



Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería Mecánica Eléctrica

PRINCIPIOS PARA EL ANÁLISIS DE CRIPTOGRAFÍA DIGITAL

Luis Gustavo Méndez Aparicio

Asesorado por el Ing. Enrique Edmundo Ruiz Carballo

Guatemala, octubre de 2012

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA



FACULTAD DE INGENIERÍA

PRINCIPIOS PARA EL ANÁLISIS DE CRIPTOGRAFÍA DIGITAL

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA

POR

LUIS GUSTAVO MÉNDEZ APARICIO

ASESORADO POR EL ING. ENRIQUE EDMUNDO RUIZ CARBALLO

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO ELECTRÓNICO

GUATEMALA, OCTUBRE DE 2012

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA



NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paiz Recinos
VOCAL I	Ing. Alfredo Enrique Beber Aceituno
VOCAL II	Ing. Pedro Antonio Aguilar Polanco
VOCAL III	Inga. Elvia Miriam Ruballos Samayoa
VOCAL IV	Br. Juan Carlos Molina Jiménez
VOCAL V	Br. Mario Maldonado Muralles
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADOR	Ing. Julio Rolando Barrios Archila
EXAMINADOR	Ing. Carlos Eduardo Guzmán Salazar
EXAMINADORA	Inga. María Magdalena Puente Romero
SECRETARIA	Inga. Marcia Ivónne Véliz Vargas

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

PRINCIPIOS PARA EL ANÁLISIS DE CRIPTOGRAFÍA DIGITAL

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería Mecánica Eléctrica, con fecha 29 de mayo de 2009.



Luis Gustavo Méndez Aparicio

Guatemala, 21 de septiembre de 2011

Señor Coordinador del Área de Electrotécnica
Escuela de Ingeniería Mecánica Eléctrica
Facultad de Ingeniería
Universidad de San Carlos de Guatemala

Señor Coordinador

Por medio de la presente, me permito informarle que he revisado completamente el trabajo de graduación titulado "PRINCIPIOS PARA EL ANALISIS DE CRIPTOGRAFIA DIGITAL", desarrollado por el señor Luis Gustavo Méndez Aparicio, dicho trabajo cumple con los objetivos propuestos en el anteproyecto de tesis.

Por lo tanto, el autor de este trabajo y yo, como su asesor, nos hacemos responsables por el contenido y conclusiones de la misma.

Atentamente,


In g. Enrique Edmundo Ruiz Carballo

Asesor Nombrado

Colegiado No. 2225

ENRIQUE E. RUIZ C.
INGENIERO ELECTRICISTA
C.O.L. No. 2225



Ref. EIME 77. 2011
Guatemala, 28 de OCTUBRE 2011.

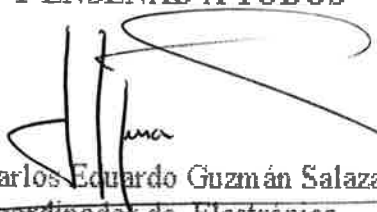
Señor Director
Ing. Guillermo Antonio Puente Romero
Escuela de Ingeniería Mecánica Eléctrica
Facultad de Ingeniería, USAC.

Señor Director:

Me permito dar aprobación al trabajo de Graduación titulado:
"PRINCIPIOS PARA EL ANÁLISIS DE CRIPTOGRAFÍA
DIGITAL", del estudiante Luis Gustavo Méndez Aparicio, que
cumple con los requisitos establecidos para tal fin.

Sin otro particular, aprovecho la oportunidad para saludarle.

Atentamente,
ID Y ENSEÑAD A TODOS


Ing. Carlos Eduardo Guzmán Salazar
Coordinador de Electrónica

CEGS /sro





FACULTAD DE INGENIERIA

Ref. EIME 132. 2009.
29 de mayo 2009.

Estudiante
Luis Gustavo Mèndez Aparicio
Carnè 2001-13240
Presente.

Estimado señor Mèndez:

Me permito hacer de su conocimiento que, la Direcciòn de Escuela ha aprobado su protocolo de tesis titulado: **PRINCIPIOS PARA EL ANÁLISIS DE CRIPTOGRAFÍA DIGITAL**, Se aprobò que este trabajo sea asesorado por el **Ing. Enrique Ruiz Carballo**.

Sin otro particular, aprovecho la oportunidad para saludarle.

