débiles del programa de claves, en contraste con Fiestel y muchos otros cifrados, en SAFER las operaciones usadas para encriptación difieren de aquellas para desencriptar, SAFER puede verse como una red SP. L y R denotan entradas de 8 bit en la izquierda y derecha y se define.

- $f(L,R)=(2L+R,L+R)$, la suma aquí es módulo 256, también denotada por $\boxplus$.
- Tablas $S$ y $S_{inv}$, y la tabla constante para la predisposición de claves $B_i[j]$.

**S-cajas y la predisposición de claves en SAFER.** La S-caja, la S-caja inversa y la predisposición de claves para el algoritmo de SAFER son tablas constantes y se detalla a continuación. $g \leftarrow 45$, $S[0] \leftarrow 1$, $S_{inv}[1] \leftarrow 0$, para i desde 1 a 255 hacer:

$t \leftarrow g \bullet S[i-1] \bmod 257$, $S[i] \leftarrow t$, $S_{inv}[t] \leftarrow i$. Finalmente $S[128] \leftarrow 0$, $S_{inv}[0] \leftarrow 128$ (g es generada desde $Z_{257}^{*}$, $S[i]$ es una biyección sobre $\{0,1,...,255\}$, notar que $g^{128} \equiv 256 \pmod{257}$ y asociando 256 con 0 genera S un mapeo con una entrada y una salida de 8 bit. La adición de una clave predispuesta de 8 bit constante es usada en el programa de claves pretendiendo comportarse como número aleatorio, y definiendo $B_i[j] = S[S[9i + j]]$ para i desde 2 a $2r + 1$ y j desde 1 a 8.

**SAFER SK-64 fortalecimiento del programa de claves.** Un fortalecimiento al programa de claves resulta en este modelo, involucra principalmente 3 cambios.

1. Después de inicializar $R[i]$ en el paso 1 del algoritmo para generar las subclaves, fijar $R[9] \leftarrow R[1] \oplus R[2] \oplus ... \oplus R[8]$.
2. Cambiar el rebote superior sobre el ciclo indexado desde 8 a 9 en el paso 4 (a) del algoritmo para generar las subclaves.
3. Reemplazar la línea iterativa en el paso 4 (b) del algoritmo para generar las subclaves por $K_i[j] \leftarrow R[((i + j - 2) \bmod 9) + 1] \boxplus B_i[j]$.

Así claves de 1,…,8 bytes de $R[\bullet]$ son usadas para $K_1$, de 2,…,9 bytes para $K_2$, de 3,…,9,1 para $K_3$, y así sucesivamente. Aquí y originalmente $\boxplus$ denota una adición módulo 256.

*Algoritmo SAFER K-64 de r ciclos*

Entrada:

- Texto plano $M = m_1...m_{64}$ de 64 bit
- Claves $K = k_1...k_{64}$ de 64 bits
- r con $6 \le r \le 10$

Salida:

- Bloques de texto cifrado $Y = (Y_1,...,Y_8)$ de 64 bits

Encriptación:

- Programa de claves, subclaves $K_1,...,K_{2r+1}$ de 64 bits, con entradas K y r.
- $(X_1, X_2,..., X_8) \leftarrow (m_1...m_8, m_9...m_{16},..., m_{57}...m_{64})$
- *for* $i = 1$ *to* $r$ , (XOR-Suma, S-Cajas, Suma-XOR y tres capas lineales)

1. Para $j = 1,4,5,8$:

$$X_j \leftarrow X_j \oplus K_{2i-1}[j]$$

Para $j = 2,3,6,7$:

$$X_j \leftarrow X_j \boxplus K_{2i-1}[j]$$

2. Para $j = 1,4,5,8$:

$$X_j \leftarrow S[X_j]$$

Para $j = 2,3,6,7$:

$$X_j \leftarrow S_{inv}[X_j]$$

3. Para $j = 1,4,5,8$:

$$X_j \leftarrow X_j \boxplus K_{2i}[j]$$

Para $j = 2,3,6,7$:

$$X_j \leftarrow X_j \oplus K_{2i}[j]$$

4. Para $j = 1,3,5,7$:

$$(X_j, X_{j+1}) \leftarrow f(X_j, X_{j+1})$$

5. $(Y_1, Y_2) \leftarrow f(X_1, X_3)$

$(Y_3, Y_4) \leftarrow f(X_5, X_7)$

$(Y_5, Y_6) \leftarrow f(X_2, X_4)$

$(Y_7, Y_8) \leftarrow f(X_6, X_8)$

*for* $j = 1$ *to* 8

$X_j \leftarrow Y_j$

6. $(Y_1, Y_2) \leftarrow f(X_1, X_3)$

$(Y_3, Y_4) \leftarrow f(X_5, X_7)$

$(Y_5, Y_6) \leftarrow f(X_2, X_4)$

$(Y_7, Y_8) \leftarrow f(X_6, X_8)$

*for* $j = 1$ *to* 8

$X_j \leftarrow Y_j$

Esto es similar al paso anterior

- Transformación de salida

Para $j = 1,4,5,8$:

$$Y_j \leftarrow X_j \oplus K_{2r+1}[j]$$

Para $j = 2,3,6,7$:

$$Y_j \leftarrow X_j \boxplus K_{2r+1}[j]$$

Desencriptación:

- Se logra utilizando la misma clave K y las mismas subclaves generadas para la encriptación. Cada paso de encriptación es deshecho en orden inverso, desde el ultimo al primero, comenzando con la transformación de entrada (etapa de XOR-resta) con clave $K_{2r+1}$ para deshacer la transformación de salida,

reemplazando la suma modular con resta, siguiendo con r ciclos de desencriptación usando claves desde $K_{2r}$ a $K_1$ (dos por ciclo) invirtiendo a su vez cada ciclo. Todo comienza con una capa lineal inversa de 3 niveles usando $f_{inv}(L,R) = (L-R, 2R-L)$, con una resta módulo 256, en el paso 3 la secuencia se define como sigue (para invertir la permutación de byte entre etapas de encriptación):

Nivel 1: Para $j = 1,3,5,7$:

$$(X_j, X_{j+1}) \leftarrow f_{inv}(X_j, X_{j+1})$$

Nivel 2 y 3: Para $j = 1,3,5,7$:

$$(Y_1, Y_2) \leftarrow f_{inv}(X_1, X_5)$$

$$(Y_3, Y_4) \leftarrow f_{inv}(X_2, X_6)$$

$$(Y_5, Y_6) \leftarrow f_{inv}(X_3, X_7)$$

$$(Y_7, Y_8) \leftarrow f_{inv}(X_4, X_8)$$

$$for\ j = 1\ to\ 8$$

$$X_j \leftarrow Y_j$$

Seguido de una etapa resta-XOR (reemplazo de una suma modular por una resta), además de una etapa de sustitución inversa (intercambiar $S$ por $S^{-1}$) y una etapa de XOR-resta.

*Algoritmo Generador de Claves SAFER K-64*

Entrada:

- Claves $K = k_1 ... k_{64}$ de 64 bits

- Número de ciclos r

Salida:

- Subclaves $K_1,...,K_{2r+1}$ de 64 bit, donde $K_i[j]$ es un byte j de $K_i$ (numerado de izquierda a derecha).

Proceso:

- Definir $R[i]$ de 8 bit para almacenar datos y $B_i[j]$ para denotar un byte j de $B_i$.
- $(R[1],R[2],...,R[8]) \leftarrow (k_1...k_8,k_9...k_{16},...,k_{57}...k_{64})$
- $(K_1[1],K_1[2],...,K_1[8]) \leftarrow (R[1],R[2],...,R[8])$
- *for* $i = 2$ *to* $2r+1$ (rotar bytes de claves 3 bit a la izquierda y agregar en el sesgo)

1. *for* $j = 1$ *to* 8

$$R[j] \leftarrow (R[j] \lfloor 3)$$

2. *for* $j = 1$ *to* 8

$$K_i[j] \leftarrow R[j] \boxplus B_i[j]$$

### 2.2.1.7 Algoritmo RC5

Es un algoritmo con una arquitectura orientada a la palabra para palabras de largo variables, con w=16, 32 o 64 bit. El número de ciclos r y la clave b con largo en byte, la cual igual puede ser variable. Esto es conocido más ampliamente como RC5-w, RC5-w/r y RC5-w/r/b. RC5-32/16/12 es considerado una elección común de parámetros, es recomendado r=12 ciclos para RC5-32 y r=16 para RC5-64.

El texto plano y texto cifrado son bloques de largo en bit 2w, para cada ciclo r actualizar ambas partes de datos de w bit usando dos subclaves en una transformación

de entrada y dos más para cada ciclo. Las operaciones usadas, todas sobre palabras de w bit son: suma módulo $2^w$ ($\boxplus$), XOR ($\oplus$) y rotaciones (izquierda $\overset{L}{\leftarrow\!\!\lrcorner}$, derecha $\overset{R}{\leftarrow\!\!\lrcorner}$). La operación XOR es lineal, mientras que la suma se puede considerar no lineal dependiendo de la métrica, las rotaciones presentadas aquí son dependientes de los datos y son las principales operaciones no lineales usadas: $x \overset{L}{\leftarrow\!\!\lrcorner} y$ denota un cambios cíclicos en la izquierda de y bits en una palabra de w bit, el contador de rotación y puede ser reducido a módulo w (bajo el orden suficiente de $\lg(w)$ bit de y). El programa de claves expande una clave de b bytes en $2r+2$ subclaves $K_i$ de w bit cada una. Respecto al empaquetado y desempaquetado de bytes en palabras, el orden de los bytes es guardado según su importancia, por ejemplo para w=32 bit, el primer byte de texto plano entra al final de la parte de abajo de A, el cuarto entra al final de la parte superior de A, el quinto entra al final de la parte de abajo de B, y así sucesivamente [Riv+, 1995].

*Algoritmo RC5-w/r/b*

Entrada:

- Texto plano $M = (A,B)$ de 2w bit
- Claves $K = k[0]..k[b-1]$
- r ciclos

Salida:

- Texto cifrado $C$ de 2w bits

Encriptación:

- Programa de claves, calcular $2r+2$ subclaves $K_0,...,K_{2r+1}$ , con entradas K y r.

- $A \leftarrow A \boxplus K_0$ (usar suma módulo $2^w$)
- $B \leftarrow B \boxplus K_1$ (usar suma módulo $2^w$)
- *for* $i = 1$ *to* $r$

    1. $A \leftarrow ((A \oplus B) \overset{L}{\hookleftarrow} B) \boxplus K_{2i}$
    2. $B \leftarrow ((B \oplus A) \overset{L}{\hookleftarrow} A) \boxplus K_{2i+1}$
- $C \leftarrow (A, B)$

Desencriptación:

- Se logra utilizando las subclaves empleadas en la encriptación, operando sobre texto cifrado $C \leftarrow (A, B)$ operando como sigue (resta es módulo $2^w$ y es denotada por $\boxminus$):

    1. *for* $i = r$ *downto* 1
        a. $B \leftarrow ((B \boxminus K_{2i+1}) \overset{R}{\hookleftarrow} A) \oplus A$
        b. $A \leftarrow ((A \boxminus K_{2i}) \overset{R}{\hookleftarrow} B) \oplus B$
    2. Finalmente $M \leftarrow (A \boxminus K_0, B \boxminus K_1)$

*Algoritmo Generador de Claves RC5*

Entrada:

- Palabra w de tamaño en bit
- Clave $K[0]..K[b-1]$ de b byte
- Número de ciclos r

Salida:

- Subclaves $K_0,...,K_{2r+1}$ , donde $K_i$ es de w bits

Proceso:

- Definir $u = w/8$ (número de bytes por palabra), $c = \lceil b/u \rceil$ (número de palabras K suficientes). Llenar K por la derecha con cero bytes si es necesario para conseguir un contador de byte divisible por u (ejemplo: $K[j] \leftarrow 0$ para $b \le j \le c \bullet u - 1$).
- *for* $i = 0$ *to* $c - 1$

  a. $L_i \leftarrow \sum_{j=0}^{u-1} 2^{8j} K[i \bullet u + j]$,

- $K_0 \leftarrow P_w$ , usar tabla
- *for* $i = 1$ *to* $2r + 1$

  a. $K_i \leftarrow K_{i-1} \boxplus Q_w$ , usar tabla
- $i \leftarrow 0,\ j \leftarrow 0, A \leftarrow 0, B \leftarrow 0,\ t \leftarrow max(c, 2r + 2)$
- *for* $s = 1$ *to* $3t$

  a. $K_i \leftarrow (K_i \boxplus A \boxplus B) \overset{L}{\lhd} 3$, $A \leftarrow K_i$ y

