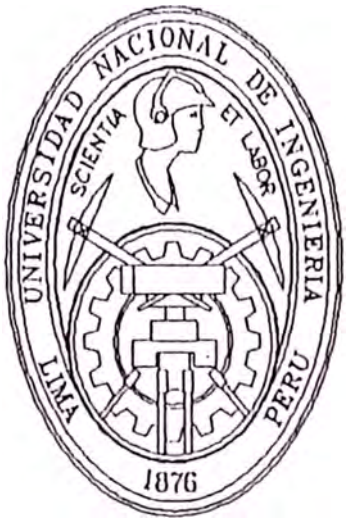
UNIVERSIDAD NACIONAL DE INGENIERIA

FACULTAD DE INGENIERIA ELECTRICA, ELECTRONICA



Implementación de un Sistema Cliente/Servidor para Transmisión Encriptada de Voz y Datos sobre una Red Local utilizando DSP's

TESIS

PARA OPTAR EL TITULO PROFESIONAL DE

Ingeniero Electrónico

PRESENTADA POR:

Marlon Joel Jiménez Bazán

PROMOCION 1999-I

LIMA - PERU

2000

A Gilberto y Graciela,

mis Padres.

IMPLEMENTACION DE UN SISTEMA CLIENTE/SERVIDOR PARA TRANSMISION ENCRIPTADA DE VOZ Y DATOS SOBRE UNA RED LOCAL UTILIZANDO DSP'S

SUMARIO

En este trabajo de tesis se propone un Sistema para transferencia de Voz y Datos, en el cual se utiliza técnicas de encriptación que proporcionan seguridad en la transferencia de información. Además se consideran técnicas de compresión de datos para conseguir mayor eficiencia en la transmisión de la aplicación de voz. El Sistema de encriptación utilizado se basa en el International Data Encryption Algorithm (IDEA), método que garantiza la correcta recuperación de los datos procesados mediante microoperaciones aritméticas. La implementación del sistema se realiza sobre la tarjeta de Procesamiento Digital de Señales DSP56002 de Motorola, la que proporciona características apropiadas para el procesamiento de datos. El método de compresión utilizado es ADPCM, que es implementado en base a rutinas de software. Finalmente se monta una aplicación de transferencia de información sobre una red local utilizando sockets TCP/IP para la comunicación, sobre el entorno Linux, aplicándose tanto para el caso de transferencia de archivos como para el caso de voz, obteniéndose resultados adecuados y sin pérdida de información.

Un agradecimiento especial al Dr. José Paz Campaña, asesor en todo momento y quien más contribuyo conmigo en la orientación necesaria. Para el Dipl. Ing. Javier Donayre Sánchez, mi asesor; para el Ph.D. Luis Herrera Bendezú; PhD. Arturo Rojas Moreno, y todos aquellos que de una u otra manera participaron en este trabajo

INDICE

PROLOGO

En el presente trabajo de Tesis se propone un sistema para la transferencia encriptada de datos, pudiendo transmitirse datos estáticos (archivos de datos) asi como voz, la cual es comprimida a fin de mejorar la eficiencia de la transmisión. Un método de encriptación es utilizado, el que garantiza la total seguridad en la transmisión, debido a que los datos son encriptados en la etapa cliente, y desencriptados por la etapa servidora luego de la recepción. El sistema planteado es aplicable para la transmisión de cualquier tipo de datos, implementándose la etapa de encriptación y recuperación de datos sobre una tarjeta DSP, de modo que se posee un componente de hardware especializado para esta labor.

En el Capítulo 1 denominado **Fundamentos Matemáticos** se hace un repaso de algunos conceptos teóricos que son importantes para poder comprender totalmente las propiedades de encriptación y posterior recuperación de los datos transmitidos. En el Capítulo 2 denominado **Métodos de Encriptación** se analizan y describen los métodos mas utilizados para conseguir un buen nivel de seguridad en la transmisión de datos, analizando las ventajas y diferencias entre uno y otro. El Capítulo 3 denominado **El Algoritmo IDEA,** estudia el International Data Encryption Algorithm, algoritmo estándar de encriptación que es finalmente considerado para la implementación del sistema. En este capítulo se estudia detalladamente la forma en la que IDEA distribuye los datos iniciales y los combina

con la llave de encriptación adoptada, para generar los datos encriptados, los que finalmente son transmitidos. El Capítulo 4 denominado **Técnicas de Compresión de Voz** analiza los métodos de codificación de datos, enfatizando el análisis para el caso de muestras digitalizadas de voz, y demostrando las capacidades de compresión que estos proporcionan. La implementación del sistema de transmisión y recepción, que utilizará compresión de voz y que se basará en IDEA para protejer los datos se describe en el Capítulo 5 denominado **Implementación del Sistema**. Este capítulo muestra la concepción del sistema de comunicación, la interfase entre la PC y el DSP, el módulo encriptador, el módulo compresor/descompresor, y el método de comunicación mediante sockets TCP/IP. Se explica detalladamente las partes cliente y servidor de la aplicación, asi como el protocolo de señalización que se utilizará para la transmisión, debido a que es necesario manejar conceptos de comunicación asíncrona tales como el acuse de recibo.

Los resultados obtenidos en la Tésis, que son en base a la transmisión de archivos de datos, ya sean de texto ASCII o de caracteres binarios en una primera etapa, asi como de transmisión de voz comprimida en la parte final, son mostrados y comentados en el Capítulo 6 denominado **Resultados**. En este capítulo también se analiza los casos de retardos en la transmisión y casos de congestión de la red de datos. Las conclusiones de la presente Tésis y trabajos futuros que pudieran contribuir con diversas aplicaciones a este trabajo, e incluso aportes en su performance y funcionalidades se presentan en el Capítulo denominado **Conclusiones**. Finalmente se considera un **Apéndice** en el que se detallan algunos conceptos generales y terminología utilizados en el trabajo.

CAPITULO I
FUNDAMENTOS MATEMATICOS

He considerado necesario presentar este primer capítulo denominado Fundamentos Matemáticos a fin de presentar algunos conceptos teóricos relacionados al álgebra de módulos y operaciones aritméticas, los cuales serán de suma utilidad en capítulos posteriores en donde se apreciará que son estas operaciones elementales la base de la metodología de encriptación planteada en el presente trabajo.

1.1 Introducción al Algebra

El propósito de esta sección es familiarizar al lector en los conceptos elementales del álgebra. Los conceptos son tratados de una manera básicamente descriptiva y no rigurosamente matemática, a fin de proporcionar el conocimiento básico necesario sobre los tópicos a tratar.

1.1.2 Grupos.- Sea G un conjunto de elementos. Una operación binaria @ en G es una regla de correspondencia que asigna a cada par de elementos a y b un solo tercer elemento dado que c=a@b en G. Cuando se define una operación @ en G, podemos decir que "G es cerrado bajo @". Una operación binaria @ sobre G se dice que es asociativa si para cualquier a, b y c en G, a@(b@c)=(a@b)@c

Ahora nosotros tenemos un útil sistema algebraico denominado grupo.

Definición 1.1: Un conjunto G sobre el cual una operación binaria @ es definida, es denominado grupo si satisface las siguientes condiciones:

- La operación binaria es asociativa
- G contiene un elemento e tal que, para cualquier valor a en G, a@e = e@a = a

  Este elemento es denominado elemento identidad de G.

Para cualquier elemento a en G, existe otro elemento a' en G tal que, a@a' = a'@a = e

El elemento a' es denominado inverso de a (a es también el inverso de a').

G se denomina conmutativo si la operación @ también satisface la siguiente condición, para cualquier a y b en G, a@b = b@a

Ejemplo 1:

Sea m un entero positivo. Tenemos un grupo G=0,1,2,...,m-1. Consideremos + como la suma real. Definimos [+] en G como sigue: Para cualquier i y j en G, i[+]j=r, donde r es el resto de dividir i+j por m. El resto r es un entero entre 0 y m-1 y por lo tanto pertenece a G. De aquí G es cerrado bajo la operación [+], la que se denomina suma módulo-m. De esta forma el conjunto G=0,1,2,...,m-1 es un grupo bajo la suma módulo-m. Se aprecia fácilmente que el elemento identidad es el 0. Además se aprecia que: i[+](m-i) = (m-i)[+]i = 0. De esta manera i y m-i son inversos respecto a [+]. También se demuestra fácilmente que la operación es conmutativa y asociativa [3], y de este modo G es un grupo bajo la suma módulo-m. La sgte. tabla muestra la operación para el caso m=5.

[+]	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	2	2
4	4	0	1	2	3

Ejemplo 2:

Sea p un número primo (por ejemplo p=2,3,5,7,..). Consideremos un conjunto de enteros G=1,2,3,..,p-1. Sea * la operación multiplicación real. Definimos

la operación [*] en G como sigue: Para i y j en G: i [*] j = r, donde r es el resto de dividir i*j por p. Primero notamos que i*j no es divisible por p (porque p es primo), así que 0<r<p y r es un elemento de G. De esta manera el conjunto G es cerrado bajo la operación [*] llamada multiplicación módulo-p. De la misma manera que el caso anterior es sencillo demostrar que [*] es conmutativa y asociativa. El elemento inverso es 1. Un tanto más complejo es demostrar que todo número en G posee un inverso; esto se consigue utilizando los teoremas de Euclides[1]

De esta manera G es un grupo en [*]. Al igual que el caso anterior la tabla mostrada a continuación muestra la operación [*] para p=5.

[*]	1	2	3	4
1	1	2	3	4
2	2	4	1	3
3	3	1	4	2
4	4	3	2	1

1.1.2 Campos.- Ahora mostraremos otro concepto utilizado en los sistemas algebraicos, los llamados campos. Informalmente hablando, un campo es un conjunto de elementos en el cual podemos sumar, restar, multiplicar y dividir sin salir del conjunto. La suma y la multiplicación deben satisfacer las leyes conmutativa, asociativa y distributiva. Una definición formal es dada a continuación.

Definición 1.2: Sea F un conjunto de elementos, en el que dos operaciones binarias llamadas adición "+" y multiplicación "*" son definidas. El conjunto F junto con las dos operaciones + y * es un campo si las sgtes. condiciones son satisfechas:

[1] Mayor información en la Referencia: Error Control Coding: Principles and Applications

i) F es un grupo conmutativo bajo la adición +. El elemento identidad con respecto a la adición es llamado elemento cero o identidad aditiva de F y es denotado por 0.

ii) El conjunto de elementos diferentes de cero en F es un grupo conmutativo bajo la multiplicación *. El elemento identidad con respecto a la multiplicación es llamado elemento unidad o identidad multiplicativo de F, y es denotado por 1.

iii) La multiplicación es distributiva bajo la adición, esto es, para tres elementos cualesquiera a, b y c en F:

$$a*(b+c)=a*b+b*c$$

El número de elementos de F constituye el orden del campo, el que al menos debe poseer los elementos inverso aditivo y multiplicativo. Puede demostrarse que un campo de 2 elementos no existe [3].

1.2 Aritmética de Campos Binarios:

En general, es posible construir códigos en base a cualquier Campo de Galois GF(q), en donde q es cualquier número primo p o una potencia de p. Sin embargo, códigos con símbolos de campos binarios GF(2) o GF(2^m) son ampliamente utilizados en sistemas de transmisión digital de datos o en sistemas de almacenamiento debido a que la información es universalmente codificada en binario por razones prácticas.

En la aritmética binaria nosotros utilizamos la adición y multiplicación módulo 2, las que fueron definidas anteriormente. Esta es equivalente a la aritmética ordinaria, excepto que consideramos que 2 es igual a 0. (por ejemplo 1+1=2=0). Note que $1+1=0 \Rightarrow 1=-1$, de aquí, en aritmética binaria, la sustracción es lo mismo que la adición. Para ilustrar como las ideas de la aritmética ordinaria pueden ser usadas en la aritmética binaria consideramos el siguiente conjunto de ecuaciones:

$$X+Y=1, \qquad X+Z=0, \qquad X+Y+Z=1$$

Estas pueden ser solucionadas restando la primera de la tercera, lo que da Z=0. Reemplazando en la segunda ecuación: X+Z=0, entonces tenemos X=0. Sustituyendo estos valores en la primera ecuación tenemos que Y=1. Es sencillo comprobar que los valores encontrados satisfacen las ecuaciones anteriores.

Al haber sido posible solucionar el sistema anterior, las ecuaciones deben ser linealmente independientes, y el determinante de los coeficientes en el lado izquierdo debe ser diferente de cero. Si el determinante es diferentes de cero, entonces este debe ser 1. Esto puede ser verificado de la manera mostrada:

$$\begin{vmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{vmatrix} = 1 * \begin{vmatrix} 0 & 1 \\ 1 & 1 \end{vmatrix} - 1 * \begin{vmatrix} 1 & 1 \\ 1 & 1 \end{vmatrix} + 0 * \begin{vmatrix} 1 & 0 \\ 1 & 1 \end{vmatrix} = 1 * 1 - 1 * 0 + 0 * 1 = 1$$

Estas ecuaciones pueden ser solucionados utilizando la regla de Cramer:

$$X = \frac{\begin{vmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{vmatrix}}{\begin{vmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{vmatrix}} = \frac{0}{1} = 0, \quad Y = \frac{\begin{vmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{vmatrix}}{\begin{vmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{vmatrix}} = \frac{1}{1} = 1, \quad Z = \frac{\begin{vmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{vmatrix}}{\begin{vmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{vmatrix}} = \frac{0}{1} = 0$$

Ahora consideraremos cálculos con polinomios cuyos coeficientes pertenecen al campo binario GF(2). Un polinomio f(X) con una variable X y con coeficientes de GF(2) posee la siguiente forma:

$$f(X)=f_0 + f_1X + f_2X^2 + \ldots + f_nX^n,$$

donde f_i =0 ó 1 para $0 \leq i \leq n$. El grado del polinomio es la mayor potencia de X con coeficiente no nulo. A partir de ahora utilizaremos el término: "Un polinomio sobre GF(2)" para expresar "un polinomio con coeficientes de GF(2)"". Existen dos

polinomios de grado 2 sobre GF(2): X y 1+X. Existen 4 polinomios de grado 2 sobre GF(2): X^2 , $1+X^2$, $X+X^2$ y $1+X+X^2$. En general existen 2^n polinomios de grado n sobre GF(2).

Los polinomios sobre GF(2) pueden ser sumados (o restados), multiplicados y divididos de la manera usual. De esta forma:

$$g_{(X)} = g_0 + g_1X + g_2X^2 + \ldots + g_m X^m$$

es otro polinomio sobre GF(2). Estos polinomios pueden sumarse y los coeficientes deben considerarse bajo las reglas del álgebra binaria. Por ejemplo si sumamos $a_{(X)} = 1 + X + X^3 + X^5$ y $b(X) = 1 + X^2 + X^3 + X^4 + X^7$ obtenemos:

$$a_{(X)} + b_{(X)} = (1+1) + X + X^2 + (1+1)X^3 + X^4 + X^5 + X^7 = X + X^2 + X^4 + X^5 + X^7$$

Teorema: Cualquier polinomio irreductible de grado m sobre GF(2) divide a $X^{2^m-1}+1$.

Como un ejemplo del teorema podemos mostrar que $X^3 + X + 1$ divide a $X^{2^3-1}+1 = X^7+1$

$$\begin{array}{rl} X^7+0X^6+0X^5+0X^4+0X^3+0X^2+0X+1 & \vert X^3+X+1 \\ \underline{X^7+0X^6+X^5+X^4+0X^3+0X^2+0X+0} & X^4+X^2+X \\ X^5+X^4+0X^3+0X^2+0X+1 & \\ \underline{X^5+0X^4+X^3+X^2+0X+0} & \\ X^4+X^3+X^2+0X+1 & \\ \underline{X^4+0X^3+X^2+X+0} & \\ X^3+X+1 & \\ \underline{X^3+X+1} & \\ 0 & \end{array}$$

Un polinomio irreductible p(X) de grado m es llamado una primitiva si el menor entero positivo n para el que p(X) divide a X^n+1 es $n=2^m-1$. Por ejemplo podemos verificar que $p_{(X)} = X^4+X+1$ divide a $X^{15}+1$ pero no divide a cualquier

polinomio X^n+1 para $1 \leq n \leq 15$. De aquí tenemos que X^4+X+1 es un polinomio primitivo. El polinomio $X^4+X^3+X^2+X+1$ es irreductible pero no es primitivo, debido a que divide a X^5+1. No es sencillo reconocer un polinomio primitivo, por ello se presentan los primeros polinomios primitivos en la tabla mostrada a continuación.

m	
3	$1+X+X^3$
4	$1+X+X^4$
5	$1+X^2+X^5$
6	$1+X+X^6$
7	$1+X^3+X^7$
8	$1+X^2+X^3+X^4+X^8$
9	$1+X^4+X^9$
10	$1+X^3+X^{10}$

Construcción de los Campos de Galois GF(2^m)

A continuación presentaremos la forma de contruir los campos de Galois de elementos ($m > 1$) de un campo binario GF(2). Empezaremos con los dos elementos de GF(2): 0 y 1, y un nuevo símbolo α. Ahora definiremos una multiplicación "." para introducir una secuencia de potencias de α como sigue:

$$0.0 = 0,$$
$$0.1 = 1.0 = 0,$$
$$1.1 = 1,$$
$$0 .\alpha = \alpha . 0 = 0,$$
$$1 . \alpha = \alpha .1 = \alpha,$$
$$\alpha^2 = \alpha . \alpha,$$
$$\alpha^3 = \alpha . \alpha . \alpha ,$$
$$\vdots$$
$$\alpha^j = \alpha . \alpha .... \alpha \text{ (j veces)},$$
$$\vdots$$

De este modo tenemos un conjunto de elementos en el que la operación multiplicación está definida: $F= \{ 0,1, \alpha, \alpha^2, ... , \alpha^j, ... \}$

Ahora tenemos un polinomio primitivo de grado m sobre GF(2) p(X). Consideramos que p(α)=0. También se sabe que p(X) divide a $X^{2^m-1}+1$ (de acuerdo al teorema). Por lo tanto tenemos:

$$X^{2^m-1}+1 = q(X)p(X).$$

Reemplazando X por α, $\alpha^{2^m-1}+1 = q(\alpha)\,p(\alpha).$

Si ahora consideramos que p(α)=0 tenemos: $\alpha^{2^m-1}+1 = q(\alpha).0 = 0.$

Esto puede finalmente expresarse como: $\alpha^{2^m-1} = 1$

De este modo, bajo la condición que p(α)=0, el conjunto F resulta finito y contiene los sgtes. elementos:

$$F^* = \{0, 1, \alpha, \alpha^2, \ldots, \alpha^{2^m-2}\}$$

Se puede demostrar que el conjunto F* es cerrado para la adición y multiplicación[2] , y de este modo el conjunto $F^* = \{0, 1, \alpha, \alpha^2, \ldots, \alpha^{2^m-2}\}$ es un Campo de Galois de 2^m elementos, $GF(2^m)$. Cabe resaltar que la suma y multiplicación sobre $F^* = GF(2^m)$ implica suma y multiplicación módulo-2.

Ejemplo

Para m=4, $p(X)=1+X+X^4$ es el polinomio primitivo sobre GF(2). Si hacemos que $p(\alpha)= 1+\alpha+\alpha^4 = 0$ tenemos que $\alpha^4 = 1+\alpha$. Utilizando esto contruiremos el Campo de Galois $GF(2^4)$.

$$\alpha^5 = \alpha \,.\, \alpha^4 = \alpha\,.(1+\alpha) = \alpha + \alpha^2,$$

$$\alpha^6 = \alpha \,.\, \alpha^5 = \alpha\,.(\alpha+\alpha^2) = \alpha^2 + \alpha^3,$$

$$\alpha^7 = \alpha \,.\, \alpha^6 = \alpha\,.(\alpha^2+\alpha^3) = \alpha^3 + \alpha^4 = 1 + \alpha + \alpha^3$$

---

[2] Esto es demostrado en la Referencia: "Error Control Coding, Fundamental and Applications"

Los demás resultados se muestran en la siguiente tabla. Además utilizando la expresión encontrada anteriormente $\alpha^{2^m-1} = 1$ podemos ver que los términos mayores de $2^4 = 16$ son reducidos. De esta manera se calcula el Campo de Galois para cualquier valor de m.

Potencia representada	Representación Polinomial	Representación Bloque-4
0	0	0000
1	1	1000
α	α	0100
α^2	α^2	0010
α^3	α^3	0001
α^4	$1+\alpha$	1100
α^5	$\alpha+\alpha^2$	0110
α^6	$\alpha^2+\alpha^3$	0011
α^7	$1+\alpha+\alpha^3$	1101
α^8	$1+\alpha^2$	1010
α^9	$\alpha+\alpha^3$	0101
α^{10}	$1+\alpha+\alpha^2$	1110
α^{11}	$\alpha+\alpha^2+\alpha^3$	0111
α^{12}	$1+\alpha+\alpha^2+\alpha^3$	1111
α^{13}	$1+\alpha^2+\alpha^3$	1011
α^{14}	$1+\alpha^3$	1001

CAPITULO II
METODOS DE ENCRIPTACION

La transmisión de datos es un proceso que posee un fin específico: lograr que dos terminales (transmisor y receptor) se conecten, y que finalmente puedan intercambiar información. El avance de la tecnología en hardware de comunicaciones ha hecho posible que el concepto de dato se generalice, y que pueda ser utilizado indistintamente para transmisión de datos, tales como archivos de información, correo electrónico, etc, y también para el caso de voz, siempre que se cuente con la arquitectura adecuada para conseguir una buena digitalización y adquisición de datos[3]

Cuando se realiza una transferencia de información sobre una infraestructura de comunicaciones (una Red de Area Local por ejemplo) por lo general el establecimiento y culminación de la comunicación está garantizado, debido a que las infraestructuras actuales poseen tecnología adecuada. Para el caso de transmisión sobre una Red Local, que es el que nos interesa analizar, la comunicación se montan sobre un conjunto de protocolos de control que aseguran el correcto arribo de datos. Son estas técnicas las que permiten la transferencia de "pesados datos" sobre la Red, tales como Voz sobre IP, transmisión de audio e incluso video.