Atentamente,

ID Y ENSEÑAD A TODOS

Ing. Mario Renato Escobedo Martinez
Director

Escuela de Ingeniería Mecánica Eléctrica

MREM/ sro
cc. archivo



DTG. 513.2012

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería Mecánica Eléctrica, al Trabajo de Graduación titulado: **PRINCIPIOS PARA EL ANÁLISIS DE CRIPTOGRAFÍA DIGITAL**, presentado por el estudiante universitario **Luis Gustavo Méndez Aparicio**, autoriza la impresión del mismo.

IMPRÍMASE:

Ing. Murphy Olympo Paiz Recinos
Decano

Guatemala, 18 de octubre de 2012

/gdech



AGRADECIMIENTOS A:

- Dios** La fuerza vital que ha estado conmigo en todo momento y me guió por el camino del entendimiento para encontrar el conocimiento y la felicidad.
- Mis padres** Carlos Adolfo Méndez Alonso y Silvia Leonor Aparicio Jordán, quienes me apoyaron y dieron el aliento necesario para lograr mis objetivos.
- Mi abuela** Antonia Alonzo Ochoa, quien me brindo apoyo y consejos ayudándome a lograr mis objetivos.
- Mis compañeros y amigos** Quienes hicieron el camino en la universidad una experiencia agradable y única.
- Facultad de Ingeniería** Por haberme brindado la oportunidad de estudiar una carrera universitaria.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES	V
GLOSARIO	VII
RESUMEN.....	XI
OBJETIVOS.....	XIII
INTRODUCCIÓN	XV
1. PRINCIPIOS BÁSICOS	1
1.1. Reseña histórica.....	1
1.1.1. Escitalo Espartano.....	1
1.1.2. Criptografía por reordenamiento.....	2
1.1.3. Criptosistema de César	3
1.1.4. Criptosistema de Vigenere.....	5
1.1.5. La máquina Enigma.....	7
1.1.6. La máquina Sigaba.....	9
1.1.7. Cifrado de Lorenz	11
1.2. Reglas básicas del criptoanálisis.....	14
1.2.1. Reglas de Kerck-hoffs.....	14
1.2.2. Medida de la información de Shanon	18
1.2.3. Ataque criptoanalítico	23
1.3. Herramientas del criptoanálisis.....	25
1.3.1. Estadísticas en el criptoanálisis	25
1.3.2. Aritmética modular	29
1.3.3. Algoritmo de Euclides.....	31
1.3.4. Complejidad de las operaciones aritméticas en Z_n	32

	1.3.4.1.	Cálculo de inversas en Z_n	33
	1.3.5.	Función de Euler	34
	1.3.6.	Teorema del Chino Resto	36
	1.3.7.	Exponenciación	37
	1.3.7.1.	Algoritmo rápido de exponenciación...	38
	1.3.7.2.	El problema de los logaritmos discretos	38
	1.3.7.3.	El problema de Diffie-Hellman	39
	1.3.7.4.	Importancia de los números primos....	40
1.4.		Criptografía y números aleatorios	42
	1.4.1.	Tipos de secuencias aleatorias	42
	1.4.1.1.	Secuencias estadísticamente aleatorias	43
	1.4.1.2.	Secuencias criptográficamente aleatorias	44
	1.4.1.3.	Secuencias totalmente aleatorias.....	45
	1.4.1.4.	Utilidad de las secuencias aleatorias en criptografía	45
	1.4.2.	Generadores aleatorios criptográficamente seguros.....	46
	1.4.2.1.	Generador X9.17	47
	1.4.2.2.	Generador Blum Blum Shub.....	48
2.		SISTEMAS SECRETOS.....	49
	2.1.	Alfabetos	49
	2.2.	<i>Plain text source</i> (fuente de texto plano)	51
	2.3.	Sistemas criptográficos	66
	2.4.	Claves débiles.....	68
	2.5.	Propiedades de los sistemas de información	74

