Capítulo 3: Protección de los Datos:Control de Errores. Seguridad en Redes.

Objetivos: Establecer la importancia de proteger los datos,tanto de errores,mediante el control de errores, como de acciones no autorizadas,mediante la seguridad en redes.Describir los métodos de detección de errores: Redundancia. Codificación de cuenta exacta. Chequeo de paridad vertical (VRC). Chequeo de paridad horizontal (LRC). Chequeo de paridad bidimensional (VRC/LRC). Checksum. Chequeo de redundancia cíclica (CRC). Describir los métodos de corrección de errores: Requerimiento automático de repetición: (ARQ) (Automatic Request for Repeat): Pare y Espere y Continuos. Corrección de errores hacia adelante: FEC (Forward Error Correction): de Hamming, de bloques, convolucionales.Describir los protocolos de transferencia de archivos: Xmodem, Ymodem, Kermit, Zmodem y la detección,corrección de errores y compresión en modems analógicos: MNP, LAP-M, V42, V42bis, Vfast, etc.Enumerar las técnicas de compresión. Describir los mecanismos de Seguridad en Redes.Criptografía. Autenticación. Firewalls.

3.1.- Protección de los Datos.

Los datos,cuya utilidad radica en su **integridad** y en su **confidencialidad**,están sujetos a dos tipos de amenazas:

Errores

En el Capítulo precedente (Sección 2.3) hemos enumerado y descripto las causas por las que la señal eléctrica se deteriora al viajar por el canal de comunicación,ellas son: distorsión, atenuación, limitación del ancho de banda, ruido, interferencia y diafonía. Esta degradación de la señal puede hacer que recibamos en el receptor un caracter distinto al que fue emitido por el extremo transmisor, diremos entonces que se ha producido un **error.**

Si bien es imposible evitar que ocurran errores, un buen diseño los minimizará. Hecho esto la tarea es en primer lugar determinar la presencia de los errores, aquí es donde aparecen las técnicas de **detección de errores,** y luego tratar de corregirlos, lo que da lugar a la **corrección de errores**,la denominación genérica de estas técnicas es **Control de Errores.**

Acciones no autorizadas

Ya se ha hablado extensamente sobre el valor de **la información**,asegurar que esta información:

- **No sea vista o copiada por personas no autorizadas ó no calificadas para ello.**
- **No sea alterada en ningun sentido(modificada,destruída,alterados los equipos donde reposa,etc)por personas ó máquinas no autorizadas.**
- **No sea creada y/o difundida engañosamente simulando fuentes reales ó inexistentes.**

Esto de **vital** importancia dado el auge de Internet y de las conexiones en red y entre redes.
De ello se ocupa la **Seguridad en Redes.**

Comencemos por el **Control de Errores.**

3.2.-Control de Errores

3.2.1- Detección de errores.

La **detección de errores** consiste en monitorear la información recibida y a través de técnicas implementadas en el Codificador de Canal ya descrito, determinar si un carácter, caso asincrónico, ó un grupo de datos, caso sincrónico, presentan algún ó algunos errores.

Las técnicas más comunes son:

- **Redundancia.**
- **Codificación de cuenta exacta.**
- **Chequeo de paridad vertical (VRC).**
- **Chequeo de paridad horizontal (LRC).**
- **Chequeo de paridad bidimensional (VRC/LRC).**
- **Checksum**
- **Chequeo de redundancia cíclica (CRC).**

3.2.1.1.- Redundancia.

La **redundancia** significa transmitir cada carácter dos o tres veces, o si se emplea a nivel de mensaje repetir el mensaje dos o tres veces, en caso que las versiones difieran habrá error ó errores. Obviamente la **eficiencia** con este método se reduce a ½ ó 1/3 según corresponda.

3.2.1.2- Codificación de cuenta exacta.

En esta técnica de **codificación de cuenta exacta,** lo que se hace es configurar el código de manera que cada carácter esté representado por una secuencia de unos y ceros que contiene un número fijo de unos, por ejemplo tres de ellos. Tal es el caso del Código de cuenta exacta ARQ que se muestra en la **Tabla 3.1**, en caso de recibirse un carácter cuyo número de unos no sea tres, se tratará de un error.

Es claro que este método, al igual que los demás, tiene limitaciones: cuando se recibe un 0 en vez de un 1 y un 1 en vez de un 0 dentro del mismo carácter los errores no serán detectados. Ésta no es la única posibilidad de errores indetectados el lector podrá imaginar varias más.

	Código Binario							Carácter	
Bit:	1	2	3	4	5	6	7	Letra	Cifra
	0	0	0	1	1	1	0	Cambio a letras	
	0	1	0	0	1	1	0	Cambio a cifras	
	0	0	1	1	0	1	0	A	-
	0	0	1	1	0	0	1	B	?
	1	0	0	1	1	0	0	C	:
	0	0	1	1	1	0	0	D	(WRU)
	0	1	1	1	0	0	0	E	3
	0	0	1	0	0	1	1	F	%
	1	1	0	0	0	0	1	G	@
	1	0	1	0	0	1	0	H	£
	1	1	1	0	0	0	0	I	8
	0	1	0	0	0	1	1	J	(campana)
	0	0	0	1	0	1	1	K	(
	1	1	0	0	0	1	0	L	)
	1	0	1	0	0	0	1	M	.
	1	0	1	0	1	0	0	N	,
	1	0	0	0	1	1	0	O	9
	1	0	0	1	0	1	0	P	0
	0	0	0	1	1	0	1	Q	1
	1	1	0	0	1	0	0	R	4
	0	1	0	1	0	1	0	S	´
	1	0	0	0	1	0	1	T	5
	0	1	1	0	0	1	0	U	7
	1	0	0	1	0	0	1	V	=
	0	1	0	0	1	0	1	W	2
	0	0	1	0	1	1	0	X	/
	0	0	1	0	1	0	1	Z	+
	0	0	0	0	1	1	1		(blanco)
	1	1	0	1	0	0	0		(espacio)
	1	0	1	1	0	0	0		Alimentar línea
	1	0	0	0	0	1	1		Regreso de línea

Tabla 3.1. Código de cuenta exacta ARQ

3.2.1.3.- Chequeo de paridad vertical ó paridad de carácter (VRC).

Este método, como todos los que siguen, hace uso del agregado de **bits de control.**

Se trata de la técnica más simple usada en los sistemas de comunicación digitales (Redes Digitales, Comunicaciones de Datos) y es aplicable a nivel de byte ya que su uso está directamente relacionado con el código ASCII.

Como se recordará, el código ASCII utiliza 7 bits para representar los datos, lo que da lugar a 128 combinaciones distintas. Si definimos un carácter con 8 bits (un byte) quedará un bit libre para **control,** ese bit se denomina **bit de paridad** y se puede escoger de dos formas:

- **Paridad par**
- **Paridad impar**

según que el número **total** de unos en esos 8 bits, incluyendo el octavo bit (el de paridad), sea par ó impar, tal como se muestra en la **Figura 3.1.** Por sus características la técnica se denomina también **paridad de carácter.**

El uso de un bit adicional para paridad disminuye la eficiencia, y por lo tanto la velocidad en el canal, el cálculo es sencillo pasamos de 7 bits de datos a 7+1, ello conduce de acuerdo a la expresión 2.10 a un **overhead** de: (1 - 7/8)100 % = 12.5% de disminución en la eficiencia.

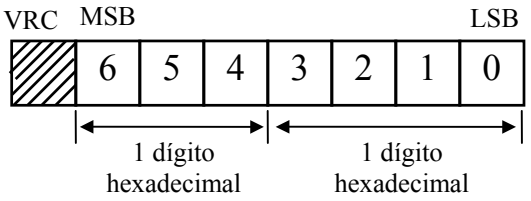


Figura 3.1. Bit de paridad del Código ASCII.

En el extremo de transmisión el Codificador de Canal calcula el bit de paridad y lo adosa a los 7 bits de datos. El Decodificador de Canal recibe los 8 bits de datos calcula la paridad y la compara con el criterio utilizado, tal como describe la **Figura 3.2.**

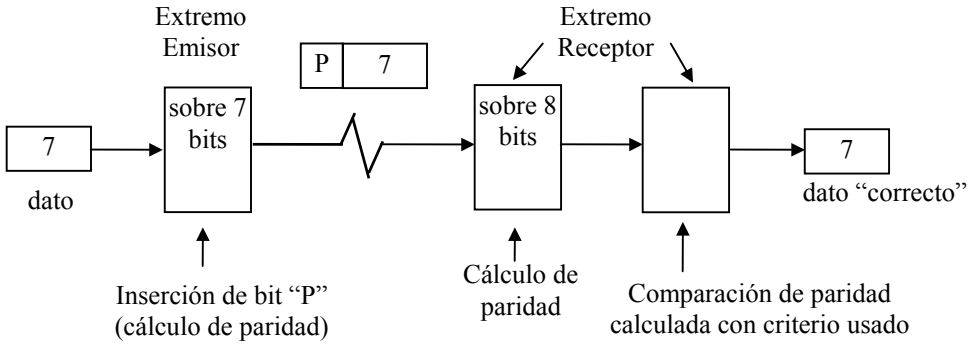
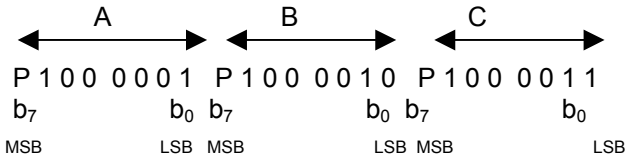


Figura 3.2. Método de chequeo de paridad vertical (VRC).

Este método tampoco asegura inmunidad a errores, basta con que dos bits cambien su valor simultáneamente para que el error no sea detectado pues la paridad será correcta y el dato no. Sin embargo, este sencillo sistema permite que en una línea telefónica discada que transmite entre 10^3 y 10^4 bps con una tasa de error (BER) de 10^{-5} mejore a 10^{-7}.

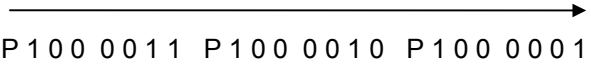
Debe mencionarse que en transmisión serial la interpretación de un secuencia 0 y 1 presenta un problema, **pues el bit menos significativo (LSB) se transmite primero y el más significativo (MSB) de último.**

En una secuencia de tres letras ABC no hay duda de identificar a la A como la primera letra, sin embargo si escribimos sus códigos ASCII (debe hacerse notar que no todos los autores numeran de b_0 a b_7 algunos lo hacen de b_1 a b_8) con el bit de paridad tendremos:



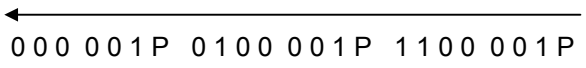
Esta presentación induce a confusión pues no es la de transmisión serial. Tenemos dos alternativas para mejorarla:

Caso 1: flecha a la derecha



el bit del extremo derecho del primer caracter es el primero en ser transmitido (b_0 de A) y el último corresponderá al extremo izquierdo del último carácter (b_7 de C, bit de paridad P de C), la flecha indica el sentido en que fluyen los bits. **Los datos para ser interpretados deben tomarse en grupos de 8 y ser leídos de derecha a izquierda.**

Caso 2: flecha a la izquierda



este caso, común cuando se utiliza un osciloscopio para monitorear líneas de datos ya que el primer bit recibido queda en el extremo izquierdo de la pantalla,la flecha indicará aquí también el sentido en que fluyen los bits,**requiere para interpretar correctamente los caracteres tomar grupos de 8 bits y leerlos de izquierda a derecha**.

3.2.1.4.- Chequeo de paridad horizontal(LRC),longitudinal ó de columna.