---

[3] Ver Apéndice: Encriptación en Sistemas de Comunicación.

Por otro lado, si bien es cierto las Redes nos garantizan una comunicación confiable, no nos garantizan la seguridad de los datos transmitidos. La transferencia de información sobre una Red Local entre dos puntos es usualmente realizada utilizando los métodos convencionales de transferencia de información, por ejemplo aplicaciones consideradas dentro del protocolo TCP/IP tales como FTP, Telnet, etc., ellas lamentablemente no consideran la existencia de un canal seguro de comunicación. Algunas políticas de comunicación utilizan servidores de validación para controlar el acceso hacia ciertas zonas restringidas, las cuales dejan de tener efecto luego de la validación ya que se permite el acceso irrestricto. La información de uso común (el correo electrónico por ejemplo) es enviado sin ningún tipo de protección en la gran mayoría de los casos, aún cuando la importancia de la información compartida ameritaría un mayor cuidado. De este modo puede surgir la necesidad de poseer mecanismos que permitan proteger los datos enviados, de manera que solo los elementos interesados (transmisor y receptor) puedan tener acceso a la información correcta. El presente capítulo tiene como objetivo mostrar y discutir las técnicas más comunes de seguridad en la comunicación y transmisión de datos, analizando las metodologías y algoritmos mas comunes y seguros.

Como un ejemplo tenemos un sistema de comunicación de voz que considere métodos y políticas de seguridad de datos, así como una eventual codificación y compresión, a fin de mejorar la eficiencia de la comunicación, debe considerar la estructura mostrada en la Fig. 2.1.

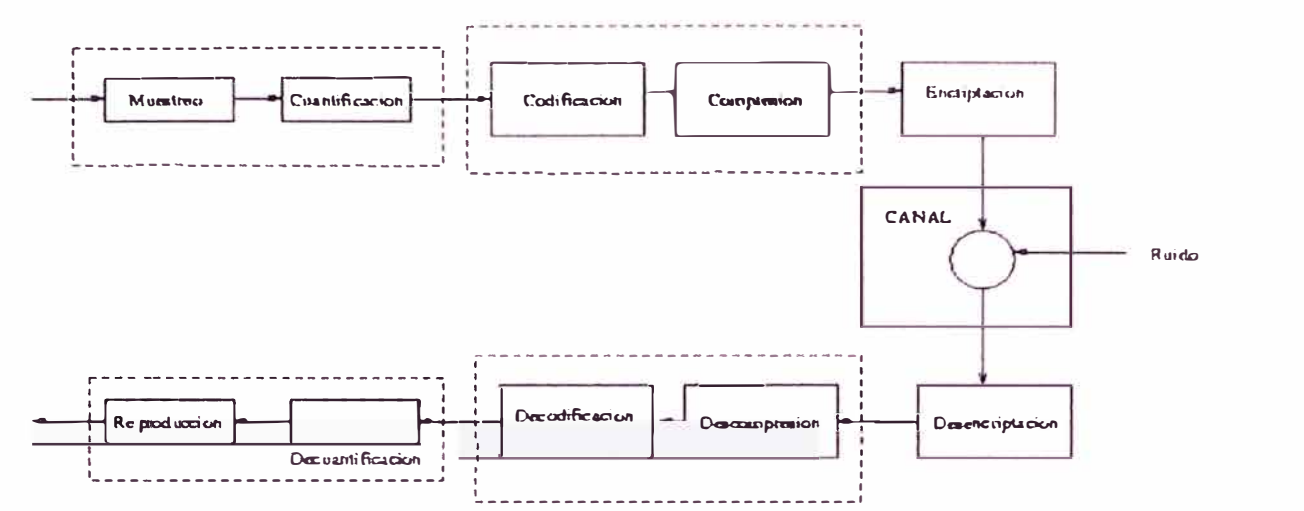


Figura 2.1: Esquema Funcional de una Tranmisión segura de Voz

2.1 Criptografía

Se denomina criptografía al conjunto de procesos necesarios para conseguir que la información que desea ser transmitida se transforme en una salida codificada aparentemente sin relación alguna a los datos originales, la que posteriormente es enviada y finalmente reconstruída por el receptor. Un sistema de encriptación estándar se representa en la Fig. 2.2

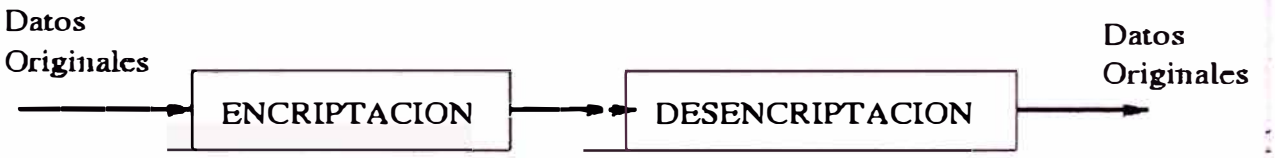


Figura 2.2: Proceso Estándar de Encriptación

Bajo este contexto podemos apreciar que la información viaja segura por el canal de información. El análisis de que tan confiable es el algoritmo de encriptación utilizado, el contenido secreto adicionado al proceso(la llave), los procesos a realizarse, etc, constituyen el criptoanálisis.

2.1.1 La Llave de Encriptación.- El concepto más importante en el proceso de encriptación está constituído sin lugar a dudas por la llave de encriptación. Es la que va a permitir que la información sea posteriormente recuperada satisfactoriamente. El nivel de seguridad que un método en particular proporcione depende directamente de la llave de encriptación y la cantidad de bits de la que esté constituída. Los características más importantes de cada algoritmo en particular que deben ser tomadas en cuenta durante el criptoanálisis son las siguientes:

- Longitud de Bloque de Datos
- Longitud de la Llave
- Método de combinación de los datos con la llave
- Número de Iteraciones
- El Efecto "Avalancha"

Este último es particularmente importante. Un buen algoritmo debe ser extremadamente "sensible"' a pequeñas variaciones de la llave y/o datos. De este modo un dato de salida con una llave determinada debe ser completamente diferente a los datos obtenidos con una llave que sólo difiere en un dígito. Posteriormenente veremos que algoritmos como IDEA cumplen a cabalidad con esta norma.

Con estos conceptos podemos redefinir el modelo de comunicación que la criptografía propone, y podemos apreciarlo en la Fig. 2.3.

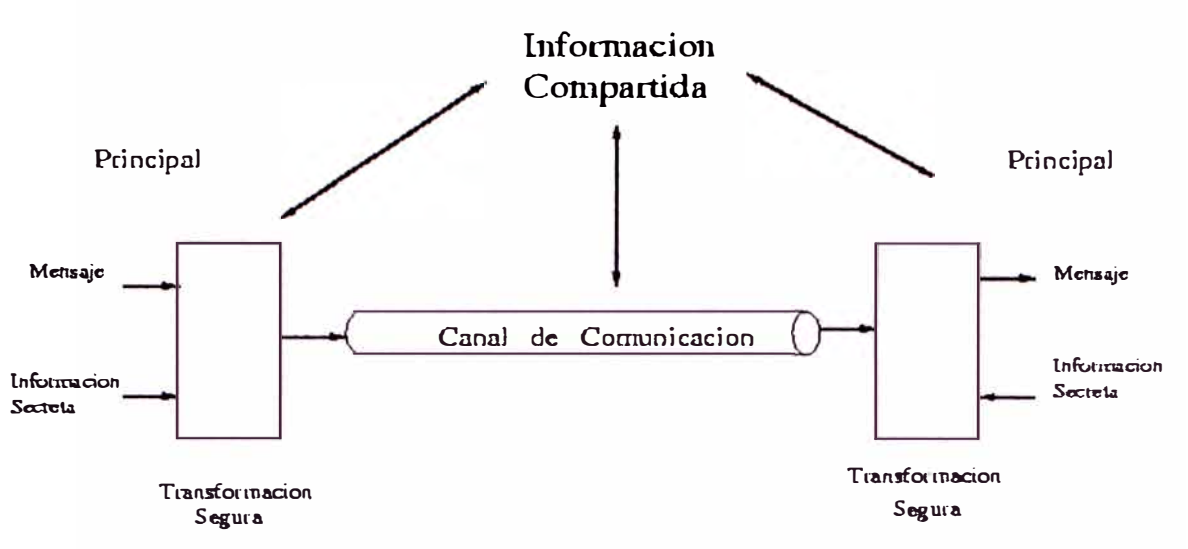


Figura 2.3: Proceso de Transmisión Segura de Datos

2.2 Algoritmos de Encriptación

Ahora ya se tiene el panorama mas claro acerca de la seguridad que puede conseguirse en la transmisión de información utilizando la criptografía. Ahora nos enfrentamos a otro problema: ¿que metodología debo emplear para conseguirlo?.

La primera idea que surge es considerar que para cada problema en particular se creará una solución, constituyendo soluciones por cada usuario final. Estas soluciones de tipo propietarias implicarían que cada usuario final interesado en la protección de su información deba diseñar los sistemas de encriptación en un primer momento, y mejorar continuamente para mantener sus aplicaciones en etapas posteriores cuando debido a un nuevo requerimiento de comunicación aparezca, lo que implicaría nuevos desarrollos, o modificación de los ya existentes. De este modo la difusión de los métodos para conseguir su generalización no resulta del todo clara.

Para uniformizar esta idea se pensó en definir técnicas de encriptación de datos, es decir metodologías de diseño y algoritmos, que sean conocidos y probados, y que garanticen un alto nivel de seguridad de datos en la comunicación, además de

proporcionar bondades para su implementación, las que irán de acuerdo a la metodología adoptada y al algoritmo finalmente escogido.

Dos de las tendencias más resaltantes y que poseen mayor aceptación en los sistemas de comunicación, donde la seguridad es determinante (aplicaciones con cajeros bancarios por ejemplo) son la Public Key Infraestructure (PKI) , y Pretty Good Privace (PGP). A continuacion trataremos acerca de ellas.

2.2.1 Public Key Infraestructure (PKI).- Viene a ser la combinación de software, técnicas de encriptación, y servicios que permitan a empresas o corporaciones poseer seguridad en sus comunicaciones y transacciones de negocios sobre el Internet. PKI incluye certificación digital, criptografía con llave pública, certificación y autorización que en total constituyen una arquitectura de red segura de gran nivel empresarial.

Una típica arquitectura PKI para una empresa, abarca la emisión de certificación digital, tanto para usuarios individuales como para servidores; software de manejo de inscripciones para usuarios finales, integración con directorios corporativos certificados; herramientas de administración, renovación y revocación de certificados y documentos; servicios y ayuda relacionados. Una representación gráfica de la arquitectuta PKI se muestra a continuación en la Fig. 2.4.

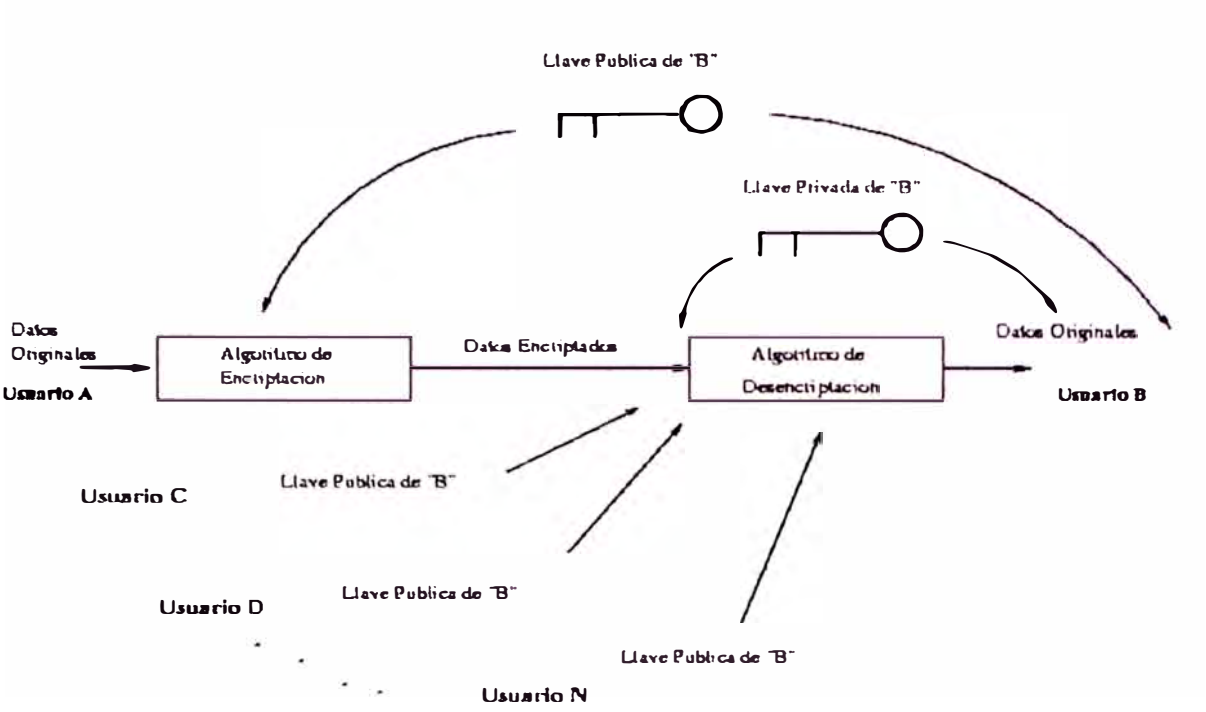


Figura 2.4: Public Key Infraestructure

De este modo se maneja el concepto de ``par de llaves" para el proceso de seguridad de datos, de modo tal que una llave es mantenida en secreto, mientras que la otra es pública. Con este modelo de seguridad si una llave encripta, la otra se encarga de la recuperación de los datos, y viceversa.

Dos aplicaciones muy utilizadas en las transacciones sobre el Internet y que se basan en PKI son:

- Para Confidencialidad.- El emisor encripta el mensaje con la llave pública del receptor, mientras que el receptor desencripta el mensaje con la llave privada del receptor.
- Para Autentificación de identidad.- El emisor encripta el mensaje con la llave privada del emisor, mientras que el receptor desencripta el mensaje con la llave pública del emisor.

2.2.2 Pretty Good Privace (PGP).- Es el estándar de facto para envío seguro de correo electrónico y encriptación de archivos sobre el Internet. Permite y facilita que las personas puedan evitar la lectura no autorizada de sus mensajes, y la posibilidad de adicionar firmas digitales a los mensajes que garanticen su autenticidad.

PGP está basado en algoritmos que han "sobrevivido" una extensa revision pública, y que son considerados extremadamente seguros. Entre los algoritmos PGP más conocidos se encuentran el RSA (para el caso de encriptación con llave pública), IDEA(para encriptación convencional con llave privada), y MD5(para codificación de contraseñas por ejemplo). La metodología de encriptación utilizada por los sistemas PGP se representa en la Fig. 2.5.

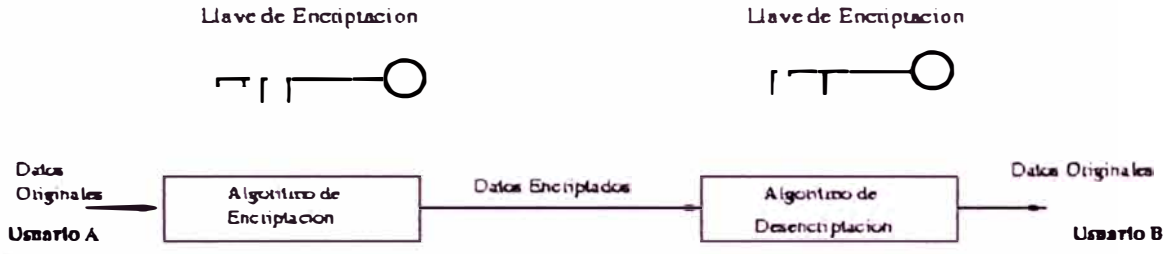


Figura 2.5: Pretty Good Privace

Para el caso de utilización de PGP con sistemas convencionales de encriptación (una llave simple compartida y privada) se cuenta con IDEA. Este algoritmo permite implementar un sistema de seguridad de datos extremadamente confiable y seguro. La cantidad de información que procesa, y que permite la generación de los datos a la salida, lo hace muy seguro para la transmisión de información.

2.3 Algoritmos

Ahora nombraremos algunos de los algoritmos de encriptación más utilizados. Estos poseen características especiales que los hacen adecuados y altamente apropiados para determinados tipo de aplicaciones.

2.3.1 Rivest-Shamir-Adleman (RSA).- Este algoritmo fue desarrollado inicialmente en 1977 por Ron Rivest, Adi Shamir y Len Adleman en el MIT y publicado en 1978 (RIVE78). Perteneciente al grupo de algoritmos PGP, RSA un algoritmo de Llave Pública, de modo que posee información conocida e información que se mantiene en secreto.

En una comunicación en la que se desea transmitir los datos originales M, los datos encriptados C están definidos por: $C= M^e \bmod (n)$

$$M= C^d \bmod (n) = (M^e)^d \bmod (n)= M^{ed} \bmod (n) \quad \text{.....(2.1)}$$

Donde a mod b es el operador modificador igual al resto de dividir a entre b [1].

Tanto el transmisor como el receptor conocen el valor de n. El transmisor conoce el valor de e, pero solo el receptor conoce el valor de d, lográndose de este modo la encriptación con llave pública.

2.3.2 Message-Digest Algorithm (MD5).- Fue desarrollado por Ron Rivest(El mismo de RSA). Es un algoritmo basado en técnicas "hash", es decir, la generación de los datos de salida siguen la sgte. regla de formación: h= H(M), donde M es el mensaje de longitud variable, y h la salida de longitud constante. La función H es denominada Función Hash, y para el caso de MD5 recibe mensajes de longitud arbitraria, y produce a la salida mensajes de 128 bits. La entrada de datos es

procesada en bloques de 512 bits. El proceso de realiza en base a correspondencias entre la llave y los datos, mediante tablas definidas.

2.3.3 Secure Hash Algorithm (SHA).- Fué desarrollado por el National Institute of Standars and Technology (NIST) y publicado en 1993 (FIPS PUB 180). Su esquema básico es basado en MD4(el antecesor de MD5). De este modo estos dos algoritmos son muy parecidos en su estructura. SHA recibe datos de una longitud menor a 2^{24}, y produce salidas de datos de 160 bits de longitud. Los datos de entrada son procesados también en bloques de 512 bytes. Debido a su parecido, MD5 y SHA pueden ser comparados algunos aspectos, tales como la seguridad, en donde SHA proporciona mayor seguridad que MD5 debido a la cantidad de datos que produce, y en el aspecto de procesamiento de datos obviamente SHA posee un proceso mas lento, teniéndose que en una misma plataforma de aplicación SHA se procesa aproximadamente 25% mas lento que MD5.

2.3.4 Data Encryption Standard (DES).- Es uno de los algoritmos más difundidos en los sistemas de seguridad de datos. Se basa en el Data Encryption Standard adoptado en 1977 por el National Bureau of Standard, ahora el National Institute of Standards and Technology (NIST). El método utilizado por DES procesa bloques de datos de 64 bits de longitud, los cuales son combinados con una llave de 56 bits. El algoritmo transforma paso a paso los 64 bits de entrada en otros 64 de salida. Los mismos procesos, con la misma llave, son utilizados posteriormente para la recuperación de la información.

DES es un algoritmo basado en permutaciones. Esto quiere decir que los datos de entrada son desordenados y reagrupados de acuerdo a los valores de la llave de encriptación utilizada. El proceso implica un total de 16 iteraciones de los datos

de entrada, para finalmente obtener los datos encriptados a la salida. El esquema completo de DES es mostrado en la Fig. 2.6.