  $i \leftarrow i + 1 \mod (2r + 2)$

  b. $L_j \leftarrow (L_j \boxplus A \boxplus B) \overset{L}{\lhd} (A \boxplus B)$, $B \leftarrow L_j$ y

  $j \leftarrow j + 1 \mod c$

- La salida es $K_0, K_1, ..., K_{2r+1}$ ($L_i$ no es usado)

| $w$ : | 16 | 32 | 64 |
|---|---|---|---|
| $P_w$ : | B7E1 | B7E15163 | B7E15162 8AED2A6B |
| $Q_w$ : | 9E37 | 9E3779B9 | 9E3779B9 7F4A7C15 |

**Tabla 5.2.3 Constantes mágicas RC5 (en Hexadecimal)**

### 2.2.2 Cifrado por flujo (stream cipher)

#### 2.2.2.1 Algoritmo de vernam

Es el principal algoritmo “one-time pad”, el cual esta definido sobre un alfabeto $A = \{0,1\}$, un mensaje binario $m_1 m_2 ... m_t$ es operado sobre cadenas de claves binarias $k_1 k_2 ... k_t$ del mismo largo para producir cadenas de texto cifrado un cifrado $c_1 c_2 ... c_t$ donde la función de encriptación queda definida como $c_i = m_i \oplus k_i$ para $1 \le i \le t$ y la función de desencriptación se define como $m_i = c_i \oplus k_i$. Existen dos sustituciones de cifrado sobre el conjunto A, una es simplemente un mapeo de identidad $E_0$ que envía 0 para 0 y 1 para 1, la otra es $E_1$ que envía 0 para 1 y 1 para 0. Cuando el flujo de claves contiene un 0 aplicar $E_0$ para el correspondiente símbolo de texto plano, en otro caso aplicar $E_1$. Su seguridad radica en que cada mensaje es encriptado con una clave distinta, de lo contrario se produce redundancia y puede permitir un criptoanálisis [Van+, 1996].

#### 2.2.2.2 Algoritmo SEAL (Optimized Encryption for Software)

Es un cifrado por flujo aditivo binario que fue propuesto en 1993, este algoritmo todavía no ha sido admitido por la mayoría de la comunidad criptográfica, pero sin embargo fue diseñado especialmente para una implementación eficiente de software,

en particular para microprocesadores de 32 bit. SEAL es una función Seudoaleatoria de largo creciente la cual mapea una secuencia de n números de 32 bit para un flujo de claves de L bit bajo el control de una clave secreta a de 160 bit. En la etapa de preprocesado, la clave es distendida dentro de grandes tablas utilizando la función generadora de tabla $G_a$, esta función esta basada sobre el algoritmo SHA-1 [Van+, 1996]. Siguiendo a este preprocesado, la generación del flujo de claves requiere aproximadamente de 5 instrucciones de maquina por byte. Las siguientes notaciones se usan en SEAL para cantidades $A,B,C,D,X_i,Y_j$ de 32 bit.

- $\overline{A}$ complemento de A , a nivel de bit.
- $A \wedge B, A \vee B, A \oplus B$ AND, OR-inclusivo, OR-exclusivo, a nivel de bit.
- $A \overset{L}{\hookleftarrow} s$ , resultado de 32 bit de la rotación izquierda de A en s posiciones.
- $A \overset{R}{\hookleftarrow} s$ , resultado de 32 bit de la rotación derecha de A en s posiciones.
- $A + B$ , suma $\bmod\ 2^{32}$ de enteros sin signo Ay B.
- $f(B,C,D) \overset{def}{=} (B \wedge C) \vee (\overline{B} \wedge D)$
- $g(B,C,D) \overset{def}{=} (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
- $h(B,C,D) \overset{def}{=} B \oplus C \oplus D$
- $A \,||\, B$ , concatenación de A y B.
- $(X_1,...,X_j) \leftarrow (Y_1,...,Y_j)$, asignación simultanea $(X_i \leftarrow Y_i)$, donde $(Y_1,...,Y_j)$ es evaluado anteriormente a alguna asignación.

La tabla de generación usa la función de compresión de SHA-1 para expandir la clave secreta a dentro de los largos de tabla T, S y R. Estas tablas pueden ser precalculadas pero solo después que la clave secreta a haya sido establecida. Las

tablas T y S son de 2 Kbytes y 1 Kbyte de tamaño respectivamente, el tamaño de la tabla R depende del largo en bit L del flujo de claves deseado, cada 1Kbyte de flujo de claves requiere 16 bytes de R. En cuanto a la diferencia entre SEAL 1.0 y SEAL 2.0 radica principalmente en la función generadora de tabla, para SEAL 1.0 se basa sobre SHA, y para la segunda versión se basa sobre SHA-1.

*Algoritmo de la Función Generadora de Tabla para SEAL 2.0*

$G_a(i)$

Entrada:

- Una cadena a de 160 bit
- Un entero i con $0 \le i \le 2^{32}$

Salida:

- Una cadena de 160 bit denotada por $G_a(i)$

Proceso:

- Definición de constantes, definir 4 constantes de 32 bit en Hexadecimal. $y_1 = 0x5a827999, y_2 = 0x6ed9eba1$ y $y_3 = 0x8f1bbcdc, y_4 = 0xca62c1d6$.
- Función Generadora de Tabla
  1. Inicializar 80 palabras de 32 bit $X_0, X_1, ..., X_{79}$.
  2. $X_0 \leftarrow i$.
  3. *for* $j = 1$ *to* $15$ : $X_j \leftarrow 0x00000000$
  4. *for* $j = 16$ *to* $79$
     - o $X_j \leftarrow ((X_{j-3} \oplus X_{j-8} \oplus X_{j-14} \oplus X_{j-16}) \overset{L}{\lhd} 1)$

5. Inicializar variables de trabajo, dividir la cadena a de 160 bit en 5 palabras de 32 bit $a = H_0 H_1 H_2 H_3 H_4$.
6. $(A,B,C,D,E) \leftarrow (H_0,H_1,H_2,H_3,H_4)$
7. Ejecutar 4 ciclos de 20 pasos para actualizar, t es una variable temporal.
8. *for* $j = 0$ *to* 19
   - o $t \leftarrow ((A \overset{L}{\hookleftarrow} 5) + f(B,C,D) + E + X_j + y_1)$
   - o $(A,B,C,D,E) \leftarrow (t,A,B \overset{L}{\hookleftarrow} 30,C,D)$
9. *for* $j = 20$ *to* 39
   - o $t \leftarrow ((A \overset{L}{\hookleftarrow} 5) + h(B,C,D) + E + X_j + y_2)$
   - o $(A,B,C,D,E) \leftarrow (t,A,B \overset{L}{\hookleftarrow} 30,C,D)$
10. *for* $j = 40$ *to* 59
    - o $t \leftarrow ((A \overset{L}{\hookleftarrow} 5) + g(B,C,D) + E + X_j + y_3)$
    - o $(A,B,C,D,E) \leftarrow (t,A,B \overset{L}{\hookleftarrow} 30,C,D)$
11. *for* $j = 60$ *to* 79
    - o $t \leftarrow ((A \overset{L}{\hookleftarrow} 5) + h(B,C,D) + E + X_j + y_4)$
    - o $(A,B,C,D,E) \leftarrow (t,A,B \overset{L}{\hookleftarrow} 30,C,D)$
12. $(H_0,H_1,H_2,H_3,H_4) \leftarrow (H_0 + A,H_1 + B,H_2 + C,H_3 + D,H_4 + E)$, actualizar cambios de valores.
13. El valor de $G_a(i)$ es la cadena de 160 bit $H_0 \,||\, H_1 \,||\, H_2 \,||\, H_3 \,||\, H_4$.

*Algoritmo Generador de Flujo de Claves para SEAL 2.0*

$SEAL(a,n)$

Entrada:

- Una cadena a de 160 bit (clave secreta)
- Un entero n (no secreto) con $0 \le n \le 2^{32}$ (la secuencia de números) y el largo L en bit del flujo de claves deseado.

Salida:

- Flujo de claves y de largo $L'$, donde $L'$ es el mínimo múltiplo de 128, el cual es $\ge L$.

Proceso:

- Generación de tabla, generar las tablas T, S y R cuyas entradas son palabras de 32 bit. La función F usada abajo es definida por $F_a(i) = H^i_{i \bmod 5}$ donde $H^i_0 H^i_1 H^i_2 H^i_3 H^i_4 = G_a(\lfloor i/5 \rfloor)$ y donde la función $G_a$ fue definida anteriormente.

  1. $for\ i = 0\ \ to\ \ 511$
     - $T[i] \leftarrow F_a(i)$
  2. $for\ j = 0\ \ to\ \ 255$
     - $S[j] \leftarrow F_a(0x00001000 + j)$
  3. $for\ k = 0\ \ to\ \ 4 \bullet \lfloor (L-1)/8192 \rfloor - 1$
     - $R[k] \leftarrow F_a(0x00002000 + k)$

- Procedimiento de inicialización, la siguiente es la descripción de la subrutina $Initialize(n,t,A,B,C,D,n_1,n_2,n_3,n_4)$ la que toma como entrada una palabra n de 32 bit y un entero t y con 8 palabras

salidas de 32 bit $A,B,C,D,n_1,n_2,n_3,n_4$, esta rutina es usada en el paso 4.

1. $A \leftarrow n \oplus R[4t], B \leftarrow (n \overset{R}{\hookleftarrow} 8) \oplus R[4t+1]$
2. $C \leftarrow (n \overset{R}{\hookleftarrow} 16) \oplus R[4t+2]$
3. $D \leftarrow (n \overset{R}{\hookleftarrow} 24) \oplus R[4t+3]$
4. *for* $j = 1$ *to* 2
    - $P \leftarrow A \wedge 0x000007fc$
    - $B \leftarrow B + T[P/4], A \leftarrow (A \overset{R}{\hookleftarrow} 9)$
    - $P \leftarrow B \wedge 0x000007fc$
    - $C \leftarrow C + T[P/4], B \leftarrow (B \overset{R}{\hookleftarrow} 9)$
    - $P \leftarrow C \wedge 0x000007fc$
    - $D \leftarrow D + T[P/4], C \leftarrow (C \overset{R}{\hookleftarrow} 9)$
    - $P \leftarrow D \wedge 0x000007fc$
    - $A \leftarrow A + T[P/4], D \leftarrow (D \overset{R}{\hookleftarrow} 9)$
5. $(n_1,n_2,n_3,n_4) \leftarrow (D,B,A,C)$
6. $P \leftarrow A \wedge 0x000007fc$
7. $B \leftarrow B + T[P/4], A \leftarrow (A \overset{R}{\hookleftarrow} 9)$
8. $P \leftarrow B \wedge 0x000007fc$
9. $C \leftarrow C + T[P/4], B \leftarrow (B \overset{R}{\hookleftarrow} 9)$
10. $P \leftarrow C \wedge 0x000007fc$

11. $D \leftarrow D + T[P/4], C \leftarrow (C \overset{R}{\hookleftarrow} 9)$

12. $P \leftarrow D \wedge 0x000007fc$

13. $A \leftarrow A + T[P/4], D \leftarrow (D \overset{R}{\hookleftarrow} 9)$

- Inicializar $y$ como una cadena vacía y $t \leftarrow 0$.
- Repetir lo siguiente

1. ejecutar $Initialize(n,t,A,B,C,D,n_1,n_2,n_3,n_4)$.
2. $for\ i = 1\ to\ 64$

   - $P \leftarrow A \wedge 0x000007fc$
   - $B \leftarrow B + T[P/4], A \leftarrow (A \overset{R}{\hookleftarrow} 9)$
   - $B \leftarrow B \oplus A$
   - $Q \leftarrow B \wedge 0x000007fc$
   - $C \leftarrow C \oplus T[Q/4], B \leftarrow (B \overset{R}{\hookleftarrow} 9)$
   - $C \leftarrow C + B$
   - $P \leftarrow (P + C) \wedge 0x000007fc$
   - $D \leftarrow D + T[P/4], C \leftarrow (C \overset{R}{\hookleftarrow} 9)$
   - $D \leftarrow D \oplus C$
   - $Q \leftarrow (Q + D) \wedge 0x000007fc$
   - $A \leftarrow A \oplus T[Q/4], D \leftarrow (D \overset{R}{\hookleftarrow} 9)$
   - $A \leftarrow A + D$
   - $P \leftarrow (P + A) \wedge 0x000007fc$
   - $B \leftarrow B \oplus T[P/4], A \leftarrow (A \overset{R}{\hookleftarrow} 9)$

- $Q \leftarrow (Q + B) \wedge 0x000007fc$
- $C \leftarrow C + T[Q/4], B \leftarrow (B \overset{R}{\hookleftarrow} 9)$
- $P \leftarrow (P + C) \wedge 0x000007fc$
- $D \leftarrow D \oplus T[P/4], C \leftarrow (C \overset{R}{\hookleftarrow} 9)$
- $Q \leftarrow (Q + D) \wedge 0x000007fc$
- $A \leftarrow A + T[Q/4], D \leftarrow (D \overset{R}{\hookleftarrow} 9)$
- $y \leftarrow y \,||\, (B + S[4i-4]) \,||\, (C \oplus S[4i-3]) \,||\, (D + S[4i-2]) \,||\, (A \oplus S[4i-1])$
- Si $y$ es $\geq L$ bits de largo, entonces retornar $y$ y parar.
- Si $i$ es impar entonces $(A,C) \leftarrow (A + n_1, C + n_2)$, en otro caso $(A,C) \leftarrow (A + n_3, C + n_4)$.