Este chequeo de paridad horizontal ó longitudinal (HRC ó LRC) en vez de estar orientado al carácter lo está al **mensaje**, y consiste en que cada posición de bit de un mensaje tiene bit de paridad, así por ejemplo se toman todos los bits b_0 de los caracteres que componen el mensaje y se calcula un bit de paridad par o impar, según el criterio definido, este bit de paridad es el bit b_0 de un carácter adicional que se transmite al final del mensaje, y se procede luego sucesivamente con los demás bits incluyendo el de paridad. El carácter así construido se denomina BCC (Block Check Character), también se le denomina BCS (Block Character Sequence), ver **Figura 3.3.**

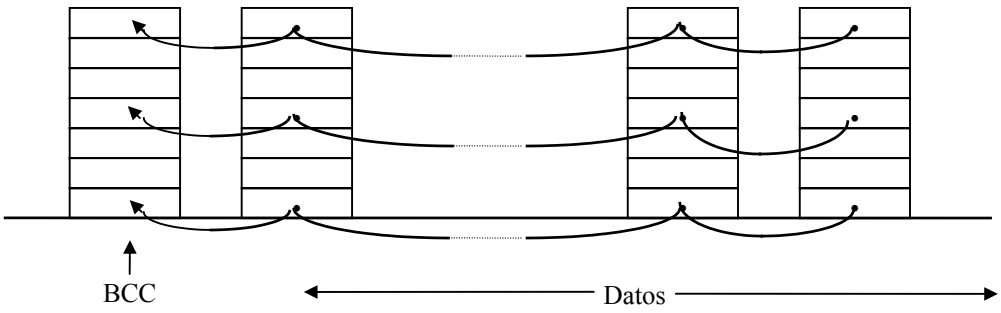


Figura 3.3.Chequeo lomgitudinal LRC

Históricamente entre el 75 y el 98% de los errores presentes son detectados por LRC, los que pasan desapercibidos se deben a limitaciones propias del método, así por ejemplo un error en b_2 en dos diferentes caracteres simultáneamente produce un LRC válido.

3.2.1.5.- Chequeo de paridad bidimensional (VRC/LRC).

La combinación de los dos métodos precedentes proporciona mayor protección y no supone gran consumo de recursos y, aunque tiene la misma sencillez conceptual de los métodos de paridad lineal, es más complicado y por ello menos popular.

El uso simultáneo de VRC y LRC hace que pasen indetectados errores en un número par de bits que ocupan iguales posiciones en un número par de caracteres, circunstancia muy poco probable.

Aunque no es el objeto de esta Sección debe hacerse notar que en caso que se trate de un solo error el uso simultáneo de VRC y LRC permite determinar con precisión **cual es el bit erróneo** y por lo tanto **corregirlo**. Otras combinaciones de errores pueden ser detectadas y algunas además corregidas. Las **Figuras 3.4 y 3.5** ilustran algunas circunstancias del chequeo bidimensional.

Paridad de carácter computada en el receptor	bit de paridad trabsmitido ↓ Bit # 8	7	6	5	4	3	2	1		
0	0	1	0	0	0	0	0	1	A	Carácter #1
0	0	1	0	0	1	1	0	1	M	Carácter #2
1	1	1	0	0	0	1	0	1	E	Carácter #3
1	1	1	0	1	0	0	1	0	R	Carácter #4
1	1	1	0	0	1	0	0	1	I	Carácter #5
1	1	1	0	0	0	0	1	1	C	Carácter #6
1*	0	1	0	**1**	0	0	0	1	A	Carácter #7
0	0	1	0	1	0	0	0	0		Paridad de columna recibida
1	0	1	0	**0***	0	0	0	0		Paridad de columna computada en el extremo receptor

Diferencias de paridad

Figura 3.4 Mensaje con un error en la segunda letra A

Parid. de car. Computada en el receptor	Bit # 8	7	6	5	4	3	2	1		
0	0	1	0	0	0	0	0	1	**A**	Carácter # 1
1*	0	1	0	0	1	1	1	1	**M**	Carácter # 2
1	1	1	0	0	0	1	0	1	**E**	Carácter # 3
1	1	1	0	1	0	0	1	0	**R**	Carácter # 4
1	1	1	0	0	1	0	0	1	**I**	Carácter # 5
1	1	1	0	0	0	0	1	1	**C**	Carácter # 6
0	0	1	1	1	0	0	0	1	**A**	Carácter # 7
0	0	1	0	1	0	0	0	0		Paridad de columna recibida
0	0	1	**1** *	**0** *	0	0	**1** *	0		Paridad de columna computada en el extremo receptor

Figura 3.5 Mensaje con tres errores, uno es detectado e identificado, los otros dos solo detectados.

3.2.1.6.- Checksums

Es otro método simple orientado al **mensaje**, en él los valores (por ejemplo decimales) que corresponden a cada carácter en el código ASCII son sumados y la suma es enviada al final del

mensaje. En el extremo receptor se repite el procedimiento de sumar los valores de los caracteres y se compara el resultado obtenido con el recibido al final del mensaje, ver **Figura 3.6.**

# Carácter	Carácter	Valor decimal
1	A	65
2	M	77
3	E	69
4	R	82
5	I	73
6	C	67
7	A	65
CHEKSUM		498

Figura 3.6. Método de checksums

3.2.1.7.- Código de redundancia cíclica[13].

Los métodos basados en el uso de paridad son sencillos de comprender y de implementar, suministran cierto grado de protección contra los errores pero son limitados y su efectividad es cuestionable en determinadas aplicaciones. Por ello se utilizan solamente cuando resulta muy complicado ó muy costoso implementar otros métodos. Además, el de paridad vertical requiere que cada carácter lleve su protección contra errores, lo que lo hace adecuado en entornos asíncronos, en entornos síncronos el uso de tantos bits de detección de errores consume un porcentaje importante de la capacidad del canal y resulta oneroso. Por ello es necesario, en entornos síncronos, emplear métodos que tengan en cuenta dos factores importantes:

1. **Detección más segura de los errores**. Dado que los datos se envían en bloques un solo error corrompe toda la información contenida en él, que es considerable,además muchas veces los errores se presentan en "ráfagas" **[14],** por ello se requieren esquemas más poderosos
2. **Eficiencia.** No se deben consumir demasiados recursos dejando libre la mayor parte del canal para datos.

Un grupo de métodos que cumplen con dichos requisitos son los llamados **códigos de redundancia cíclica**, que se basan en propiedades matemáticas de los códigos empleados para la transmisión de datos, para dar una idea del método veremos un ejemplo sencillo.

Deseamos transmitir al extremo receptor, mediante un canal de comunicación muy vulnerable a errores, un número. Dadas las circunstancias es muy posible que si enviamos, digamos el número 23, llegue al extremo receptor un número distinto, una solución es elegir un número **clave**, por ejemplo el 5. Ahora dividimos el número a transmitir entre la clave y calculamos el resto: 23/5 = 4 resto 3 y enviamos conjuntamente con el 23 el resto, o sea, transmitimos 233. En el extremo receptor se efectúa el proceso inverso, supongamos que hemos recibido 253 al dividir 25/5 el resto es 0 y 0 es distinto de 3 lo que indica error.

Lo mismo se hace en los **métodos de redundancia cíclica**, que están basados en la propiedades de la operación módulo (se define módulo de dos números *a* mod *b* al resto de dividir a por b).

Para ello se considera la cadena de bits a transmitir como el conjunto de coeficientes de un polinomio, por ejemplo si enviamos 1100100110, el polinomio equivalente P(x) es:

$$1\,x^9 + 1\,x^8 + 0\,x^7 + 0\,x^6 + 1\,x^5 + 0\,x^4 + 0\,x^3 + 1\,x^2 + 1\,x + 0$$

sea

$$1100100110 = P(x) = x^9 + x^8 + x^5 + x^2 + x.$$

Debemos ahora especificar **la clave** para efectuar la división. La selección de esta clave es esencial para la capacidad de respuesta del código frente a los diversos tipos de errores. El CCITT especifica algunas claves, que como se van a emplear para dividir un polinomio serán también polinomios, denominados **polinomio generador**. En el CRC denominado **CRC-16** correspondiente a la norma CCITT V.41, se utiliza el siguiente **polinomio generador**:

$$G(x) = x^{16} + x^{12} + x^5 + x^0$$

donde $x^0 = 1$.

El procedimiento es el siguiente: se toma el polinomio de datos P(x) y se multiplica por x^k, donde k es el exponente más alto de G(x), este polinomio así construido se divide por G(x) y se obtiene un **polinomio resto** R(x),llamado BCS (Block Character Sequence), luego se procede a enviar el polinomio T(x) construido así:

$$T(x) = x^k P(x) + R(x)$$

En el extremo receptor se procederá a extraer lo que suponemos es $x^k P(x)$, lo dividimos por G(x) y calculamos un polinomio resto que si coincide con el R(x) recibido indicará que no hay errores.

La longitud de P(x) es variable,algunos autores hablan de 4 bytes para CRC-16 y otros dicen que se aplica a la trama ó bloque ó paquete.

La operación de división empleada en CRC se denomina de **módulo 2**. Esta división es diferente de la que estamos acostumbrados y funciona así:

- La restas durante la división(o sea la obtención del resto parcial ó final) no son aritméticas sino **módulo 2**, lo que significa una operación XOR entre los dígitos binarios que se están restando(1 y 0 da 1, 0 y 1 da 1, 0 y 0 dá 0, 1 y 1 dá 0).
- Sí el primer bit del resto parcial es 1 y queda uno ó más bits del dividendo se baja el primer bit de la izquierda no usado y se hace el XOR.En caso contrario se bajan bits de la izquierda del dividendo hasta que este resto con los bits bajados esté encabezado por un 1 y tenga la misma longitud del divisor. De no lograrse el resto con todos los bits bajados será el resto final de la división

Un ejemplo sencillo ilustra esta técnica, ver **Ejemplo 3.1.**

EJEMPLO 3.1:

Determine el BSC(Block Character Sequence), para los siguientes polinomios generadores de datos y CRC.

$$\text{datos } P(x) = x^7 + x^5 + x^4 + x^2 + x^1 + x^0 \quad \text{ó} \quad 10110111$$

$$\text{CRC } G(x) = x^5 + x^4 + x^1 + x^0 \quad \text{ó} \quad 110011$$

Solución

Primero P(x) es multiplicado por el número de bits en el código CRC, 5.

$$x^5(x^7 + x^5 + x^4 + x^2 + x^1 + x^0) = x^{12} + x^{10} + x^9 + x^7 + x^6 + x^5$$

$$= 1011011100000$$

Al dividir este polinomio por G(x) obtenemos R(x) ó BCS que resulta:

```
1 1 0 0 1 1 _____________ 1 1 0 1 0 1 1 1
           ) 1 0 1 1 0 1 1 1 0 0 0 0 0
             1 1 0 0 1 1
             -----------
               1 1 1 1 0 1
               1 1 0 0 1 1
               -----------
                 1 1 1 0 1 0
                 1 1 0 0 1 1
                 -----------
                   1 0 0 1 0 0
                   1 1 0 0 1 1
                   -----------
                     1 0 1 1 1 0
                     1 1 0 0 1 1
                     -----------
                       1 1 1 0 1 0
                       1 1 0 0 1 1
                       -----------
                           1 0 0 1 = Resto
```

Resto 01001 significa $R(x) = 0x^4 + 1x^3 + 0x^2 + 0x + 1$

Se transmite entoces $T(x) = x^k P(x) + R(x)$ o sea 1011011101001

Suponiendo que se reciba $T_R(x)$ 1011011101001 (lo mismo que se transmitió),al dividir $T_R(x)$ por G(x) el resto será cero indicando que se recibió correctamente ó bien se separa lo que se supone es x^k P(x) se divide por G(x) y el resto se compara con R(x)(se recomienda al lector efectuar la división) .

Es posible demostrar**[1]** que la condición para que un código polinomial no detecte un error es que el polinomio que lo representa sea múltiplo del polinomio generador. Por ello la selección del polinomio generador es muy importante ya que si se elige adecuadamente es muy difícil que un error pase indetectado. En el caso del polinomio generador de 17 bits descripto como CCITT V.41 puede demostrarse que detecta:

- Todos los bits erróneos que se produzcan en número impar
- Todos los errores que afecten en bloque 16 ó menos bits (single-error bursts, seb)
- 99.9969% de todos los posibles bloques de error(seb) de 17 bits.
- 99.9984% de todos los posibles bloques de error(seb) de longitud superior a 17 bits.

A modo de comentario, la línea telefónica de la Subsección 3.2.3 con CRC mejoraría su BER a 10^{-14}.

Sin embargo detectar los errores no es suficiente, **hay que corregirlos.**

3.2.2.- Corrección de errores.[4]

Dos formas principales de corrección de errores son:

- **Requerimiento automático de repetición**: *(ARQ) (Automatic Request for Repeat).*
- **Corrección de errores hacia adelante**: *FEC (Forward Error Correction).*

3.2.2.1.- ARQ, Requerimiento automático de repetición[12].

Se trata de **sistemas de corrección hacia atrás**, en estos sistemas la estación receptora que ha detectado la recepción de caracteres ó bloques con errores procede a pedir a la estación emisora que repita lo recibido con error. Debe observarse que esto requiere dar al sistema de comunicación algún medio para facilitar el diálogo entre la estación emisora y la estación receptora, de donde se deduce que el extremo receptor abandona el papel pasivo en la comunicación para participar en forma activa en el proceso.