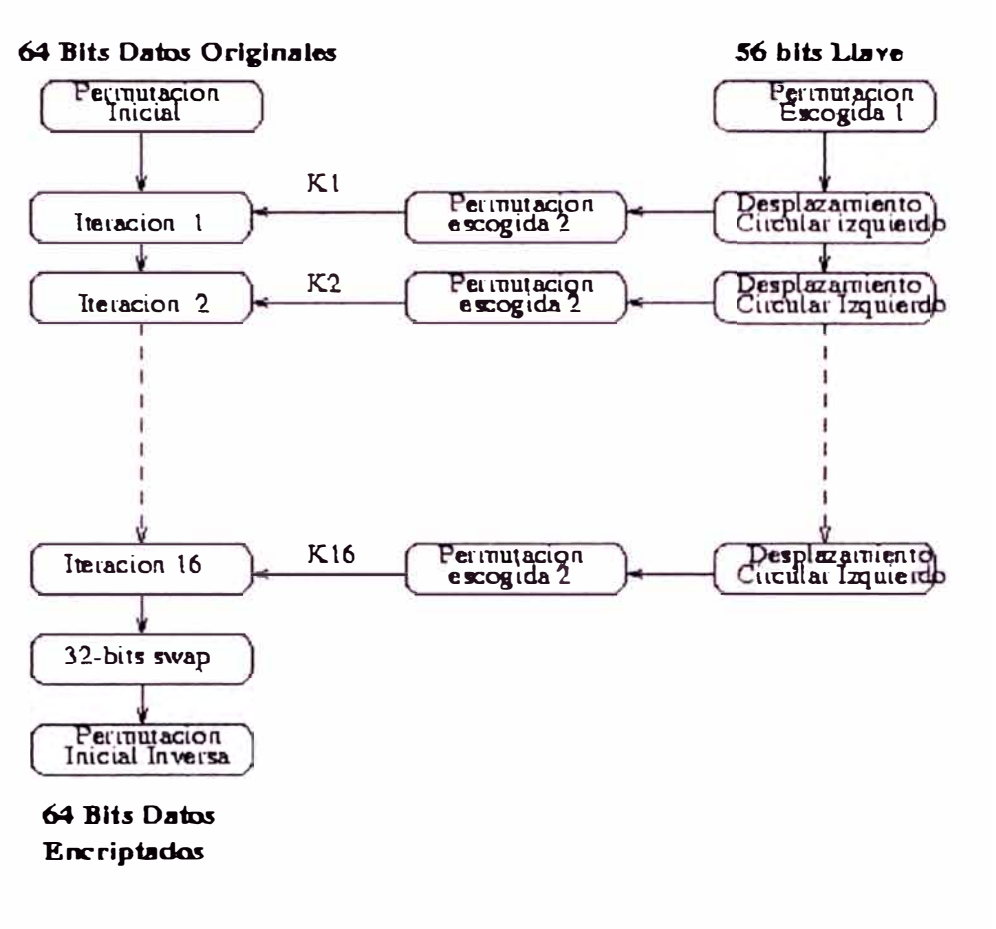


Figura 2.6: Descripción del DES

De este modo los datos son encriptados mediante un largo proceso, lo que lo hace lento y no lo sufucientemente seguro, debido a que como se comentó, su método se basa solo en permutaciones, de este modo son los mismos datos de entrada los que son producidos a la salida. Por otro lado es bastante sólido al "efecto avalancha", debido a que pequeñas variaciones de la llave provocan significativos cambios en la salida.

2.3.5 Tripe DES.- Surge como una alternativa para compensar las limitaciones que presenta DES a nivel de encriptación de datos. La información no se hace del todo segura, y antes de pensar en otros algoritmos que pudieran reemplazar a DES se

asimiló la posibilidad de utilizar mas de una llave en el proceso. Por ello se creó Doble DES, que encripta la información con dos llaves. Por otra parte se pensó en tener tres bloques de encriptación pero con solo 2 llaves. A este estándar generado se le denominó Tripe DES, y su estructura de proceso, así como la de Doble DES se muestra en la Fig. 2.7.

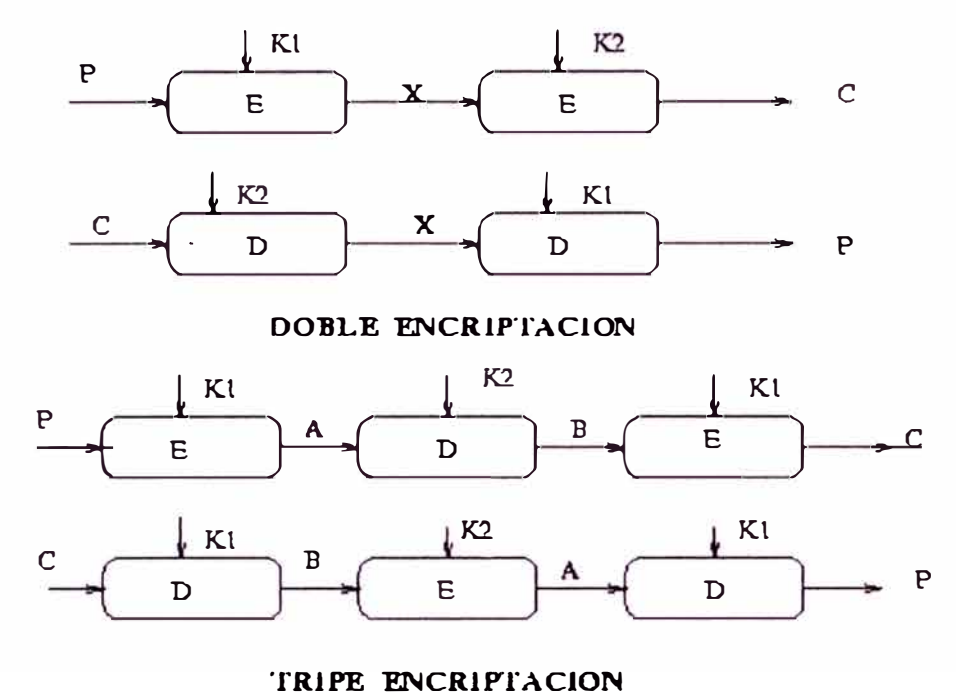


Figura 2.7: Encriptación Múltiple

2.3.6 International Data Encryption Algorithm (IDEA).- IDEA es el nuevo algoritmo convencional de encriptación orientado a bloques, desarrollado por Xeejia Lai and James Massey del Swiss Federal Institute of Technology. La versión original fue presentada en [LAI90], una versión revisada del algoritmo optimizada para ser "muy fuerte" fue presentado en [LAI91] y detallado de mejor manera en [LAI92]. IDEA es el algoritmo que en los últimos años ha sido propuesto para reemplazar a DES. IDEA es largamente superior a DES en conceptos de seguridad, ya que utiliza combinación de datos con la llave y no solo permutaciones, mayor cantidad de datos

a procesar (datos de 64 bits y llave de 128 bits), etc. El principal objetivo de IDEA es contar con un sistema extremadamente seguro y sencillo de implementar. La estructura básica de IDEA es mostrada en la Fig. 2.8.

El principal acierto de IDEA es que consigue la encriptación mediante combinaciones entre los bits de datos y de la llave, pero en base a operaciones aritméticas y lógicas sencillas, lo que hace que el proceso sea rápido y eficiente, ya que son las propiedades del álgebra de módulos[4], las que garantizan la total reconstrucción de los datos.

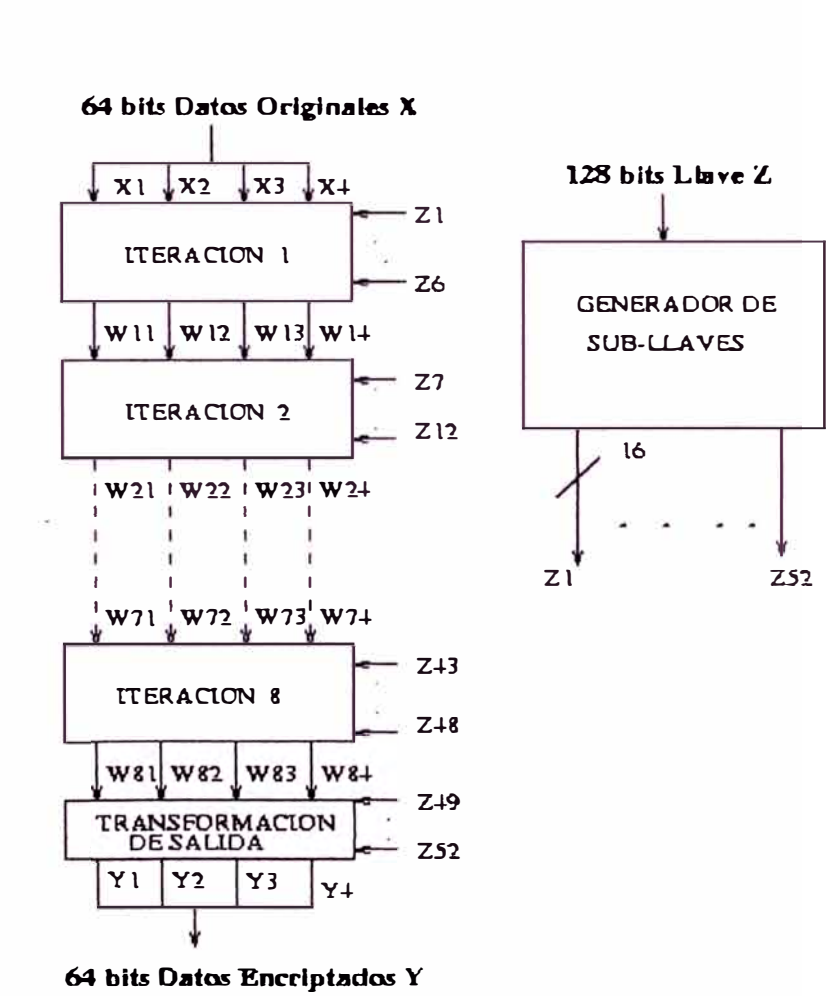


Figura 2.8: Implementación de IDEA

[4] Ver Capítulo 1: Fundamentos Matemáticos

CAPITULO III
EL ALGORITMO IDEA

En el capítulo anterior se describió de manera general los más importantes y difundidos algoritmos estándar de encriptación, los que son ampliamente utilizados para aplicaciones específicas. Cada uno proporciona niveles de seguridad propios y una arquitectura definida de implementación.

El algoritmo más utilizado actualmente es el DES. Este algoritmo se basa en permutaciones, siendo la llave de encriptación la que controla y define como es que los bits de información deben ser distribuidos. Sin embargo no proporciona un nivel de seguridad lo suficientemente confiable, su implementación requiere de muchas etapas iterativas, lo que lo hace lento. Por ello se pensó en mejorarlo, surgiendo las tendencias de encriptación Doble y Tripe para DES.

IDEA aparece como una alternativa más adecuada para reemplazar a DES. Su metodología de encriptación no solo se limita a la utilización de rutinas de permutaciones y desplazamientos, sino que adiciona procesos en los que combina los datos de entrada con valores obtenidos de la llave de encriptación adoptada. Estos conjuntos de operaciones constituyen la unidad básica de encriptación denominada Célula de Encriptación. IDEA utiliza 8 células en cascada, proporcionando una adecuada decorrelación de los datos originales con los datos de salida, terminando su proceso con una transformación de salida, tal como se apreció en la Fig. 2.8 del

capítulo anterior. Estas cualidades para el procesamiento, seguridad de datos e implementación fueron factor primordial para elegirlo como estándar de encriptación para el presente trabajo, y en este capítulo especificaremos en forma detallada el proceso del que se vale IDEA para la encriptación de información.

3.1 Principios del Diseño

IDEA es un encriptador de bloques de información, que utiliza una llave de 128 bits de longitud, y que encripta bloques de datos de 64 bits, superior a DES que utiliza 64 bits de datos y 56 bits de llave.

3.2 Requisitos para garantizar seguridad

Las siguientes características hacen de IDEA un robusto sistema de encriptación.

- Longitud de Bloque: El tamaño de bloque debería ser lo suficientemente grande para evitar análisis estadístico, para evitar que los bloques aparezcan muchas veces y que sirvan de ayuda para descifrar los datos. La longitud de bloque de 64 bits es considerado altamente confiable en este sentido.
- Longitud de llave: La longitud de la llave debe ser lo suficiente grande para evitar procesos de búsqueda y reconstrucción de la información. Con 128 bits IDEA garantiza este requisito largamente.
- Confusión: La información a proteger debe combinarse con la llave de una manera confusa y complicada, de modo que no sea sencillo encontrar dependencias. IDEA consigue esto debido a la utilización de 3 operaciones diferentes de combinación, en contraste con DES que utiliza principalmente XOR y pequeños bloques no lineales de permutación de bits.
- Difusión: Los datos de salida deben ser extremadamente sensibles a los pequeños cambios en la llave. IDEA es especialmente efectivo en esta característica.

El tema de Confusión es muy importante. IDEA implementa tres tipos de operaciones combinacionales, las cuales se listan a continuación.

- OR-Exclusivo Bit a Bit, denotado por (+).
- Adición de enteros módulo 2^{16}(módulo 65336), denotado por [+].
- Multiplicación de enteros módulo $2^{16}+1$ (modulo 65537), considerando entradas y salidas como enteros sin signo, excepto para el bloque de ceros que es considerado 2^{16}. La operación se denota por (*) [5]

Por ejemplo: 0000000000000000(*)1000000000000000=1000000000000001

Debido a que: $2^{16} \times 2^{15} \bmod (2^{16}+1)=2^{15}+1$ [2]

En resumen, IDEA es un algoritmo que garantiza una total seguridad en los datos, al cumplir largamente los requerimientos que un buen algoritmo debe presentar.

3.3 Consideraciones para la Implementación

IDEA puede ser implementado tanto en hardware como en software. La solución de software proporciona flexibilidad, bajo costo, y facilidad de implementación de las operaciones. La solución en hardware puede ser implementado con técnicas VLSI o en Procesadores Digitales de Señales (DSP's), y proporciona alta velocidad en el procesamiento y similitud de implementación para las etapas de encriptación y desencriptación. Además la utilización de operaciones aritméticas a nivel de bits hacen que sea sencilla y muy rápida la implantación y desarrollo en lenguaje ensamblador del algoritmo de encriptación, para el caso de soluciones basadas en DSP.

3.4 Encriptación IDEA

Como se trató en el Capítulo anterior, IDEA se basa en bloques funcionales para el procesamiento de los datos. El sistema propuesto por IDEA es el mostrado en la Fig. 3.1. Como se aprecia existen dos entradas al sistema: los datos originales y la llave de encriptación. Para este caso se considera datos de 64 bits y una llave de 128 bits. Analizando la estructura, IDEA se compone de 8 bloques en cascada, los cuales reciben 4 bloques de datos de 16 bits cada uno, provenientes de los datos de entrada, y utiliza 6 sub-llaves de 16 bits, los que fueron generados de la llave original. Finalmente una transformación de salida genera 4 bloques de datos de 16 bits, los que concatenados resultan los 64 bits de datos encriptados, y para ello utiliza 4 sub-llaves, haciendo un total de 52 sub-llaves, las que deben ser obtenidas de la llave original.

3.4.1 Un bloque funcional en detalle.- Ahora analizaremos con más detalle una celda (o célula) de encriptación. Las 8 celdas posee la misma estructura, solo que reciben como entradas otros datos y otras sub-llaves. Se aprecian que las 3 operaciones elementales de IDEA son aplicadas (XOR, Suma módulo 2^{16} y multiplicación módulo $2^{16}+1$), consiguiendo la combinación de datos y las permutaciones, al combinar los datos y situarlos posteriomente en otras posiciones del bloque funcional. Este proceso es detallado en la Fig. 3.2.

[5] Ver Capítulo 1: Fundamentos Matemáticos

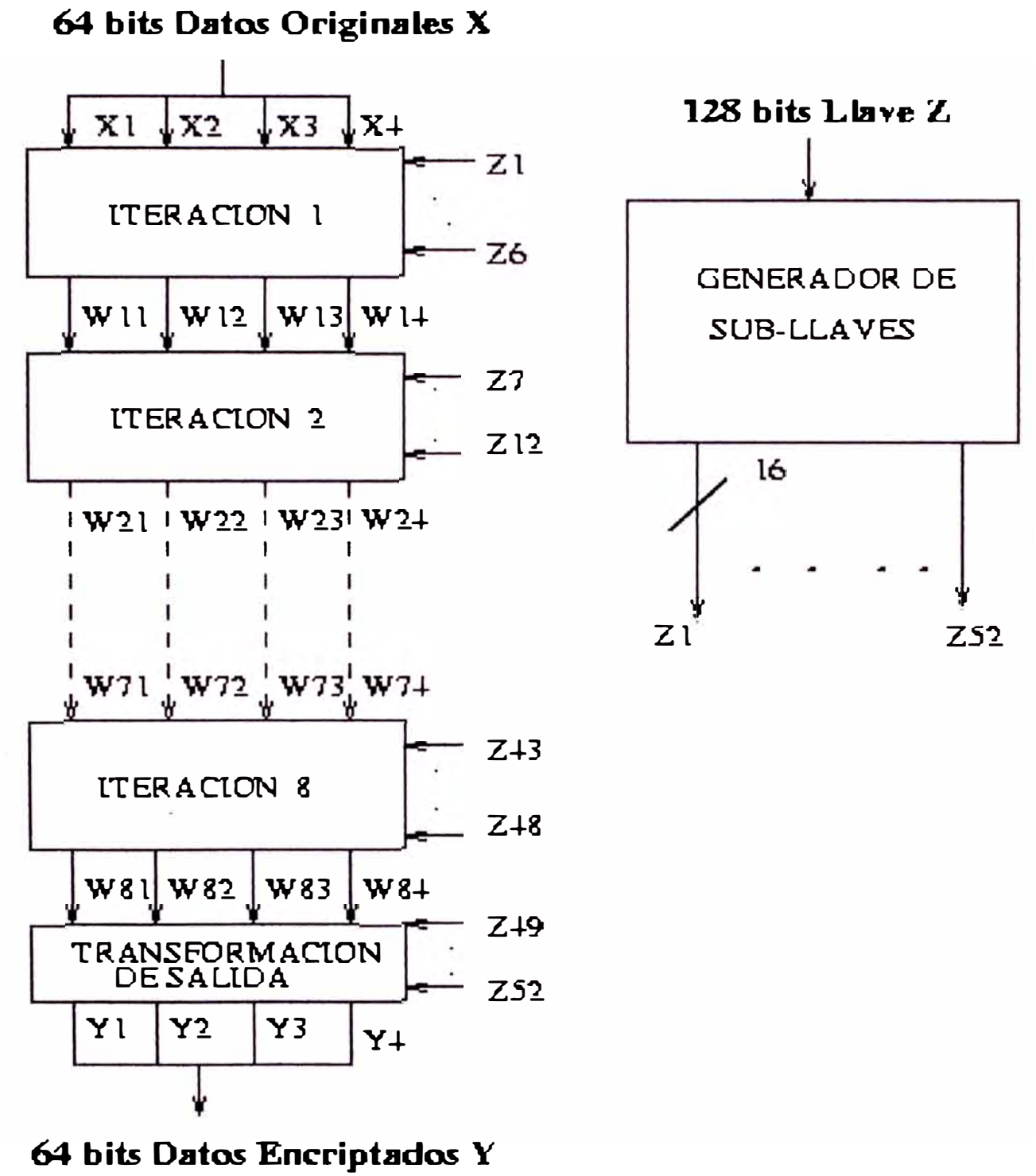


Figura 3.1: Estructura de IDEA

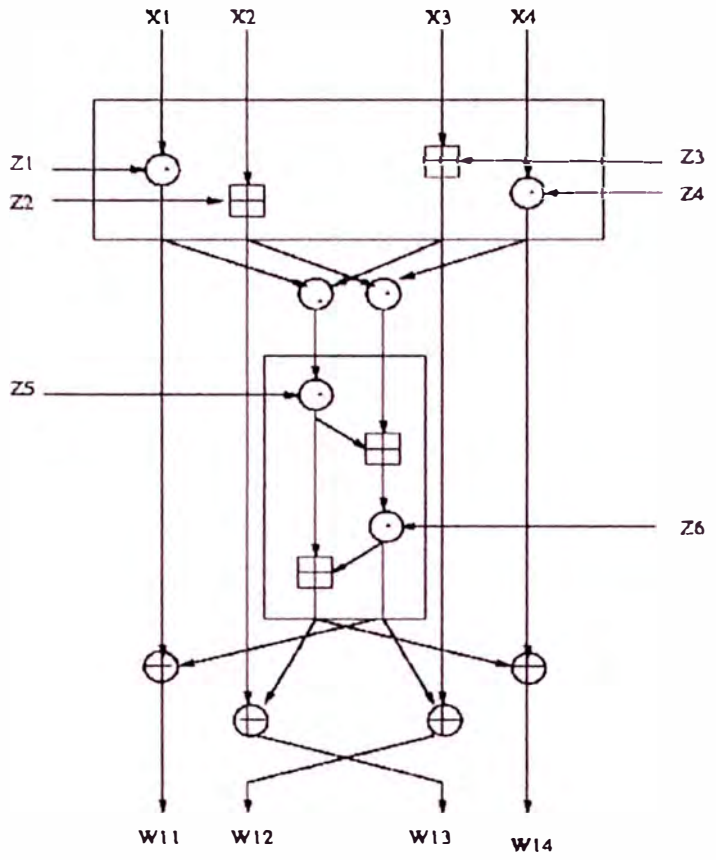


Figura 3.1: Primera Celda de Encriptación IDEA

Esta celda de encriptación es repetida 8 veces y los datos de entrada como las sub-llaves de encriptación son las detalladas en la Fig. 3.1. Finalmente se hace uso de una Transformación de Salida, la que finalmente genera los datos encriptados. Esta transformación es mostrada en la Fig. 3.3.