2.6.	Problemas de los sistemas de información	76
2.7.	Tipos de vulnerabilidades	78
2.8.	Criptoanálisis por el oponente bayesiano	80
2.9.	Secretividad perfecta	89
2.10.	Sistemas criptográficos aleatorios.....	95
3.	APLICACIONES	101
3.1.	Seguridad en archivos y comunicación	101
3.2.	Administración de la clave	106
3.2.1.	Aspectos generales	106
3.2.2.	Claves de sesión	111
3.3.	Sistemas de clave pública	114
3.4.	Firmas digitales y autenticación.....	120
3.5.	Curvas elípticas	126
3.6.	Funciones Hash.....	129
3.7.	Timestamping	137
	CONCLUSIONES	141
	RECOMENDACIONES	143
	BIBLIOGRAFÍA	145
	APÉNDICE.....	147
	ANEXO	153

ÍNDICE DE ILUSTRACIONES

FIGURAS

1.	Escítalo Espartano	2
2.	Máquina Enigma	8
3.	Máquina Sigaba	10
4.	Máquina SZ40.....	12
5.	Codificación binaria cifrado de Lorenz	13
6.	Gráfica H(x) medida de la información de Shanon.....	20
7.	Secuencias aleatorias en criptografía	45
8.	Gráfica de ocurrencias en ejemplo plain text source.....	58
9.	Diagrama de transmisión en claves débiles	73
10.	Diagrama de flujo en criptoanálisis por oponente bayesiano	81
11.	Diagrama encriptador DES	113
12.	Diagrama desencriptador DES.....	114
13.	Diagrama de flujo firmas digitales	121
14.	Diagrama mensaje estándar utilizando autenticación	124

TABLAS

I.	Guía para criptosistema de César.....	4
II.	Guía para criptosistema Vigenere.....	6
III.	Alfabetos	51
IV.	Probabilidades ejemplo 1	56
V.	Clasificación de las letras por probabilidad de ocurrencia	59

GLOSARIO

Algoritmo	Conjunto ordenado y finito de operaciones que permite hallar la solución de un problema.
Anagrama	Transposición de las letras de una palabra o sentencia, de la que resulta otra palabra o sentencia distinta.
Asimetría	Se refiere a la propiedad de determinados cuerpos, funciones matemáticas y otros elementos a los cuales al aplicarles una regla de transformación efectiva, se observan cambios respecto al elemento original.
Autenticación	Acto de establecimiento o confirmación de algo como autentico, es decir, la confirmación de la procedencia de un objeto.
Bit	Es un dígito del sistema de numeración binario.
Byte	Es una secuencia de bits contiguos, cuyo tamaño depende del código de información en que sea definido.

Criptoanálisis	Estudio de los métodos para obtener el sentido de una información cifrada, sin acceso a la información secreta requerida para obtener este sentido normalmente.
Criptografía	Técnica que altera las representaciones lingüísticas de un mensaje.
Ecuación	Es una igualdad matemática entre dos expresiones algebraicas.
Informática	Es la ciencia aplicada que abarca el estudio y aplicación del tratamiento automático de la información utilizando sistemas computacionales.
Logaritmos	En matemática la base determinada es el exponente al cual hay que elevar la base para obtener dicho número.
Permutación	Dado un conjunto finito se llama permutación a cada una de las posibles ordenaciones de todos los elementos de dicho conjunto.
Tecnología	Conjunto de conocimientos técnicos, ordenados científicamente, que permiten diseñar y crear bienes y servicios.

Transposición

Proceso por el que un saber sabio o saber científico se convierte en objeto de enseñanza.

Vectores

Herramienta geométrica que se utiliza para representar una magnitud física definida por un módulo y una dirección.

RESUMEN

En los procesos de almacenamiento y transmisión de la información normalmente aparece el problema de la seguridad. En el almacenamiento, el peligro lo representa el robo del mensaje o simplemente acceso no autorizado a la información de dicho mensaje, mientras que en las transmisiones el riesgo consiste en la intervención del canal. Para proteger dicha información se varía su forma, a este proceso se le llama cifrado, análogamente se le llama descifrado al proceso de recuperar el texto original.

El objetivo general de este trabajo de graduación es: describir los diversos métodos existentes para la encriptación de datos. Se logró haciendo desde una reseña histórica mostrando los diversos métodos utilizados en la antigüedad hasta los métodos logarítmicos que se utilizan en la era de la información, haciendo uso de los sistemas digitales de almacenamiento y transmisión de datos. Así como sus distintas aplicaciones y reglas para encriptar y desencriptar mensajes.

OBJETIVOS

General

Describir diversos métodos existentes para la encriptación de datos.