3. $t \leftarrow t + 1$.

### 2.2.3 Cifrado por funciones hash (hash functions)

#### 2.2.3.1 Funciones hash sin claves (MDC)

##### 2.2.3.1.1 Funciones hash basadas sobre cifrado por bloques

###### 2.2.3.1.1.1 Algoritmo Matyas-Meyer-Oseas de largo simple

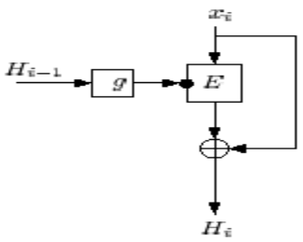



**Fig. 5.2.3 Diagrama del algoritmo Matyas-Meyer-Oseas**

*Algoritmo*

Entrada:

- Cadena de bit x

Salida:

- Cifrado hash de n bit de x

Proceso:

- La entrada x es dividida en bloques de n bit y rellenado si es necesario para completar el último bloque. Designar el rellenado del mensaje consistiendo de t bloques de n bit $x_1 x_2 \ldots x_t$, un valor inicial constante IV de n bit debe ser pre-especificado.
- La salida es $H_t$ definida por:

  1. $H_0 = IV$
  2. $H_i = E_{g(H_{i-1})}(x_i) \oplus x_i$ , $1 \le i \le t$

### 2.2.3.1.1.2 Algoritmo Davies-Meyer de largo simple

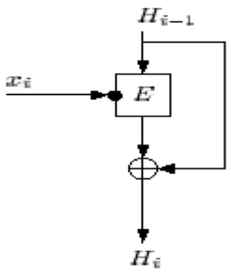



**Fig. 5.2.4 Diagrama del algoritmo Davies-Meyer**

*Algoritmo*

Entrada:

- Cadena de bit x

Salida:

- Cifrado hash de n bit de x

Proceso:

- La entrada x es dividida en bloques de k bit donde k es el tamaño de la clave y rellenado si es necesario para completar el último bloque. Designar el rellenado del mensaje consistiendo de t bloques de k bit $x_1 x_2 \ldots x_t$ , un valor inicial constante IV de n bit debe ser pre-especificado.
- La salida es $H_t$ definida por:

  1. $H_0 = IV$
  2. $H_i = E_{x_i}(H_{i-1}) \oplus H_{i-1}, 1 \le i \le t$

### 2.2.3.1.1.3 Algoritmo Miyaguchi-Preneel de largo simple

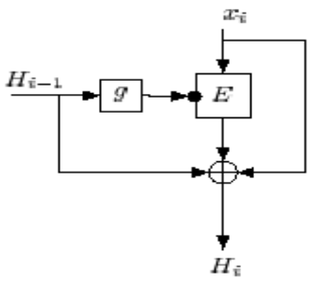



**Fig. 5.2.5 Diagrama del algoritmo Miyaguchi-Preneel**

*Algoritmo*

Entrada:

- Cadena de bit x

Salida:

- Cifrado hash de n bit de x

Proceso:

- La entrada x es dividida en bloques de n bit y rellenado si es necesario para completar el último bloque. Designar el rellenado del mensaje consistiendo de t bloques de n bit $x_1 x_2 \ldots x_t$, un valor inicial constante IV de n bit debe ser pre-especificado.
- La salida es $H_t$ definida por:

  1. $H_0 = IV$
  2. $H_i = E_{g(H_{i-1})}(x_i) \oplus x_i \oplus H_{i-1}, 1 \leq i \leq t$

### 2.2.3.1.1.4 Algoritmo MDC-2 de largo doble

En general la construcción suele hacerse en base al cifrado por bloques DES, sin embargo puede ser usada con otros cifrados por bloques, MDC-2 y MDC-4 hacen uso de los siguientes componentes.

- DES como el cifrado por bloques $E_K$ de largo en bit $n = 64$ parametrizado por una clave K de 56 bit.
- Dos funciones $g$ y $\tilde{g}$ las cuales mapean valores U de 64 bit para claves DES apropiadas de 56 bit como sigue.
  - Para $U = u_1 u_2 ... u_{64}$ borrar todos los octavos bit comenzando con $u_8$ y fijar el segundo y tercer bit a $'10'$ para g y $'01'$ para $\tilde{g}$, quedando $g(U) = u_1 10 u_4 u_5 u_6 u_7 u_9 .. u_{63}$ y $\tilde{g}(U) = u_1 01 u_4 u_5 u_6 u_7 u_9 .. u_{63}$.

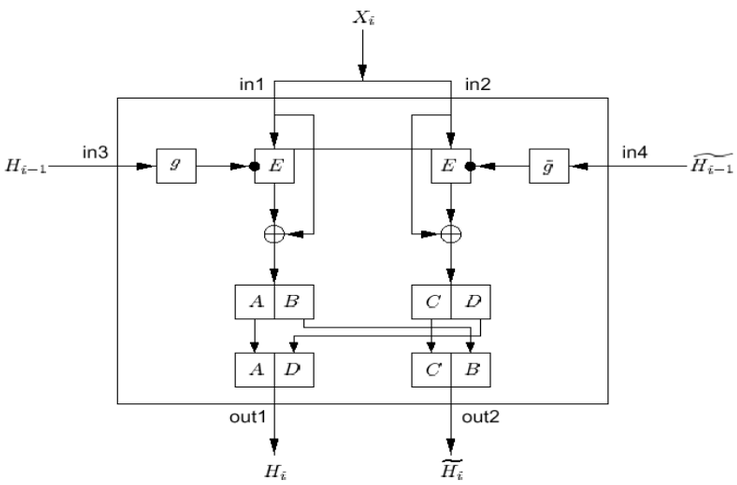



**Fig. 5.2.6 Función de compresión de una función hash MDC-2, E=DES**

*Algoritmo MDC-2 basado en DES*

Entrada:

- Cadena de x de largo en bit $r = 64t$ para $t \geq 2$.

Salida:

- Cifrado hash de 128 bit de x

Proceso:

- Partición de x en bloques $x_i$ de 64 bit $x = x_1 x_2 ... x_t$.
- Escoger las constantes no secretas de 64 bit IV, $\tilde{IV}$ (la misma constante debe ser usada para la verificación MDC) desde un conjunto de valores prescritos, un conjunto de valores prescritos por defecto (Hexadecimal) son:

  $IV = 0x5252525252525252$ $\tilde{IV} = 0x2525252525252525$.
- Permitir $||$ denotando concatenación y $C_i^L, C_i^R$ como las partes de la izquierda y derecha respectivamente de 32 bit. La salida es $h(x) = H_t \,||\, \tilde{H}_t$ para $1 \le i \le t$ y definida como sigue:

  1. $H_0 = IV, \tilde{H}_0 = \tilde{IV}$
  2. $k_i = g(H_{i-1}), \tilde{k}_i = \tilde{g}(\tilde{H}_{i-1})$
  3. $C_i = E_{K_i}(x_i) \oplus x_i, \tilde{C}_i = E_{\tilde{K}_i}(x_i) \oplus x_i$
  4. $H_i = C_i^L \,||\, \tilde{C}_i^R, \tilde{H}_i = \tilde{C}_i^L \,||\, C_i^R$

**2.2.3.1.1.5 Algoritmo MDC-4 de largo doble**

Es construido usando la función de compresión de MDC-2, una iteración de la función de compresión de MDC-4 consiste de dos ejecuciones secuenciales de la función de compresión de MDC-2., donde.

- Los dos datos de entrada de 64 bit para la primera compresión MDC-2 son ambos el mismo para el siguiente bloque de mensaje de 64 bit.
- Las claves para la primera compresión MDC-2 son derivadas desde las salidas (cambiando variables) de la anterior compresión MDC-4.
- Las claves de la segunda compresión MDC-2 son derivadas desde las salidas (cambiando variables) de la primera compresión MDC-2.
- Los dos datos de entrada de 64 bit para la segunda compresión MDC-2 son la salida (cambiando variables) de los lados opuestos de la anterior compresión MDC-4.

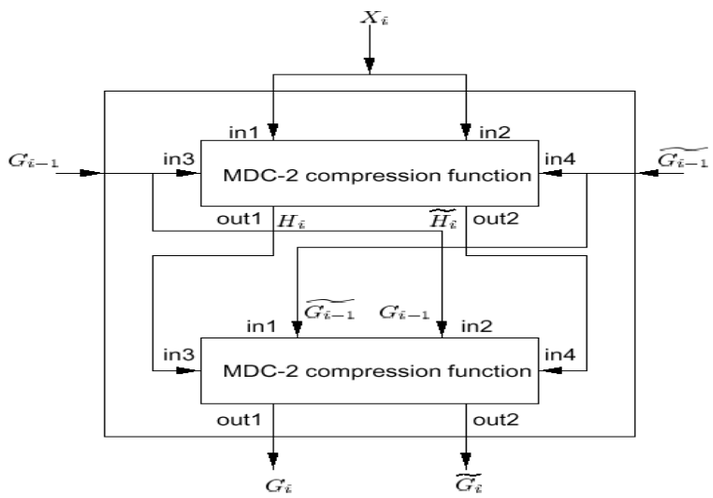



**Fig. 5.2.7 Función de compresión de una función hash MDC-4**

*Algoritmo MDC-4 basado en DES*

Entrada:

- Cadena de x de largo en bit $r = 64t$ para $t \geq 2$.

Salida:

- Cifrado hash de 128 bit de x

Proceso:

- Partición de x en bloques $x_i$ de 64 bit $x = x_1 x_2 \ldots x_t$.

- Escoger las constantes no secretas de 64 bit IV, $\tilde{IV}$ (la misma constante debe ser usada para la verificación MDC) desde un conjunto de valores prescritos, un conjunto de valores prescritos por defecto (Hexadecimal) son:

  $IV = 0x5252525252525252$ $\tilde{IV} = 0x2525252525252525$.

- Permitir $||$ denotando concatenación y $C_i^L, C_i^R$ como las partes de la izquierda y derecha respectivamente de 32 bit. La salida es $h(x) = G_t \,||\, \tilde{G}_t$ para $1 \le i \le t$ y definida como sigue:

  1. $G_0 = IV, \tilde{G}_0 = \tilde{IV}$
  2. $k_i = g(G_{i-1}), \tilde{k}_i = \tilde{g}(\tilde{G}_{i-1})$
  3. $j_i = g(H_i), \tilde{j}_i = \tilde{g}(\tilde{H}_i)$
  4. $C_i = E_{K_i}(x_i) \oplus x_i, \tilde{C}_i = E_{\tilde{K}_i}(x_i) \oplus x_i$
  5. $D_i = E_{j_i}(\tilde{G}_{i-1}) \oplus \tilde{G}_{i-1}, \tilde{D}_i = E_{\tilde{j}_i}(G_{i-1}) \oplus G_{i-1}$
  6. $H_i = C_i^L \,||\, \tilde{C}_i^R, \tilde{H}_i = \tilde{C}_i^L \,||\, C_i^R$
  7. $G_i = D_i^L \,||\, \tilde{D}_i^R, \tilde{G}_i = \tilde{D}_i^L \,||\, D_i^R$

**2.2.3.1.2 Funciones hash optimizadas basadas sobre MD4 (Message Digest)**

Toda la familia de algoritmos de MD4 usan algunas de las notaciones de la tabla siguiente.

| Notation | Meaning |
|---|---|
| $u, v, w$ | variables representing 32-bit quantities |
| 0x67452301 | hexadecimal 32-bit integer (least significant byte: 01) |
| $+$ | addition modulo $2^{32}$ |
| $\overline{u}$ | bitwise complement |
| $u \hookleftarrow s$ | result of rotating $u$ left through $s$ positions |
| $uv$ | bitwise AND |
| $u \vee v$ | bitwise inclusive-OR |
| $u \oplus v$ | bitwise exclusive-OR |
| $f(u, v, w)$ | $uv \vee \overline{u}w$ |
| $g(u, v, w)$ | $uv \vee uw \vee vw$ |
| $h(u, v, w)$ | $u \oplus v \oplus w$ |
| $(X_1, \ldots, X_j) \leftarrow (Y_1, \ldots, Y_j)$ | simultaneous assignments $(X_i \leftarrow Y_i)$, where $(Y_1, \ldots, Y_j)$ is evaluated prior to any assignments |

**Tabla 5.2.4 Notaciones para MD4**

### 2.2.3.1.2.1 Algoritmo MD4

*Algoritmo de Función Hash MD4*

Entrada:

- Cadena de bit x de largo arbitrario en bit $b \geq 0$.