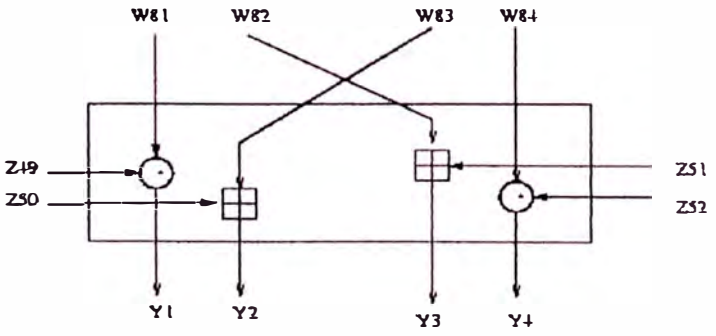


Figura 3.3: Transformación de Salida

Se aprecia que esta transformación realiza una permutación previa de los datos antes de combinarlos con las sub-llaves. Finalmente los fragmentos Y1, Y2, Y3 e Y4 constituyen la salida encriptada.

3.4.2 Generación de las sub-llaves.- En las figuras anteriores se aprecia la presencia de 52 bloques de 16 bits denominadas sub-llaves. Estos fragmentos tienen una directa relación con la llave original adoptada. El proceso de generación se presenta a continuación.

Las primeras 8 sub-llaves, denotadas como Z_1, Z_2,...,Z_8 se toman directamente de la llave, con Z_1 como los 16 bits más significativos, Z_2 como los siguientes 8 y así hasta completar las 8 sub-llaves. Luego de ello se realiza un desplazamiento circular hacia la izquierda de 25 posiciones y se vuelve a elegir las 8 siguientes sub-llaves. El proceso continúa hasta conseguir los 52 fragmentos de llave. De este modo, para los 8 bloques, la primera sub-llave en cada caso será:

$Z_1 = Z[1..16]$, $Z_{25} = Z[76..91]$

$Z_7 = Z[97..112]$, $Z_{31} = Z[44..59]$

$Z_{13} = Z[90..105]$, $Z_{37} = Z[37..52]$

$Z_{19} = Z[83..98]$, $Z_{43} = Z[30..45]$

Haciendo un análisis mas general, considerando un número n de celdas en cascada tenemos que el número de sub-llaves necesarias es:

$$\text{Numero}_{\text{sub-llaves}} = 6n+4 \quad \text{.....(3.1)}$$

3.5 Proceso de Desencriptación

Una de las principales características de IDEA es que el proceso de desencriptación es en esencia idéntico al de encriptación. Los datos de entrada a la primera celda son los datos encriptados, y son las sub-llaves las que cambian para

conseguir la recuperación de datos. Las sub-llaves de desencriptación U_1, U_2,...,U_{52} se derivan de las sub-llaves de encriptación y se relacionan de la siguiente manera:

- Las primeras 4 sub-llaves de desencriptación para la iteración i son derivadas de las 4 sub-llaves de encriptación de la iteración (10-i), donde la transformación final es considerada como la iteración 9. La primera y la cuarta sub-llave es igual al inverso multiplicativo módulo $(2^{16}+1)$ [1] de la primera y cuarta sub-llave de encriptación respectivamente. Para las iteraciones 2 hasta la 8, la segunda y tercera sub-llave de desencriptación son iguales a los inversos aditivos módulo (2^{16}) de las correspondientes tercera y segunda sub-llaves de encriptación. Para las iteraciones 1 y 9 la segunda y tercera sub-llaves son igual al inverso aditivo módulo (2^{16}) de las correspondientes segunda y tercera sub-llaves.
- Para las 8 primeras iteraciones, las dos últimas sub-llaves de desencriptación para la iteración i son iguales a las últimas 2 sub-llaves de la iteración (9-i).

Se utiliza la notación Z_j^{-1} para denotar el inverso multiplicativo, y por propiedades del álgebra de módulos: $Z_j \,(*)\, Z_j^{-1} = 1$

Debido a que $2^{16}+1$ es primo y cada número entero diferente de cero $Z_j \leq 2^{16}$ tiene un único inverso multiplicativo [1]. Para el caso aditivo consideramos la notación $-Z_j$, y se cumple: $-Z_j \,[+]\, Z_j = 0$

Para verificar la veracidad del algoritmo consideraremos la última celda y la transformación final, analizando el comportamiento de los datos encriptados al realimentarse a la primera celda, pero utilizando las llaves de desencriptación halladas. El esquema es apreciado en la Fig. 3.4.

Podemos apreciar que para la transformación de salida se tiene:

$$Y_1 = W_{81} \,(*)\, Z_{49} \qquad Y_3 = W_{82} \,[+]\, Z_{51}$$

$$Y_2 = W_{83}\ [+]\ Z_{50} \qquad Y_4 = W_{84}\ (*)\ Z_{52}$$

Ahora, para la primera iteración en el lado de desencriptación tenemos:

$$J_{11} = Y_1\ (*)\ U_1 \qquad J_{13} = Y_3\ [+]\ U_3$$

$$J_{12} = Y_2\ [+]\ U_2 \qquad J_{14} = Y_4\ (*)\ U_4$$

De acuerdo a las reglas dadas anteriormente sobre la generación de las sub-llaves de desencriptación, tenemos que $U_1 = Z_{49}^{-1}$, $U_2 = -Z_{50}$, $U_3 = -Z_{51}$ y $U_4 = Z_{52}^{-1}$. En base a esto podemos sustituir en la expresión anterior y obtenemos:

$$J_{11} = Y_1\ (*)\ Z_{49}^{-1} = W_{81}\ (*)\ Z_{49}\ (*)\ Z_{\_49}^{-1} = W_{81}$$

$$J_{12} = Y_2\ [+]\ -Z_{50} = W_{83}\ [+]\ Z_{50}\ [+]\ -Z_{50} = W_{83}$$

$$J_{13} = Y_3\ [+]\ -Z_{51} = W_{82}\ [+]\ Z_{51}\ [+]\ -Z_{51} = W_{82}$$

$$J_{14} = Y_4\ (*)\ Z_{52}^{-1} = W_{84}\ (*)\ Z_{52}\ (*)\ Z_{52}^{-1} = W_{84}$$

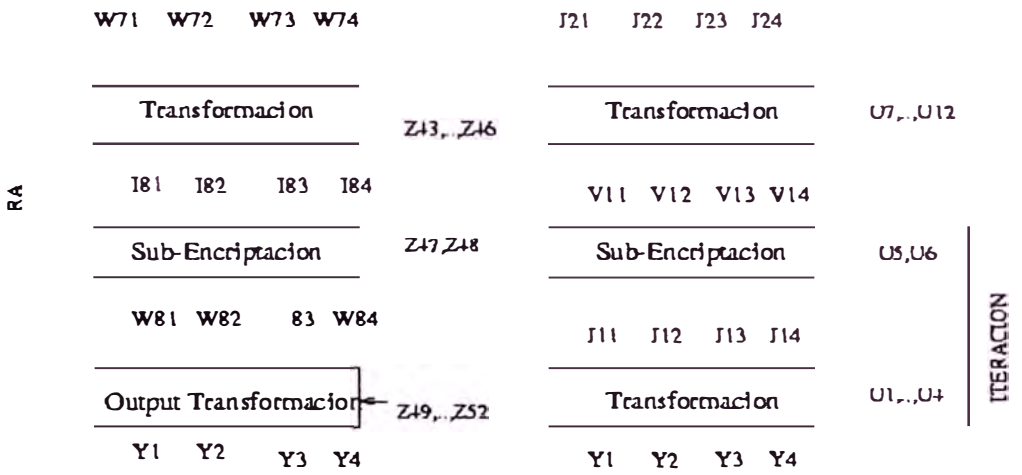


Figura 3.4: Proceso de Recuperación de Datos

De este modo los valores de salida del primer bloque de la etapa de encriptación es la misma que los datos que ingresaron a la última etapa de encriptación (transformación de salida), excepto que por el intercambio de los sub-bloques segundo y tercero. Análogamente se pueden demostrar que al avanzar con las celdas se obtiene los valores correspondientes de datos y el intercambio existente

es remediado en la última iteración en el proceso de desencriptación, consiguiéndose de este modo la recuperación total de la información inicial.

De esta manera se ha detallado el proceso de encriptación que utiliza IDEA, y demostrado la recuperación de los datos, apreciando que el número de celdas puede ser variable debido a que el algoritmo no es cerrado en cuanto al número de iteraciones, solo que ambas etapas posean la misma cantidad. Esto nos servirá en el capítulo próximo para conseguir la adaptación de IDEA para los fines específicos del presente trabajo de Tesis.

CAPITULO IV
TECNICAS DE COMPRESION DE VOZ

Este capítulo trata sobre los métodos más usuales de codificación y comprensión de señales analógicas adquiridas (la voz por ejemplo), a fin de tener una visión general de estos métodos, lo que nos servirá para posteriores capítulos.

4.1 Técnicas de Codificación de Señales Analógicas

Un gran número de técnicas de codificación para señales analógicas han sido desarrolladas los últimos 40 años, la mayoría de estas aplicadas a la codificación de voz e imágenes. En esta sección describiremos brevemente algunos de estos métodos y la aplicación para voz como un ejemplo para verificar su rendimiento.

Resulta conveniente para el análisis sub-dividir los métodos de codificación en tres tipos: Codificación Temporal, en el que la codificación es en base a las características en el tiempo de la señal; Codificación Espectral, en el que la señal es subdividida en diferentes bandas de frecuencia, y la codificación se hace en base a las características en frecuencia de la señal; y la Codificación Basada en Modelos, en el que se aplica transformaciones matemáticas a la señal a fin de codificarla. En este capítulo por fines de aplicación solo analizaremos el primero de ellos, debido a que es este el utilizado en la aplicación planteada en el presente trabajo.

Existen muchos métodos de codificación de datos que han sido diseñados para representar la señal en el dominio del tiempo. Los más utilizados son tratados a continuación.

4.1.1 Pulse Code Modulation (PCM).- Sea x(t) una señal muestreada, y sea x_n una muestra tomada a una frecuencia de muestreo $f_s \geq 2W$, donde W es la máxima frecuencia del espectro de x(t). En PCM, cada muestra de la señal es cuantizada en uno de los 2^R niveles de amplitud, donde R es el número de dígitos binarios utilizados para representar cada muestra. De esta manera la tasa de la muestra es Rf_s bits/s.

El proceso de cuantización puede ser modelado matemáticamente como:

$$\overline{X^n} = x_n + q_n,$$

donde X^n representa el valor cuantizado de x_n y q_n representa el error de cuantización, el cual tratamos como un ruido aditivo. Asumimos que un cuantizador uniforme es utilizado, teniendo la característica de entrada-salida ilustrada en la Fig. 4.1, el ruido de cuantización es estadísticamente caracterizado como:

$$p(q) = \frac{1}{\Delta}, \qquad -\frac{1}{2}\Delta \leq q \leq \frac{1}{2}\Delta$$

donde el paso del cuantizador es $\Delta=2^{-R}$. El valor cuadrático medio del error de cuantización es: $E(q^2) = \frac{1}{12}\Delta^2 = \frac{1}{12}x2^{-2R}$

Midiendo en decibelios, el valor cuadrático medio del ruido es:

$$10\log\left(\frac{1}{12}\Delta^2\right) = 10\log\left(\frac{1}{12}x2^{-2R}\right) = -6R - 10.8dB$$

Observamos que el ruido de cuantización decrece a 6 dB/bit utilizado en el cuantizador. Por ejemplo, para una cuantización a 7 bits tenemos una potencia de error de cuantización de -52.8 dB.

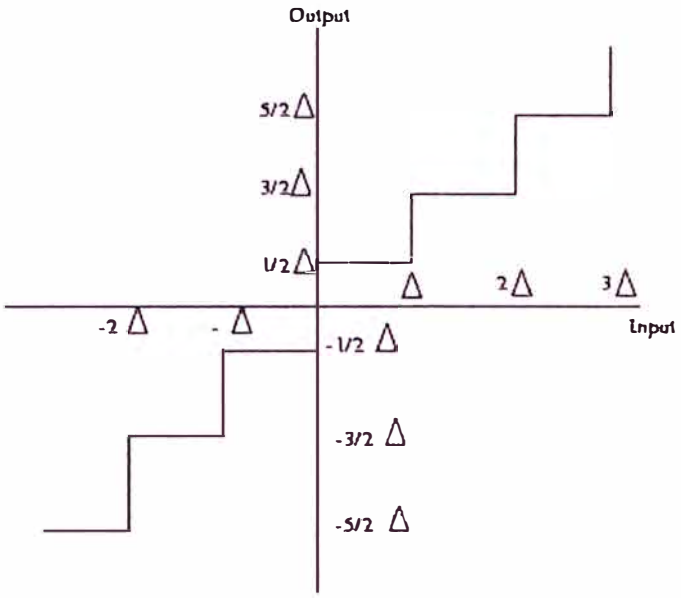


Figura 4.1: Característica de Entrada – Salida del Cuantizador Uniforme

Muchas señales tales como señales de voz tienen la característica que los valores de amplitud pequeña ocurren mas a menudo que los valores grandes. Sin embargo, una cuantización uniforme proporciona el mismo espaciamiento entre niveles sucesivos en todo el rango de la señal. Una mejor aproximación es la cuantización no uniforme. Una característica de cuantización no uniforme es usualmente obtenida haciendo pasar la señal por un dispositivo no lineal que comprima la señal, seguido de una cuantización uniforme. Por ejemplo, un compresor logarítmico tiene una característica de magnitud entrada-salida de la forma:

$$|y| = \mathrm{sgn}(x)\frac{\log(1+\mu|x|)}{\log(1+\mu)}$$

donde $|x| \leq 1$ es la magnitud de la entrada, $|y|$ es la magnitud de la salida, y μ es un parámetro que es seleccionado de acuerdo a las características de compresión

deseadas. La Fig. 4.2 muestra esta relación para varios valores de μ. El valor μ=0 implica no compresión.

Para la codificación de voz, el valor de μ=255 ha sido adoptado como estándar en USA y Canadá. Este valor proporciona 24 dB menos que la potencia de ruido de cuantización uniforme. Por ejemplo, una cuantización en 7 bits produce una potencia de ruido de aproximadamente -77 dB comparada con los -53 dB de una cuantización uniforme. Este método de codificación es conocido como Ley μ.

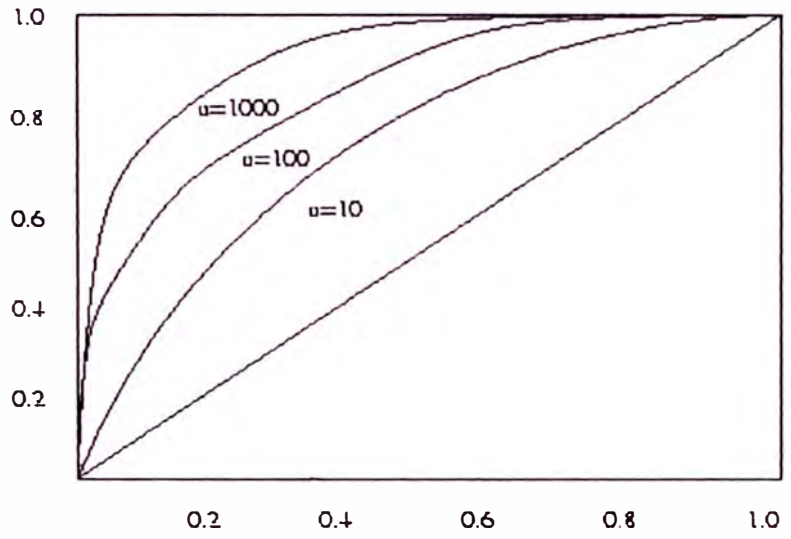


Figura 4.2: Característica de Entrada – Salida del Compresor Logarítmico

El estándar utilizado en Europa se define de una manera similar a la Ley μ y tiene la sgte. expresión:

$$F(x) = \begin{cases} \operatorname{sgn}(x)\dfrac{1+\ln(A|x|)}{1+\ln A}, & \dfrac{1}{A} \le |x| \le 1 \\ \operatorname{sgn}(x)\dfrac{A|x|}{1+\ln|x|}, & 0 \le |x| \le \dfrac{1}{A} \end{cases}$$

Esta es conocida como la Ley A. Para la reconstrucción de los valores cuantizados de la señal se utiliza la inversa de la función logarítmica para expandir la

amplitud. La combinación del compresor-decompresor es conocido como compandor.

4.1.2 Differential Pulse Code Modulation (DPCM).- En PCM, cada muestra es codificada independientemente de las demás. Sin embargo, muchas señales muestreadas al nivel de Nyquist o mayor presentan significante correlación entre muestras sucesivas. En otras palabras, el promedio de cambio de amplitud entre señales sucesivas es relativamente pequeño.

Una solución relativamente simple es codificador las diferencias entre muestras sucesivas. Desde que se espera que las diferencias entre las muestras sean pequeñas que las actuales amplitudes muestreadas, menos bits son requeridos para representar las diferencias. Un refinamiento de esta aproximación general es predecir la actual muestra en base a la muestra anterior. Para ser específico, $\{x_n\}$ denota la actual muestra de la señal, y sea X^n denota el valor predecido de x_n, definido como:

$$X^n = \sum_{i=1}^{p} a_i x_{n-i}$$

Así X^n es una combinación lineal ponderada de las pasadas p muestras, y los $\{a_i\}$ son los coeficientes predictores. Los $\{a_i\}$ son seleccionados para minimizar las funciones de error entre X_n y X^n.

Una conveniente función matemática de error es el error cuadrático medio (ECM). Con el ECM como medida de la performance del predictor nosotros seleccionamos los a_i para minimizar:

$$E_p = E(e^2{}_n) = E\left[\left(x_n - \sum_{i=1}^{p} a_i x_{n-i}\right)^2\right] = E(x^2 n) - 2\sum_{i=1}^{p} E(x_n x_{n-i}) + \sum_{i=1}^{p}\sum_{j=1}^{p} a_i a_j E(x_{n-i} x_{n-j})$$

Asumiendo que la señal de salida es estacionaria, la expresión anterior es equivalente a:

$$E_p = \phi(0) - 2\sum_{i=1}^{p} a_i \phi(i) + \sum_{i=1}^{p}\sum_{j=1}^{p} a_i a_j \phi(i-j)$$

donde $\phi(m)$ es la función de autocorrelación de la secuencia de muestras x_n. La minimización de E_p con respecto a los coeficientes predictores $\{a_i\}$ resulta del siguiente conjunto de ecuaciones lineales:

$$\sum_{i=1} pa_i \varphi(i - j) = \varphi(j), \qquad j = 1,2,\ldots,p$$

De esta manera, los valores de los coeficientes predictores son establecidos[6]

El diagrama de bloques del DPCM es mostrado en la Fig. 4.3. Se aprecia que el predictor es implementado con un lazo realimentado alrededor del cuantizador. La entrada al predictor es denotada por $\overline{X^n}$, el cual representa a la muestra x_n, modificada por el proceso de cuantización, y la salida del predictor es:

$$X^n = \sum_{i=1} pa_i \overline{x_{n-i}}$$

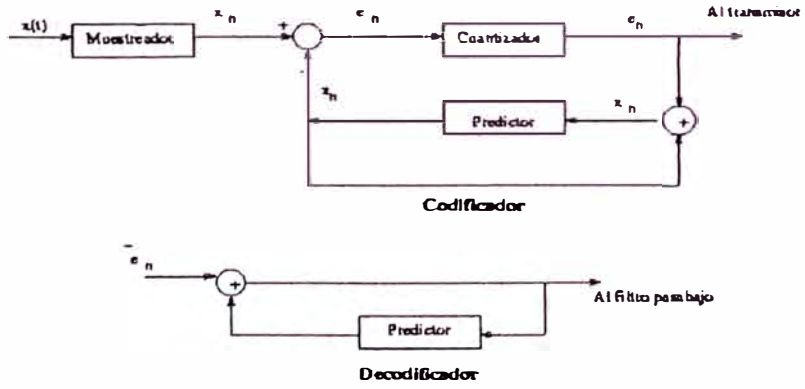


Figura 4.3: Diagrama de Bloques del Codificador y Decodifacor DPCM

[6] El método de solución de estas ecuaciones está basado en el método de Yule-Walker, el cual es explicado en ``Digital Communication'' de John Proakis.

Con esto se ha demostrado que la codificación DPCM permite codificar señales en base a la diferencia entre sus muestras, requiriendo para ello menor cantidad de bits, lo que constituye una compresión de datos.

4.1.3. Adaptive PCM y DPCM.- Muchas de la señales en la naturaleza son cuasi-estacionarias. Un aspecto de la característica cuasi-estacionaria es que las funciones varianza y auto-correlación de la señal de salida varían lentamente con el tiempo. Los codificadores PCM y DPCM, sin embargo, están diseñados sobre la base que las señales a procesar son estacionarias. La eficiencia y performance de estos codificadores pueden ser mejorados si pueden adaptarse a los lentos cambios con el tiempo de las funciones estadísticas de la señal.