Específicos

1. Determinar cuáles son los métodos que se aplican mejor a la criptografía digital.
2. Describir las aplicaciones digitales básicas y determinar si hay un método que se ajuste mejor a las necesidades de la paliación.

INTRODUCCIÓN

Del antiguo Egipto a la era digital, los mensajes cifrados han jugado un papel destacado en la historia. Arma de militares, diplomáticos y espías, son la mejor defensa de las comunicaciones y datos que viajan por Internet. Los criptogramas han protagonizado buena parte de los grandes episodios históricos y un sinnúmero de anécdotas. Los espartanos utilizaron en el 400 a.C., la cítala, que puede considerarse como el primer sistema de criptografía por transposición, es decir, que se caracteriza por enmascarar el significado real de un texto alterando el orden de los signos que lo conforman.

Los militares de la ciudad griega, escribían sus mensajes sobre una tela que envolvía una vara, el mensaje sólo podía leerse cuando se enrollaba sobre un bastón del mismo grosor, que poseía el destinatario lícito. El método de la cítala era extremadamente sencillo, como también lo era el que instituyó Julio César, basado en la sustitución de cada letra por la que ocupa tres puestos más allá en el alfabeto.

La criptografía resurgió en la Europa de la edad media, impulsada por las intrigas del papado y las ciudades-estado italianas. Fue un servidor del papa Clemente VII, Gabriele de Lavinde, quien escribió el primer manual sobre la materia en el viejo continente. El siglo XX ha revolucionado la criptografía. Retomando el concepto de las ruedas concéntricas de Alberti, a principios de la centuria se diseñaron teletipos equipados con una secuencia de rotores móviles. Estos giraban con cada tecla que se pulsaba. De esta forma, en lugar de la letra elegida, aparecía un signo escogido por la máquina según diferentes reglas en un código polialfabético complejo.

Estos aparatos, se llamaron traductores mecánicos. Una de sus predecesoras fue la rueda de Jefferson, el aparato mecánico criptográfico más antiguo que se conserva. La primera patente data de 1919 y es obra del holandés Alexander Koch, que comparte honores con el alemán Arthur Scherbius, el inventor de Enigma una máquina criptográfica que los nazis creyeron inviolable, sin saber que a partir de 1942, propiciaría su derrota. En efecto, en el desenlace de la contienda, hubo un factor decisivo y apenas conocido: los aliados eran capaces de descifrar todos los mensajes secretos alemanes.

Hoy por hoy, se utilizan métodos que combinan dígitos del mensaje con otros o bien algoritmos de gran complejidad. Un computador tardaría 200 millones de años en interpretar las claves más largas, de 128 bits. En palabras de un apasionado de la criptografía, Edgar Allan Poe, “Es dudoso que el género humano logre crear un enigma que el mismo género humano no resuelva”. En la actualidad en muchas de las actividades cotidianas del ser humano tales como la compra de insumos, el pago de servicios, inscripción en colegios, etcétera.

Se requiere que se transmitan datos a través de la Internet o en una red local, debido a que la información viaja de un punto A hacia un punto B y puede ser vista o copiada por terceras personas, se requiere que esta información lleve cierto tipo de protección de manera que aunque un tercero la obtenga este no sea capaz de descifrar qué es lo que se dice en el mensaje, esto se consigue encriptado el mensaje de manera que el código que recibe en el punto B se sepa que va de A y qué es lo que dice para que las transacciones puedan llevarse a cabo exitosamente, a continuación se hace un análisis de los principios básicos para llevar a cabo esta encriptación.