Existen dos estrategias principales en el diseño de los sistemas de corrección hacia atrás:

- **Pare y espere (stop and wait ARQ).**
- **Envío continuo (Continuos ARQ).**

Estas suponen el uso de estructuras de información llamadas **bloques** ó **tramas,** ya se trate de bloques de bits con bit de paridad, ó un conjunto de bits con CRC, en cualquier caso la trama se envía como un todo.

3.2.2.1.1.- Pare y espere.

Cuando se hace uso de este método **pare y espere**, lo que ocurre es precisamente eso, las tramas se van intercambiando una a una. Cuando el receptor recibe una trama procede a **validarla**, si resulta que no contiene errores envía una señal de confirmación hacia el emisor, esta señal se denomina ACK (acrónimo del término ingles *acknowledge*: confirmación). Por el contrario, si hay errores se envía hacia el emisor una señal de recepción errónea denominada NACK (por *negative acknowledge*).

Mientras espera la recepción de ACK ó de NACK el emisor mantiene el mensaje enviado en un buffer, cuando recibe NACK vuelve a enviar el contenido del buffer, si por el contrario recibe un ACK copia en el buffer la trama ó bloque siguiente y procede a enviarla.

En muchos casos, y mientras espera la llegada de NACK ó ACK, el emisor arranca un temporizador. El temporizador se detendrá al llegar cualquiera de las señales de confirmación, si el temporizador llega a su término sin recibirlas pueden ocurrir dos cosas: la primera consiste en abortar el proceso de comunicación dado que no hay respuesta del receptor, la segunda en enviar nuevamente la trama sin confirmar y arrancar nuevamente el temporizador. Si esta situación se repite varias veces el sistema aborta la comunicación.

En el receptor también se arranca un temporizador ya sea al recibir la trama o al enviar una señal ACK, al vencer este tiempo el receptor procede a reenviar una señal de ACK ú otra predeterminada, si se repite esto una cierto número de veces se procede a abortar la comunicación pues no se recibe respuesta del emisor.

Generalmente tanto el receptor como el emisor dispondrán de un contador para determinar el número de veces que se ha intentado retransmitir una trama sin éxito. De alcanzar el contador el valor prefijado se procede a abortar la comunicación. Normalmente esta situación es transitoria y puede deberse a picos de tráfico, de persistir señalan anomalías en los equipos.

3.2.2.1.2.-Envío continuo (Continuos ARQ).

El método de envío y espera tiene inconveniente de reducir el tiempo de utilización efectiva de los canales de comunicación dado que cada mensaje debe ser confirmado individualmente y todo se paraliza hasta que ello ocurre. Para corregir esto los métodos de **envío continuo** utilizan el mecanismo de enviar continuamente la información sin esperar confirmación, cada bloque ó trama

contiene un número (ó varios) de secuencia que la identifica. Existen diversos métodos para enviar al ACK ó el NACK, ellos son:

- **Variante retroceda 2 (GO BACK 2).**
  Mientras se envía un mensaje se está confirmando el anterior por otra vía
- **Variante retroceda n.**
  Previamente se conviene en un número "m" que dará el número de mensajes al cabo de los cuales se va a enviar respuesta ACK ó NACK. En caso de error en el mensaje x, se pide que se retransmita la secuencia a partir de x retrocediendo n = m - x. La variante se denomina **"ventana fija"** y también **rechazo no selectivo** pues se procede a enviar la trama defectuosa y todas las pendientes de conformación. Es un método usado en protocolos orientados al bit como HDLC
- **Rechazo selectivo**: se reenvía solamente la trama defectuosa ,se denomina también **"ventana deslizante"[12].**

Obviamente deberán establecerse buffers que contengan las tramas a confirmar y límites al número máximo de tramas a confirmar.La Referencia **[1]** calcula la efectividad de cada método y muestra que **Rechazo no selectivo** y **Rechazo selectivo** son **equivalentes**, mientras que **Pare y Espere** es un 30% más ineficiente.

3.2.2.2.- Corrección de errores hacia adelante: FEC(Forward Error Correction)[16][17][18].

Se basan en la idea de reconstruir la información deteriorada por los errores, obviamente la reconstrucción tiene lugar en el equipo receptor, para ello deben emplearse en los códigos un gran número de bits lo que disminuye la efectividad del código.Por lo tanto debemos hablar de **codificación [16]**,existen varios esquemas que pueden clasificarse en **códigos de bloque** y **códigos de árbol**:

3.2.2.2.1.-Códigos de bloque.

Un **código de bloques** convierte k bits de entrada en n bits de salida con n>k,este es un código **sin memoria**,veamos algunos ejemplos:

3.2.2.2.1.1.- Códigos de Hamming[13].

En 1950 R. W. Hamming, de los Laboratorios Bell, comenzó a utilizar el concepto de **distancia**, que se define como el número de posiciones en las que dos dígitos binarios de igual longitud defieren. Por ejemplo en 10010 y 01011 la distancia de Hamming es 3, pues difieren en 3 de las 5 posiciones.

La idea entonces fue establecer un set de caracteres en el que todos tienen entre sí la misma distancia de Hamming, luego en el extremo receptor se verifica cada carácter recibido para determinar si es uno de los caracteres válidos, si no lo es se busca el carácter que tenga la menor distancia de Hamming con este y se le asigna ese valor como "correcto". Un código de Hamming de distancia 3 puede detectar errores dobles y corregir los simples (de un solo bit). En el caso de errores dobles la "corrección" es errónea ya que como no se sabe si el error es simple o doble se presupone simple y como tal se corrige.

Los códigos de Hamming a propuesta de este se mejoraron agregando a los bits de información una serie de bits de comprobación, a partir de estos últimos se puede detectar la posición de bits erróneos y corregirlos.

Como ejemplo veremos uno de estos códigos de Hamming. Supongamos que tenemos una palabra de información de 8 bits, D1 a D8, agregamos a ella cuatro bits de comprobación, C1 a C4 con la siguiente estructura:

1	2	3	4	5	6	7	8	9	10	11	12
C1	C2	D1	C3	D2	D3	D4	C4	D5	D6	D7	D8

donde C1 comprueba con paridad par D1, D2, D4, D5 y D7. C2 se usa para D1, D3, D4 y D6, análogamente se usan C3 y C4, lo que se describe en la tabla siguiente:

1	2	3	4	5	6	7	8	9	10	11	12
C1	C2	D1	C3	D2	D3	D4	C4	D5	D6	D7	D8
+		*		*		*		*		*	
	+	*			*	*			*		
			+	*	*	*					*
							+	*	*	*	*

Cuando se envían los datos se calculan C1, C2, C3 y C4 y se van comparando los valores recibidos empezando con C4 y terminando con C1, si los valores recibidos y los calculados coinciden se asigna a la comprobación el valor 0 y si difieren se asigna 1. Si todas las comprobaciones dan resultado positivo tendremos un valor 0000 para la secuencia de comprobación. Pero si ha habido un error alguna de las comprobaciones fallará. Supongamos que se ha producido un error en D3, al realizar las comprobaciones de paridad C1 y C4 darán 0 mientras que C2 y C3 darán 1, tendremos una secuencia de comprobación 0110 que representa en binario 6 que es la posición de D3 el bit erróneo..!!!, solo queda corregirlo.

El número de bits de comprobación necesaria depende de la longitud de la palabra a transmitir**[2] [13]** y aunque su funcionamiento parezca engorroso puede implementarse con “hardware” no muy costoso.

3.2.2.2.1.2.- Otros códigos de bloque.

Además de los códigos de Hamming existen muchos otros tipos de códigos de bloque,una clase popular son los **códigos cíclicos**,en los que otra palabra del código puede ser obtenida tomando una palabra cualquiera,desplazando los bits hacia la derecha y colocando los bits sobrantes a la izquierda,esto permite el uso de "shift registers" lineales con realimentación en la codificación y una sencilla decodificación.Ejemplos de estos códigos son Bose-Chaudhuri-Hocquenhem(BCH),Reed-Solomon,Reed-Müller y Golay**[18]**.

Un tipo de código Reed-Solomon es muy utilizado en los reproductores de CDs y soporta hasta 100 Mbps. Tiene una eficiencia de 7/8 ó 15/16,una demora típica de 2000 ó más duraciones de bits y corrige bloques de hasta 100 bits.

Reed-Solomon son un subconjunto lineal de los códigos de bloque BCH y se especifican como RS(*n,k*) donde *n* es la longitud de las palábras de código y *k* es la longitud de los datos, *n-k* es el número de bits adicionales. Así el estándar G.709 FEC RS(255.239) tiene 255 bits en la palábra de código, con 239 bits de datos y 16 bits adicionales para implementar la detección y corrección de errores.

Tanto los códigos BCH como los subconjuntos RS mejoran el BER sustancialmente tal como lo muestra la **Figura 3.7** que dá las BER en función de la relación señal ruido para enlaces opticos de 40Gbps.

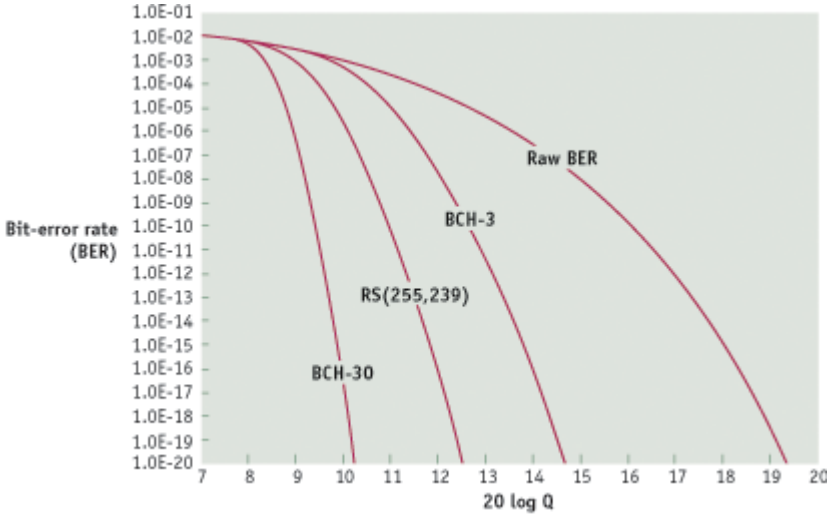


Figura 3.7. BER en función de la relación señal a ruido para enlaces ópticos a 40 Gbps.

3.2.2.2.2.-Códigos de árbol.

Un **código de árbol** es producido por un codificador con memoria,a este grupo pertenecen los **códigos convolucionales**.

3.2.2.2.2.1.Códigos Convolucionales[14][16].

Mediante la codificación convolucional cada bit de una secuencia es “convolucionado”(al aplicar una operación binaria específica)con uno ó varios bits enviados inmediatamente antes, además, se agrega un bit redundante a cada grupo de bits comparados de esta manera.

Cuando un bit es solo comparado con el que le precede, el número de bits redundantes necesarios para asegurar la decodificación es muy alto, aún cuando la complejidad del procesamiento es minimizada.

Por el contrario, cuando el bit es comparado con gran número de bits precedentes (Constraint Length) el número de bits redundantes es minimizado, pero la complejidad del procesamiento en ambos extremos es alta (los más comunes son Viterbi, Feedback y Secuencial).

En un codificador convolucional**[16 pag.28]** k bits(una trama ó paquete) de entrada son convertidos en n bits de salida(trama de salida) con n>k,la conversión aquí es diferente que en los códigos de bloque ya que el codificador tiene **memoria**,y la trama de salida depende de las K tramas de entrada anteriores(K>1),K es la "constraint lenght"

El decodificador tiene como función determinar la secuencia de salida *más probable* dado un flujo de bits recibido(que pueden contener errores)y sabiendo la codificación empleada. Equivale a comparar la secuencia recibida con todas las secuencias posibles de ese codificador y seleccionar la que más se le acerque según una distancia de Hamming,otro esquema de decodificación es el de Viterbi.

Estos códigos normalmente trabajan hasta 50 Mbps, con una eficiencia de ½, 3/4 ó 7/8, tienen una demora típica de 30 a 1000 duraciones de bit y la secuencia de bits corregidas son inferiores a 20, el estándar V.32 del CCITT usa Trellis Coding y tiene una eficiencia de 4/5.

3.3.- Protocolos de transferencia de archivos.

La habilidad para intercambiar archivos es una de las grandes cualidades de las comunicaciones entre computadores y en general de las redes digitales. Deseamos transferir

documentos, bases de datos, hojas de cálculo, programas, gráficos, sonido, imágenes en movimiento, etc. y queremos hacerlo directamente sin tener que enviar el transporte físico. Para ello se usan los protocolos de transferencia de archivos, muchos de ellos aparecen en los programas (ó software) de comunicaciones, en general estos traen alternativas de modo de que se pueda seleccionar uno de los más populares (y efectivos). Han aparecido grandes cantidades de protocolos pero un grupo no muy grande ha alcanzado aceptación general. Veremos los más importantes.