En ambos casos, PCM y DPCM, el error de cuantización q_n resulta de la aplicación de un cuantizador uniforme sobre una señal cuasi-estacionaria, teniendo de esta manera una varianza cambiante con el tiempo (Potencia de Ruido de Cuantización). Una mejora, que reduce el rango dinámico de la potencia de ruido es la utilización de un cuantizador de tipo adaptivo. Aunque un cuantizador puede hacerse adaptivo de diversas maneras, un relativamente simple método consiste en utilizar un cuantizador uniforme, pero que varíe su paso de acuerdo a la varianza de las muestras de la señal. Por ejemplo, un término estimado de la varianza de x_n puede ser calculado de la secuencia de entrada x_n y el tamaño del paso puede ser ajustado en base de tal estimado. De esta manera, el algoritmo para el ajuste del tamaño del paso emplea solo la anterior muestra. Un algoritmo ha sido desarrollado por Jayant (1974) para la codificación de señales de voz. La Fig. 4.4. ilustra un cuantizador (de 3 bits) en el que el paso es ajustado recursivamente de acuerdo a la relación

$$\Delta_{n-1} = \Delta_n M(n)$$

Donde M(n) es un factor, cuyo valor depende del nivel de cuantización para la muestra x_n, y Δ_n es el tamaño del paso del cuantizador para procesar x_n. Los valores optimizados de los factores de multiplicación para la codificación de voz han sido dados por Jayant (1974). Estos valores se muestran en la siguiente tabla, para cuantizadores adaptivos de 2, 3 y 4 bits.

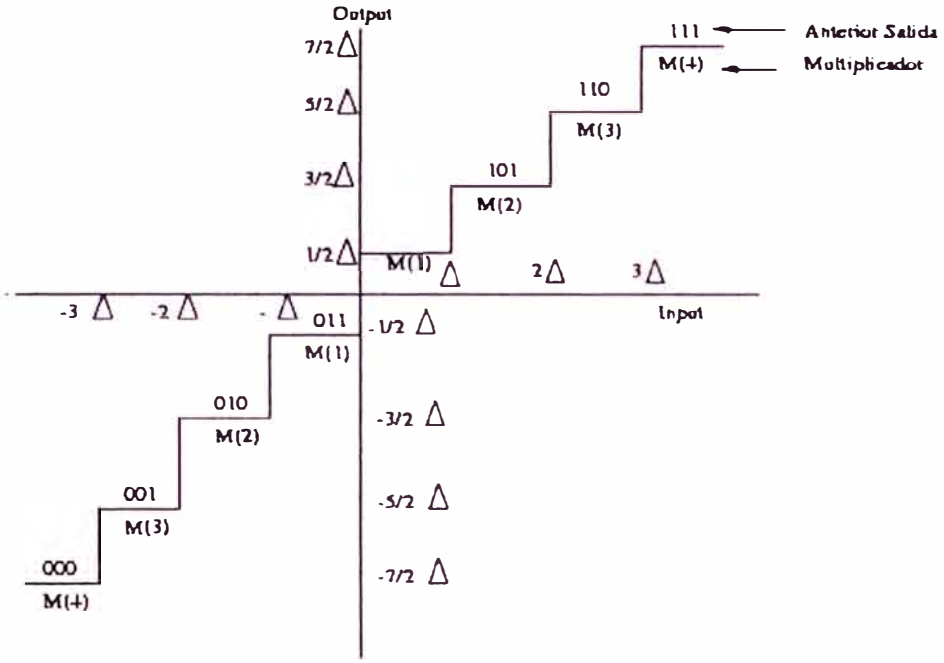


Figura 4.3: Cuantificación de 3 bits con tamaños de paso adaptivo

	PCM			DPCM		
	2	3	4	2	3	4
M(1)	0.60	0.85	0.80	0.80	0.90	0.90
M(2)	2.20	1.00	0.80	1.60	0.90	0.90
M(3)		1.00	0.80		1.25	0.90
M(4)		1.50	0.80		1.70	0.90
M(5)			1.20			1.20
M(6)			1.60			1.60
M(7)			2.00			2.00
M(8)			2.40			2.40

Factores Multiplicativos para un ajuste adaptivo del paso

CAPITULO V
IMPLEMENTACION DEL SISTEMA

Se ha demostrado en el capítulo 3 que IDEA es definitivamente la mejor opción en lo que a algoritmos de encriptación se refiere. A continuación analizaremos el caso particular de la aplicación a implementar: transmisión de datos y voz sobre una Red Local, sobre la cual se montará una solución de encriptación.

5.1 Planteamiento del Problema

Lo primero es definir cuales son los requerimientos y funcionalidades que deseamos que nuestro sistema posea, luego analizar las soluciones existentes y adecuar alguna de ellas para nuestro caso particular y finalmente decidir e implementar la mejor de las alternativas estudiadas. De este modo, aplicando la metodología pensada analizaremos los casos en una manera secuencial.

5.1.1 Objetivos y Requerimientos.- El Objetivo del Trabajo es implementar una plataforma de transferencia de información, la cual sea montada sobre una Red Local, sobre la cual se pueda transmitir archivos de datos y voz en tiempo real, adicionando seguridad en la transmisión a través de métodos de encriptación y compresión de voz. De este modo los requerimientos son los siguientes:

- Transferencia de archivos de información, con aplicaciones Cliente/Ser- vidor.
- Transferencia de Voz, la que debe ser transmitida en tiempo real utilizando técnicas de compresión y con aplicaciones de AudioConferencia.

- Aplicación del Sistema sobre una Red de Area Local (LAN), sobre la que se pueda implementar y probar.
- Utilización de métodos de encriptación para garantizar la seguridad de los datos transmitidos.

A continuación discutiremos los puntos más importantes, a fin de determinar las mas adecuadas soluciones para cada punto.

5.1.2 Plataforma de Aplicación.- La aplicación de transferencia de información debe ser aplicable sobre una Red de Area Local. De este modo se debe definir el método de comunicación a utilizar para conseguir el intercambio de información, utilizando protocolos robustos que aseguren el normal establecimiento de la comunicación y soporte a control de errores en algunas circunstancias (sobrecarga y congestión de red por ejemplo).

En base a esto la aplicación se implementará sobre una Red TCP-IP, debido a su gran difusión y amplia utilización en prácticamente todo campo. Además debemos considerar el control en la comunicación, debido a que debe ser un sistema robusto, el que debe montarse sobre arquitecturas con manejo de control de flujo y corrección de errores. En base a estos puntos se decide entablar la comunicación entre el cliente y el servidor utilizando sockets TCP, debido a que este método de comunicación permite realizar conexiones tipo Cliente-Servidor, es aplicable sobre TCP/IP sobre Areas Locales, y además el utilizar sockets TCP nos garantiza el control de flujo y sesión que no garantiza la utilización de sockets UDP.

Otro aspecto no menos importante es definir la plataforma de aplicación. Si bien es cierto se cuenta con el escenario adecuado para montar la aplicación (Red LAN), es necesario definir el Sistema Operativo que controlará la aplicación y que

será finalmente quien implemente las aplicaciones, tanto el cliente como el servidor. El sistema de mayor utilización actualmente, Microsoft Windows, es un sistema que no está orientado hacia aplicaciones de servicio. El manejo y programación de los dispositivos de comunicaciones Ethernet está gobernado por un conjunto de aplicaciones de software denominada Funciones WINSOCK, las cuales requieren de manejo de API's y programación sobre esta plataforma, además que el sistema como tal requiere de licencias para su normal utilización. Por otro lado la utilización de servidores especializados (SUN Solaris, HP, etc) implicaría una inversión exhorbitante por concepto de costos de los servidores, además como aplicación solo para entornos de desarrollo o producción, no pudiendo aplicarse a nivel de usuario donde la aplicación podría mostrar interesantes funcionalidades.

De este modo la solución debe ser implementado sobre PC's compatibles, de modo que pueda proporcionar una alternativa para un mayor rango de usuarios. La aplicación finalmente es implementada sobre el Sistema Operativo Linux, debido a que es totalmente compatible y de muy buen rendimiento sobre computadoras personales, además de proporcionar versiones de libre distribución, lo que implica un costo nulo en este aspecto.

Por otro lado Linux proporciona al usuario primitivas de programación directa de sus interfases, incluyendo sockets de comunicación, siendo sencillo su control y programación. La distribución adoptada es RedHat, versión de Linux que ha experimentado el más grande desarrollo y difusión en los últimos años debido a las innumerables contribuciones que ha recibido. De este modo la utilización de sockets nativos (sockets BSD) es la solución mas adecuada para la implementación de nuestras aplicaciones.

5.1.3 Compresión de Voz.- El procesamiento seguro de voz implica la adquisición de muestras de voz a una determinada frecuencia de muestreo y resolución de bits. Posteriormente se aplica el método de encriptación a estas muestras digitalizadas, las cuales son transmitidas. Sin embargo, las señales de voz poseen la característica que sus muestras poseen una gran correlación entre dos muestras sucesivas, es decir, las diferencias entre muestras son muy pequeñas. De esta manera los sistemas comunes de adquisición, para el caso de voz, son en extremo redundantes. La solución propuesta implementa la codificación ADPCM, la que se encarga de codificar precisamente las diferencias de las muestras de voz, requiriendo para ello menor cantidad de bits, y por ende se genera menor cantidad de datos, garantizándose la no pérdida de la información. Por ello se decide utilizar ADPCM para la transmisión y de esta forma conseguir mayor eficiencia.

5.1.4 Sistema de Encriptación.- El sistema de encriptación es de suma importancia. Este se montará sobre la arquitectura de comunicación y en conjunto con las aplicaciones Cliente - Servidor implementadas deberá completar la transferencia segura de la información. Dos puntos a considerar son: 1. ¿Qué método de encriptación utilizar? y 2. ¿Sobre que arquitectura debe implementarse?

Sobre el primer punto debemos considerar en primer lugar que la aplicación implica transferencia de datos importantes, como archivos de información valiosa y comunicaciones de voz, las cuales claramente definen una política de comunicación privada. De este modo la metodología óptima de aplicación es Pretty Good Privace (PGP)[7] . Como se apreció en el capítulo 2, IDEA es la mejor solución para aplicaciones de transferencia de archivos y su arquitectura permitirá el

procesamiento de voz si consideramos a la transferencia de voz como una transferencia de muestras de voz digitalizada, los que representan datos finalmente. Estas razones, además del excelente nivel de encriptación, hacen que finalmente IDEA sea adoptado para el desarrollo del presente trabajo.

La segunda interrogante nos hace reflexionar sobre la manera de implementar IDEA. Las posibles soluciones son implementación en software y en hardware. La implementación en software implicaría la facilidad de implementación al permitir desarrollos en lenguajes de alto nivel que faciliten la labor de programación, pero no proporcionan una adecuada performance en el proceso debido a que la PC sobre la que se implementa, que gestiona su aplicación Cliente y/o Servidor, será la encargada del control del proceso, lo que puede ser "pesado" debido al tipo de información a manejar (voz). La solución en hardware proporciona las ventajas que no ofrecen las soluciones de software, es decir la velocidad y precisión en el procesamiento de datos, al poseer un componente especializado y dedicado a la función de encriptación. Por otro lado implicaría una mayor complicación en la implementación, debido a que la programación de un dispositivo de hardware es orientada hacia el manejo a bajo nivel, pero debido a que la estructura que IDEA utiliza es basada en operaciones aritméticas a nivel de bits hace que la implementación en hardware no sea en extremo complicada y que garantice satisfactoriamente la velocidad de procesamiento. En este contexto la aplicación es montada sobre una tarjeta de Procesamiento Digital de Señales (DSP), específicamente sobre la DSP56002 de Motorola [6]. Las principales especificaciones de la tarjeta son:

[7] **Revisar Capítulo: Fundamentos Matemáticos**

- Velocidad de Procesamiento: 40 MHZ
- Número de Bits de Procesamiento: 24 bits
- Interfases de comunicación: Serial (UART de 62500 baudios), e Interfase Paralela.
- Arquitectura: Punto Fijo
- Capacidad: Dos Memorias RAM de 64K, y 64K De Memoria de Programa

Con estas características la DSP56002 ofrece precisión, alta velocidad de procesamiento, e integración con la PC con las interfases más comúnmente utilizadas. De este modo la aplicación se implementará en base a los siguientes parámetros:

1. Implementación del Sistema de Comunicación sobre una Red de Area Local con protocolo TCP/IP
2. Arquitectura Cliente-Servidor, con la comunicación basada en sockets, utilizando como plataforma de aplicación Linux, siendo utilizada para este caso RedHat.
3. Implementación de IDEA, sobre un DSP56002, el que se comunicará con la PC mediante RS-232 y/o transferencia paralela.
4. Utilización de codificación ADPCM para la aplicación de voz, logrando una compresión de 4 a 1.

Finalmente hemos conseguido definir el escenario sobre el que aplicará la solución presentada, además de los medios para conseguir la transmisión segura de la información.

5.1.5 Redefinición de IDEA.- Se ha conseguido definir la estructura que tendrá el sistema a implementar. La solución de implementar IDEA sobre un DSP proporciona precisión y alta velocidad de procesamiento. El problema que surge ahora es analizar

un problema inherente a todo microprocesador o dispositivo de hardware: la precisión. IDEA es uno de los más confiables y seguros métodos de encriptación debido a que utiliza 64 bits de datos y 128 bits de llave de encriptación [1], pero esto no es aplicable como tal en un dispositivo de hardware.

El manejo de datos que el DSP56002 posee es mostrado en la Fig. 5.1. Posee 2 acumuladores de 56 bits, los que representan 2 registros de 24 bits, además de 8 bits de signo. De este modo se posee un procesamiento de 24 bits, que es superior a la mayoría de los DSP's existentes en el medio, pero que no es suficiente para implementar IDEA en su estructura original.

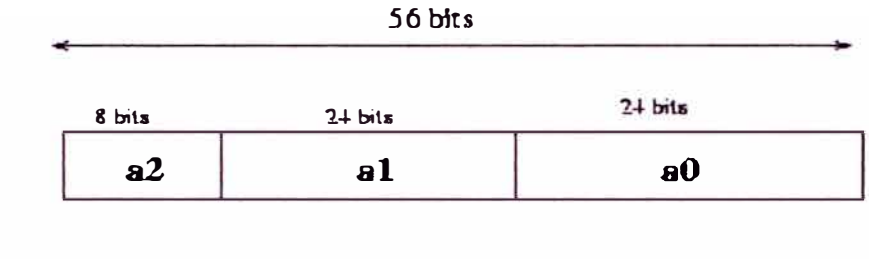


Figura 5.1: Estructura del Acumulador del DSP56002

Para solucionar esto debemos adaptar IDEA para que sea aplicable a nuestro caso particular. De este modo debemos aprovechar al máximo los 24 bits que proporciona el DSP para procesamiento y almacenamiento de datos. De este modo planteamos la sgte. estructura de bloques de datos y llave para la encriptación.

Longitud de Bloque de Datos: 24 bits

Longitud de la Llave: 48 bits

Con esto mantenemos la relación Datos/Llave que IDEA propone. Este cambio naturalmente implica la modificación de los métodos de obtención de las sub-llaves y el tamaño de los bloques de datos. Para ello consideramos que la entrada de datos tendrá una longitud de 24 bits y esta es la que ingresa a la primera celda de

encriptación[8], la cual recibe 4 bloques de datos. Las condiciones del problema hacen que optemos por 4 bloques de 6 bits, con sub-llaves también de 6 bits.

La implementación original de IDEA implica la presencia de 8 celdas de encriptación en cascada y una transformación a la salida. En el capítulo III se demostró que el algoritmo de IDEA no depende del número de etapas, solo que el mismo número de celdas se utilice en las etapas de encriptación y de recuperación de datos. La transformación de salida es fundamental y debe considerarse de todas maneras.

Debido a que IDEA proporciona un muy buen nivel de encriptación y considerando la aplicación de voz que implicará un sistema rápido y capaz de procesar los datos a frecuencia de muestreo se implementa una etapa de encriptación mas la transformación de salida, con la previa justificación de la calidad de encriptación. Este punto se explica con detalle en las conclusiones del trabajo. Finalmente optamos por una solución que es mostrada en la Fig. 5.2.

5.1.6 Generación de las sub-llaves.- Solo nos queda definir el mecanismo de obtención de las llaves de encriptación y de desencriptación, el que se efectuará en base a un proceso similar al mostrado en el capítulo III. El desplazamiento de 25 posiciones a la izquierda es reemplazado por desplazamientos de 7 posiciones, y de acuerdo a la fórmula 3.1 el número de sub-llaves requeridas es:

$$\#\ \text{sub-llaves} = 6 * 1 + 1 = 10$$

[8] Ver Capítulo III: El Algoritmo IDEA

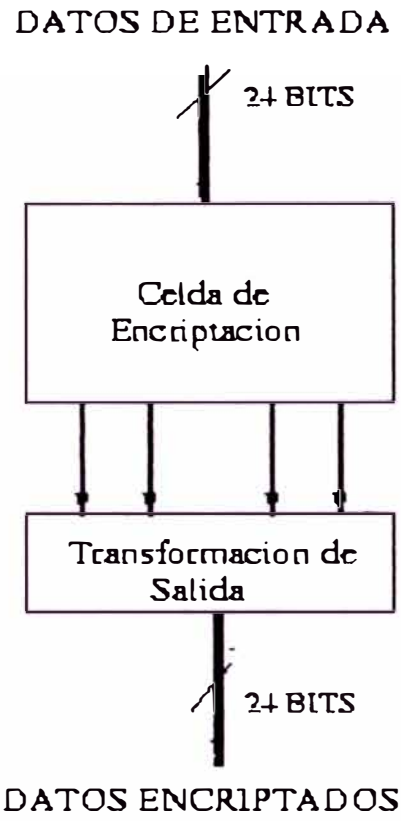


Figura 5.2: Sistema de Encriptación a implementar

De acuerdo a esto, las sub-llaves de encriptación que se derivan de la llave principal, se obtienen de la manera mostrada en la Fig. 5.3.

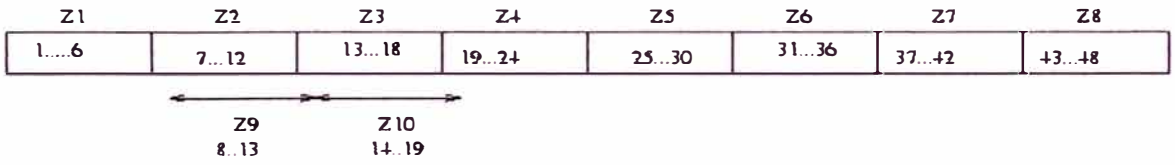


Figura 5.3: Obtención de las sub-llaves de encriptación.

De este modo la etapa de encriptación es realiza de igual manera que el algoritmo original. El cálculo de las sub-llaves de desencriptación se realiza análogamente a las recomendaciones dadas en el capítulo III. De esta manera finalmente se obtiene lo sgte.:

$$U_1 = Z_7^{-1}, \qquad U_6 = Z_6$$

$$U2 = -Z_8, \qquad U_7 = Z_1^{-1}$$

$$U_3 = -Z_9, \qquad U_8 = -Z_2$$

$$U_4 = Z_{10}^{-1}, \qquad U_9 = -Z_3$$

$$U_5 = Z^5, \qquad U_{10} = Z_4^{-1}$$

Con esto hemos conseguido adaptar IDEA a nuestros requerimientos, siendo posible su implementación. A continuación detallaremos la implementación en sí del sistema, tratando sobre el módulo encriptador, las aplicaciones Cliente y Servidor y la interfase de comunicación PC-DSP.

5.2 Implementación

El sistema de comunicación a implementar proporcionará seguridad en la transmisión mediante su módulo encriptador implementado en hardware. El sistema planteado es como se aprecia en la Fig. 5.4.

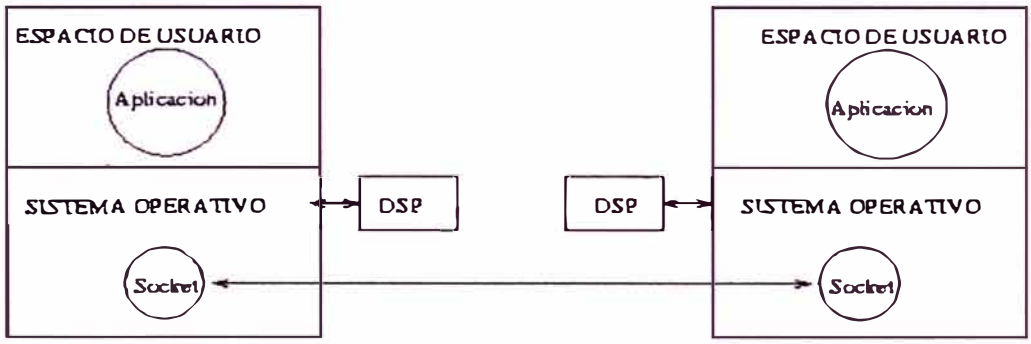


Figura 5.4: Sistema de Comunicación planteado

Como se aprecia en la figura, la implementación requiere de los siguientes componentes:

- Interfase de Comunicación PC - DSP
- Sockets de Comunicación.
- Implementación de IDEA sobre el DSP56002

A continuación detallaremos cada uno de estos componentes y mostraremos la manera en la que se integran y forman el sistema deseado.