Salida:

- Cifrado hash de 128 bit de x

Proceso:

- Definición de constantes, definir 4 valores de cambio iniciales (IV) de 32 bit.

$h_1 = 0x67452301$
$h_2 = 0xefcdab89$
$h_3 = 0x98badcfe$
$h_4 = 0x10325476$

Definir constantes aditivas de 32 bit.

$y[j] = 0 , 0 \leq j \leq 15$

$y[j] = 0x5a827999 , 16 \leq j \leq 31$ ,(constante = $\sqrt{2}$ )

$y[j] = 0x6ed9eba1 , 32 \leq j \leq 47$ ,(constante = $\sqrt{3}$ )

Definir orden para acceder palabras de origen (cada lista contiene de 0 a 15)

$$z[0..15] = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]$$
$$z[16..31] = [0,4,8,12,1,5,9,13,2,6,10,14,3,7,11,15]$$
$$z[32..47] = [0,8,4,12,2,10,6,14,1,9,5,13,3,11,7,15]$$

Finalmente definir el número de posiciones de bit para cambios a la izquierda (rotaciones)

$$s[0..15] = [3,7,11,19,3,7,11,19,3,7,11,19,3,7,11,19]$$
$$s[16..31] = [3,5,9,13,3,5,9,13,3,5,9,13,3,5,9,13]$$
$$s[32..47] = [3,9,11,15,3,9,11,15,3,9,11,15,3,9,11,15]$$

- Preprocesado, rellenar x como sigue tal que su largo en bit sea múltiplo de 512. Anexar 1 bit, entonces anexar $r - 1 (\geq 0)$ bit para un pequeño r resultando en un múltiplo de 512 con largo en bit de menos de 64. Finalmente agregar la representación $b \mod 2^{64}$ de 64 bit, como dos palabras de 32 bit con la primera palabra menos significante. Fijar m como el número de bloques de 512 bit en la cadena de resultado $(b + r + 64 = 512m = 32 \bullet 16m)$. El formateo de entrada consiste de 16m palabras de 32 bit $X_0 X_1 ... X_{16m-1}$. Inicializar $(H_1, H_2, H_3, H_4) \leftarrow (h_1, h_2, h_3, h_4)$.
- Procesando, para cada i desde 0 a m-1, copiar el i-ésimo bloque de 16 palabras de 32 bit en un almacenamiento temporal $X[j] \leftarrow x_{16i+j}$, $0 \leq j \leq 15$, entonces procesar estos en 3 ciclos de 16 pasos antes de actualizar las variables de cambio.
    1. $(A, B, C, D) \leftarrow (H_1, H_2, H_3, H_4)$
    2. *for* $j = 0$ *to* 15

- $t \leftarrow (A + f(B,C,D) + X[z[j]] + y[j])$
- $(A,B,C,D) \leftarrow (D,t \lfloor s[j], B, C)$

3. *for* $j = 16$ *to* 31
   - $t \leftarrow (A + g(B,C,D) + X[z[j]] + y[j])$
   - $(A,B,C,D) \leftarrow (D,t \lfloor s[j], B, C)$
4. *for* $j = 32$ *to* 47
   - $t \leftarrow (A + h(B,C,D) + X[z[j]] + y[j])$
   - $(A,B,C,D) \leftarrow (D,t \lfloor s[j], B, C)$
5. $(H_1,H_2,H_3,H_4) \leftarrow (H_1 + A, H_2 + B, H_3 + C, H_4 + D)$

- Completación, el valor Hash final es la concatenación de $H_1 \| H_2 \| H_3 \| H_4$.

**2.2.3.1.2.2 Algoritmo MD5**

Fue diseñado como un fortalecimiento de MD4, producto de las coalisiones encontradas en MD4, los cambios hechos a MD4 para obtener MD5 fueron.

- Adición de un cuarto ciclo de 16 pasos y una función de ciclo 4.
- Reemplazo de la función de ciclo 2 por una nueva función.
- Modificación del orden de acceso para palabras de mensaje dentro de los ciclos 2 y 3.
- Modificación de la cantidad de cambios (tal que los cambios difieran en los distintos ciclos).
- Usar un aditivo constante único dentro de los cada 4x16 pasos, basados sobre la parte entera de $2^{32} \bullet sin(j)$ para el paso j.
- Adición de salida desde el paso anterior en cada uno de los 64 pasos.

*Algoritmo de Función Hash MD5*

Entrada:

- Cadena de bit x de largo arbitrario en bit $b \geq 0$.

Salida:

- Cifrado hash de 128 bit de x

Proceso:

- Notación, reemplazar la función de ciclo 2 por $g(u,v,w) \overset{def}{=} uw \vee v\overline{w}$.

  Definir una función de ciclo 4 como $k(u,v,w) \overset{def}{=} v \oplus (u \vee \overline{w})$.

- Definición de constantes, Redefinir constantes aditivas únicas. $y[j] =$ primeros 32 bit del valor binario $abs(sin(j+1))$, $0 \leq j \leq 63$, donde j esta en radianes y "abs" denota valor absoluto. Redefinir orden de acceso para palabras en el ciclo 2 y 3 definir para ciclo 4.

  $z[16..31] = [1,6,11,0,5,10,15,4,9,14,3,8,13,2,7,12]$
  $z[32..47] = [5,8,11,14,1,4,7,10,13,0,3,6,9,12,15,2]$
  $z[48..63] = [0,7,14,5,12,3,10,1,8,15,6,13,4,11,2,9]$

  Redefinir el número de posiciones de bit para cambios a la izquierda (rotaciones)

  $s[0..15] = [7,12,17,22,7,12,17,22,7,12,17,22,7,12,17,22]$
  $s[16..31] = [5,9,14,20,5,9,14,20,5,9,14,20,5,9,14,20]$
  $s[32..47] = [4,11,16,23,4,11,16,23,4,11,16,23,4,11,16,23]$
  $s[48..63] = [6,10,15,21,6,10,15,21,6,10,15,21,6,10,15,21]$

- Preprocesado, igual a MD4

- Procesado, en cada uno de los ciclos 1, 2 y 3 reemplazar $"B \leftarrow (t \lfloor s[j] \rfloor)"$ por $"B \leftarrow B + (t \lfloor s[j] \rfloor)"$, también inmediatamente después del ciclo 3 agregar.

  1. *for* $j = 48$ *to* 63

     - $t \leftarrow (A + k(B,C,D) + X[z[j]] + y[j])$
     - $(A,B,C,D) \leftarrow (D, B + (t \lfloor s[j] \rfloor), B, C)$

- Completación, el valor Hash final es la concatenación de $H_1 \| H_2 \| H_3 \| H_4$.

**2.2.3.1.2.3 Algoritmo SHA-1 (Secure Hash Algorithm)**

Esta basado sobre MD4, y fue propuesto por U.S. NIST (National Institute for Standards and Technology) para la seguridad de aplicaciones del gobierno federal de Estados Unidos [Van+, 1996], las principales diferencias de SHA-1 desde MD4 son.

- El valor Hash es de 160 bit y cinco (versus cuatro) variables de cambio de 32 bit son usadas.
- La función de compresión tiene 4 ciclos en lugar de 3, usando las funciones de paso de MD4 f, g y h como sigue: f es la primera, g es la tercera y h esta entre el segundo y el cuarto ciclo. Cada ciclo tiene 20 pasos en lugar de 16.
- En el interior de la función de compresión, cada bloque de mensaje de 16 palabras es expandido para un bloque de 80 palabras por un proceso por el cual las últimas 64 de las 80 palabras constituyen un XOR de 4 palabras desde las primeras posiciones en el bloque expandido. Esas 80 palabras son la entrada (una palabra por paso) para los 80 pasos.
- El núcleo del paso es modificado como sigue: la única rotación usada es una rotación constante de 5 bit, la quinta variable de trabajo es agregada dentro de

cada paso, palabras de mensajes desde el bloque de mensaje expandido son accesadas secuencialmente y C es actualizada como una rotación izquierda de B de 30 bit, mejor dicho simplemente B.

- SHA-1 usa cuatro constantes aditivas no cero, mientras que MD4 usaba 3 constantes y solo 2 de las cuales eran no cero.

*Algoritmo SHA-1*

Entrada:

- Cadena de bit x de largo arbitrario en bit $b \geq 0$.

Salida:

- Cifrado hash de 160 bit de x

Proceso:

- Notación, igual a MD4.
- Definición de constantes, definir un quinto IV y los otros corresponden a MD4 $h_5 = 0xc3d2e1f0$, definir por ciclos constantes aditivas enteras

  $y_1 = 0x5a827999$
  $y_2 = 0x6ed9eba1$
  $y_3 = 0x8f1bbcdc$
  $y_4 = 0xca62c1d6$

  (sin orden para acceder fuentes de palabras, especificaciones de posiciones de bit para cambios a la izquierda es requerida)
- Preprocesamiento Global, rellenar como en MD4, excepto al final ya que dos palabras de 32 bit específicamente de largo en bit b son anexadas por medio de la palabra mas significante precediendo la menos significante. Como en MD4 el formateo de entrada son

16m palabras de 32 bit $x_0 x_1 ... x_{16m-1}$. Inicializar variables de cambio $(H_1, H_2, H_3, H_4, H_5) \leftarrow (h_1, h_2, h_3, h_4, h_5)$.

- Procesamiento, para cada i desdo 0 hasta $m-1$ copiar el i-esimo bloque de 16 palabras de 32 bit en un almacenamiento temporal $X[j] \leftarrow x_{16i+j}$ , $0 \le j \le 15$ y procesar estos debajo en cuatro ciclos de 20 pasos antes de actualizar las variables de cambio (expandir un bloque de 16 palabras en un bloque de 80 palabras, $X_j$ significa $X[j]$).

1. *for* $j = 16$ *to* 79
   - $X_j = ((X_{j-3} \oplus X_{j-8} \oplus X_{j-14} \oplus X_{j-16}) \lhd 1)$
2. Inicializar variables de trabajo

   $(A, B, C, D, E) \leftarrow (H_1, H_2, H_3, H_4, H_5)$
3. *for* $j = 0$ *to* 19
   - $t \leftarrow ((A \lhd 5) + f(B, C, D) + E + X_j + y_1)$
   - $(A, B, C, D, E) \leftarrow (t, A, B \lhd 30, C, D)$
4. *for* $j = 20$ *to* 39
   - $t \leftarrow ((A \lhd 5) + h(B, C, D) + E + X_j + y_2)$
   - $(A, B, C, D, E) \leftarrow (t, A, B \lhd 30, C, D)$
5. *for* $j = 40$ *to* 59
   - $t \leftarrow ((A \lhd 5) + g(B, C, D) + E + X_j + y_3)$
   - $(A, B, C, D, E) \leftarrow (t, A, B \lhd 30, C, D)$
6. *for* $j = 60$ *to* 79
   - $t \leftarrow ((A \lhd 5) + h(B, C, D) + E + X_j + y_4)$

- $(A,B,C,D,E) \leftarrow (t,A,B \lfloor\!\!\lfloor 30,C,D)$

7. Actualizar valores de cambio

$$(H_1,H_2,H_3,H_4,H_5) \leftarrow (H_1+A,H_2+B,H_3+C,H_4+D,H_5+E)$$

- El valor Hash es $H_1 \,||\, H_2 \,||\, H_3 \,||\, H_4 \,||\, H_5$.