3.3.1.- Xmodem[6].

Este protocolo es un primer paso en la comprensión de otros más recientes. Fue inventado por Ward Christiansen en 1977 y está basado en su MODEM7 usado en los viejos sistemas CP/M. Aunque lento y limitado,frecuentemente es el único que funciona entre máquinas disimiles.

Xmodem divide un archivo en bloques ó paquetes de 128 caracteres y le agrega caracteres de paquetización que denotan el comienzo del bloque, su final, el número de este y el "checksum" de los datos**[6]**, emplea ARQ pare y espere para corrección de errores y se implementa fácilmente por software de dominio público, lo que contribuyó a su rápida aceptación.

A pesar de su simplicidad tiene algunos inconvenientes, los más notables son: usa caracteres de 8 bits de paquetización lo que le hace incompatible con mainframes de 7 bits, sólo envía el archivo, no su nombre, fecha, hora, etc.; no mantiene la dimensión original de los archivos pues redondea al incremento más cercano de 128 bytes y finalmente el esquema de detección de errores no es el más eficiente.

Además al aparecer modems más rápidos lo pequeño del bloque y el método de pare y espere se constituyeron en "cuellos de botella".

El Xmodem original ha tenido mejoras mediante la sustitución de "checksum" por CRC dando lugar a Xmodem CRC y al Xmodem Auto que conmuta a checksum si el computador remoto no soporta CRC.

Existe un Xmodem 1K que utiliza bloques de 1K y por lo demás es idéntico al Xmodem CRC.

3.3.2.- Kermit[6].

Kermit fue creado por Frank da Cruz en los 80, mantiene la división en bloques y el pare y espere pero permite enviar múltiples archivos con nombres, fechas y horas y mantiene las dimensiones de los archivos; permite el uso de *wild cards* (uso de asteriscos para transmitir secuencialmente y automáticamente muchos archivos con nombres ó extensiones comunes), además fue diseñado tanto para microcomputadores (8 bits) como para mainframes (7 bits).

En este protocolo los paquetes son más pequeños, 10 a 96 bytes, e incluyen más caracteres de paquetización que Xmodem. La compatibilidad con mainframes requiere de expandir un carácter en dos, existen paquetes de inicialización y el ACK consiste de varios caracteres, todo ello hace que Kermit tengo un "throughput" que es de un medio a un tercio el de Xmodem.

Versiones posteriores han mejorado Kermit aumentando la dimensión de los paquetes hasta 9K y utilizando "ventana deslizante" en lugar de "pare y espere".

3.3.3.- Ymodem[6].

Introducido por Chuck Forsberg en 1981 ofrece varias mejoras respecto de Xmodem: utiliza CRC, envía nombres de archivos, días y horas; usa bloques de 1K, pero mantiene el pare y espere.

Con esto se logra una mejora del 60% en líneas *no ruidosas*, en las que lo son el efecto es desastroso en Ymodem ya que el reenvío de un bloque dañado puede tomar 8 veces el tiempo que emplea Xmodem.

3.3.4.- Zmodem.

En 1986 Chuck Forsberg introdujo el Zmodem que alcanza una eficiencia del 98% enviando los datos en un flujo continuo insertando códigos detectores de error a intervalos pero deteniéndose solo al final del archivo para ACK. En el receptor los códigos detectores de error son verificados e inmediatamente solicita el almacenamiento de los datos dañados y su reenvío. Además Zmodem introdujo “recuperación de archivos” que es la habilidad del protocolo de reiniciar una transmisión abortada en el punto donde se interrumpió evitando retransmisiones desde cero que pudieran volverse inacabables en entornos ruidosos.

3.3.5.- FTP(File Transfer Protocol)[14].

FTP(File Transfer Protocol)es un **programa** de transferencia de archivos en entornos TCP/IP(Transmission Control Protocol/Internet Protocol) como Internet. El programa tiene su versión **cliente,**que puede activarse en la máquina del usuario(por ejemplo WSFTP),y la **servidor** en la máquina que posee los archivos a transferir,si se accede a está máquina por Telnet ó vía otro servidor,deben introducirse los comandos respectivos.FTP es componente fundamental en los sistemas de TCP/IP y está ubicado en la **capa de aplicación** del modelo OSI.Su forma de trabajo se basa en el programa Telnet y en TCP.

Las utilidades de FTP incorporan las siguientes:

- FTP permite especificar la estructura del archivo(**no estructurados,estructurados y de acceso aleatorio**)
- Soporte para archivos **texto(ASCII y EBCDIC**),**binarios de ocho bits** y **binarios con formato libre.**
- Capacidad de lectura y escritura donde los usuarios visualicen,creen,y eliminen archivos y directorios.
- Protección de contraseñas.
- Compresión de datos.

Existe también**[14]** un **TFTP**(Trivial FTP)para uso en LANs donde la probabilidad de errores es mucho menor que en otro tipo de redes.

3.4.- Detección, corrección de errores y compresión en modems analógicos[5].

El advenimiento de modems de más alta velocidad esta vinculado a señalización multinivel, esto es consecuencia del siguiente análisis.

3.4.1.- Capacidad de una línea telefónica.

Una línea telefónica (circuito de voz) se aproxima a un canal Gaussiano de banda limitada (GBLC) con ruido Gaussiano aditivo, para ese caso es aplicable el Teorema de Shannon que establece que si la velocidad a que introducimos los datos en el canal es menor que su capacidad C, existe un código para el cual la tasa de error tiende a cero si la longitud del mensaje es infinita. Esa capacidad C se calcula con la expresión:

$$C = B \log_2 (1 + S/N)$$

donde B es el ancho de banda limitado del canal en Hz, S/N la relación señal a ruido y C la capacidad del canal en bits por segundo(bps).

En el caso de la línea telefónica con un ancho de banda de 3100 Hz y una S/N de 1023 (aproximadamente 30 dB) resulta la capacidad del canal 31.000 bps.

Los valores de B y S/N son razonables y sabemos que rara vez podemos alcanzar más de 3000 baudios de velocidad de símbolos en una línea telefónica. Ello se debe a la interferencia intersimbólica producida por la respuesta no uniforme del canal al pulso.

Por lo que si empleamos un sistema de dos niveles (binario) estaríamos hablando de 3000 bps, el valor límite de Shannon puede aproximarse más mediante sistemas multinivel, sin embargo, al emplear estos sistemas debe incrementarse la potencia pues de otro modo la relación S/N se deteriora rápidamente tal como se explicó en el Capítulo 2.

3.4.2.- Detección y Corrección de errores.

Al implementarse los primeros modems a 1200 y 2400 bps se observó que el ruido en las líneas discadas dificultaba grandemente la comunicación al introducir errores severos en los textos (*garbage* ó basura) haciendo inútiles los archivos binarios. Surgió entonces la necesidad de implementar **modems con detección y corrección de errores.**

La empresa Microcom desarrollo un protocolo llamado MNP (Microcom Networking Protocol) que se ha convertido en el estándar en modems**[6]**.

MNP se puede implementar en software ó en hardware, sin embargo, opera más eficientemente cuando se coloca en el “firmware” del módem (que es software grabado en chips ROM que se encuentran en la tarjeta del módem). Los modems en ambos extremos deben usar el mismo MNP (veremos que hay varios) y eso se logra porque al iniciarse la comunicación el módem que la solicita “informa” que está usando MNP y el otro extremo automáticamente responde que sí,si tiene esa opción,lo que activa el software del módem solicitante. Cuando se implementa así el MNP **cambia** una serie de datos **asincrónicos** por una **equivalente** de datos **sincrónicos** y forma **paquetes de datos continuos con CRC** para detección de errores.

Ya se dijo que hay varios MNP:

- MNP2: en este MNP inicial los paquetes de caracteres asincrónicos se transmiten con un encabezamiento, que reduce la velocidad (throughput) a 84%, por ello prácticamente no se ha usado.
- MNP3: en él los bits de start y stop del byte son eliminados por lo que se pasa de 10 a 8 bits por carácter, con lo que se gana un 20%, sin embargo al convertirlo en una serie sincrónica de datos (paquetes) se agregan una serie de bits que suman un 12%, por lo que se está ganando un 8%, lo que lleva la velocidad de 2400 bps a 2600 bps cuando no hay ruido que produzca errores. Los protocolos basados en software (no en

firmware) no quitan los bits de start y stop, por lo que su eficiencia es menor. El método de detección de errores de MNP es CRC de 16 bits sobre bloques de igual longitud en MNP3 (en otros la longitud es variable), una vez detectado un error se emplea retransmisión para corregirlo, o sea es ARQ. Sin embargo se transmiten 8 bloques y al final de ellos es que se espera la información de si hubo errores y de cual fue el primer bloque con errores, ordenándose la retransmisión a partir del primer bloque erróneo, es entonces un método n-atrás (go back-n).

- MNP4: es casi idéntico al 3 sólo que tiene un mejor rendimiento (throughput) debido a que ajusta la longitud de los bloques permitiendo bloques más largos durante lapsos de pocos errores, lo que aumenta la eficiencia a 120%.
- MNP5: incluye las características descriptas de las clases 3 y 4, pero usa además compresión de datos para mejorar la eficiencia. La mejora depende del tipo de datos, si se trata de textos de lectura la eficiencia puede llegar a duplicarse (mejora típica 85%), en datos y archivos ya comprimidos la mejora es ínfima. El método de compresión empleado hace dos cosas: primero, busca caracteres repetidos, cuando tres o más caracteres aparecen en sucesión los comprime; segundo, usa tabulación dinámica de caracteres repetidos, lo que le da habilidad de aprendizaje.
- MNP7: comprime 3 a 1, y emplea técnicas de codificación de longitud en el algoritmo de compresión, de modo que registra la recurrencia de caracteres y ajusta el código en consecuencia.

LAP-M (Link Access Procedure for Modems): es un estándar de detección y corrección de errores del CCITT basado también en CRC.

Como existen millones de modems con MNP, CCITT adoptó en 1988 nuevos estándares:

- V.42: incluye dos protocolos de detección y corrección de errores: LAP-M y MNP 2, 3 y 4. Además permite comprimir los datos usando MNP 5 ó 7.
- V.42bis: es un estándar de compresión de datos destinado a reemplazar MNP5 en modems que usan detección y corrección de errores de V.42.Se logran relaciones de compresión de hasta 4 a 1 mediante un algoritmo llamado de Lempel-Ziv (en él los caracteres usados frecuentemente son codificados y se construye un diccionario, y se transmiten solo los símbolos del diccionario en lugar del carácter completo, los diccionarios son construidos en ambos extremos usando encabezamientos mínimos), con esto se logra que un módem de 9600 bps pase a 56700 bps.
- V.Fast (llamado también V.32turbo): es un estándar que aprobado por el CCITT soporta velocidades hasta 28800 bps para datos sin comprimir, y usando V42bis podrá llegar a 86400 bps en líneas discadas comunes (de voz).

3.5.- Compresión.

La compresión es el “Holy Grail” de las telecomunicaciones ya que en la medida que se desarrollan mejores esquemas de compresión es posible utilizar más eficientemente el canal de comunicaciones. Esquemas como la codificación de Huffman**[7][14]** tiene más de 20 años. La compresión se utiliza para voz y para video, esquemas**[14]** como **JPEG,QCIF,MPEG 1,2,3 y 4**(200:1)**[15]** y **Wavelets** (480:1)son muy interesantes.

El esquema de compresión **MP3**,utilizado para música y video en Internet,es en realidad un **MPEG-1 capa 3** con compresiones que van desde 96:1 en sonido telefónico a 12:1 en calidad CD **[20].**

Es necesario consultar publicaciones recientes ya que es un campo en el que se está trabajando a diario.

3.6 - Seguridad en Redes.

3.6.1- Introducción.

Un sistema de comunicación de datos que no presente una protección adecuada está bajo la amenaza de un uso indebido o ilegal de la información presente en él. Por esta razón todos los

sistemas de comunicación de datos deberían poseer cierto grado de seguridad o privacidad de acorde a la importancia de la información manejada. Para esto existe una gran variedad de técnicas tales como: protección física, sistemas programados, encriptamiento, etc.

Hoy en día un especialista en comunicaciones de datos no puede estar armado solamente de sus conocimientos de modulación, debe tener como tarea fundamental mantener un traslado de datos seguro y confiable de un sitio a otro, por lo que es muy importante el conocimiento y el manejo de las técnicas de encriptamiento para la seguridad y confiabilidad del traslado de datos.