5.2.1 Interfase de Comunicación PC-DSP.- Está definido como el medio de comunicación e intercambio de información entre el elemento supervisor y de control (PC) con el módulo encriptador (DSP56002). La PC necesita mantener una comunicación fluida y constante con el DSP debido a que deben ser transferidos a este último los datos adquiridos antes de su transmisión, para luego de recepcionados ser enviados al receptor (Servidor). La interfase de comunicación puede ser cualquiera de los dos estándares mas utilizados: La comunicación serial RS232, y la comunicación Paralela.

Breve Descripción del DSP56002

EL DSP56002 pertenece a la familia de Procesadores Digitales de Señales de Motorola [6]. Es un componente de hardware especializado en labores como procesamiento de audio en tiempo real, filtros digitales, cancelación de ruido, adquisición de datos, etc. Sus características principales son:

Características del CPU DSP56002

- 20 Millones de Instrucciones por Segundo (MIPS) a 40 MHZ.
- Procesamiento Paralelo en ciclos simples de 24x24 bits.
- Instrucciones con Procesamiento Paralelo con Modos de Direccionamiento DSP.
- Rápidos Auto-Retorno a Interrupciones.
- Diseño CMOS con muy bajo consumo de potencia.

Especificaciones del Módulo DSP56002

- Memoria de Programa de 512x24.
- Dos Memorias RAM 256x24.

- Dos Memorias ROM 256x24 (Tablas de Senos y Cosenos).
- Alta Velocidad de Expansión y Comunicación con puertos mediante buses de datos de 16 y 24 bits.
- Soporte para DMA.
- Interfase Serial Sincrónica (SSI).
- Interfase de Comunicación Serial (SCI).
- Pines de 24 bits I/O de Propósito General.
- Timer/Event Counter de 24 bits.
- On-Chip Emulator (OnCE) para comunicación con la PC.

La distribución física de los componentes de la Tarjeta DSP56002 es mostrada en la Fig 5.5.

COMUNICACION SERIAL

La Tarjeta DSP56002 posee soporte para la Interfase de Comunicación Serial estándar. De este modo la integración con la PC utilizando este estándar es posible. Como todo microprocesador, el DSP56002 necesita de una programación en bajo nivel que permita configurar sus dispositivos de propósito general a realizar determinadas tareas y de una determinada manera. De este modo, para poder utilizar una comunicación de tipo serial es necesario previamente definir los siguientes parámetros: Protocolo de Comunicación Serial y Velocidad de Transmisión.

El DSP56002 posee integrado un Universal Asinchronus Rate Transmition (UART), que realiza las transformaciones Serie/Paralelo y Paralelo/Serie necesarias para la comunicación, además de poseer un reloj interno que permite especificar múltiples velocidades de transmisión. La velocidad síncrona del DSP56002 es 62500 baudios.

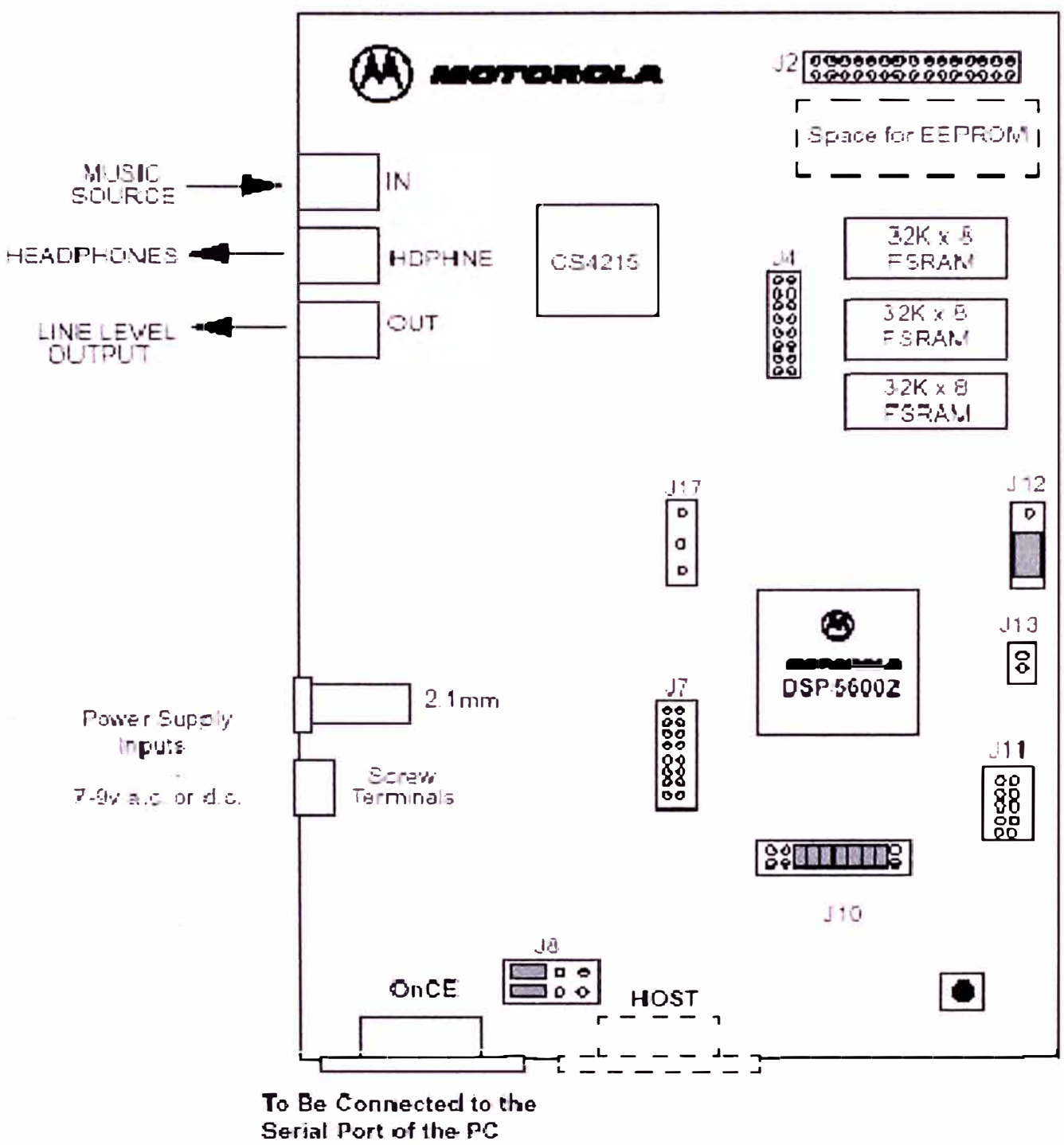


Figura 5.5: Componentes de la Tarjeta DSP56002

Por otro lado las PC Compatibles actuales poseen el 16650A, que es un UART que soporta hasta 115200 baudios. De este modo la velocidad de transmisión a utilizar debe ser común para ambas interfases, y por ello elegimos la velocidad de 10400 baudios, que es alcanzable tanto por la PC y por el DSP. Elegimos este valor no estándar debido a que otros valores, como por ejemplo 4800 baudios resultan muy lentos para la transmisión. De esta manera los parámetros de la comunicación son:

- Protocolo de Transmisión Serial: 8 bits de datos, 1 de inicio y 1 de parada, sin control de flujo y sin bits de paridad.
- Velocidad de Transmisión: 10400 baudios.

De esta forma la comunicación serie está totalmente definida, y se encuentra lista para ser utilizada. Este proceso será explicado posteriormente.

COMUNICACION PARALELA

La transferencia de información utilizando comunicación serial es muy segura y confiable, pero no proporciona velocidades muy alta de transmisión. Además existe dependencia del manejo de relojes internos en cada uno de los componentes del sistema, lo que limita la velocidad tal cual sucedió en el caso anterior, donde al tener relojes de 62500 y 115200 baudios la velocidad máxima es solo 10400 baudios.

La interfase paralela soluciona este inconveniente al proporcionar transferencia de datos en paralelo, traduciéndose en mayor velocidad de transmisión. La PC posee soporte de comunicación paralela mediante su puerto LPT1, el que posee un comportamiento de transmisión y recepción de datos mediante la utilización y programación de algunos puertos reservados de entrada y salida de la PC, los cuales son:

Tipo	Puerto I/O	Cantidad de Bits de Datos	I/O
Puerto de Datos	0x378	8 Bits	Output
Puerto de Estado	0x379	5 Bits	Input
Puerto de Control	0x37A	5 Bits	Output

En base a esto la transmisión de datos puede conseguirse a través del envío de un byte de datos (8 bits), pero con una recepción de un máximo de 5 bits. Por otro lado el DSP56002 posee una interfase paralela conocida como el Port B [6], que proporciona 15 pines de propósito general, los que pueden ser configurados tanto como entrada como de salida. En este contexto definimos 8 pines de datos de recepción en el DSP (que recibirán los 8 bits de salida de la PC), 4 bits de salida (que serán recepcionados por la PC), además de un bit de entrada y uno de salida que servirá para la señalización , y que son conocidos como bits de STROBE (STR) y ACKNOWLEDGE (ACK).

El protocolo de comunicación paralela es mostrado en la Fig. 5.6.

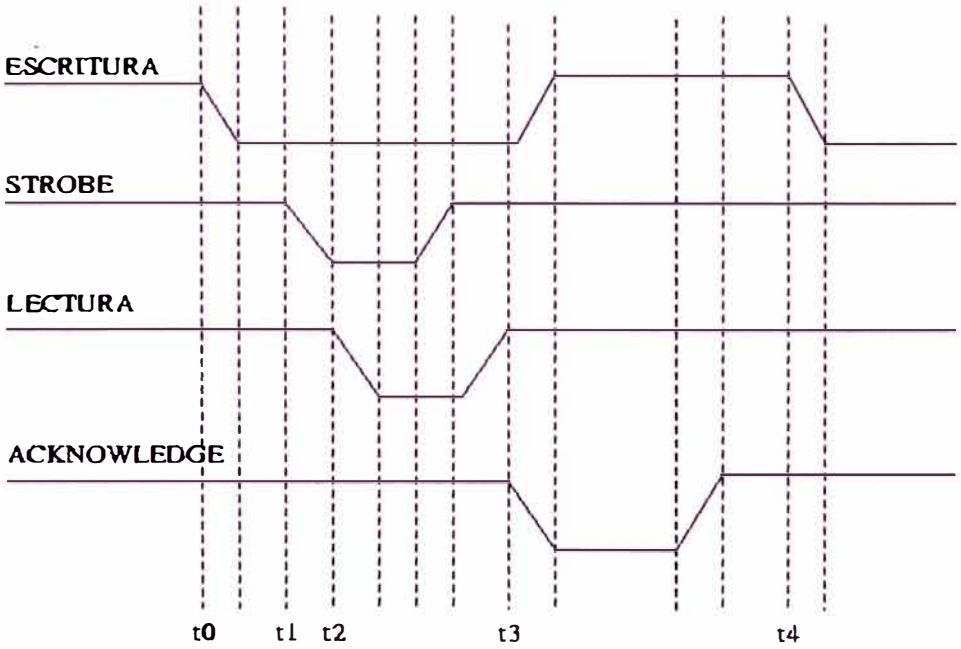


Figura 5.6: Comunicación Paralela

Como se aprecia los datos son escritos en el bus de comunicaciones en el tiempo t_0. Estos datos no serán leídos hasta que el transmisor indique que es posible la lectura confiable, por lo que baja su STROBE en el tiempo t_1. Luego de esto los datos son leídos por el receptor en el tiempo t_2. Por otro lado el transmisor no puede transmitir nuevamente si no está seguro que el receptor leyó los datos, pues espera un ACKNOWLEDGE (acuse de recibo) desde el receptor. Este es enviado en el tiempo t_3, y de este modo el transmisor vuelve a colocar datos en el tiempo t_4, reanudándose el proceso anterior.

5.2.2 Sockets de Comunicación.- La comunicación entre el Cliente y el Servidor se efectuará sobre la arquitectura de una Red de Area Local, sobre la cual se efectuará una transferencia de datos utilizando el protocolo TCP/IP. Bajo este contexto la comunicación se efectúa utilizando sockets TCP en lugar de sockets UDP con el fin de ganar en control de sesión y manejo de errores.

El sistema operativo de control es SO Linux. De este modo el Servidor deberá abrir un socket disponible, y quedar a la espera del establecimiento de alguna comunicación con el cliente. Durante este tiempo el Servidor deberá esperar indefinidamente, manejando el sockets mediante un demonio de red [9]. Tanto las aplicaciones Cliente como Servidor son implementadas en lenguaje ANSI C, debido a que es un lenguaje ligero y de buen control de periféricos de hardware.

Cuando el cliente desea transmitir datos establece una petición de conexión con el servidor, es decir intenta realizar una conexión en el socket del servidor. El canal de comunicación es bidireccional, debido a que aparte del flujo de información del Cliente hacia el Servidor también existe flujo de información del Servidor hacia

el Cliente, la que es concerniente a la señalización y acuse de recibo. La señalización de la aplicación de transmisión es mas notoria para el caso de transmisión de archivos, la que se muestra en la Fig. 5.7.

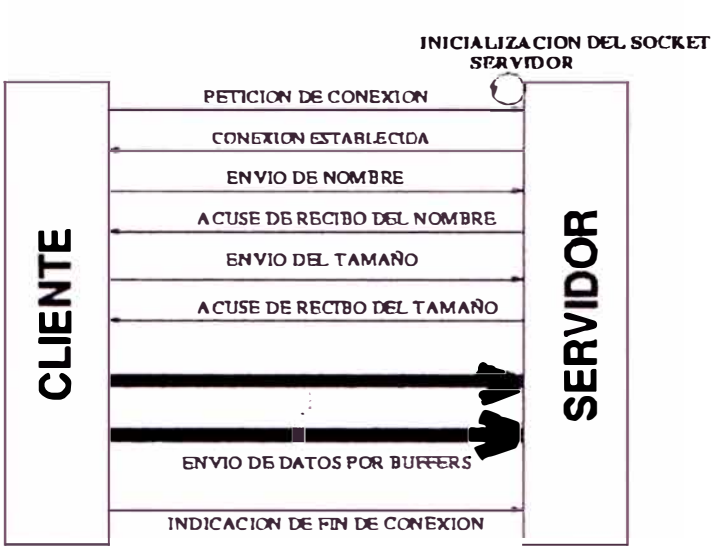


Figura 5.7: Esquema de Señalización de la Transferencia de Archivos

De esta manera se efectúa la comunicación con la respectiva señalización. En este caso luego de establecida la comunicación es necesario enviar el nombre y la talla del archivo a transferir. La información es enviada en bloques debido a que cada socket maneja un tamaño de buffer de almacenamiento. Esto también es provechoso para el proceso de encriptación, debido a que el archivo de información será segmentado y procesado por el módulo encriptador antes de su envío, y de igual modo, cada paquete recibido será desencriptado a su arribo al servidor.

5.2.3 Implementación de IDEA sobre el DSP56002.- Esta sección describe la parte mas importante del presente trabajo: la implementación del sistema de encriptación de datos sobre el DSP56002. Hasta ahora la plataforma de comunicación ya ha sido definida, incluso discutido los dos posibles tipos de

[9] Demonio: Aplicación de Sistema que se ejecuta siempre y en background

comunicación entre la PC y el DSP56002, pero recién ahora especificaremos la implementación de IDEA en hardware.

El proceso de adquisición de datos por el DSP está ligado directamente a la labor de la PC. Es la PC la que dispone en que momento enviará datos a ser procesados hacia el DSP56002, y luego de ser recibidos el DSP deberá continuar en su proceso de espera. De esta manera el DSP se comporta como un elemento pasivo de comunicación pero que está siempre a la espera del arribo de datos.

Por otro lado el DSP puede trabajar en dos maneras: como encriptador y como desencriptador. Como se apreció en los capítulos anteriores ambos procesos solo se diferencian en las llaves a utilizarse. De esta manera el DSP debe ser informado acerca del proceso que debe realizar. Para ello debe existir una señalización inicial entre la PC y el DSP para definir que proceso utilizar. Luego de esto se debe realizar el envío de datos desde la PC hacia el DSP y posteriormente el DSP debe devolver los valores resultantes, para luego regresar a su estado inicial de espera.

El proceso de envío y recepción de datos está basado en el concepto de tramas. Como se vió en la sección anterior la comunicación por sockets es realizada en base paquetes de transmisión. Estos son obtenidos por el cliente (ya sea del buffer de la tarjeta de sonido para el caso de voz o del archivo de información para el caso de transmisión de datos), distribuidos en paquetes de bits, enviados al DSP, recuperados y finalmente transmitidos. El mecanismo de comunicación se muestra en el diagrama de tiempo de la Fig. 5.8.

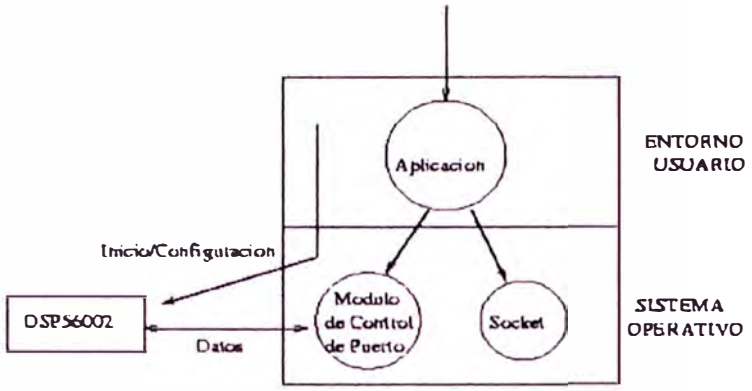


Figura 5.8: Proceso de Comunicación PC-DSP56002

El tamaño del paquete de transmisión entre los sockets Cliente y Servidor no es necesariamente igual al tamaño del buffer de transmisión entre la PC y el DSP. Esto es debido a que para conseguir un tamaño dado para el buffer de transmisión de los sockets puede hacerse varios intercambios de información entre la PC y el DSP, de modo que en base a paquetes mas pequeños se consiga el tamaño deseado.

MODULO ENCRIPTADOR

El DSP56002 proporciona todo lo necesario para implementar una aplicación de procesamiento de datos en tiempo real [10]. Para ello posee un espacio de memoria RAM programable denominada "Memoria de Programa", sobre la cual se cargan las instrucciones en lenguaje de bajo nivel que permitan realizar la aplicación deseada. Estos códigos son generados mediante el lenguaje ensamblador que proporciona el fabricante y son transferidos hacia el DSP mediante la interfase de programación On-Chip Emulator (OnCE) [6].

Como se apreció anteriormente el DSP requiere de una notificación que le indique el tipo de procesamiento que debe realizar. De esta manera programa las

[10] Estas características fueron detalladas al principio del presente capítulo

llaves adecuadas y procesa los datos recibidos. Por lo tanto el módulo encriptador poseerá las sgtes. etapas:

- Recepción de la Palabra de Configuración.
- Módulo Receptor de datos.
- Módulo Encriptador - Desencriptador.
- Módulo de Transmisión.

Este proceso puede apreciarse de mejor manera en la Fig. 5.9, en la que se presenta un diagrama de flujo del proceso al interior del DSP56002.

De esta manera siempre es el DSP el que está a la espera de la comunicación, siendo la PC la que posee el control de las aplicaciones y de los procesos.

5.3 Sistema Desarrollado

5.3.1 Aplicación de Transferencia de Datos.- Para este caso de la aplicación desarrollada, los archivos de datos son leídos por el cliente. El Servidor ha inicializado el socket correspondiente y espera por la comunicación, la que es efectuada por el cliente, proporcionándole los datos de nombre de archivo y tamaño del fichero. Luego la información es paquetizada en bloques de 255 bytes, los que son enviados al DSP para su encriptación. Este número de bytes es elegido debido a que es necesario un buffer de transmisión múltiplo de 3, debido a que el DSP56002 almacena los datos recibidos en sus registros de 24 bits=3 bytes. Luego se espera el retorno de los datos, los que son transmitidos al socket servidor que se encuentra esperando. Al arribar el primer paquete el Servidor desencripta la información y la almacena en el fichero que previamente abrió, con el nombre de archivo recibido. El proceso continúa hasta que el servidor se da cuenta que el archivo está completo

debido a que conoce el tamaño del mismo, cierra el archivo, lo guarda en disco y termina la aplicación. De esta manera la aplicación final es mostrada en la Fig. 5.10.