**2.2.3.1.2.4 Algoritmo RIPEMD-160**

Esta basado sobre MD4, tomando en cuenta conocimientos ganados en el análisis de MD4, MD5 y RIPEMD. La función de compresión global de RIPEMD-160 mapea entradas de 21 palabras (5 palabras de variables de cambio más 16 palabras de bloques de mensajes, con palabras de 32 bit) para salidas de 5 palabras [Van+, 1996]. Cada bloque de entrada es procesado en paralelo por distintas versiones (linea izquierda y linea derecha) de la función de compresión, las salidas de 160 bit de las lineas separadas son combinadas para obtener una única salida de 160 bit.

| Notation | Definition |
|---|---|
| $f(u,v,w)$ | $u \oplus v \oplus w$ |
| $g(u,v,w)$ | $uv \vee \overline{u}w$ |
| $h(u,v,w)$ | $(u \vee \overline{v}) \oplus w$ |
| $k(u,v,w)$ | $uw \vee v\overline{w}$ |
| $l(u,v,w)$ | $u \oplus (v \vee \overline{w})$ |

**Tabla 5.2.5 Funciones de ciclo de RIPEMD-160**

La función de compresión de RIPEMD-160 difiere de MD4 en el número de palabras de variables de cambio, el número de de ciclos, las funciones de ciclo, el orden en el cual las palabras de entrada son accesadas y las cantidades con la cual los resultados son rotados. Las lineas de cálculo de la izquierda y derecha difieren en dos item, en sus constantes aditivas y el orden en el cual las funciones de ciclo son aplicadas.

| Variable | Value |
|---|---|
| $z_L$[ 0..15] | [ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15] |
| $z_L$[16..31] | [ 7, 4,13, 1,10, 6,15, 3,12, 0, 9, 5, 2,14,11, 8] |
| $z_L$[32..47] | [ 3,10,14, 4, 9,15, 8, 1, 2, 7, 0, 6,13,11, 5,12] |
| $z_L$[48..63] | [ 1, 9,11,10, 0, 8,12, 4,13, 3, 7,15,14, 5, 6, 2] |
| $z_L$[64..79] | [ 4, 0, 5, 9, 7,12, 2,10,14, 1, 3, 8,11, 6,15,13] |
| $z_R$[ 0..15] | [ 5,14, 7, 0, 9, 2,11, 4,13, 6,15, 8, 1,10, 3,12] |
| $z_R$[16..31] | [ 6,11, 3, 7, 0,13, 5,10,14,15, 8,12, 4, 9, 1, 2] |
| $z_R$[32..47] | [15, 5, 1, 3, 7,14, 6, 9,11, 8,12, 2,10, 0, 4,13] |
| $z_R$[48..63] | [ 8, 6, 4, 1, 3,11,15, 0, 5,12, 2,13, 9, 7,10,14] |
| $z_R$[64..79] | [12,15,10, 4, 1, 5, 8, 7, 6, 2,13,14, 0, 3, 9,11] |
| $s_L$[ 0..15] | [11,14,15,12, 5, 8, 7, 9,11,13,14,15, 6, 7, 9, 8] |
| $s_L$[16..31] | [ 7, 6, 8,13,11, 9, 7,15, 7,12,15, 9,11, 7,13,12] |
| $s_L$[32..47] | [11,13, 6, 7,14, 9,13,15,14, 8,13, 6, 5,12, 7, 5] |
| $s_L$[48..63] | [11,12,14,15,14,15, 9, 8, 9,14, 5, 6, 8, 6, 5,12] |
| $s_L$[64..79] | [ 9,15, 5,11, 6, 8,13,12, 5,12,13,14,11, 8, 5, 6] |
| $s_R$[ 0..15] | [ 8, 9, 9,11,13,15,15, 5, 7, 7, 8,11,14,14,12, 6] |
| $s_R$[16..31] | [ 9,13,15, 7,12, 8, 9,11, 7, 7,12, 7, 6,15,13,11] |
| $s_R$[32..47] | [ 9, 7,15,11, 8, 6, 6,14,12,13, 5,14,13,13, 7, 5] |
| $s_R$[48..63] | [15, 5, 8,11,14,14, 6,14, 6, 9,12, 9,12, 5,15, 8] |
| $s_R$[64..79] | [ 8, 5,12, 9,12, 5,14, 6, 8,13, 6, 5,15,13,11,11] |

**Tabla 5.2.6 Orden de acceso a palabras y conteo de rotación de RIPEMD-160**

*Algoritmo RIPEMD-160*

Entrada:

- Cadena de bit x de largo en bit $b \geq 0$.

Salida:

- Cifrado hash de 160 bit de x

Proceso:

- Notación, redefinir funciones de ciclo f, g, h según tabla y además de las nuevas funciones de ciclo k y l.
- Definición de constantes, definir un quinto IV y los otros corresponden a MD4 $h_5 = 0xc3d2e1f0$. En adición.
    1. Usar las constantes aditivas de MD4 para la linea izquierda, renombrar:

$y_L[j] = 0 , 0 \le j \le 15$
$y_L[j] = 0x5a827999 , 16 \le j \le 31$
$y_L[j] = 0x6ed9eba1 , 32 \le j \le 47$
$y_L[j] = 0x8f1bbcdc , 48 \le j \le 63$
$y_L[j] = 0xa953fd4e , 64 \le j \le 79$

2. Definir cinco nuevas constantes aditivas para la linea derecha.

$y_R[j] = 0x50a28be6 , 0 \le j \le 15$
$y_R[j] = 0x5c4dd124 , 16 \le j \le 31$
$y_R[j] = 0x6d703ef3 , 32 \le j \le 47$
$y_R[j] = 0x7a6d76e9 , 48 \le j \le 63$
$y_R[j] = 0 , 64 \le j \le 79$

3. Ver la tabla de valores de RIPEMD-160 para constantes para el paso j de la función de compresión: $z_L[j]$, $z_R[j]$ especificar el orden de acceso para palabras de origen en las lineas de la izquierda y derecha. $s_L[j]$, $s_R[j]$ el número de posiciones de bit para rotaciones.

- Preprocesamiento, igual que en MD4 con la suma de una quinta variable de cambio $H_5 \leftarrow h_5$.
- Procesamiento, para cada i desdo 0 hasta $m-1$ copiar el i-esimo bloque de 16 palabras de 32 bit en un almacenamiento temporal $X[j] \leftarrow x_{16i+j}$ , $0 \le j \le 15$, entonces.

1. Ejecutar cinco ciclos de 16 pasos de la linea izquierda como sigue:

   o $(A_L, B_L, C_L, D_L, E_L) \leftarrow (H_1, H_2, H_3, H_4, H_5)$

- o *for* $j = 0$ *to* 15
  - a. $t \leftarrow (A_L + f(B_L, C_L, D_L) + X[z_L[j]] + y_L[j])$
  - b. $(A_L, B_L, C_L, D_L, E_L) \leftarrow (E_L, E_L + (t \lll s_L[j]), B_L, C_L \lll 10, D_L)$
- o *for* $j = 16$ *to* 31
  - a. $t \leftarrow (A_L + g(B_L, C_L, D_L) + X[z_L[j]] + y_L[j])$
  - b. $(A_L, B_L, C_L, D_L, E_L) \leftarrow (E_L, E_L + (t \lll s_L[j]), B_L, C_L \lll 10, D_L)$
- o *for* $j = 32$ *to* 47
  - a. $t \leftarrow (A_L + h(B_L, C_L, D_L) + X[z_L[j]] + y_L[j])$
  - b. $(A_L, B_L, C_L, D_L, E_L) \leftarrow (E_L, E_L + (t \lll s_L[j]), B_L, C_L \lll 10, D_L)$
- o *for* $j = 48$ *to* 63
  - a. $t \leftarrow (A_L + k(B_L, C_L, D_L) + X[z_L[j]] + y_L[j])$
  - b. $(A_L, B_L, C_L, D_L, E_L) \leftarrow (E_L, E_L + (t \lll s_L[j]), B_L, C_L \lll 10, D_L)$
- o *for* $j = 64$ *to* 79
  - a. $t \leftarrow (A_L + l(B_L, C_L, D_L) + X[z_L[j]] + y_L[j])$
  - b. $(A_L, B_L, C_L, D_L, E_L) \leftarrow (E_L, E_L + (t \lll s_L[j]), B_L, C_L \lll 10, D_L)$

2. Ejecutar en paralelo los 5 ciclos de la derecha de forma analogica a los citados anteriormente con $(A_R, B_R, C_R, D_R, E_R)$, $y_R[j]$, $z_R[j]$, $s_R[j]$ reemplazando las correspondientes cantidades con el subíndice L y el orden de las funciones de ciclo invertirlas, de esta manera sus ordenes son l, k, h, g y f. Comenzar incializando las variables de trabajo de la linea derecha $(A_R, B_R, C_R, D_R, E_R) \leftarrow (H_1, H_2, H_3, H_4, H_5)$.

3. Despues de ejecutadas ambas lineas izquierda y derecha, actualizar los valores de cambio como sigue:

$$t \leftarrow H_1$$
$$H_1 \leftarrow H_2 + C_L + D_R$$
$$H_2 \leftarrow H_3 + D_L + E_R$$
$$H_3 \leftarrow H_4 + E_L + A_R$$
$$H_4 \leftarrow H_5 + A_L + B_R$$
$$H_5 \leftarrow t + B_L + C_R$$

- El valor Hash es $H_1 \,||\, H_2 \,||\, H_3 \,||\, H_4 \,||\, H_5$.

**2.2.3.1.3 Funciones hash basadas sobre aritmética modular**

**2.2.3.1.3.1 Algoritmo MASH-1 (Modular Arithmetic Secure Hash)**

Este habia sido propuesto por la incorporación de un borrador de un estándar ISO/IEC. MASH-1 involucra el uso de un parecido módulo M RSA, cuyo largo en bit afecta la seguridad, M podria ser dificil de factorizar y para un M de factorización desconocida la seguridad esta basada en parte sobre la dificultad de estraer las raices de los módulos [Van+, 1996]. El largo en bit de M también determina el tamaño del bloque para procesamiento de mensajes y el tamaño del resultado Hash.

*Algoritmo MASH-1*

Entrada:

- Dato x de largo en bit $0 \leq b < 2^{n/2}$.

Salida:

- Cifrado hash de n bit de x (n es aproximadamente el largo en bit del módulo M)

Proceso:

- Configuración de sistema y definición de constantes, fijar un módulo M=pq semejante a RSA de largo en bit m, donde p y q son valores primos secretos seleccionados aleatoriamente tal que la factorización de M sea intratable. Definir el largo en bit n del resultado Hash para ser un múltiplo grande de m de menos de 16. $H_0 = 0$ es definido como un IV y una constante entera $A = 0xf0...0$ de n bit, "$\vee$" denota OR inclusivo en bits, "$\oplus$" denota OR exclusivo en bits.
- Rellenado, rellenar x con bits 0 si es necesario para obtener una cadena de largo en bit $t \bullet n/2$ para el t más pequeño posible con $t \geq 1$. Dividir el texto rellenado en bloques $x_1,...,x_t$ de $(n/2)$ bit y agregar al final un bloque $x_{t+1}$ conteniendo la representación de b de $(n/2)$ bit.
- Expansión, expandir cada $x_i$ para un bloque $y_i$ de n bit por medio de particiones de medio byte (4 bit) y insertando 4 bit 1, excepto para $y_{t+1}$ en donde el medio byte insertado es 1010 (y no 1111).
- Procesamiento de la función de compresión, para $1 \leq i \leq t+1$ mapear dos entradas $(H_{i-1}, y_i)$ de n bit para una salida de n bit, este proceso es el siguiente:
    1. $H_i \leftarrow ((((H_{i-1} \oplus y_i) \vee A)^2 \bmod M) \prec n) \oplus H_{i-1}$

       Donde $\prec n$ denota conservación del extremo derecho de n bit del resultado de m bit para su izquierda.
- Completación, el Hash es un bloque $H_{t+1}$ de n bit.

MASH-2 es definido de la misma forma que MASH-1, excepto porque el exponente $e = 2$ usado para elevar al cuadrado la función de compresión, es reemplazada por $e = 2^8 + 1$.