3.6.2- Aplicaciones de los Sistemas de Comunicación de Datos (SCD).

Hasta hace 30 años el uso y los usuarios de SCD eran muy limitados. Hoy en día esta técnica se utiliza en casi todas partes: negocios, institutos financieros y agencias de gobierno dependen de los SCD. Por ejemplo:

- **Negocios.** Usan los SCD para todas sus tareas diarias: Registros financieros; recepción, expedición y mantenimiento de cuentas; transferencia de fondos electrónicamente; inventarios y control de stock; control de procesos; órdenes de compra, comercio electrónico, reservaciones aéreas, hoteleras,etc.
- **Institutos Financieros.** Varios de los primeros sistemas de computadoras comerciales fueron usados en bancos y finanzas para operaciones financieras, traslado de fondos, terminales financieros, cajeros automáticos, redes de datos.
- **Agencias de Gobierno.** Aparte del uso militar están los sistemas de base de datos usados por la policía, el impuesto sobre la renta, etc.
- **Empresas.** Intranet (redes internas de las empresas con correo electrónico, conferencias (audio ó tele), acceso a bases de datos, bibliotecas, etc.).
- **Público.** Internet.

3.6.3- Acceso no autorizado de datos.

Cuando más y más datos se mueven entre computadores, mantener la confidencialidad de los datos comienza a ser un problema. Para nuestros propósitos, ***seguridad de datos*** significa "**proteger los datos de alteración o acceso por partes no autorizadas**".

La información es un artículo único. Puede ser copiada sin alterar la información original, esta característica hace del robo de información algo difícil de detectar. O alterada, sin técnicas especiales es difícil de detectar si la información de un SCD a sido modificada sin el conocimiento del usuario del SCD.

3.6.4- Técnicas para seguridad de datos.

3.6.4.1- Introducción.

La seguridad de los datos se complica debido a que cuando se hace una copia de la información en una computadora la información no es alterada ni destruida. Por lo tanto, descubrir esta clase de hurto de información es muy difícil. Sin embargo, varias cosas pueden hacerse para mejorar la seguridad de los datos y de este modo decrementar la posibilidad de acceso no autorizado a la información. Existen ciertas recomendaciones para mejorar la seguridad de los datos, tales como:

- **Revisión del Personal:** Para evitar el acceso de personas no autorizadas a la información, se debe decidir primero quien está autorizado al acceso de la información y a que clase de información.

- **Seguridad Física:** Se refiere a los métodos de "boxing out" para evitar el fisgoneo. La medida más directa debe ser el de asegurarse de mantener los computadores y los terminales en áreas de acceso controlado. Esto reduce grandemente la oportunidad de que alguien haga una visita no autorizada a los datos y haga una copia por sí mismo.
- **Validación de Acceso:** Dado que hay un número de personas con acceso al sistema se debe mantener una clase de control, ya que no todos los usuarios tienen la misma necesidad de información, por lo que deben ser implementadas bases de datos segmentadas y passwords (contraseñas).
- **Passwords:** Permiten que los terminales y computadores den acceso a cierta clase de información permitida previo conocimiento del respectivo password. Deben ser cambiados frecuentemente para mayor efectividad.
- **Sistemas de Call-Back:** La mayoría de los sistemas de computadoras que pueden ser accedidos por línea telefónica son vulnerables a Break-ins por parte de un llamador anónimo. Por esto se requiere que el sistema detecte llamadas no autorizadas,regrese las llamadas para transferir datos,utilice passwords y exista chequeo de identidad del que realizó la llamada.
- **Codificación:** Si no se puede mantener los datos fuera del alcance de manos no autorizadas, lo mejor que sigue es realizar una transmisión disfrazada. Una forma de realizar esto es codificar los datos de manera que parezcan sin sentido a los ojos del ladrón de datos. Por largo tiempo simplemente se codificaban los datos con códigos de computadoras que servían para todo propósito, ya que no había muchas personas que aprendieran a *romper* el código. Al surgir la literatura sobre computadoras estas técnicas quedaron obsoletas.

3.6.4.2- Encriptación {21].

Encriptación es el proceso de codificación de la información, de tal manera de hacer difícil para el ladrón de información que lo robado tenga sentido para él. La codificación no debe ser demasiado compleja pues al usuario le sería engorroso entenderla,ni muy simple como para que un lector dedicado pueda aprender a descifrarla. En términos de encriptación, el mensaje original es referido como ***plain text***, debido a que está en lenguaje sencillo o llano. La salida del proceso de encriptación se llama ***cipher text***, debido a que es una versión codificada del texto sencillo de entrada (un cipher es un sistema de escritura secreto que usa un esquema prearreglado o clave).

El uso de equipos de codificación automática datan desde 1930. Hasta hace poco tales equipos sólo estaban disponibles para los gobierno y usos militares, esto ya no es así.

3.6.4.3- Métodos de Encriptación.

a) Sustitución. Un cifrado por sustitución reemplaza los caracteres originales uno a uno por otro set de caracteres. Los dos sets carácter a carácter como en el siguiente ejemplo:

Alfabeto original:	ABCDEFGHIJKLMNOPQRTSUVWXYZ
Alfabeto de sustitución:	BCDEFGHIJKLMNOPQRSTUVWXYZA
Mensaje original:	MEET ME AT EIGHT ON FRIDAY
Mensaje codificado:	NFFU NF BU FJHIU PO GSJEBZ

En este ejemplo se ilustra un cifrado de sustitución monoalfabético donde el alfabeto de sustitución es la clave, la cual produce el mensaje en clave. Así un cifrado por sustitución

monoalfabético reemplaza cada letra del mensaje original con una letra del código clave. En nuestro ejemplo el código es un caso especial llamado *Caesar cipher*, llamado así por Julius Caesar. Un código Caesar usa un alfabeto de sustitución único, consistiendo en el alfabeto básico (A a la Z), que es desplazado en posición. El corrimiento es prearreglado. Se puede notar en el ejemplo que el corrimiento es de una posición. Para otros cifrados por sustitución de alfabeto el orden de los caracteres del texto original no es retenido.

Una mirada al código de salida nos puede revelar que este método no cambia la longitud del mensaje, ni la longitud de cada palabra, por lo que conociendo el idioma en que esta el mensaje es relativamente fácil saber que dice el mensaje final basándonos en la repetibilidad de las letras en cada idioma como veremos a continuación:

Mensaje codificado: NFFU NF BU FJHIU PO GSJEBZ

Número de letras: 21

Número de veces usadas cada letras:

F4 U3 B2 N2 J2 G1 H1 I1 O1 P1 S1 Z1

En inglés la letra E es la letra más usada comúnmente. En orden estadístico de repetición tenemos que las letras siguen el siguiente orden: ETAON RISHD, así como algunas palabras también son de uso común como “THE”, con todo esto podemos descifrar fácilmente el texto. Por la facilidad de descifrar el texto el código de sustitución no es usado actualmente.

b)Cifrado por transposición. Otro método simple de cifrado es el de transposición. Un poco mejor que el de sustitución de caracteres, usa los caracteres actuales escritos de otra forma. En el siguiente ejemplo podemos apreciar un método simple de transposición, donde el mensaje se escribe de derecha a izquierda

Mensaje original:	MEET ME AT THE MOVIES
Mensaje codificado:	SEIVOM EHT TA EM TEEM

Este ejemplo ilustra la naturaleza del cifrado, pero el ejemplo es sencillo a propósito. Es posible reescribir el mensaje de otras formas, por lo tanto la transposición no es tan obvia como se muestra:

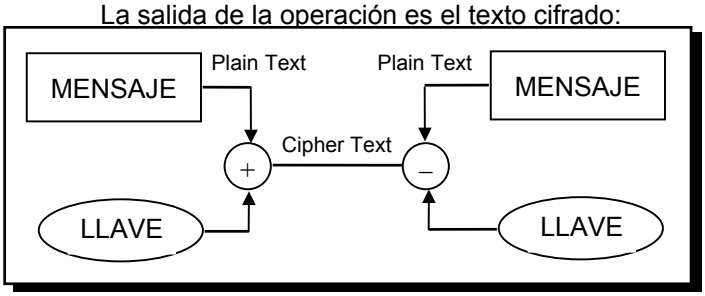
Mensaje original:	BEGIN THE ATTACK AT DAWN
Reescribiendo:	B G N H A T C A D W ↓↑↓↑↓↑↓↑↓↑↓↑↓↑↓↑↓↑↓↑↓ E I T E T A K T A N
Mensaje a transmitir:	BGNHA TCADW EITET AKTAN

En este ejemplo se puede ver que el mensaje es transpuesto en un patrón repetitivo, arriba y abajo, izquierda a derecha. El mensaje fue separado en posiciones pares e impares, la fila superior se escribe primero que la inferior y el mensaje final fue separado en bloques de cinco letras para la transmisión. Mientras este mensaje parece mucho más seguro que el de sustitución, realmente no ofrece mucho. La longitud de las palabras y los puntos de ruptura desaparecen, pero la frecuencia de las letras tiende a ser retenida. Además, no es necesario un código de cifrado por sustitución, el texto original está aquí, solo que rearreglado.

c) Sistemas de llave privada. Los más modernos métodos de cifrado de texto dan forma al texto original a partir de cómputos matemáticos. La operación más común consiste en añadir al texto original otra cadena de caracteres, la cual sirve como llave. O sea:

Plain Text + Key (Data Stream) = Cipher Text

La salida de la operación es el texto cifrado:



Veamos ahora un ejemplo de como un sistema basado en técnicas computacionales codifica. Aquí cada carácter puede ser representado por su valor en ASCII, tanto la clave, el texto original y el texto cifrado. Vamos a considerar una clave *random* corta con el fin de explicar el ejemplo. Supongamos que el texto original es "*Meet me on friday*" y la clave aleatoria es "*ALMQV1YXRGNB95FPO*" como se muestra a continuación:

Mensaje original:	Meet me on friday
Clave:	ALMQV1YXRGNB95FPO

Ahora construyamos el texto cifrado por codificación del código ASCII del mensaje original con la clave:

Código ASCII										
Mensaje:	77	101	101	116	32	77	101	32	111	110 ...
Clave:	65	76	77	81	86	49	89	88	82	71...
Suma:	142	177	178	197	118	126	190	120	193	181...
-128*:	14	49	50	69	118	126	62	120	65	53...
Cifrado:	SO	1	2	E	v	~	>	x	A	5...

**Se sustrae solamente si Suma > 127*

Al analizar este ejemplo se puede observar que el valor ASCII de cada carácter del mensaje original (incluidos los espacios) se le añade el valor ASCII de cada carácter de la clave. Nótese que el resultado de la suma se reduce si esta excede a 127. Por lo tanto son retransladados al código ASCII y no necesariamente todos pueden ser impresos: al primer carácter del cifrado del ejemplo corresponde el código SO (Shift Out) del ASCII. Éste es un carácter para control de periféricos y no tiene representación impresa, pero puede ser transmitido fácilmente como dato.

La salida de este proceso parece ser más *random* que la salida de los otros métodos de cifrado estudiados hasta ahora. En parte esto es debido a que los espacios entre las palabras

también son encriptados (lo que no podía hacerse con los otros métodos) y también debido a que el mensaje se encuentra en mayúsculas y minúsculas. El texto cifrado puede ser diferente para el mismo mensaje si éste es escrito totalmente en mayúsculas y con la misma clave. La verdadera razón de que la salida parezca más aleatoria es la naturaleza de la clave. Si la clave es verdaderamente aleatoria, entonces la secuencia de caracteres de la clave nunca se repetiría. Si los caracteres no se repiten, entonces nunca se podría descifrar dos segmentos del mensaje original con la misma secuencia de la clave. Si se genera una clave absolutamente aleatoria, el mensaje no puede ser descifrado sin la clave.

Sin embargo, no es posible producir una verdadera secuencia aleatoria de números. En su lugar se obtienen diferentes puntos de partida o semillas (a través de algoritmos) en el programa que genera la secuencia de números. Para un observador, estos números parecen ser aleatorios, sin patrón alguno. Los matemáticos se refieren a ellos como números pseudo-aleatorios, porque el patrón no es realmente aleatorio por lo menos para quien conozca el algoritmo.

Tarde o temprano, una clave producida por un generador de números pseudo-aleatorios puede repetirse. Si el “rompe-códigos” puede comparar dos secciones de texto cifrado que fue producido con la misma secuencia de la clave para dos mensajes diferentes,puede comenzar a romper la clave. Así, si el rompe-códigos tiene suficiente paciencia, puede eventualmente duplicar la clave.