1. PRINCIPIOS BÁSICOS

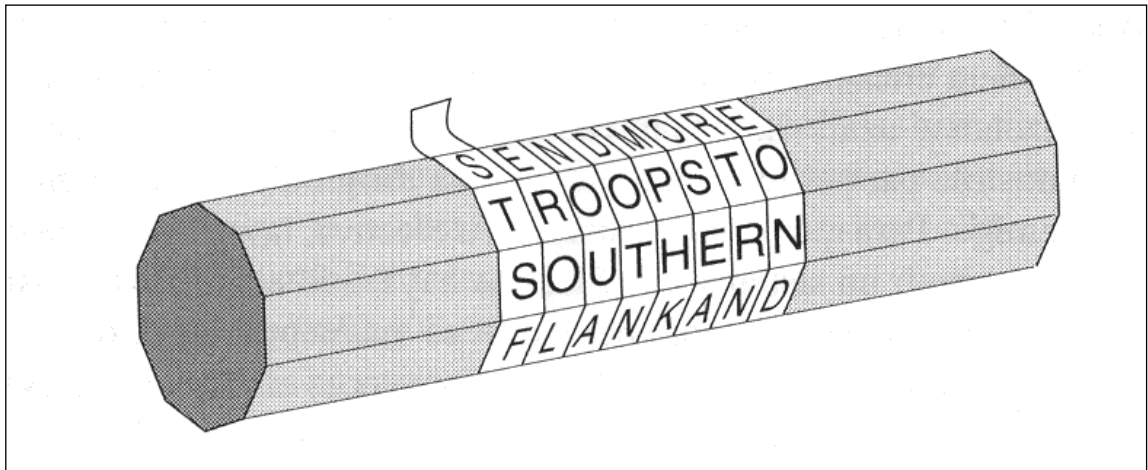
1.1. Reseña histórica

A continuación se describirán algunos de los métodos que se han utilizado desde la antigüedad y que marcaron un desarrollo dentro del campo de la criptografía, explicando sus bases principales y situaciones en las cuales fueron utilizados.

1.1.1. Escitalo Espartano

Es el primer aparato criptográfico de la historia, el Escitalo Espartano, que se remonta al siglo V a.C. El escitalo es una vara de madera sobre la que se enrolla una tira de cuero o un pergamino, tal como se muestra bajo estas líneas, el emisor escribe un mensaje a lo largo de la longitud del escitalo y luego desenrolla la tira, que ahora parece llevar una lista de letras sin sentido, el mensaje ha sido codificado. El mensajero llevaba la tira de cuero, para recuperar el mensaje, el receptor simplemente enrolla la tira de cuero en torno a un escitalo del mismo diámetro que el usado por el emisor.

Figura 1. **Escítalo Espartano**



Fuente: <http://personal.telefonica.terra.es/web/jms32/Cifra/CodSecretos/Cap01/ImgEscan/01-03.gif>. Consulta: febrero de 2012.

1.1.2. **Criptografía por reordenamiento**

En la criptografía por transposición las letras simplemente se colocan en un orden diferente generando así un anagrama. Para mensajes muy cortos este método es inseguro ya que el número de combinaciones posibles es reducido y puede ser fácilmente roto por un ataque de fuerza bruta, pero a medida que el número de letras se va incrementando el número de combinaciones posibles se dispara, haciendo que sea virtualmente imposible decodificar el mensaje original a menos que se conozca que método se utilizó exactamente.

Por ejemplo en una frase de 35 letras existen más de 50 000 000 000 000 000 000 000 000 000 000 000 combinaciones posibles haciendo que un ataque de fuerza bruta tome demasiado tiempo de efectuar, provocando que sea muy segura la encriptación.

Pero hay un inconveniente; la transposición genera eficazmente un anagrama increíblemente difícil y si las letras se mezclan al azar, sin pies ni cabeza, la decodificación del anagrama es tan imposible para el recipiente a quien va dirigido como para un interceptor enemigo. Para que la transposición sea efectiva, la combinación de letras necesita seguir un sistema sencillo, que haya sido acordado previamente por el emisor y el receptor, pero que se mantenga secreto frente al enemigo.

1.1.3. Criptosistema de César

Este sistema recibe su nombre en honor a Julio César, quien lo utilizaba para comunicarse con sus generales, el método consiste en primero numerar las letras del alfabeto, por ejemplo, la letra A tiene asignado el 0, la letra B el 1, ..., la letra Z el 25. Luego se define la clave k como un número entre 0 y 25, para encriptar un mensaje lo que se hace es sumarle a cada letra, la clave k y reducir módulo 26. Para desencriptar el mensaje se debe restar k a cada letra y reducir módulo 26. Por ejemplo, se puede encriptar el mensaje:

El ataque será mañana

De la siguiente manera:

Según la definición de $A=0, B=1, \dots, Z=26$; con $k=10$ y módulo = 27

Tabla I. **Guía para criptosistema de César**

E	L	A	T	A	Q	U	E	S	E	R	A	M	A	Ñ	A	N	A
4	1	0	2	0	1	2	4	1	4	1	0	1	0	1	0	1	0
	1		0		7	1		9		8		2		4		3	
1	2	1	3	1	0	4	1	2	1	1	1	2	1	2	1	2	1
3	1	0		0			3		3		0	2	0	4	0	3	0
N	U	K	D	K	A	E	N	C	N	B	K	V	K	X	K	W	K

Fuente: G. KONHEIM, Alan. Cryptography: A Primer.