### 2.2.3.2 Funciones hash con claves (MAC)

#### 2.2.3.2.1 MAC basadas sobre cifrados por bloque

##### 2.2.3.2.1.1 Algoritmo CBC-MAC

Este algoritmo hace uso de los encadenamientos de bloques de cifrado, cuando DES es usado como un cifrado por bloque E con n=64 y la clave MAC es una clave DES de 56 bit [Van+, 1996].

*Algoritmo CBC-MAC*

Entrada:

- Dato x
- Especificación de cifrado por bloque E
- Clave secreta MAC k para E.

Salida:

- MAC sobre x de n bit (n es el largo del bloque de E)

Proceso:

- Rellenado, rellenar x si es necesario, dividir el texto rellenado en bloques $x_1,\ldots,x_t$ de n bit, el algoritmo de rellenado es:
    1. Entrada, Dato x, largo en bit n para el tamaño de bloque de datos de entrada para etapa de procesamiento.
    2. Salida, dato rellenado $x'$ con largo en bit a múltiplo de n.

3. Proceso

   - Anexar a x un bit.
   - Entonces anexar algunos bits 0 como sea necesario para obtener una cadena $x'$ cuyo largo en bit es múltiplo de n.

- Procesamiento CBC, $E_K$ denota encriptación usando E con clave k, calcular el bloque $H_t$ como sigue:

$$H_1 \leftarrow E_K(x_1)$$
$$H_i \leftarrow E_K(H_{i-1} \oplus x_i) \quad , \; 2 \le i \le t$$

(Este es el encadenamiento estándar del cifrado por bloques, $IV = 0$, descartando bloques de texto cifrado $C_i = H_i$).

- Proceso opcional para incrementar fortaleza de MAC, usar una segunda clave secreta $k' \neq k$, calcular opcionalmente:

$$H'_t \leftarrow E^{-1}_{K'}(H_t)$$
$$H_t \leftarrow E_K(H'_t)$$

(Usar dos claves es equivalente una triple encriptación sobre el último bloque)

- Completación, la MAC es el bloque $H_t$ de n bit.

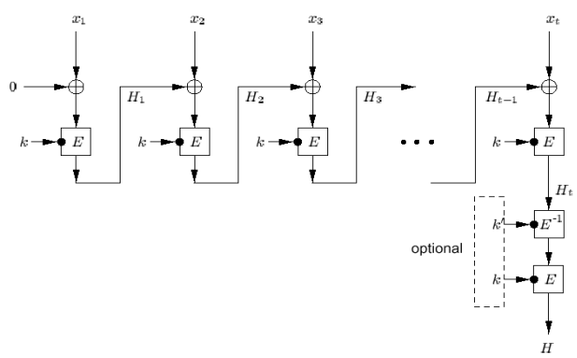



**Fig. 5.2.8 Diagrama de algoritmo CBC-MAC**

#### 2.2.3.2.2 Construcciones MAC desde MDC

Una sugerencia común es la construcción de un algoritmo MAC desde un algoritmo MDC, simplemente incluyendo una clave secreta k como parte de la entrada MDC. Inquietudes con este acercamiento son a menudo hechas acerca de las propiedades que tiene MDC, en particular mientras más MDC son diseñados para proveer un camino o resistencia a la coalisión, los requerimientos de un algoritmo MAC van haciendose distintos. Incluso en el caso de que una función Hash de un solo camino imposibilite recuperar la clave secreta usada como un mensaje parcial de entrada, esto hace no garantizar factibilidad de producir MAC para nuevas entradas.

#### 2.2.3.2.3 MAC personalizadas

##### 2.2.3.2.3.1 Algoritmo MAA (Message Authenticator Algorithm)

Este algoritmo data de 1983, su loop principal consiste de dos flujos de calculos paralelos interdependientes, los mensajes son procesados en bloques de 4 bytes usando variables de cambio de 8 bytes, su tiempo de ejecución es proporcional al largo del mensaje, MAA es dos veces más lento que MD4 [Van+, 1996].

*Algoritmo MAA*

Entrada:

- Dato x de largo en bit 32j con $1 \leq j \leq 10^6$ .
- Clave secreta MAC $Z = Z[1]..Z[8]$ de 64 bit.

Salida:

- MAC sobre x de 32 bit.

Proceso:

- Expansión de clave independiente del mensaje, expandir la clave Z para 6 cantidades X, Y, V, W, S y T de 32 bit (X, Y son valores iniciales; V, W son variables del loop principal; S, T son agregadas para el mensaje) de la siguiente forma:

  1. Reemplazar algunos bytes 0x00 o 0xff en Z de la siguiente forma:
     - $P \leftarrow 0$
     - $for\ i = 1\ to\ 8$
       a. $P \leftarrow 2P$
       b. $if\ Z[i] = 0x00\ \ or\ \ 0xff\ then$
          - $P \leftarrow P + 1$
          - $Z[i] \leftarrow Z[i]\ OR\ P$
  2. Dejar J y K ser los primeros 4 bytes y los últimos 4 byte a Z y calcular:
     - $X \leftarrow J^4 (\text{mod } 2^{32} - 1) \oplus J^4 (\text{mod } 2^{32} - 2)$
     - $Y \leftarrow \left[K^5 (\text{mod} 2^{32} - 1) \oplus K^5 (\text{mod} 2^{32} - 2)\right] \bullet (1 + P)^2 (\text{mod } 2^{32} - 2)$
     - $V \leftarrow J^6 (\text{mod } 2^{32} - 1) \oplus J^6 (\text{mod } 2^{32} - 2)$
     - $W \leftarrow K^7 (\text{mod } 2^{32} - 1) \oplus K^7 (\text{mod } 2^{32} - 2)$
     - $S \leftarrow J^8 (\text{mod } 2^{32} - 1) \oplus J^8 (\text{mod } 2^{32} - 2)$
     - $T \leftarrow K^9 (\text{mod } 2^{32} - 1) \oplus K^9 (\text{mod } 2^{32} - 2)$

3. Procesar el resultado en parejas $(X,Y),(V,W),(S,T)$ para remover algunos bytes 0x00 o 0xff como en Z. Definir las constantes del AND y OR:
    - $A = 0x02040801$
    - $B = 0x00804021$
    - $C = 0xbfef7fdf$
    - $D = 0x7dfefbff$

- Inicialización y preprocesamiento, inicializar el vector de rotación $v \leftarrow V$ y las variables de cambio $H_1 \leftarrow X, H_2 \leftarrow Y$. Agregar los bloques S, T derivados de la clave para x, y $x_1 \ldots x_t$ denotan el segmento de resultados aumentados de bloques de 32 bit.
- Procesamiento de bloques, procesar cada bloque $x_i$ de 32 bit.
    1. $for\ i = 1\ to\ t$
        - $v \leftarrow (v \lfloor\!\leftarrow 1)$
        - $U \leftarrow (v \oplus W)$
        - $t_1 \leftarrow (H_1 \oplus x_i) \times_1 (((H_2 \oplus x_i) + U) ORA) ANDC)$
        - $t_2 \leftarrow (H_2 \oplus x_i) \times_2 (((H_1 \oplus x_i) + U) ORB) ANDD)$
        - $H_1 \leftarrow t_1$
        - $H_2 \leftarrow t_2$

Donde $\times_i$ denota una multiplicación especial $\bmod\ 2^{32} - i$ con i=1 y i=2, "+" es una adición $\bmod\ 2^{32}$ y "$\lfloor\!\leftarrow 1$" denota rotación a la izquierda en 1 bit (cada operación combinada AND-OR sobre cantidades de 32 bit coloca 4 bit a 1 y 4 a 0).

- El resultado MAC es $H = H_1 \oplus H_2$.

#### 2.2.3.2.3.2 Algoritmo MD5-MAC

Una aproximación más conservadora para construir una MAC desde un MDC es para definir que la función de compresión MAC dependa de si misma sobre k, implicando que la clave secreta este involucrada en todas las iteraciones, esto suministra una protección adicional en el caso de que la vulnerabilidad de la función Hash fundamental llegue a ser conocida [Van+, 1996].

*Algoritmo MD5-MAC*

Entrada:

- Cadena de bit x de largo arbitrario en bit $b \geq 0$.
- Clave k de largo en bit $\leq 128$.

Salida:

- MAC sobre x de 64 bit.

Proceso:

- Este algoritmo es obtenido desde MD5 haciendo los siguientes cambios.
- Constantes, la constante $T_i$ es derivada sobre si misma usando MD5 sobre constantes predefinidas y $U_i = T_i \,||\, T_{i+1} \,||\, T_{i+2} \,||\, T_i \,||\, T_{i+1} \,||\, T_{i+2}$.
- Expansión de clave.
    1. Si k es más pequeño que 128 bit, concatenar k consigo mismo para un número suficiente de tiempos y redefinir k con los 128 bit más a la izquierda.

2. $\overline{MD5}$ denota a MD5 mediante el rellenado y anexado omitiendo el largo, expandir k en 3 subclaves $K_0, K_1$ y $K_2$ de la siguiente forma:
    - $for\ i = 0\ \ to\ \ 2$
        a. $K_i \leftarrow \overline{MD5}(k \,||\, U_i \,||\, k)$
3. Particionar $K_0, K_1$ en cuatro subcadenas $K_j[i]$ de 32 bit con $0 \leq i \leq 3$.

- $K_0$ reemplaza los cuatro IV de MD5 de 32 bit (ejemplo: $h_i = K_0[i]$).
- $K_1[i]$ es agregado a $\bmod 2^{32}$ para cada constante $y[j]$ usada en el ciclo i de MD5.
- $K_2$ es usada para construir el siguiente bloque de 512 bit, el que es anexado al relleno de la entrada x subsiguiente para el rellenado regular, el largo del bloque es definido por MD5: $K_2 \,||\, K_2 \oplus T_0 \,||\, K_2 \oplus T_1 \,||\, K_2 \oplus T_2$.
- El valor MAC son los 64 bit más a la izquierda de los 128 bit de la salida.

**2.2.3.2.4 MAC para cifrados por flujo**

**2.2.3.2.4.1 Algoritmo CRC basado sobre MAC**

Una familia Hash H (b, m) es una colección de funciones Hash mapeando mensajes de b bit para valores Hash de m bit, una familia de funciones Hash es $\xi$ balanceada si para todos los mensajes $B \neq 0$ y todos los valores Hash c de m bit se

tiene $prob_h(h(B) = c) \leq \xi$ donde la probabilidad son en general funciones $h \in H$ seleccionadas aleatoriamente.

*Algoritmo CRC sobre MAC*

Entrada:

- Mensaje B de b bit.
- Clave compartida entre la MAC de origen y el verificador.

Salida:

- MAC sobre B de m bit.

Proceso:

- Asociar $B = B_{b-1}...B_1B_0$ con el polinomio $B(x) = \sum_{i=0}^{b-1} B_i x^i$ .
- Selección de la clave MAC
    1. Seleccionar un binario irreducible polinomial $p(x)$ aleatorio de grado m (este representa una función aleatoria h desde una familia Hash (b, m)).
    2. Selección de una clave k one-time aleatoria de m bit (para ser usada como one-time-pad).

  La clave secreta MAC consiste de $p(x)$ y k, ambas deberian ser compartidas a priori entre el originador MAC y el verificador.
- Calcular $h(B) = coef(B(x) \bullet x^m \bmod p(x))$, la cadena de coeficientes de m bit desde el grado $m-1$corresponde al residuo polinomial después de dividir $B(x) \bullet x^m$ por $p(x)$.
- El valor MAC para B es $h(B) \oplus k$ de m bit.

## 2.3 Análisis de algoritmos asimétricos

### 2.3.1 Algoritmo RSA

Este criptosistema debe su nombre a sus creadores R. Rivest, A. Shamir y L. Adleman y corresponde al criptosistema más ampliamente usado, puede ser usado para proveer confidencialidad y firma digital. En cuanto a su seguridad, esta basada sobre la inflexibilidad del problema de factorización entera [RSA+, 1998].

*Algoritmo EE (Eucledíano Extendido)*

- Entrada, dos enteros no negativos a y b con $a \geq b$.
- Salida, $d = gcd(a,b)$ y enteros x e y satisfaciendo $ax + by = d$.
- Proceso.
    - *if* $b = 0$ *then*
        - $d \leftarrow a$
        - $x \leftarrow 1$
        - $y \leftarrow 0$
        - Retornar d, x, y.
    - $x_2 \leftarrow 1, x_1 \leftarrow 0$
    - $y_2 \leftarrow 0, y_1 \leftarrow 1$
    - Mientras $b > 0$ hacer
        - $q \leftarrow [a/b], r \leftarrow a - qb$
        - $x \leftarrow x_2 - qx_1, y \leftarrow y_2 - qy_1$
        - $a \leftarrow b, b \leftarrow r, x_2 \leftarrow x_1$
        - $x_1 \leftarrow x, y_2 \leftarrow y_1, y_1 \leftarrow y$

- $d \leftarrow a$ , $x \leftarrow x_2$ , $y \leftarrow y_2$
- Retornar d, x, y.