Ésta no es la única mala noticia. Si el patrón fuese totalmente aleatorio, ¿cómo podría ser recreada la clave por el receptor para que el mensaje fuera descifrado?. Se podría pensar en enviar la clave (el algoritmo) como dato, pero esto sería frustrar el propósito de la encriptación, ya que alguien casualmente podría capturar tanto la clave como el texto cifrado. Las secuencia pseudo-aleatorias, sin embargo, pueden ser recreadas por el receptor para producir una copia de la llave usada para encriptar el mensaje. Esta es una de las razones por las cuales se usa.

Por último, el método de sumar la clave no es el único método para encriptar. Se pueden usar diversos algoritmos, tales como multiplicar, lógica complementaria, etc. y la encriptación puede ser realizada por software o hardware automáticamente.

d) Sistemas de llave pública[9]. La distribución y uso de las llaves de cifrado en una red de datos puede ser un trabajo difícil. El primer problema es como distribuir las llaves. Obviamente, no se pueden distribuir las llaves por el enlace de comunicaciones. En lugar de eso, las llaves deben ser enviadas por rutas más seguras. Otro problema es que si la llave no es obtenida uniformemente debido a problemas de sincronismo, la red no puede trabajar apropiadamente hasta que la llave correcta esté en su sitio. Una forma fácil de resolver el problema del manejo de llaves es el uso de sistemas de *llave pública*.

En un sistema de llave pública (desarrollado por Whitfield Diffie y Martin Hellman en 1976, usa algoritmos sobre campos finitos para asegurar y desencriptar los datos) la llave se divide en dos partes. Una parte de la llave es información pública. La segunda parte es información privada conocida solamente por las personas que quieren utilizar la llave para comunicarse. La teoría de los sistemas de llave pública es compleja y todavía no es usada al máximo. Sin embargo, estos sistemas ofrecen la promesa de comunicaciones seguras con un reducido problema de manejo de llaves.

Para usar un sistema de llave pública cada usuario construye una función de una vía que se usa para encriptar los mensajes. Ésta es una porción pública de la llave. El usuario también construye la función inversa de esta función, la cual es necesaria para descifrar los mensajes encriptados con la porción pública de la llave. Esta función inversa es mantenida en secreto y es la porción privada de la llave. Aún si se conoce la porción de la llave pública, un ladrón de información no puede calcular la porción privada.

Los sistemas de llave pública están basados en el concepto conocido como **one-way trap door functions**. Las funciones son fáciles de desarrollar, aunque la teoría fundamental no es

sencilla. Las funciones inversas son únicas y no pueden ser derivadas de funciones trampas conocidas. Para derivar la función inversa, el rompe-códigos debe conocer como la función fue construida y no sólo conocer el resultado que la construcción produce.

Para entrar a un sistema de llave pública, cada usuario primero debe publicar una llave pública de encriptación en un directorio. La llave consiste en dos números. Entonces, cuando dos usuarios A y B quieren comunicarse, cada uno de ellos va al directorio. Usando la llave pública que A ha publicado en el directorio, B cifra un mensaje y lo envía a A. A, conociendo la función inversa de esta única función trampa, puede descifrar este mensaje fácilmente y reconstruir el texto original. De manera similar, si A desea enviar una respuesta a B, el mensaje es cifrada con la llave pública de B, la cual fue colocada por B en el directorio. B puede descifrar este mensaje usando la inversa de su función *trap-door*.

Esta llave de dos partes es menos segura que los que usan una sola llave completa mantenida en secreto dado que midiendo el tiempo de procesamiento de ciertas operaciones de cifrado pueden "romperse" la llave publica.[1] Esto es verdad, sin embargo, recuerden que la información no necesita mantenerse segura por siempre. De esta manera, si un sistema de llave pública puede mantener por períodos de tiempo razonable una adecuada seguridad, podemos sentirnos satisfechos. Debido a que ellos pueden llevar a cabo esta meta, son usados más y más sistemas de llave pública.

3.6.4.5- Estándares de Encriptación de Datos.

Para probar si los requerimientos de seguridad de datos se cumplen y también para proveer cierto grado de uniformidad, el *National Bureau of Standards* fue encargado en 1970 para desarrollar un *Data Encryption Standard* (DES) para ser usado en aplicaciones no relacionadas con defensa dentro del gobierno de los U.S. DES fue diseñado para satisfacer los siguientes criterios:

- Alto nivel de protección contra acceso y modificaciones de datos no autorizadas.
- Simple de entender, pero difícil de violar.
- Protección basada en llave y no dependiente de cualquier programa o proceso de encriptación.
- Económico y eficiente.
- Adaptable a diversas aplicaciones.
- Disponible para todos los usuarios y suplidores a un costo razonable.

El DES resultó aprobado por el Gobierno de los Estados Unidos en 1977.

El DES encripta bloques de 64 bits usando una llave de 56 bits dando lugar a billones (trillones en inglés) de posibles permutaciones, la llave es un algoritmo usado para codificar los datos. Independientemente de que se trate de “llave privada” ó de “llave pública” con estaciones de trabajo poderosas puede “romperse” el código, por ello se han considerado varias alternativas:

- usar llaves de 128 bits
- Utilizar “triple DES”, cada bloque es codificado tres veces con tres llaves diferentes.
- usar algoritmos diferentes del DES

El tema además de sus aspectos de seguridad de datos tiene connotaciones públicas. Los gobiernos, sus cuerpos de seguridad, organismos de inteligencia, etc. se niegan a permitir

[1] Ver Data Communications February 1996.

esquemas que ellos no conozcan ó que no puedan de algún modo controlar para poder "examinar" el contenido de los datos. Por ello el Gobierno norteamericano solo autorizaba la exportación de sistemas con llaves de 40 bits, luego autorizó a tres empresas a exportar las de DES de 56 bits.

Casos como el del profesor de matemática David J. Bernstein, de Illinois, procesado por intentar exportar un programa de cifrado de su creación han causado polémicas, intervenciones judiciales en resguardo de los derechos constitucionales y preocupación en la industria estadounidense.

En Noviembre del año 2001 la Secretaría de Comercio de los Estados Unidos aprobó un nuevo estándar de cifrado el "Advanced Encryption Standard (AES) (FIPS PUB 197)", esté AES fue aprobado por la agencia "National Institute of Standards and Technology (NIST)" y publicado como FIPS(Federal Information Processing Standards) PUB 197 y consiste en un algoritmo de cifrado simétrico de bloque y es capaz de utilizar llaves de 128, 192 y 256 bits para cifrar y decifrar los datos en bloques de 128 bits, para ello utiliza el algoritmo de **Rijndael.** Para más detalles visitar http://www.nist.gov/ donde hay abundante información sobre criptografía y sobre este nuevo algoritmo, ver especialmente fips-197.pdf.

3.6.4.6.- Autenticación.

Encriptar no es suficiente es necesario "autenticar", ó sea, asegurar que las personas ó entes participando en una "conversación" (ó intercambio de datos) son quienes dicen ser. Los mecanismos empleado van desde tarjetas de seguridad hasta autoridades de certificación.

Una técnica muy popular es la **"firma digital"**,en ella se calcula con una función criptográfica de un solo sentido un valor llamado "hash",se envía el texto original y el valor hash encriptado con una llave privada(que se denomina **"firma"**), en el extremo receptor se vuelve a calcular el valor hash,se desencripta la firma con clave pública y se compara con el hash calculado,si verifica el mensaje es auténtico.

3.6.4.7.- Firewalls.

En conexiones de servicios externos,líneas de acceso discadas e Internet cualquiera puede intentar acceder a su computadora con el propósito de ver ó modificar archivos sin estar autorizado para ello ó causar daños en directorios y aún en la operatividad del equipo,para prevenir esto se instala un equipo de acceso dedicado con medidas especiales de seguridad denominadas "firewalls" ó "muros de seguridad" que solamente dejan pasar paquetes autorizados. Se parte del principio de que dentro de la organización todos son confiables y que fuera de ella todos son "hackers"(ó intrusos),los requisitos de una buena firewall son:

Facilidad de configuración y administración,flexible en cuanto a técnicas de bloqueo,habilidad de registrarlo todo,emitir alarmas a los administradores y reglas detalladas de control de los accesos.

APÉNDICES

APÉNDICE 3A:
TÉRMINOS DE SEGURIDAD

Adaptados del Diccionario Gratuito de Cómputos en-Línea, por René Ramos. http://wombat.doc.ic.ac.uk/

Authentification: Autenticar o autentificar

La verificación de la identidad de una persona o un proceso. En sistemas de comunicaciones, la identificación verifica que la procedencia del mensaje es la que éste indica, como la firma en una carta.

Cracker: Intruso

Un individuo que trata de tener acceso no autorizado a un sistema de computación. Estas personas muchas veces actúan maliciosamente, y tienen una serie de recursos para entrar ilícitamente en un sistema.

Encryption: Encripción, codificación, cifrar

Cualquier mecanismo usado en criptografía para convertir un texto común en un texto cifrado con el fin de prevenir que nadie, excepto el destinatario, lea el texto. Existen varios tipos de cifrado de información, y ellos están basados en las normas de seguridad de las redes. Entre los tipo más conocidos están el Patrón de Cifrado de Información (Data Encryption Standard), y Cifrado de Clave Pública (Public Key Encryption).

Firewall: Cortafuegos

Equipo de acceso dedicado, con medidas especiales de seguridad, y utilizado para conexiones de servicios externos, y para líneas de acceso de discado. La idea es mantener fuera del alcance de los intrusos todos los sectores administrativos de la red. Un cortafuegos típico es un sistema operativo Unix basado en un microprocesador, de bajo costo, y sin información crítica, con modems, y puertos para red pública, pero con una conexión con la red que es cuidadosamente observada. Entre las precauciones especiales se pueden incluir seguimiento con amenaza, Call-Back, y cajas de seguridad para controlar identidades y patrones de actividad.

Hacker: Insistente, forzador, violador

Originalmente, un programador o usuario de computadora con mucha experiencia. Ahora se utiliza este nombre para denotar intrusos ilegales en sistemas de computadoras que tratan de usurpar información clasificada. De ahí viene “el violador de códigos personales” (password hacker), “el violador de redes” (network hacker). El término correcto es “cracker”, intruso.

Phreaking: Usurpación

Es el arte y la ciencia de entrometerse en las redes de teléfonos como, por ejemplo, para hacer una llamada de larga distancia sin pagar. Por extensión, se utiliza en otros contextos, especialmente, pero no exclusivamente, en redes de comunicaciones, como por ejemplo, “violación de seguridad” (security phreaking).

Public Key Encryption: Cifrado de Clave Pública

Un esquema de cifrado, introducido en 1976 por Diffie y Hellman, en el cual cada persona obtiene claves pares, una pública y otra secreta. Cada clave pública está al acceso de otros, pero no la secreta. Los mensajes son cifrados, utilizando la clave pública del recipiente, pero sólo pueden ser descifrados utilizando la clave secreta. La posibilidad de que el emisor y el receptor puedan intercambiar información sobre la clave secreta es eliminada, y las claves secretas nunca se comparten.

Sniffer: rastreador

Es una herramienta de seguimiento de red que puede tener acceso a paquetes de información y puede codificarlos con el fin de conseguir su protocolo. Es utilizado, a menudo, por los violadores de redes para robar códigos personales de acceso, así como por los detectives cibernéticos para controlar las actividades de los violadores.

SSL (Secure Sockets Layer): Capa de cerrojos de seguridad

Es un protocolo diseñado por Netscape Communications Corporation con el fin de proporcionar seguridad en Internet. SSL está colocada por debajo de la jerarquía de otros protocolos como HTTP, SMTP, Telnet, FTP, Gopher y NNTP, y colocada por encima de los protocolos de conexión de TCP/IP. Es utilizado para el método de acceso HTTP.

APÉNDICE 3B:

CRIPTOGRAFÍA

Parte 1

Introduction to Cryptography

•Preface •Basic Terminology •Basic Cryptographic Algorithms •Digital Signatures •Cryptographic Hash Functions •Cryptographic Random Number Generators •Strength of Cryptographic Algorithms •Cryptanalysis and Attacks on Cryptosystems

---

Preface

People mean different things when they talk about cryptography. Children play with toy ciphers and secret languages. However, these have nothing to do with real security and strong encryption. Strong encryption is the kind of encryption that can be used to protect information of real value against organized criminals, multinational corporations, and major governments. Strong encryption used to be only military business; however, in the information society it has become one of the central tools for maintaining privacy and confidentiality.

As we move into an information society, the technological means for global surveillance of millions of individual people are becoming available to major govenments. Cryptography has become one of the main tools for privacy, trust, access control, electronic payments, corporate security, and countless other fields.