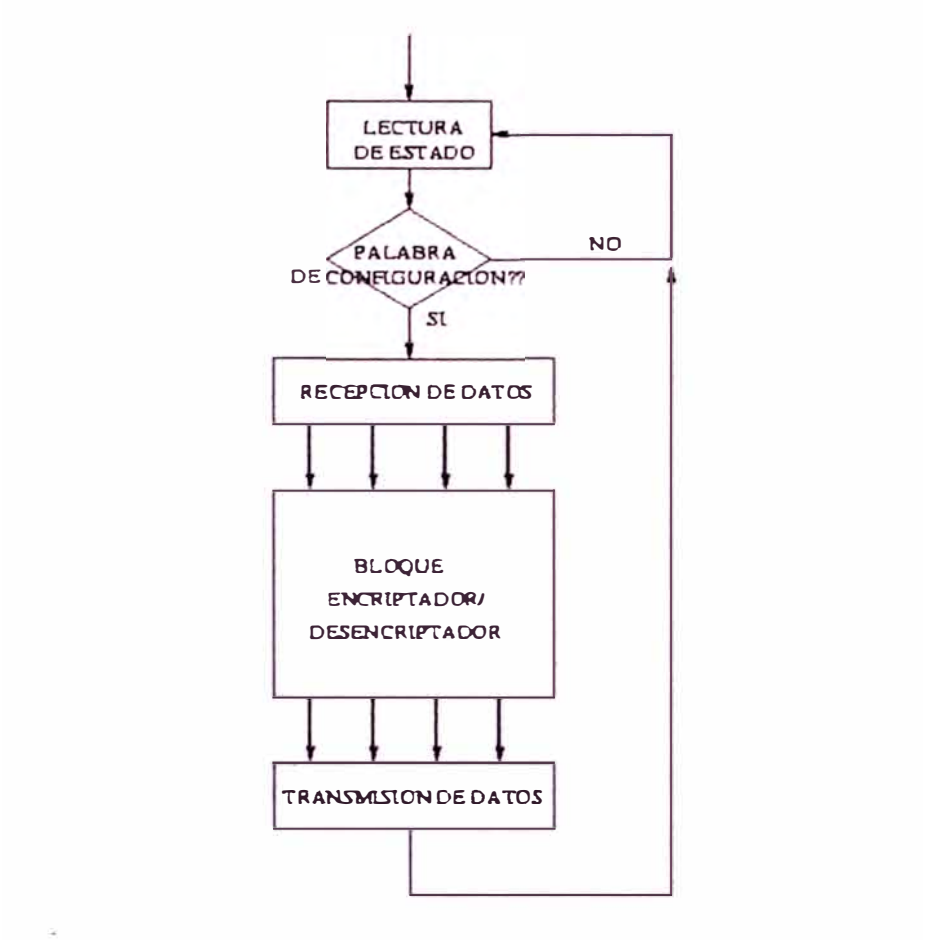


Figura 5.9: Diagrama de flujo del módulo encriptador/desencriptador

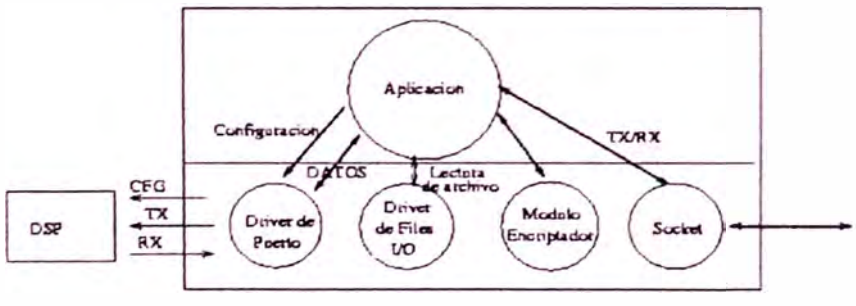


Figura 5.10: Diagrama de flujo del módulo encriptador/desencriptador

5.3.2 Aplicación de Transferencia de Voz.- La transferencia segura de información por el canal de comunicación está garantizada gracias a la estructura de encriptación montada, la que permite el procesamiento en tiempo real de los datos

recibidos y la posterior transmisión y recuperación de los mismos. La voz es también un tipo de información. Si bien es cierto el concepto de voz está relacionado a formas de onda de naturaleza analógica (como la mayoría de las señales en el mundo real) es posible considerarla como datos mediante el proceso de discretización y muestreo de señales[11].

Este concepto es el punto de partida para concebir una aplicación de transferencia de información, en las que los datos a transmitir sean muestras de voz, las que son adquiridas y procesadas en tiempo real. A continuación se detalla la aplicación, enfatizando en las etapas de adquisición y muestreo por el Cliente, encriptación, compresión, transmisión, recuperación y finalmente salida de audio en el Servidor.

* ETAPA DE ADQUISICION DE DATOS

El Proceso de adquisición de la voz en su forma natural (es decir analógica) es realizada en la etapa del Cliente. La utilización de la plataforma Linux permite que la obtención de las muestras sea realizada a través de la Tarjeta de Sonido que el Cliente posee. De esta manera la información será adquirida mediante un micrófono, posteriormente se configura los parámetros mas importantes de la Tarjeta de Sonido como son la fragmentación del DMA, el número de canales PCM a utilizar, el número de bits y la frecuencia de muestreo. La aplicación propuesta presenta los sgtes. valores para estos parámetros:

- Tamaño del DMA (por defecto): 4 Kb
- Fragmentación aplicada: 512 bytes
- Número de Canales PCM: 1
- Número de bits: 16 bits

[11] Ver Apéndice: Encriptación en Sistemas de Comunicación

- Frecuencia de Muestreo: 4KHZ

Cabe destacar que el proceso de adquisición es realizado a 16 bits debido a que se utilizará ADPCM en la codificación, el que solo requiere de 4 bits de información. De esta manera se requiere de una muy buena resolución de muestras en la codificación inicial, de modo que las diferencias entre muestras consecutivas se encuentren definidas de la mejor manera posible y una codificación a 16 bits la proporciona satisfactoriamente. De esta manera la compresión es de 4 a 1. La frecuencia de muestreo 4 KHZ es adoptada debido a que la interfase de comunicación entre la PC y el DSP es totalmente asincrónica, lo que origina problemas en transferencia a altas velocidades. Las muestras a 4 KHZ proporcionan una baja calidad de audio, pero sin embargo para fines de transmisión de voz es aún aplicable, debido a que si bien son suprimidas las componentes de frecuencia agudas se consigue mínimo retardo y recorte de la voz en la transmisión, como se demuestra en los resultados experimentales. Posteriores trabajos podrán realizar codificaciones más óptimas para transmitir mayor cantidad de información, solo con el hecho de mejorar la interfarse PC-DSP, siendo posible no solo la transmisión de voz sobre IP, sino audio de alta calidad y hasta aplicaciones de seguridad en comunicaciones telefónicas, con solo modificar ligeramente la presente aplicación. Con estos parámetros Linux nos garantiza la adquisición de datos de manera segura y confiable. Los datos codificados son almacenados en un buffer de información, el cual será procesado por el módulo encriptador, de igual manera para el caso de datos, como se apreció en páginas anteriores.

La comunicación entre el Cliente y el Servidor se realiza de igual manera para el caso de datos. El Servidor inicializa su servicio y espera la petición del Cliente, y al

darse esta se procede a la señalización y la posterior transferencia. Para el caso de voz es diferente debido a que no se maneja el concepto de talla de archivo ni final de conversación, debido a que el Cliente puede estar adquiriendo datos continuamente para el caso de una aplicación de audioconferencia, y por ello la comunicación finalizará al no detectarse mas muestras en el lado del Cliente, lo cual será notificado al Servidor para finalizar la transferencia. De esta manera, los parámetros de la comunicación PC-DSP y Cliente/Servidor son:

- Buffer de Transmisión: 255 bytes
- Buffer de Comunicación del Socket: 256 bytes.

La programación de la Tarjeta de Sonido en el lado del Servidor es análoga al Cliente. Los parámetros son los mismos, solo que en este caso el Servidor dispone de un parlante de audio conectado a la salida de la Tarjeta. De esta manera el Servidor al recibir las tramas de voz procede a desencriptarlas, descomprimirlas y enviarlas al dispositivo de audio, el que se encargará de reproducirlas en tiempo real. El esquema global de la aplicación es mostrado a continuación en la Fig. 5.11. Esto completa el proceso de transmisión de voz. Cabe recalcar que es gracias a las facilidades que ofrece un sistema como Linux que es posible concebir una aplicación de este tipo de manera tan sencilla.

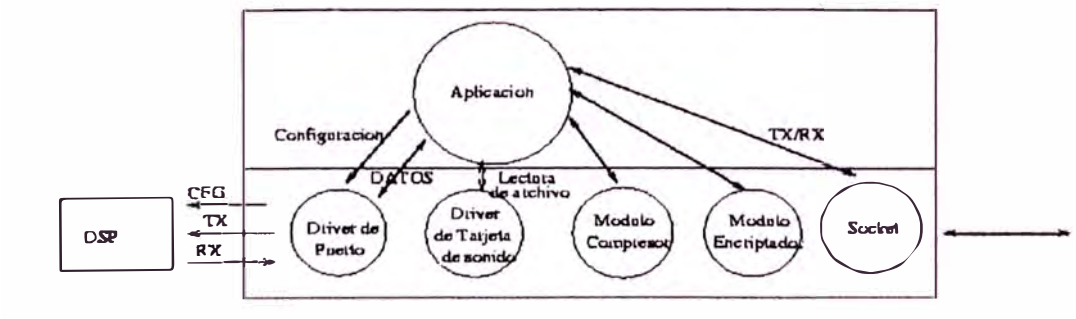


Figura 5.9: Diagrama de flujo del módulo encriptador/desencriptador

CAPITULO VI
RESULTADOS

Se ha descrito las características del sistema implementado, además de discutido el porqué de cada etapa o proceso adoptado. En esta sección destacaremos los resultados de las pruebas efectuadas y algunas características de interés en la puesta en marcha de la aplicación.

6.1 Velocidad de Transmisión

La aplicación presentada proporciona mayor velocidad de procesamiento utilizando la comunicación paralela como interfase de intercambio de información entre la PC y el DSP. De este modo los datos son transmitidos en forma paralela, y se hace necesario conocer la máxima velocidad de transmisión de datos. Para ello analizaremos los dos casos que se presentan: Transmisión PC-DSP y Transmisión DSP-PC.

6.1.1 Transmisión PC – DSP.- Este caso es el más sencillo de analizar. La comunicación es en base al puerto paralelo (por el lado de la PC) y por el Puerto Digital B por el lado del DSP56002. El Puerto B está constituído por 15 pines de propósito general, configurables mediante software para trabajar ya sea en modo entrada como salida.

De acuerdo a lo apreciado en el capítulo anterior, el puerto paralelo de PC posee 8 bits de salida de datos. De esta manera es posible realizar una transmisión

utilizando tramas de 8 bits (un byte) hacia el DSP. Por otro lado todos los pines del Puerto B son configurables y de esta manera elegimos programar los 8 primeros bits como entrada, de modo que ellos se conecten a la salida de datos de la PC, recibiendo de esta manera la información.

El protocolo de transferencia paralela maneja señalización del tipo "acuse de recibo", siendo necesario dos señales de control. El Puerto Paralelo posee adicionalmente a la salida de datos (8 bits) 4 salidas de control y 5 entradas de estado. Estos pines son controlados mediante software utilizando los registros asociados al puerto Paralelo (0x378). De este modo utilizamos un pin como entrada y otro como salida, consiguiendo las señales que trabajarán como STROBE y ACKNOWLEDGE [12]. De esta forma la distribución de los pines elegida para cada una de las interfases a utilizar es la mostrada en la Fig. 6.1.

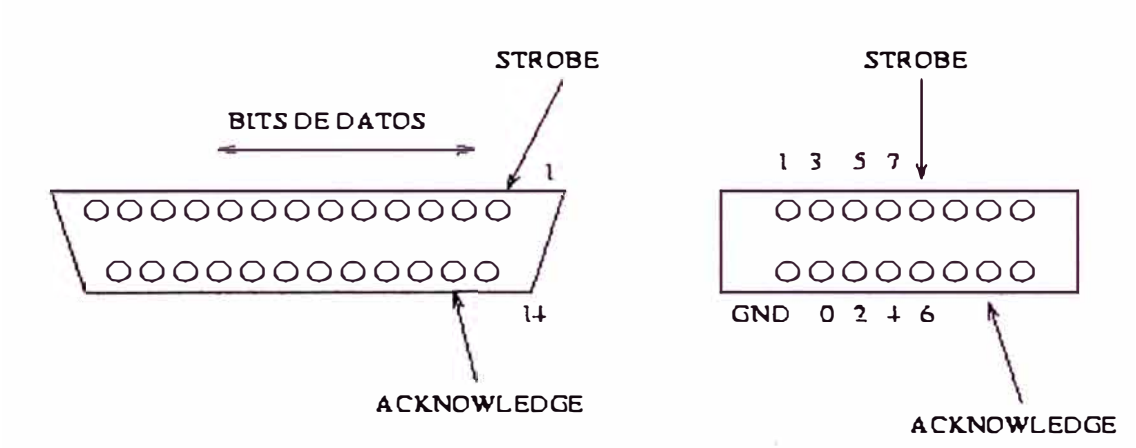


Figura 6.1: Distribución de pines para la transmisión PC-DSP

6.1.2 Transmisión DSP – PC.- Para este proceso inverso se utiliza la misma filosofía del caso anterior, es decir, la comunicación utiliza acuse de recibo para la señalización, pero utiliza otros pines en cada interfase para lograr la transmisión. Para recibir datos hacia la PC ya no se poseen los 8 bits de datos debido a que estos

son utilizados como salida. Por ello utilizamos 4 de los bits de control del puerto para conseguir la lectura, de modo que la información es transmitida word por word (4 bits en 4 bits). En la parte del DSP los pines del 8 al 11 se configuran como salida, los que son conectados a los 4 pines de entrada de la PC. Finalmente la distribución de pines para este caso es mostrada en la Fig. 6.2.

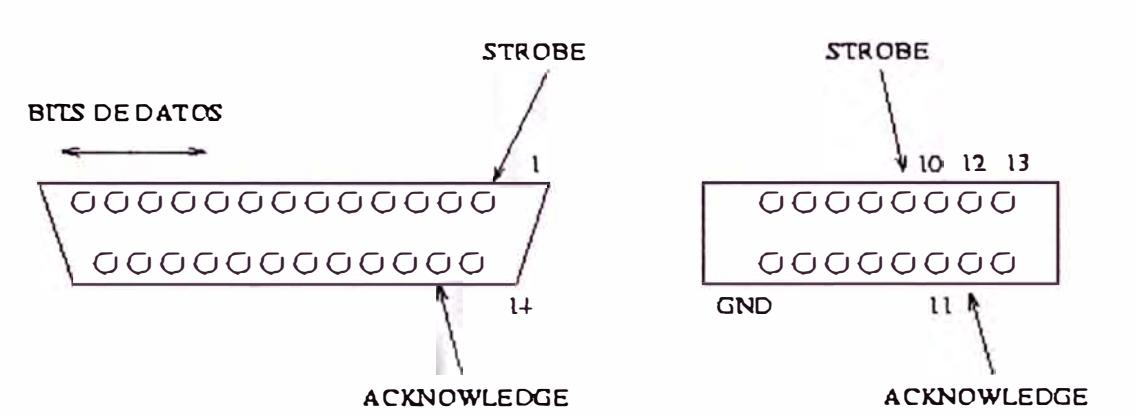


Figura 6.1: Distribución de pines para la transmisión DSP-PC

Una característica muy importante de los pines de control es que algunos de ellos trabajan con lógica negativa, como el caso de los pines 10 y 15. Según la figura 5.2. los datos serán recibidos por los pines 10, 11, 12 y 13, de modo que se debe tener en cuenta el pin que trabaja en lógica negativa. Estos pines son controlados por software mediante el registro de estado asociado al puerto paralelo. La ubicación de cada pin en este registro se muestra a continuación.

$\overline{Pin_{11}}$	Pin_{10}	Pin_{12}	Pin_{13}	$\overline{Pin_{15}}$	NN	NN	NN

Considerando que se desea obtener los 4 bits más significativos notamos que debemos considerar todos los bits de entrada como tales menos el mas significativo que se encuentra negado. Con esto la información se encuentra enmascarada a la llegada hacia la PC. Realizando un sencillo análisis notamos que la información real

[11] Revisar estos conceptos en el Capítulo: Implementación del Sistema

es la misma que se obtiene por el puerto, solo que es necesario negar al bit mas significativo, por lo que realizando la operación XOR con el valor 10001000= 0x88 conseguimos lo anterior. Debido a que no nos interesa el word menos significativo la máscara a utilizar en el puerto es 0x80. Con este concepto es posible leer la información del puerto y almacenarla correctamente.

Las velocidades de transmisión en ambos casos se pueden expresar en base a mediciones de los pulsos de STROBE que lanza la PC hacia el DSP, de modo que el tiempo transcurrido entre dos STROBE consecutivos dará el tiempo de byte. El caso de recepción se debe considerar que la transmisión requiere de dos pulsos debido a que se transmite de 4 en 4 bits. Los valores experimentales obtenidos para cada caso son:

$$V_{max-TX} = \frac{1}{28} us = 35,7KHZ, \qquad V_{max-RX} = \frac{1}{2*8} us = 62,5KHZ,$$

Cabe resaltar que las diferencias en los valores dependen de los tiempo que se requieren en cada componente de la transmisión en mantener sus señales de control en los estados transitorios. El manejo de las señales de control del lado del DSP es realizado en base a interrupciones de hardware. De esta manera los flancos negativos de la señal de STROBE que la PC envía al DSP son detectados por el microprocesador, de modo que la ejecución del programa principal es interrumpido saltando a una rutina especial de procesamiento de interrupción; de esta manera el pulso de la PC hacia el DSP es de una duración mínima, siendo su duración de 1.5 us, mientras que el pulso de ACKNOWLEDGE enviado por el DSP si debe tener una duración adecuada para que la PC pueda detectarlo. Por otro lado se pudo apreciar experimentalmente que es mucho mayor el tiempo requerido para una escritura en el

bus durante la transmisión que para realizar lecturas de datos, siendo esta la razón mas importante de que la recepción se realice mas rápidamente a pesar que la recepción requiere un ordenamiento de los words recibidos (en el lado de la PC), y la necesidad de leer en dos oportunidades

6.2 Análisis de la máxima cantidad de datos a procesar

El sistema propuesto permite encriptar cualquier tipo de información y de cualquier procedencia. En este trabajo se presenta dos aplicaciones: una aplicación de transferencia de datos utilizando técnicas de encriptación, y una aplicación de transferencia de voz, en tiempo real y con métodos similares de encriptación, además de utilizar compresión a fin de mejorar la velocidad de transmisión. A continuación analizaremos la máxima cantidad que es posible procesar sin tener problemas de pérdida de información y sin retardos considerables. Por ello planteamos el diagrama de tiempos mostrado en la Fig. 6.3.

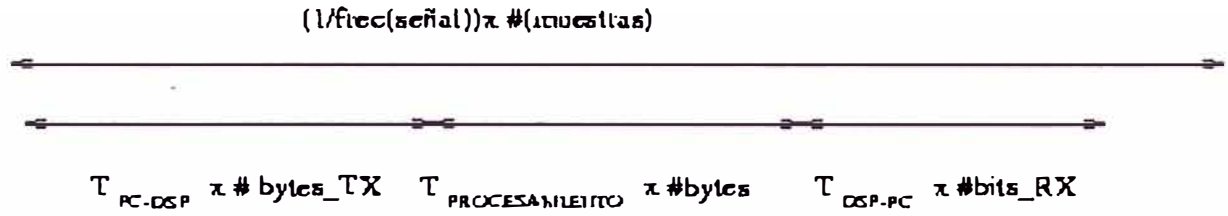


Figura 6.3: Diagrama de tiempos de la aplicación presentada

Se puede apreciar que el punto más importante a tener en consideración es que la información a procesar es adquirida en forma de paquetes de datos. Estos son recibidos, del buffer de la tarjeta de sonido en la transmisión o del socket en el caso del receptor, y deben ser procesados y transmitidos antes que el siguiente paquete arrive. De esta manera se evitará la acumulación de paquetes que implicaría grandes retardos y posible pérdida de información al excederse la capacidad de los buffers

previamente definidos. Para analizar esto, de acuerdo a la Fig. 6.3 planteamos la sgte. ecuación:

$$\left(\frac{1}{frecuencia\_señal}\right)\#muestras \geq (t_{PC-DSP})\#bytes_{TX} + t_{PROC}\#bytes + 2(t_{DSP-PC})\#bytes_{RX}$$

En nuestro caso tenemos #muestras=512. Esto es debido a que la fragmentación de la Tarjeta de Sonido hace que la talla inicial del DMA, que es 4K sea fragmentada en bloques de 512 bytes y son estos los que son leídos por la aplicación cliente. La utilización de 16 bits en la codificación implica que son 512*2 = 1K de información que se maneja y finalmente la compresión ADPCM proporciona 256 bytes que son los que serán encriptados y transmitidos. De esta manera $\#bytes_{TX}$=255, $\#bytes_{RX}$=255. El valor 255 de transmisión es debido a que el DSP56002 posee registros de almacenamiento de 24 bits de capacidad (3 bytes), de modo que el algoritmo de encriptación se basa en múltiplos enteros de 3 al considerar un ordenamiento de 3 bytes en cada registro. Mediciones sobre el DSP revelan que el tiempo requerido para encriptar y/o desencriptar el total de 255/3= 85 bytes es:

$$T_{procesamiento} = 2500us = 2{,}5\ ms.$$

De esta manera reemplazando en la ecuación anteriormente planteada se tiene:

$$\frac{1}{frec\_maxima}x512 \geq (28us)(255) + 2500us + (16us)(255)$$

De esta forma obtenemos:

$$frecuencia_{maxima} \leq \frac{(512)(10^6)}{13720} = 37.3KHZ$$

Este análisis considera el tiempo de transmisión/recepción con el DSP, pero no considera el tiempo de lectura de los 1024 bytes del dispositivo de audio, el tiempo de codificación ADPCM, y el envío de los datos procesados al socket para la transmsión. De esta manera un cálculo que se ajuste mas a la realidad sería considerando estos tiempos importantes. Mediciones experimentales demuestran que:

$$t_{lectura\text{-}audio} + t_{ADPCM} + t_{TX\text{-}socket} = 52ms$$

Con estos datos podemos replantear la ecuación anteriormente definida, obteniendo finalmente:

$$frecuencia\_max \leq \frac{512x10^6}{52000 + 28x255 + 2500 + 16x255} = 7{,}7KHZ$$

De esta manera la frecuencia teórica máxima de la señal de entrada es 7,7 KHZ. Esto demuestra que nuestra aplicación de voz a 4 KHZ de muestreo no presenta problemas de retardo y que para un muestreo de 8 KHZ aparecen algunos problemas de retardos y posibles pérdidas de datos. Tal como se expresó en el capítulo anterior, la utilización de una interfase de comunicación más óptima implicaría la posibilidad de manejar señales muestreadas a tasas mayores.