*Algoritmo Generador de Claves para RSA*

Cada entidad crea una clave pública RSA y una correspondiente clave privada.

Proceso:

- Cada entidad A debe hacer lo siguiente.
  1. Generar dos números primos aleatorios grandes y distintos $p$ y $q$ , cada uno aproximadamente del mismo tamaño.
  2. Calcular $n = pq$ y $\phi = (p-1)(q-1)$.
  3. Seleccionar un entero aleatorio e con $1 < e < \phi$, tal que $gcd(e, \phi) = 1.$
  4. Usar un algoritmo euclediano extendido para calcular el entero único d con $1 < d < \phi$, tal que $ed \equiv 1 (\mathrm{mod}\, \phi)$.
     a. Call EE()
  5. A tiene clave pública esta en (n, e), A tiene clave privada es d.

*Algoritmo RSA*

Los enteros e, d de la generación de claves son llamados exponente de encriptación y exponente de desencriptación respectivamente, mientras n se llama módulo. B encripta un mensaje m para A y A lo desencripta.

Encriptación:

- B debe hacer lo siguiente.
  1. Obtener la clave pública autorizada (n, e) de A.

2. Representar el mensaje como un entero m en el intervalo $[0, n-1]$.
3. Calcular $c = m^e \bmod n$ usando el siguiente algoritmo.
    - Entrada, $a \in Z_n$ y el entero $0 \le k < n$ cuya representación binaria es $k = \sum_{i=0}^{t} k_i 2^i$.
    - Salida, $a^k \bmod n$.
    - Proceso
        a. $b \leftarrow 1$
        b. $if\ k = 0\ then\ return(b)$
        c. $A \leftarrow a$
        d. $if\ k_0 = 1\ then\ b \leftarrow a$
        e. $for\ i = 1\ to\ t$
            - $A \leftarrow A^2 \bmod n$
            - $if\ k_i = 1 then\ b \leftarrow A \bullet b \bmod n$
        f. $return(b)$
4. Enviar el texto cifrado c a A.

Desencriptación:

- Para recuperar el texto plano m desde c, A debe hacer lo siguiente.
    1. Usar su clave privada d para recuperar $m = c^d \bmod n$.

### 2.3.2 Algoritmo RABIN

Este constituye el primer ejemplo de un esquema de encriptación de clave pública probablemente seguro, el problema confronta a un adversario pasivo que

recupera texto plano de un determinado texto cifrado y computacionalmente es equivalente a factorizar. Su seguridad esta basada sobre el problema de factorización entera compuesto de raíces cuadradas módulo n [Odl+, 1993].

*Algoritmo Raíces*

Entrada:

- Un número primo impar p.
- Un a cuadrado $\in Q_p$ .

Salida:

- Dos raíces cuadradas de a módulo p.

Proceso:

- Seleccionar un b aleatorio $\in Z_p$ hasta que $b^2 - 4a$ no sea un residuo cuadrático módulo p, ejemplo $\left(\frac{b^2 - 4a}{p}\right) = -1$.
- f debe ser polinomial $x^2 - bx + a$ en $Z_P[x]$.
- Calcular $r = x^{(p+1)/2} \bmod f$ (r podría ser entero).
- Retornar (r,-r)

*Algoritmo Raíces Cuadradas*

Entrada:

- Un número entero n.
- Sus factores primos p y q
- Un a $\in Q_p$ .

Salida:

- Cuatro raíces cuadradas de a módulo p.

Proceso:

- Call Raíces, para encontrar las dos raíces cuadradas r y –r de a módulo p.
- Call Raíces, para encontrar las dos raíces cuadradas s y –s de a módulo q.
- Usar el algoritmo euclediano extendido para encontrar enteros c y d tal que $cp + dq = 1$.
  - Call EE().
- $x \leftarrow (rdq + scp) \bmod n$
- $y \leftarrow (rdq - scp) \bmod n$
- Retornar $(\pm x \bmod n, \pm y \bmod n)$

*Algoritmo Generador de Claves para RABIN*

Cada entidad crea una clave pública y una correspondiente clave privada.

Proceso:

- Cada entidad A debe hacer lo siguiente.
  1. Generar dos números primos aleatorios grandes y distintos $p$ y $q$, cada uno aproximadamente del mismo tamaño.
  2. Calcular $n = pq$.
  3. A tiene clave pública es n, A tiene clave privada es (p, q).

*Algoritmo RABIN*

B encripta un mensaje m para A y A lo desencripta.

Encriptación:

- B debe hacer lo siguiente.

1. Obtener la clave pública autorizada n de A.
2. Representar el mensaje como un entero m en el intervalo $\{0,1,\ldots,n-1\}$.
3. Calcular $c = m^2 \bmod n$.
4. Enviar el texto cifrado c a A

Desencriptación:

- Para recuperar el texto plano m desde c, A debe hacer lo siguiente.
  1. Usar el algoritmo de siguiente para encontrar las 4 raíces cuadradas $m_1, m_2, m_3, m_4$ de $c \bmod n^2$.
     - Call Raíces Cuadradas().
  2. Cada mensaje enviado tiene $m_1, m_2, m_3, m_4$, una de estas es m.

### 2.3.3 Algoritmo ElGamal

Este esquema puede ser visto como una clave Diffie-Hellman de acuerdo con el modo de transferir la clave, su seguridad esta basada sobre la inflexibilidad del problema del logaritmo discreto y del problema de Diffie-Hellman [RSA+, 1998]. Solo se analizar el algortimo básico, ya que también se tiene una versión generalizada.

*Algoritmo RANDOM-SEARCH(k,t)*

Entrada:

- Un entero k.
- Un parámetro seguro t.

Salida:

- Un primo aleatorio de k bit.

Proceso:

- Generar un entero impar aleatorio n de k bit.
- Usar división de prueba para determinar si n es divisible por algún primo impar $\leq B$, si existe entonces volver al paso 1.
- Si la salida de MILLER-RABIN(n, t) es primo entonces retornar n, sino volver al paso 1.

*Algoritmo Generador de un Grupo Ciclico*

Entrada:

- Un grupo ciclico G de orden n.
- Una factorización prima $n = p_1^{e1} p_2^{e2} ... p_K^{ek}$.

Salida:

- Un generador $\alpha$ de G.

Proceso:

- Elegir un elemento aleatorio $\alpha$ en G.
- $for\ i = 1\ to\ k$
  - Calcular $b \leftarrow \alpha^{n/p_i}$.
  - $if\ b = 1\ then$ volver al paso 1
- Retornar $\alpha$.

*Algoritmo Generador de Claves para ElGamal*

Cada entidad crea una clave pública y una correspondiente clave privada.

Proceso:

- Cada entidad A debe hacer lo siguiente.

1. Generar un número primo aleatorio grande $p$ y un generador $\alpha$ del grupo multiplicativo $Z_P^*$ de los enteros módulo p usando el siguiente algoritmo.
    - Entrada, el largo en bit k requerido para el número primo y un parametro de seguridad t.
    - Salida, un número primo p de k bit, tal que p-1 tenga un factor primo $\geq t$ y un generador $\alpha$ de $Z_P^*$.
    - Proceso
        a. Repetir lo siguiente hasta que p-1 tenga un factor primo $\geq t$.
            - Seleccionar un número primo p aleatorio de k bit, usar RANDOM-SEARCH(k, t).
            - Factor p-1
        b. Usar algoritmo Grupo Ciclico con $G = Z_P^*$ y $n = p - 1$ para encontrar un generador $\alpha$ de $Z_P^*$.
        c. Retornar p y $\alpha$.
2. Seleccionar un entero aleatorio a con $1 \leq a \leq p - 2$ y calcular $\alpha^a \bmod p$.
3. A tiene clave pública es $(p, \alpha, \alpha^a)$, A tiene clave privada es a

*Algoritmo ElGamal*

B encripta un mensaje m para A y A lo desencripta.

Encriptación:

- B debe hacer lo siguiente.
  1. Obtener la clave pública autorizada $(p, \alpha, \alpha^a)$ de A.
  2. Representar el mensaje como un entero m en el intervalo $\{0,1,\ldots,p-1\}$.
  3. Seleccionar un entero aleatorio k con $1 \leq k \leq p-2$.
  4. Calcular $\gamma = \alpha^k \bmod p$ y $\delta = m \bullet (\alpha^a)^k \bmod p$.
  5. Enviar el texto cifrado $c = (\gamma, \delta)$ a A

Desencriptación:

- Para recuperar el texto plano m desde c, A debe hacer lo siguiente.
  1. Usar la clave privada a para calcular $\gamma^{p-1-a} \bmod p$ (notar que $\gamma^{p-1-a} = \gamma^{-a} = \alpha^{-ak}$).
  2. Recuperar m por el cálculo $(\gamma^{-a}) \bullet \delta \bmod p$.

### 2.3.4 Algoritmo McEliece

Este esquema esta basado sobre codigos de corrección de errores, la idea inicial de este esquema es que para la primera selección de un código particular para lo cual un eficiente algoritmo de decodificación es conocido para esconder el código como un código lineal general [Van+, 1996]. Hasta la fecha este esquema ha resistido criptoanálisis, es una encriptación muy eficiente pero ha recibido poca atención en la práctica por causa de sus muy largas claves públicas.

*Algoritmo Generador de Claves para McEliece*

Cada entidad crea una clave pública y una correspondiente clave privada.

Proceso:

- Enteros k, n, t son fijados como parámetros comunes del sistema.
- Cada entidad A debe hacer lo siguiente.
    1. Escoger una matriz generadora G de kxn para un código lineal binario (n, k) el cual puede corregir t errores y para ello un eficiente algoritmo decodificador es conocido.
    2. Seleccionar una matriz S aleatoria no singular binaria de kxk.
    3. Seleccionar una matriz P aleatoria de permutación de nxn.
    4. Calcular la matriz $\overline{G} = SGP$ de kxn.
    5. A tiene clave pública es $(\overline{G}, t)$, A tiene clave privada es $(S, G, P)$.

*Algoritmo McEliece*

B encripta un mensaje m para A y A lo desencripta.

Encriptación:

- B debe hacer lo siguiente.
    1. Obtener la clave pública autorizada $(\overline{G}, t)$ de A.
    2. Representar el mensaje como una cadena binaria m de largo k.
    3. Seleccionar aleatoriamente un vector de error binario z de largo n teniendo como máximo t unos (1).
    4. Calcular el vector binario $c = m\overline{G} + z$.
    5. Enviar el texto cifrado c a A

Desencriptación:

- Para recuperar el texto plano m desde c, A debe hacer lo siguiente.

  1. Calcular $\overline{c} = cP^{-1}$, donde $P^{-1}$ es la inversa de la matriz P.
  2. Usar el algoritmo decodificador para el código generado por G para decodificar $\overline{c}$ para $\overline{m}$.
  3. Calcular $m = \overline{m}S^{-1}$.

### 2.3.5 Algoritmo knapsack Merkle-Hellman

Este esquema intenta ocultar fácilmente la solución del problema de una suma de subconjunto, llamado problema de una suma de subconjunto super creciente por medio de múltiplicación modular y una permutación. Entendiéndose como una secuencia super creciente a una secuencia $(b_1, b_2, ..., b_n)$ de enteros positivos con la propiedad que $b_i > \sum_{j=1}^{i-1} b_j$ para cada i con $2 \leq i \leq n$ [Van+, 1996].

*Algoritmo para Solucionar Problema de Suma de Subconjunto Super Creciente*

Entrada:

- Una secuencia $(b_1, b_2, ..., b_n)$ super creciente.
- Un entero s, el cual es la suma de un subconjunto de los $b_i$.

Salida:

- $(x_1, x_2, ..., x_n)$ donde $x_i \in \{0,1\}$ tal que $\sum_{i=1}^{n} x_i b_i = s$.

Proceso:

- $i \leftarrow n$

- Mientras $i \geq 1$ hacer
  - *if* $s \geq b_i$ *then*
    - $x_i \leftarrow 1$
    - $s \leftarrow s - b_i$
  - *else* $x_i \leftarrow 0$
  - $i \leftarrow i - 1$
- Retornar $(x_1, x_2, ..., x_n)$.