Cryptography is no longer a military thing that should not be messed with. It is time to demystify cryptography and make full use of the advantages it provides for the modern society. Widespread cryptography is also one of the few defenses people have against suddenly finding themselves in a totalitarian surveillance society that can monitor and control everything they do.

Basic Terminology

Suppose that someone wants to send a message to a receiver, and wants to be sure that no-one else can read the message. However, there is the possibility that someone else opens the letter or hears the electronic communication.

In cryptographic terminology, the message is called plaintext or cleartext. Encoding the contents of the message in such a way that hides its contents from outsiders is called encryption. The encrypted message is called the ciphertext. The process of retrieving the plaintext from the ciphertext is called decryption. Encryption and decryption usually make use of a key, and the coding method is such that decryption can be performed only by knowing the proper key.

Cryptography is the art or science of keeping messages secret. Cryptanalysis is the art of breaking ciphers, i.e. retrieving the plaintext without knowing the proper key. People who do cryptography are cryptographers, and practitioners of cryptanalysis are cryptanalysts.

Cryptography deals with all aspects of secure messaging, authentication, digital signatures, electronic money, and other applications. Cryptology is the branch of mathematics that studies the mathematical foundations of cryptographic methods.

Basic Cryptographic Algorithms

A method of encryption and decryption is called a cipher. Some cryptographic methods rely on the secrecy of the algorithms; such algorithms are only of historical interest and are not adequate for real-world needs. All modern algorithms use a key to control encryption and decryption; a message can be decrypted only if the key matches the encryption key. The key used for decryption can be different from the encryption key, but for most algorithms they are the same.

There are two classes of key-based algorithms, symmetric (or secret-key) and asymmetric (or public-key) algorithms. The difference is that symmetric algorithms use the same key for encryption and decryption (or the decryption key is easily derived from the encryption key), whereas asymmetric algorithms use a different key for encryption and decryption, and the decryption key cannot be derived from the encryption key.

Symmetric algorithms can be divided into stream ciphers and block ciphers. Stream ciphers can encrypt a single bit of plaintext at a time, whereas block ciphers take a number of bits (typically 64 bits in modern ciphers), and encrypt them as a single unit. Many symmetric ciphers are described on the algorithms page.

Asymmetric ciphers (also called public-key algorithms or generally public-key cryptography) permit the encryption key to be public (it can even be published in a newspaper), allowing anyone to encrypt with the key, whereas only the proper recipient (who knows the decryption key) can decrypt the message. The encryption key is also called the public key and the decryption key the private key or secret key.

Modern cryptographic algorithms cannot really be executed by humans. Strong cryptographic algorithms are designed to be executed by computers or specialized hardware devices. In most applications, cryptography is done in computer software, and numerous cryptographic software packages are available.

Generally, symmetric algorithms are much faster to execute on a computer than asymmetric ones. In practice they are often used together, so that a public-key algorithm is used to encrypt a randomly generated encryption key, and the random key is used to encrypt the actual message using a symmetric algorithm.

Many good cryptographic algorithms are widely and publicly available in any major bookstore, scientific library, or patent office, and on the Internet. Well-known symmetric functions include DES and IDEA. RSA is probably the best known asymmetric algorithm. The books page lists several good textbooks on cryptography and related topics.

Digital Signatures

Some public-key algorithms can be used to generate digital signatures. A digital signature is a block of data that was created using some secret key, and there is a public key that can be used to verify that the signature was really generated using the corresponding private key. The algorithm used to generate the signature must be such that without knowing the secret key it is not possible to create a signature that would verify as valid.

Digital signatures are used to verify that a message really comes from the claimed sender (assuming only the sender knows the secret key corresponding to his/her public key). They can also be used to timestamp documents: a trusted party signs the document and its timestamp with his/her secret key, thus testifying that the document existed at the stated time.

Digital signatures can also be used to testify (or certify) that a public key belongs to a particular person. This is done by signing the combination of the key and the information about its owner by a trusted key. The reason for trusting that key may again be that it was signed by another trusted key. Eventually some key must be a root of the trust hierarchy (that is, it is not trusted because it was signed by somebody, but because you believe a priori that the key can be trusted). In a centralized key infrastructure there are very few roots in the trust network (e.g., trusted government agencies; such roots are also called certification authorities). In a distributed infrastructure there need not be any universally accepted roots, and each party may have different trusted roots (such of the party's own key and any keys signed by it). This is the web of trust concept used e.g. in PGP.

A digital signature of an arbitrary document is typically created by computing a message digest from the document, and concatenating it with information about the signer, a timestamp, etc. The resulting string is then encrypted using the private key of the signer using a suitable algorithm. The resulting encrypted block of bits is the signature. It is often distributed together with information about the public key that was used to sign it. To verify a signature, the recipient first determines whether it trusts that the key belongs to the person it is supposed to belong to (using the web of trust or a priori knowledge), and then decrypts the signature using the public key of the person. If the signature decrypts properly and the information matches that of the message (proper message digest etc.), the signature is accepted as valid.

Several methods for making and verifying digital signatures are freely available. The most widely known algorithm is RSA.

Cryptographic Hash Functions

Cryptographic hash functions are typically used to compute the message digest when making a digital signature. A hash function compresses the bits of a message to a fixed-size hash value in a way that distributes the possible messages evenly among the possible hash values. A cryptographic hash function does this in a way that makes it extremely difficult to come up with a message that would hash to a particular hash value.

Cryptographic hash functions typically produce hash values of 128 or more bits. This number is vastly larger than the number of different messages likely to ever be exchanged in the world.

Many good cryptographic hash functions are freely available. Well-known ones include MD5 and SHA.

Cryptographic Random Number Generators

Cryptographic random number generators generate random numbers for use in cryptographic applications, such as for keys. Conventional random number generators available in most programming languages or programming environments are not suitable for use in cryptographic applications (they are designed for statistical randomness, not to resist prediction by cryptanalysts).

In the optimal case, random numbers are based on true physical sources of randomness that cannot be predicted. Such sources may include the noise from a semiconductor device, the least significant bits of an audio input, or the intervals between device interrupts or user keystrokes. The noise obtained from a physical source is then "distilled" by a cryptographic hash function to make every bit depend on every other bit. Quite often a large pool (several thousand bits) is used to contain randomness, and every bit of the pool is made to depend on every bit of input noise and every other bit of the pool in a cryptographically strong way.

When true physical randomness is not available, pseudorandom numbers must be used. This situation is undesirable, but often arises on general purpose computers. It is always desirable to obtain some environmental noise - even from device latencies, resource utilization statistics, network statistics, keyboard interrupts, or whatever. The point is that the data must be unpredictable for any external observer; to achieve this, the random pool must contain at least 128 bits of true entropy.

Cryptographic pseudorandom generators typically have a large pool ("seed value") containing randomness. Bits are returned from this pool by taking data from the pool, optionally running the data through a cryptographic hash function to avoid revealing the contents of the pool. When more bits are needed, the pool is stirred by encrypting its contents by a suitable cipher with a random key (that may be taken from an unreturned part of the pool) in a mode which makes every bit of the pool depend on every other bit of the pool. New environmental noise should be mixed into the pool before stirring to make predicting previous or future values even more impossible.

Even though cryptographically strong random number generators are not very difficult to built if designed properly, they are often overlooked. The importance of the random number generator must thus be emphasized - if done badly, it will easily become the weakest point of the system.

Several examples of cryptographic random number generators are publicly available.

Strength of Cryptographic Algorithms

Good cryptographic systems should always be designed so that they are as difficult to break as possible. It is possible to build systems that cannot be broken in practice (though this cannot usually be proved). This does not significantly increase system implementation effort; however, some care and expertise is required. There is no excuse for a system designer to leave the system breakable. Any mechanisms that can be used to circumvent security must be made explicit, documented, and brought into the attention of the end users.

In theory, any cryptographic method with a key can be broken by trying all possible keys in sequence. If using brute force to try all keys is the only option, the required computing power increases exponentially with the length of the key. A 32 bit key takes 2^32 (about 10^9) steps. This is something any amateur can do on his/her home computer. A system with 40 bit keys (e.g. US-exportable version of RC4) takes 2^40 steps - this kind of computing power is available in most universities and even smallish companies. A system with 56 bit keys (such as DES) takes a substantial effort, but is quite easily breakable with special hardware. The cost of the special hardware is substantial but easily within reach of organized criminals, major companies, and governments. Keys with 64 bits are probably breakable now by major governments, and will be within reach of organized criminals, major companies, and lesser governments in a few years. Keys with 80 bits may become breakable in future. Keys with 128 bits will probably remain unbreakable by brute force for the foreseeable future. Even larger keys are possible; in the end we will

encounter a limit where the energy consumed by the computation, using the minimum energy of a quantum mechanic operation for the energy of one step, will exceed the energy of the mass of the sun or even of the universe.

However, key length is not the only relevant issue. Many ciphers can be broken without trying all possible keys. In general, it is very difficult to design ciphers that could not be broken more effectively using other methods. Designing your own ciphers may be fun, but it is not recommended in real applications unless you are a true expert and know exactly what you are doing.

One should generally be very wary of unpublished or secret algorithms. Quite often the designer is then not sure of the security of the algorithm, or its security depends on the secrecy of the algorithm. Generally, no algorithm that depends on the secrecy of the algorithm is secure. Particularly in software, anyone can hire someone to disassemble and reverse-engineer the algorithm. Experience has shown that a vast majority of secret algorithms that have become public knowledge later have been pitifully weak in reality.

The key lengths used in public-key cryptography are usually much longer than those used in symmetric ciphers. There the problem is not that of guessing the right key, but deriving the matching secret key from the public key. In the case of RSA, this is equivalent to factoring a large integer that has two large prime factors. In the case of some other cryptosystems it is equivalent to computing the discrete logarithm modulo a large integer (which is believed to be roughly comparable to factoring). Other cryptosystems are based on yet other problems.

To give some idea of the complexity, for the RSA cryptosystem, a 256 bit modulus is easily factored by ordinary people. 384 bit keys can be broken by university research groups or companies. 512 bits is within reach of major governments. Keys with 768 bits are probably not secure in the long term. Keys with 1024 bits and more should be safe for now unless major algorithmic advances are made in factoring; keys of 2048 bits are considered by many to be secure for decades. More information on RSA key length is in an article by Bruce Schneier.

It should be emphasized that the strength of a cryptographic system is usually equal to its weakest point. No aspect of the system design should be overlooked, from the choice algorithms to the key distribution and usage policies.

Cryptanalysis and Attacks on Cryptosystems

Cryptanalysis is the art of deciphering encrypted communications without knowing the proper keys. There are many cryptanalytic techniques. Some of the more important ones for a system implementor are described below.

•Ciphertext-only attack: This is the situation where the attacker does not know anything about the contents of the message, and must work from ciphertext only. In practice it is quite often possible to make guesses about the plaintext, as many types of messages have fixed format headers. Even ordinary letters and documents begin in a very predictable way. It may also be possible to guess that some ciphertext block contains a common word.

•Known-plaintext attack: The attacker knows or can guess the plaintext for some parts of the ciphertext. The task is to decrypt the rest of the ciphertext blocks using this information. This may be done by determining the key used to encrypt the data, or via some shortcut.

•Chosen-plaintext attack: The attacker is able to have any text he likes encrypted with the unknown key. The task is to determine the key used for encryption. Some encryption methods, particularly RSA, are extremely vulnerable to chosen-plaintext attacks. When such algorithms are used, extreme care must be taken to design the entire system so that an attacker can never have chosen plaintext encrypted.

•Man-in-the-middle attack: This attack is relevant for cryptographic communication and key exchange protocols. The idea is that when two parties are exchanging keys for secure communications (e.g., using Diffie-Hellman), an adversary puts himhelf between the parties on the communication line. The adversary then performs a separate key exchange with each party. The parties will end up using a different key, each of which is known to the adversary. The adversary will then decrypt any communications with the proper key, and encrypt them with the other key for sending to the other party. The parties will think that they are communicating securely, but in fact the adversary is hearing everything.

One way to prevent man-in-the-middle attacks is that both sides compute a cryptographic hash function of the key exchange (or at least the encryption keys), sign it using a digital signature algorithm, and send the signature to the other side. The recipient then verifies that the signature came from the desired other party, and that the hash in the signature matches that computed locally. This method is used e.g. in Photuris.

•Timing Attack: This very recent attack is based on repeatedly measuring the exact execution times of modular exponentiation operations. It is relevant to at least RSA, Diffie-Hellman, and Elliptic Curve methods. More information is available in the original paper and various followup articles.