6.3 Parámetros que influyen en la comunicación

Un análisis importante y que no se puede dejar de analizar es el caso de la congestión de Red. Cuando un Servidor inicializa un socket servidor lo que esta haciendo es habilitar una vía de comunicación para que cualquier cliente pueda conectarse e iniciar una sesión. Luego del establecimiento de la comunicación ellos pueden intercambiar información, la cual es empaquetada, luego se forma la trama, la que es posteriormente enviada. El servidor recibe la información y espera datos en

base al tamaño del buffer definido para este fin. La presente aplicación considera un tamaño de buffer igual para el Cliente como para el Servidor, de manera que la información que el cliente transmite completa todo el buffer receptor.

Sin embargo pueden existir problemas de congestión de Red, lo que implicaría que los paquetes sufran latencias en la red y que no lleguen de manera consecutiva al servidor. De esta manera un bloque de datos que llega al servidor puede tener una talla menor a la esperada. Para contrarestar este efecto la aplicación Servidora plantea una verificación de la talla de los datos que llegan al socket servidor. Por este mecanismo se censa el tamaño del buffer de llegada y si no es del tamaño preestablecido (lo que implica problemas sobre una Red cargada) el servidor permanece en un bucle esperando las tramas necesarias hasta que se complete el paquete y recién luego de esto se procede a iniciar el proceso de desencriptación para ese paquete.

Este sencillo análisis y corrección de este posible error facilitará la utilización de esta aplicación sobre escenarios en los que la latencia es el principal problema. La utilización de sockets TCP garantiza el correcto arribo de los paquetes de datos, por lo que la aplicación luego de varios intentos puede conseguir su información, para casos como el analizado. El esquema de la aplicación lectora del socket es mostrada en la Fig 6.4.

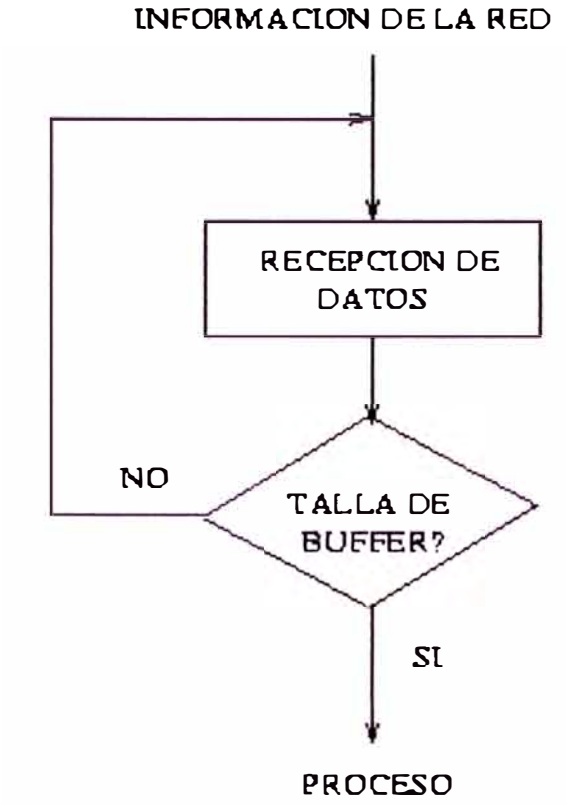


Figura 6.4: Socket Servidor para casos de congestión de red

6.4 Transmisión de Voz

El proceso más importante y que requiere el mayor cuidado en el diseño es la aplicación de transmisión de voz, debido a que no es posible salirse del márgen de retardos tolerables para la aplicación, ya que la información de llegada no sería distinguible. El método de compresión de voz ayuda sobre manera a reducir la cantidad de datos a procesar por el módulo encriptador, haciendo más óptima la aplicación. Finalmente la aplicación de voz procesa las muestras adquiridas (compresión y encriptación) y las transmite sobre un escenario común (Red LAN), teniéndose una comunicación segura y sin pérdida de información, como lo demuestran las satisfactorias pruebas experimentales.

CONCLUSIONES

A la culminación del presente trabajo, y luego de evaluar los casos y finalmente analizar los resultados, es posible dar las sgtes. conclusiones:

- Se ha analizado casos de comunicaciones en los que los sistemas tradicionales no garantizan seguridad en la transmisión y enfatizado la necesidad de implementar una plataforma de seguridad para casos específicos de comunicaciones.
- Se ha demostrado la capacidad de IDEA como método de encriptación y puesto en manifiesto la factibilidad y conveniencia de implementar una solución de IDEA sobre hardware, debido a las características del algoritmo.
- Se analizó las dos metodologías de encriptación y se concluyó que PGP es la más óptima en este caso, debido a que la aplicación está orientada a comunicaciones punto a punto, donde la transferencia de información es un sentido y no orientada a multipunto.
- La concepción inicial de IDEA no es aplicable directamente para una aplicación en hardware debido a la necesidad de considerar la cantidad de bits que el dispositivo cuenta para el procesamiento y se ha demostrado que es sencillo acondicionar IDEA a cualquier estructura de manejo de datos, sin cambiar la filosofía de encriptación.
- Ha quedado en manifiesto que Linux es una plataforma de aplicación que proporciona un control óptimo de los recursos del sistema, siendo para esta

aplicación en particular el control de sockets y del dispositivo de audio. Esto es debido a que Linux proporciona primitivas que permiten el acceso a bajo nivel del dispositivo, lo que garantiza una rápida respuesta y procesamiento sin cargar al sistema, que es el principal problema en las aplicaciones tipo Win32.

- La utilización de sockets TCP proporciona un mayor soporte en control de sesión y manejo de errores, debido a que puede trabajarse en condiciones de redes cargadas y con retardos considerables.
- Las técnicas de codificación de voz usuales proporcionan redundancia, debido a que se almacena mayor información a la necesaria. Se ha demostrado en la presente aplicación que la utilización de ADPCM proporciona una mejor codificación y adicionalmente compresión de los datos adquiridos, lo que es beneficioso para la transmisión.
- La utilización del DSP56002 ha posibilitado implementar IDEA con resultados satisfactorios, debido fundamentalmente a los 24 bits de procesamiento que proporciona, superior a los estándares en DSP's que existen en el mercado (usualmente los dispositivos microprocesadores y/o microcontroladores poseen 8, 10, 12 ó 16 bits). Por ello la solución adoptada muestra un nivel aceptable de encriptación y es garantía de una comunicación segura.
- Es importante el manejo de los dos métodos mas comunes de comunicación: Serie y Paralelo entre el DSP560002 y la PC, ya que permite que la solución sea ampliable a cualquier otro tipo de arquitectura, bastando con que soporte las interfases antes mencionadas, lo que garantiza la universalidad de la solución.
- Se ha podido apreciar las notables ventajas que ofrece la comunicación orientada a paquetes, debido a que permite la fragmentación de la información facilitando

la labor de encriptación. Además es mas óptimo en la comunicación al considerar la posibilidad de tener latencias en la Red o problemas de congestión.

Trabajos Futuros

La aplicación mostrada implementa una solución de transferencia de información sobre una Red Local. Es evidente que este sistema puede extenderse a otras aplicaciones y es susceptible a mejoras. Algunos de los puntos importantes que pueden ser base para trabajos futuros son los que a continuación se presentan.

- El sistema está concebido para una aplicación Punto a Punto. Un buen aporte al trabajo sería la implementación de una solución Punto-Multipunto, la cual no difiere mucho de la presente solución y que permitiría la implementación de aplicaciones de Audioconferencia.
- La solución adoptada necesita establecer una comunicación continua con el DSP56002. Esta etapa del sistema puede ser mejorada, implementando un prototipo de solución en hardware que sea integrado directamente al Bus de la PC y de esta manera la comunicación sería extremadamente rápida, posibilitando la implementación de una plataforma de transferencia de audio de alta calidad e incluso video sobre IP. Una posibilidad mas ambiciosa aún sería integra la solución directamente al disposible de Red de la PC (NIC), lo que no incrementaría en demasía los costos y constituiría una interesante alternativa.
- Analizar la posibilidad de implementación de IDEA en otros dispositivos DSP tales como procesadores de coma flotante, o dispositivos de menor cantidad de bits tales como 16, verificando si la posible solución proporciona los niveles de seguridad adecuados, y de ser así evaluar y posiblemente implementar esta nueva

posibilidad ya que se cuenta con una gran cantidad de dispositivos que cumplen con estas características.

- Analizar la aplicabilidad de la solución sobre Redes que sobrepasen en concepto de Area Local, tales como WAN. El proceso es el mismo sólo que es necesario analizar las características del enclace de comunicación y de esta manera tener la posibilidad de una solución de InterNetworking que implememte un canal seguro de datos entre dos Redes IP.
- Evaluar la solución sobre otras plataformas, tales como ATM, en las que solo es necesario redefinir las tramas y la naturaleza de los sockets y aplicarlo para aplicaciones de AudioConferencia sobre ATM por ejemplo[12].

[12] Esta idea es fundamentada por la Tesis de Maestría "Implementación de una aplicación de Audioconferencia sobre una Plataforma ATM en Entorno Linux" (2000)

ANEXO A
ENCRIPTACION EN SISTEMAS DE COMUNICACION

ENCRIPTACIÓN EN SISTEMAS DE COMUNICACION

En las comunicaciones cotidianas se maneja información de manera natural, es decir, que la información es transmitida por el medio de comunicación sin considerar metodologías que permitan proteger los datos. Por ejemplo, durante una conversación entre dos personas el canal de comunicación está constituido por el aire, siendo la parte transmisora la persona que habla y la parte receptora la constituye la persona que oye la conversación. La información en este caso es la voz, la que libremente viaja a través del aire. Este mismo concepto de comunicación se presenta en la transmisión de datos, donde un componente transmisor se encarga de transferir información a un componente receptor que esta esperando la información. Este proceso se denomina transmisión de información, y podemos esquematizarlo en la Fig. siguiente.

Figura A.1: Sistema de Transmisión/Recepción

Pero puede darse el caso que el tema de conversación sea muy privado, de modo que la conversación ya no puede realizarse de manera pública como en el caso anterior, sino que debe buscarse mecanismos que permitan que nadie pueda enterarse de los asuntos tratados. De este modo un primer camino podría ser hablar en un tono

un tanto mas bajo consiguiendo que sea difícil para otros receptores captar la información, o también tratar de esconderse de los demás, para conseguir privacidad en base a estar fuera de alcance de los posibles receptores. Pero un mejor y óptimo caso sería poder hablar tanto el transmisor como el receptor en un idioma totalmente desconocido, de modo tal que a pesar que los demás perciben la comunicación y tienen acceso a la misma no son capaces de digerirla ni procesarla. Este último caso es el que nos interesa, debido a que en todo sistema de transmisión/recepción el canal de comunicación es siempre el mismo para todas las comunicaciones, de modo tal que siempre todos los demás poseerán manera de interceptar los datos. Una solución será mantener comunicaciones en sistemas totalmente aislados, lo que corresponde una desventaja debido a que al ganar la privacidad se pierde la comunicación con otros componentes del sistema. Lo que debe buscarse en caso de querer proteger la información es justamente poder transmitir la misma con alguna técnica de codificación, la que permita que el transmisor como el receptor puedan comunicarse sin problemas de fuga de información y con un sistema confiable, es decir sin pérdida de información y con procesamiento en tiempo real. Este es el objetivo del presente trabajo de Tesis en el que se presenta, en un primer momento, un método para lograr la óptima codificación y recuperación de los datos y posteriormente un sistema que en base al método descrito anteriormente es capaz de transmitir datos, con un nivel de seguridad en la transmisión y con procesamiento en tiempo real.

A.1 Sistemas de Comunicación

El concepto de comunicación nace cuando se presenta la necesidad de que algún tipo de información sea enviada desde un punto emisor a un punto receptor. La comunicación puede darse de múltiples maneras, desde una tan simple y elemental

como la voz o señales, hasta altos niveles de sofisticación, donde se cuida al máximo que la información llegue de manera adecuada al destino.

A.1.1 Terminología.- Para poder hablar adecuadamente sobre comunicaciones es necesario definir previamente algunos conceptos elementales:

- Transmisor.- Es el punto inicial de la comunicación. Está constituido por el emisor de información. De ahora en adelante lo denotaremos TX .
- Canal de Comunicaciones.- Viene a ser el medio físico que une al transmisor y el receptor. Este puede ser de diversos tipos tales como aire (para el caso de voz, Radiofrecuencia, Wireless), cable (Telefonía, Redes de Datos), fibras ópticas (Transmisión de datos), etc.
- Receptor.- Es la parte terminal de la comunicación. Es el que se encarga de recibir la información enviada por el transmisor a través del canal de comunicación. Lo denotaremos como RX

A.2 Comunicaciones de Datos

Definitivamente las comunicaciones han evolucionado de una manera acelerada durante los últimos años. La tendencia es considerar que todo tipo de información a transmitir, ya sea voz, video o simplemente información sea considerado datos. Anteriormente las comunicaciones estaban bien diferenciadas unas de otras debido a que la naturaleza de la información eran totalmente diferentes, comunicaciones telefónicas en base a pulsos, enlaces de radiofrecuencia, enlaces analógicos de microondas, etc. son un ejemplo de la diversidad de medios de información que se manejaba.

La naturaleza de las señales a transmitir hacia que sea complicado conseguir que los datos lleguen de manera adecuada a la parte receptora, debido que se trataba

de señales analógicas que variaban continuamente en el tiempo. De este modo se tenía que convivir con los problemas de potencia, debido a que debido a las distancias que recorrían las señales de datos estas se desvanecían gradualmente, siendo necesario amplificarlas cada ciertos tramos de recorrido. Los amplificadores utilizados, si bien es cierto amplificaban las señales, también lo hacían con los componentes de ruido agregados a la comunicación.

El avance de la tecnología hizo que la concepción del concepto de señal cambiara. En lugar de considerar a la señal como una forma de onda, es decir, como una señal continua en todo el tiempo, se pensó que considerando la señal solo en ciertos instantes se podría representar la señal sin pérdida de información. De este modo aparece el concepto de muestreo. Este concepto puede ser representando en la siguiente Figura.

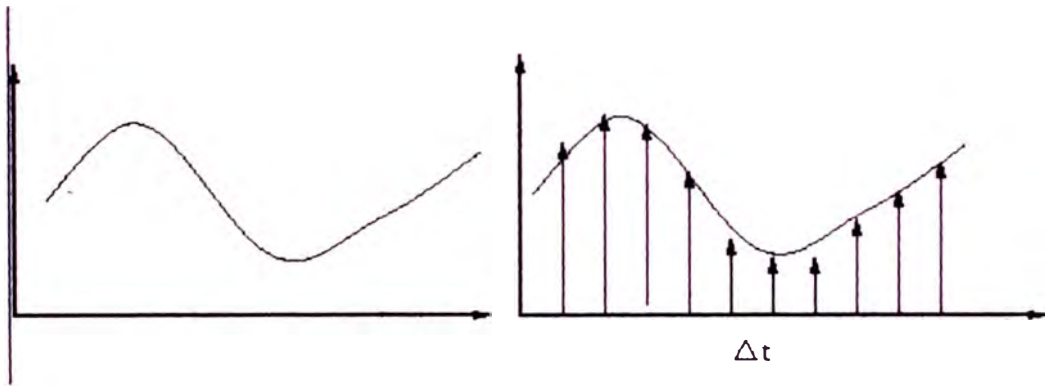


Figura A.2: Muestreo de Señales

A.2.1 Teorema del Muestreo.- Al concebirse el concepto de muestreo surge una necesidad: saber hasta que niveles podría discretizarse la señal, pues era obvio que a mayor Δ_t era menor la calidad de la señal. El Teorema del Muestreo especifica justamente ese límite a considerar.

$$\Delta_{t-minimo} = \frac{1}{2B} \quad [4][5]$$

Donde B es el ancho de banda de la señal transmitida.

De este modo queda totalmente definido el límite inferior de la frecuencia de muestreo, y en base a esto se puede diseñar los sistemas de muestreo adecuadas a cada señal a transmitir.

A.2.2 Cuantificación y Codificación.- Luego de haber conseguido el muestreo de la señal es necesario considerar a cada señal como un valor numérico de tipo discreto. La cantidad de valores discretos posibles que puede tomar la señal, viene a constituir la resolución de la señal discreta. Este proceso es conocido como cuantificación.

El objeto de la discretización es poder considerar a una señal analógica y continua como una señal de tipo discreto y poder representar la misma en base a una codificación que permita la reconstrucción sencilla de la señal, ya que se considera niveles de referencia correspondientes a los máximos y mínimos de la señal. Finalmente, para todo tiempo t se tendrá un valor discreto de señal, el cual será representado en formato binario. El esquema general de la digitalización se muestra en el diagrama funcional de bloque de la Fig. mostrada.

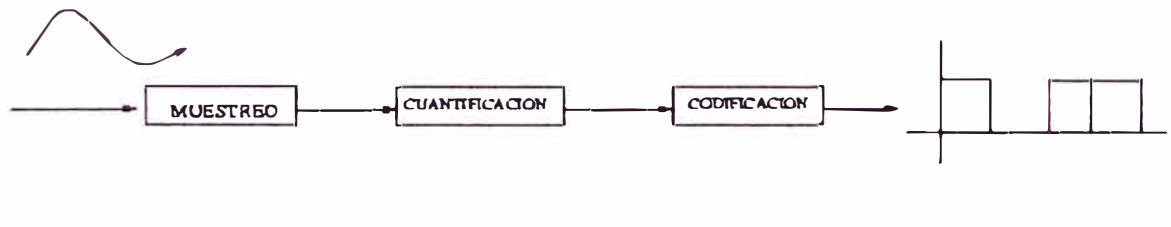


Figura A.3: Esquema del proceso de digitalización de señales

Otro punto de análisis es que al considerar cada señal como una forma de onda analógica esto obligaba a que se tenga una diferenciación de las mismas. Por ejemplo, las redes telefónicas poseen un ancho de banda requerido para la

transmisión, las señales de TV, los enlaces analógicos de microondas, las redes de datos, en fin todas son diferentes entre si y son transmitidas de forma independiente.

La digitalización de señales abre un nuevo panorama en el campo de las comunicaciones, debido a que todas las señales se pueden codificar de manera binaria. Esto conlleva a que se considere modulaciones digitales para la transmisión de señales, mejorándose de este modo el control de error y con mejor utilización al ancho de banda.

A.3 Encriptación

Luego de analizado el concepto de digitalización de señales ya estamos en condición de pensar en realizar un procesamiento digital de las mismas. Procesar digitalmente una señal implica trabajar en base a los valores codificados de las muestras, siendo posible implementar filtros digitales, realizar análisis espectral, convoluciones, etc.

Un procesamiento posible de realizar en este escenario es la encriptación de datos. Una vez que las muestras de voz (o de señal en general) se encuentran digitalizadas entonces son consideradas como simples cadenas de bits. Cualquier método estándar de encriptación es susceptible de ser aplicado, y de acuerdo a las condiciones específicas del problema encontrar el algoritmo adecuado a cada caso. De esta manera la aplicación de encriptación sería adoptada inmediatamente después de la adquisición de los datos, para que estos sean posteriormente transmitidos, consiguiéndose una comunicación que brinda seguridad y confidencialidad en los datos que se transmiten.

BIBLIOGRAFIA

[1] Network and Internetworking Security. Principles and Applications, *John Stalling Ph.D.*, Prentice Hall 1995

[2] A Proposal for a New Block Encryption Standard: *Lai X. and Massey J.* Proceedings, EUROCRYPT'90, 1990.

[3] Error Control Coding. Fundamental and Applications: *Shu Lin, Daniel J. Costello Jr.* Prentice Hall, 1983.

[4] Digital Communication: *John G. Proakis.* McGrawHill

[5] Digital Communication: *I. A. Glober - P. M. Grant.* Prentice Hall

[6] Manual de Usuario del DSP56002: *Motorola Inc*

[7] Realización de un Sistema de Transferencia Segura de Información utilizando un DSP56002: *Marlon Jiménez Bazán.* INTERCON 1999- Lima- Perú.

[8] Sistema Cliente/Servidor para Transferencia Segura de Información sobre una Red Local: *Marlon Jiménez Bazán.* CONIELECOMP 2000. Puebla-México

[9] Sistema Cliente/Servidor para Transmisión de Voz utilizando métodos de Encriptación y Compresión sobre una Red Local utilizando un DSP56002: *Marlon Jiménez Bazán.* INTERCON 2000. Lima – Perú.