*Algoritmo Generador de Claves para Knapsack Merkle-Hellman*

Cada entidad crea una clave pública y una correspondiente clave privada.

Proceso:

- Un entero n es fijado como parámetro común del sistema.
- Cada entidad A debe hacer lo siguiente.
  1. Escoger una secuencia super creciente $(b_1, b_2, ..., b_n)$ y un módulo M tal que $M > b_1 + b_2 + ... + b_n$.
  2. Seleccionar un entero aleatorio W con $1 \leq W \leq M - 1$ tal que $gcd(W, M) = 1$.
  3. Seleccionar una permutación aleatoria $\pi$ de los enteros $\{1,2,...,n\}$.
  4. Calcular $a_i = Wb_{\pi(i)} \bmod M$ para $i = 1,2,...,n$.
  5. A tiene clave pública es $(a_1, a_2, ..., a_n)$, A tiene clave privada es $(\pi, M, W, (b_1, b_2, ..., b_n))$.

*Algoritmo Knapsack Merkle-Hellman*

B encripta un mensaje m para A y A lo desencripta.

Encriptación:

- B debe hacer lo siguiente.
  1. Obtener la clave pública autorizada $(a_1, a_2, ..., a_n)$ de A.
  2. Representar el mensaje como una cadena binaria m de largo n con $m = m_1 m_2 ... m_n$.
  3. Calcular el entero $c = m_1 a_1 + m_2 a_2 + ... + m_n a_n$.
  4. Enviar el texto cifrado c a A

Desencriptación:

- Para recuperar el texto plano m desde c, A debe hacer lo siguiente.
  1. Calcular $d = W^{-1} c \mod M$.
  2. Utilizar algoritmo para solucionar el problema de suma del subconjunto super creciente, encontrar enteros $r_1, r_2, ..., r_n$ con $r_i \in \{0,1\}$ tal que $d = r_1 b_1 + r_2 b_2 + ... + r_n b_n$.
  3. El mensaje en bits es $m_i = r_{\pi(i)}$ con $i = 1,2,...,n$.

### 2.3.6 Algoritmo knapsack Chor-Rivest

Este esquema no usa una forma de multiplicación modular simple para ocultar el problema de suma de subconjunto [Van+, 1996].

*Algoritmo Generador de Claves para Knapsack Chor-Rivest*

Cada entidad crea una clave pública y una correspondiente clave privada.

Proceso:

- Cada entidad A debe hacer lo siguiente.

1. Seleccionar un area finita $F_q$ de características p, donde $q = p^h$ con $p \geq h$ y para lo cual el problema de logaritmo discreto es factible.
2. Seleccionar una función polinomial aleatoria irreducible $f(x)$ de grado h sobre $Z_p$. Los elementos de $F_q$ pueden ser representados como polinomios en $Z_p[x]$ de grado menor a h con la ejecución de la múltiplicación módulo $f(x)$.
3. Seleccionar un elemento primitivo aleatorio $g(x)$ de $F_q$, usar algoritmo Grupo Ciclico.
4. Para cada elemento de area $i \in Z_P$ encontrar el logaritmo discreto $a_i = \log_{g(x)}(x+i)$ del elemento de area $(x+i)$ para la base $g(x)$.
5. Seleccionar una permutación aleatoria $\pi$ sobre el conjunto de enteros $\{0,1,2,...,p-1\}$.
6. Seleccionar un entero aleatorio d con $0 \leq d \leq p^h - 2$.
7. Calcular $c_i = (a_{\pi(i)} + d) \bmod (p^h - 1)$ con $0 \leq i \leq p-1$.
8. A tiene clave pública es $((c_0, c_1, ..., c_{p-1}), p, h)$, A tiene clave privada es $(f(x), g(x), \pi, d)$.

*Algoritmo Knapsack Chor-Rivest*

B encripta un mensaje m para A y A lo desencripta.

Encriptación:

- B debe hacer lo siguiente.

1. Obtener la clave pública autorizada $((c_0,c_1,...,c_{p-1}),p,h)$ de A.
2. Representar el mensaje como una cadena binaria m de largo $\left\lfloor \lg \binom{p}{h} \right\rfloor$ donde $\binom{p}{h}$ es un coeficiente binomial.
3. Considere m como una representación binaria de un entero, transforme este entero en un vector binario $M=(M_0,M_1,...,M_{p-1})$ de largo p teniendo exactamente h unos (1).
   - o $l \leftarrow h$
   - o *for* $i=1$ *to* $p$
     a. *if* $m \geq \binom{p-i}{l}$ *then*
        - $M_{i-1} \leftarrow 1$
        - $m \leftarrow m-\binom{p-i}{l}$
        - $l \leftarrow l-1$
     b. *else* $M_{i-1} \leftarrow 0$
     c. (Notar que $\binom{n}{0}=1$ para $n \geq 0$ y $\binom{0}{l}=0$ para $l \geq 1$)
4. Calcular $c=\sum_{i=0}^{p-1} M_i c_i \bmod (p^h-1)$.

5. Enviar el texto cifrado c a A.

Desencriptación:

- Para recuperar el texto plano m desde c, A debe hacer lo siguiente.
  1. Calcular $r = (c - hd) \bmod (p^h - 1)$.
  2. Calcular $u(x) = g(x)^r \bmod f(x)$.
  3. Calcular $s(x) = u(x) + f(x)$, un polinomio de grado h sobre $Z_p$.
  4. Factorizar $s(x)$ en factores lineales sobre $Z_P : s(x) = \prod_{j=1}^{h}(x + t_j)$ donde $t_j \in Z_P$.
  5. Calcular el vector binario $M = (M_0, M_1, ..., M_{p-1})$ como sigue: el componente de M que es 1 tiene indices $\pi^{-1}(t_j)$ con $1 \le j \le h$. Los componentes restantes son 0.
  6. El mensaje m es recuperado desde M como sigue:
     - $m \leftarrow 0, l \leftarrow h$
     - *for* $i = 1$ *to* $p$
       a. *if* $M_{i-1} = 1$ *then*
          - $m \leftarrow m + \binom{p-i}{l}$
          - $l \leftarrow l - 1$

### 2.3.7 Encriptación asimétrica probabilística

#### 2.3.7.1 Algoritmo Goldwasser-Micali

Este esquema esta .basado sobre la inflexibilidad del problema del residuo cuadrático [Van+, 1996].

*Algoritmo Generador de Calculo Lagrange*

*Jacobi(a, n)*

Entrada:

- Un entero impar $n \geq 3$.
- Un entero a con $0 \leq a < n$.

Salida:

- El símbolo de Jacobi $\left(\frac{a}{n}\right)$, y por lo tanto el símbolo de Lagrange cuando n es primo.

Proceso:

- *if* $a = 0$ *then return*(0)
- *if* $a = 1$ *then return*(1)
- Escribir $a = 2^e a_1$ donde $a_1$ sea impar.
- *if* $e = cte$ *then*
  - $s \leftarrow 1$
- *else*
  - *if* $n \equiv 1$ *or* $n \equiv (7 \bmod 8)$ *then* $s \leftarrow 1$
  - *if* $n \equiv 3$ *or* $n \equiv (5 \bmod 8)$ *then* $s \leftarrow -1$
- *if* $n \equiv (3 \bmod 4)$ *and* $a_1 \equiv (3 \bmod 4)$ *then* $s \leftarrow -s$

- $n_1 \leftarrow n \mod a_1$
- *if* $a_1 = 1$ *then*
  - *return*$(s)$
- *else* *return*$(s \bullet JACOBI(n_1, a_1))$

*Algoritmo Generador de Claves para Goldwasser-Micali*

Cada entidad crea una clave pública y una correspondiente clave privada.

Proceso:

- Cada entidad A debe hacer lo siguiente.
  1. Seleccionar dos números primos largos y aleatorios p y q (distintos) cada uno aproximadamente del mismo tamaño.
  2. Calcular $n = pq$ .
  3. Seleccionar un $y \in Z_n$ tal que y es un residuo módulo n no cuadrático y el símbolo de Jacobi $\left(\frac{y}{n}\right) = 1$ ("y" es seudo cuadrado módulo n).
  4. A tiene clave pública es (n, y), A tiene clave privada es (p, q).

*Algoritmo Goldwasser-Micali*

B encripta un mensaje m para A y A lo desencripta.

Encriptación:

- B debe hacer lo siguiente.
  1. Obtener la clave pública autorizada (n, y) de A.
  2. Representar el mensaje m como una cadena binaria $m = m_1 m_2 \ldots m_t$ de largo t.

3. $for\ i = 1\ to\ t$

   - Elegir un $x \in Z_n^*$ aleatorio.
   - $if\ m_i = 1\ then$
     a. $c_i \leftarrow yx^2 \bmod n$
   - $else\ c_i \leftarrow x^2 \bmod n$

4. Enviar la t-tupla $c = (c_1, c_2, ..., c_t)$ a A.

Desencriptación:

- Para recuperar el texto plano m desde c, A debe hacer lo siguiente.

1. $for\ i = 1\ to\ t$

   - Calcular el símbolo de Lagrange $e_i = \left(\frac{c_i}{p}\right)$, usando el algoritmo de Lagrange $JACOBI(c_i, p)$.
   - $if\ e_i = 1\ then$
     a. $m_i \leftarrow 0$
   - $else\ m_i \leftarrow 1$

2. El mensaje desencriptado es $m = m_1 m_2 ... m_t$:

**2.3.7.2 Algoritmo Blum-Goldwasser**

Este esquema es el más eficiente dentro de los esquemas probabilisticos y es comparable con el esquema de encriptación RSA y esta .basado sobre la inflexibilidad del problema de factorización entera. Utiliza el generador de números seudo aleatorios Blum-Blum-Shub para generar secuencias de bit seudo aleatorias, las cuales se usan

con una operación XOR con el texto plano [Van+, 1996]. La secuencia de bit obtenida en conjunto con una encriptación y una semilla aleatoria es transmitida.

*Algoritmo Generador de Claves para Blum-Goldwasser*

Cada entidad crea una clave pública y una correspondiente clave privada.

Proceso:

- Cada entidad A debe hacer lo siguiente.
  1. Seleccionar dos números primos largos y aleatorios p y q (distintos) cada uno congruente para 3 módulo 4.
  2. Calcular $n = pq$ .
  3. Utilizar el algoritmo Euclediano Extendido para calcular los enteros a y b tal que $ap + bq = 1$. Call EE().
  4. A tiene clave pública es n, A tiene clave privada es $(p,q,a,b)$.

*Algoritmo Blum-Goldwasser*

B encripta un mensaje m para A y A lo desencripta.

Encriptación:

- B debe hacer lo siguiente.
  1. Obtener la clave pública autorizada n de A.
  2. $k = \lfloor \lg n \rfloor, h = \lfloor \lg k \rfloor$
  3. Representar el mensaje m como una cadena $m = m_1 m_2 ... m_t$ de largo t, donde cada $m_i$ es una cadena binaria de largo h.

4. Seleccionar una semilla $x_0$, un residuo cuadrático módulo n aleatorio (este puede ser seleccionado de un entero aleatorio $r \in Z_n^*$ y fijado como $x_0 \leftarrow r^2 \bmod n$).
5. *for* $i = 1$ *to t*
    - Calcular $x_i = x_{i-1}^2 \bmod n$.
    - Dejar $p_i$ ser el h de los bits menos significativos de $x_i$.
    - Calcular $c_i = p_i \oplus m_i$.
6. Calcular $x_{t+1} = x_t^2 \bmod n$.
7. Enviar el texto cifrado $c = (c_1, c_2, ..., c_t, c_{t+1})$ a A.

Desencriptación:

- Para recuperar el texto plano m desde c, A debe hacer lo siguiente.
    1. Calcular $d_1 = ((p+1)/4)^{t+1} \bmod (p-1)$.
    2. Calcular $d_2 = ((q+1)/4)^{t+1} \bmod (q-1)$.
    3. Calcular $u = x_{t+1}^{d1} \bmod p$.
    4. Calcular $v = x_{t+1}^{d2} \bmod q$.
    5. $x_0 = vap + ubq \bmod n$
    6. *for* $i = 1$ *to t*
        - $x_i = x_{i-1}^2 \bmod n$
        - Dejar $p_i$ ser el h de los bits menos significativos de $x_i$.
        - $m_i = p_i \oplus c_i$