There are many other cryptographic attacks and cryptanalysis techniques. However, these are probably the most important ones for a practical system designer. Anyone contemplating to design a new encryption algorithm should have a much deeper understanding of these issues. One place to start looking for information is the excellent book Applied Cryptography by Bruce Schneier.

Disclaimer: Any opinions and evaluations presented here are speculative, and the author cannot be held responsible for their correctness.

Parte 2

IAB and IESG statement on
cryptographic technology and the Internet

July 24, 1996

The Internet Architecture Board (IAB) and the Internet Engineering Steering Group (IESG), the bodies which oversee architecture and standards for the Internet, are concerned by the need for increased protection of international commercial transactions on the Internet, and by the need to offer all Internet users an adequate degree of privacy.

Security mechanisms being developed in the Internet Engineering Task Force to meet these needs require and depend on the international use of adequate cryptographic technology. Ready access to such technology is therefore a key factor in the future growth of the Internet as a motor for international commerce and communication.

The IAB and IESG are therefore disturbed to note that various governments have actual or proposed policies on access to cryptographic technology that either:

•(a) impose restrictions by implementing export controls; and/or •(b) restrict commercial and private users to weak and inadequate mechanisms such as short cryptographic keys; and/or •(c) mandate that private decryption keys should be in the hands of the government or of some other third party; and/or •(d) prohibit the use of cryptology entirely, or permit it only to specially authorized organizations.

We believe that such policies are against the interests of consumers and the business community, are largely irrelevant to issues of military security, and provide only a marginal or illusory benefit to law enforcement agencies, as discussed below.

The IAB and IESG would like to encourage policies that allow ready access to uniform strong cryptographic technology for all Internet users in all countries.

The IAB and IESG claim:

The Internet is becoming the predominant vehicle for electronic commerce and information exchange. It is essential that the support structure for these activities can be trusted.

Encryption is not a secret technology monopolized by any one country, such that export controls can hope to contain its deployment. Any hobbyist can program a PC to do powerful encryption. Many algorithms are well documented, some with source code available in textbooks.

Export controls on encryption place companies in that country at a competitive disadvantage. Their competitors from countries without export restrictions can sell systems whose only design constraint is being secure, and easy to use.

Usage controls on encryption will also place companies in that country at a competitive disadvantage because these companies cannot securely and easily engage in electronic commerce.

Escrow mechanisms inevitably weaken the security of the overall cryptographic system, by creating new points of vulnerability that can and will be attacked.

Export controls and usage controls are slowing the deployment of security at the same time as the Internet is exponentially increasing in size and attackers are increasing in sophistication. This puts users in a dangerous position as they are forced to rely on insecure electronic communication.

TECHNICAL ANALYSIS

KEY SIZE

It is not acceptable to restrict the use or export of cryptosystems based on their key size. Systems that are breakable by one country will be breakable by others, possibly unfriendly ones. Large corporations and even criminal enterprises have the resources to break many cryptosystems. Furthermore, conversations often need to be protected for years to come; as computers increase in speed, key sizes that were once out of reach of cryptanalysis will become insecure.

PUBLIC KEY INFRASTRUCTURE

Use of public key cryptography often requires the existence of a "certification authority". That is, some third party must sign a string containing the user's identity and public key. In turn, the third party's key is often signed by a higher-level certification authority.

Such a structure is legitimate and necessary. Indeed, many governments will and should run their own CAs, if only to protect citizens' transactions with their governments. But certification authorities should not be confused with escrow centers. Escrow centers are repositories for private keys, while certification authorities deal with public keys. Indeed, sound cryptographic practice dictates that users never reveal their private keys to anyone, even the certification authority.

KEYS SHOULD NOT BE REVEALABLE

The security of a modern cryptosystem rests entirely on the secrecy of the keys. Accordingly, it is a major principle of system design that to the extent possible, secret keys should never leave their user's secure environment. Key escrow implies that keys must be disclosed in some fashion, a flat-out contradiction of this principle. Any such disclosure weakens the total security of the system.

DATA RECOVERY

Sometimes escrow systems are touted as being good for the customer because they allow data recovery in the case of lost keys. However, it should be up to the customer to decide whether they would prefer the more secure system in which lost keys mean lost data, or one in which keys are escrowed to be recovered when necessary. Similarly, keys used only for conversations (as opposed to file storage) need never be escrowed. And a system in which the secret key is stored by a government and not by the data owner is certainly not practical for data recovery.

SIGNATURE KEYS

Keys used for signatures and authentication must never be escrowed. Any third party with access to such keys could impersonate the legitimate owner, creating new opportunities for fraud and deceit. Indeed, a user who wished to repudiate a transaction could claim that his or her escrowed key was used, putting the onus on that party. If a government escrowed the keys, a defendant could claim that the evidence had been forged by the government, thereby making prosecution much more difficult. For electronic commerce, non-repudiation is one of the most important uses for cryptography; and non-repudiation depends on the assumption that only the user has access to the private key.

PROTECTION OF THE EXISTING INFRASTRUCTURE

In some cases, it is technically feasible to use cryptographic operations that do not involve secrecy. While this may suffice in some cases, much of the existing technical and commercial infrastructure cannot be protected in this way. For example, conventional passwords, credit card numbers, and the like must be protected by strong encryption, even though some day more sophisticated techniques may replace them. Encryption can be added on quite easily; wholesale changes to diverse systems cannot.

CONFLICTING INTERNATIONAL POLICIES

Conflicting restrictions on encryption often force an international company to use a weak encryption system, in order to satisfy legal requirements in two or more different countries. Ironically, in such cases either nation might consider the other an adversary against whom commercial enterprises should use strong cryptography. Clearly, key escrow is not a suitable compromise, since neither country would want to disclose keys to the other.

MULTIPLE ENCRYPTION

Even if escrowed encryption schemes are used, there is nothing to prevent someone from using another encryption scheme first. Certainly, any serious malefactors would do this; the outer encryption layer, which would use an escrowed scheme, would be used to divert suspicion.

ESCROW OF PRIVATE KEYS WON'T NECESSARILY ALLOW DATA DECRYPTION

A major threat to users of cryptographic systems is the theft of long-term keys (perhaps by a hacker), either before or after a sensitive conversation. To counter this threat, schemes with "perfect forward secrecy" are often employed. If PFS is used, the attacker must be in control of the machine during the actual conversation. But PFS is generally incompatible with schemes involving escrow of private keys. (This is an oversimplification, but a full analysis would be too lengthy for this document.)

CONCLUSIONS

As more and more companies connect to the Internet, and as more and more commerce takes place there, security is becoming more and more critical. Cryptography is the most powerful single tool that users can use to secure the Internet. Knowingly making that tool weaker threatens their ability to do so, and has no proven benefit.

The Internet Architecture Board is described at http://www.iab.org/iab

The Internet Engineering Task Force and the Internet Engineering Steering Group are described at http://www.ietf.org

Parte 3

Introduction to Cryptography

Why is Encryption important?

Do you know who may be reading your E-Mail? It is transmitted in plain text over unknown pathways and resides for various periods of time on computer files over which you have no control. Whether you're planning a political campaign, discussing your finances, having an affair, completing a business deal, or engaging in some totally innocuous activity, your messages have less privacy than if you sent all of your written correspondence on postcards.

Why? Because of the nature of the Internet and the electronic medium. It allows effective scanning of message contents using sophisticated filtering software. Electronic mail is gradually replacing conventional paper mail and messages can be easily and automatically intercepted and scanned for interesting keywords.

Another problem with E-Mail is that it is very easy to forge the identity of the sender.

The solution to these problems is to use cryptography. However, there are restrictions on the export and use of strong cryptography, particularly in the USA, but now gaining momentum in other countries. Furthermore, some governments, and again the USA is the most prominant, want decryption keys lodged with escrow agents, so that law enforcement agencies can, with the appropriate authorisation, intercept and decrypt private messages. It is often claimed that this facility is no different from powers that the government has always possessed to wiretap telephones. There is however, a vital difference. Citizens are now being asked to take action to make themselves available for surveillance.

Why should Encryption be used?

Cryptography today involves more than encryption and decryption of messages. It also provides mechanisms for authenticating documents using a digital signature, which binds a document to the possessor of a particular key, while a digital timestamp binds a document to its creation at a particular time. These are important functions which must take the place of equivalent manual authentication procedures as we move into the digital age. Cryptography also plays an important part in the developing field of digital cash and electronic funds transfer.

The major applications for encryption may then be summarised as:

•To protect privacy and confidentiality.
•To transmit secure information (e.g. credit card details)
•To provide authentication of the sender of a message. •To provide authentication of the time a message was sent.

How does it work?

Up until the mid 1970's cryptography was an arcane science practised largely by government and military security experts. That situation changed dramatically following the development of public key cryptography by Hellman and Diffie in 1975. This development solved a major problem with most cryptographic systems - that of exchanging keys, and preceded a rapid escalation in civil involvement in this field of endeavour.

Public key cryptography systems work with public and secret (or private) keys. You generate these yourself as a once-only task. You distribute your public key to anyone who may need to send you encrypted information, or you can place it on one of the many public-key repositories around the globe. Your public key is then used by others to encrypt messages sent to you. Only you can decrypt such a message since the secret key is needed to perform this task. In practice, because public key encryption is a time-consuming process, many cryptosystems only use the public key to encrypt a random session key, which is then used to encrypt the actual message.

As an example, to exchange secure communications with someone the procedure would be as follows. Here we will introduce Alice and Bob, the renowned "first couple" of cryptography.

•Alice and Bob exchange their respective public keys or obtained them from a public key repository. •Alice transmits the message encrypted with the Bob's public key. •Bob decrypts it with his secret key. •Only Bob can decipher the message. Even Alice will be unable to decipher the message once encrypted, unless she has included herself as a recipient (using multiple keys to encrypt the message). •Cryptography can also be used to produce a digital signature which proves that the transmission is unchanged and can authenticate the sender. In this case the Bob would use Alice's public key to read the signature created by Alice's secret key.

BIBLIOGRAFÍA

[1] Ale R. Rafael, Cuellar M. Fernando, ”Teleinformática”, McGraw-Hill/Interamericana de España.

[2] González Sainz, Nestor, ”Comunicaciones y Redes de Procesamiento de Datos, McGraw-Hill/Interamericana de España.

[3] Tomasi Wayne, ”Sistemas de Comunicaciones Electrónicas”, Prentice Hall, 1996.

[4] Gibson Jerry, ”Principles of Digital and Analog Communications”, Second Edition, Macmillan.

[5] Dvorak John, ”Guide to PC Telecommunications”, Osborne Mc-Graw-Hill.

[6] Jordan L. y Churchill B., ”Communications and Networking for IBM PC and Compatibles, Third Edition, Brady.

[7] Hioki Warren, ”Telecommunications”, Second Edition, Prentice Hall 1995.

[8] Mendillo Vicenzo, Curso *Seguridad en Redes y Criptografía*, Universidad Central de Venezuela 1996, ftp://elecrisc.ing.ucv.ve.

[9] Losada Elsa y Fajardo Luis, Tesis de Grado: Software de Comunicaciones, para un Enlace vía HF con Computador PC 386/25, Universidad de Carabobo, Valencia, septiembre de 1992.

[10]"Internet Security: How much is enough?", Lee Bruno, Data Communications, April 1996.

[11]"Firewalls: Don´t Get Burned", Newman, Holzbaur & Bishop, Data Communications, March 1997.

[12] Hsu John Y.,Computer Networks:Architecture,protocols and software,Artech House 1996

[13] Tanenbaum Andrew S.,Redes de Ordenadores,Tercera Edición,Prentice Hall Hispanoamericana.

[14] Halsall Fred,Comunicación de datos,redes de computadores y sistemas abiertos.Cuarta Edición. Addison Wesley Iberoamericana.1998.

[15] "MPEG-4:multimedia for our time",**Koenen Rob,**IEEE Spectrum February 1999.

[16] Couch II Leon,Digital and Analog Communications Systems.Fourth Edition,MacMillan 1993.

[17] Clark G. & Cain J.,Error-Correction Coding for Digital Communications,Plenum Publishing Corporation,New York 1981.

[18] "Forward Error Correction Schemes for Digital Communications",**Bhargava V.**,IEEE Communications Magazine,vol.21,January 1983,pp.11-19.

[19] Spragins,Hammond & Pawlikowski,Telecommunications,Protocols and Design, Addison-Wesley 1992.

[20] "MP3 Sonido Digital al Alcance de Todos",**Rincón Rivera David,**Burata(Revista de la Rama Estudiantil IEEE de Barcelona España http://citel.upc.es/users/burata)Año 7,número 14.

[21] Criptografía: excelentes trabajos y enlaces en: http://www.criptored.upm.es