Lo cual daría como resultado el siguiente texto:

nu kdkaen cnbk vkxkww

Para evitar que se note cada fin de palabra se puede asignar un número al carácter espacio y ahora cifrar como si se tuviera un alfabeto de módulo 28, pero aún con esto este método puede ser fácilmente roto por un ataque de fuerza bruta, ya que únicamente requiere de la prueba de 28 diferentes claves y ver cual tiene sentido.

La función de encriptado de este sistema se puede definir así:

$$E_{Z_n} \text{---} Z_n E x = x+k \pmod{n}$$

(Ecuación 1)

Donde k es la clave utilizada y x la letra a encriptar.

La función de descryptación de este sistema se puede definir así:

$$DZ_i = Z_i E x = x - k \pmod{n}$$

(Ecuación 2)

1.1.4. Criptosistema de Vigenere

Este criptosistema es una mejora del criptosistema de César, aquí en lugar de sustituir en el texto letra por letra se sustituye bloque por bloque. Acá la clave corresponde a una palabra, la cual se utiliza para cifrar en forma similar al método César, la palabra se escribe las veces que sea necesario bajo el texto a cifrar y se suma letra por letra realizando como si fuera un criptado de César, el resultado es un texto más difícil de decodificar ya que ahora la clave es variante para cada posición del mensaje.

Ejemplo.

Utilizando el mismo texto anterior el ataque será mañana y de palabra clave la palabra prueba, asignando al carácter espacio el número 27, el módulo es 28, por lo que se tendría el siguiente resultado:

Tabla II. **Guía para criptosistema Vigenere**

E	L		A	T	A	Q	U	E		S	E	R	A		M	A	Ñ	A	N	A
4	1	2	0	2	0	1	2	4	2	1	4	1	0	2	1	0	1	0	1	0
	1	7		0		7	1		7	9		8		7	2		4		3	
P	R	U	E	B	A	P	R	U	E	B	A	P	R	U	E	B	A	P	R	U
1	1	2	4	1	0	1	1	2	4	1	0	1	1	2	4	1	0	1	1	2
6	8	1				6	8	1				6	8	1				6	8	1
2	1	2	4	2	0	5	1	2	3	2	4	6	1	2	1	1	1	1	3	2
0		0		1			1	5		0			8	0	6		4	6		1
T	B	T	E	U	A	F	L	Y	D	T	E	G	R	T	P	B	Ñ	P	D	U

Fuente: G. KONHEIM, Alan. Cryptography: A Primer.

La palabra cifrada sería TBTEUAFLYDTEGRTPBÑPDU, para descifrar este texto simplemente se repite la palabra clave bajo el texto cifrado pero en esta ocasión en lugar de sumar se resta y siempre se opera módulo 28.

Una mejora sobre el cifrado Vigenere fue introducida por el sistema de Vernam, utilizando una clave aleatoria de longitud igual a la del mensaje; la confianza en este nuevo criptosistema hizo que se utilizase en las comunicaciones confidenciales entre la Casa Blanca y el Kremlin, hasta, por lo menos 1987.

1.1.5. La máquina Enigma

En 1923, un ingeniero alemán llamado Arthur Scherbius patentó una máquina diseñada para facilitar las comunicaciones seguras. Se trataba de un instrumento de apariencia simple, parecida a una máquina de escribir. Quien deseará codificar un mensaje sólo tenía que teclearlo y las letras correspondientes al texto cifrado se irían iluminando en un panel.

El destinatario copiaba dichas letras en su propia máquina y el mensaje original aparecía de nuevo. La clave la constituían las posiciones iniciales de tres tambores o rotores. En la figura se puede apreciar un esquema de esta máquina, llamada Enigma. Los rotores no son más que tambores con contactos en su superficie y cableados en su interior, de forma que con cada pulsación del teclado, la posición de estos determina cuál es la letra que se ha de iluminar.

Cada vez que se pulsa una tecla el primer rotor avanza una posición; el segundo avanza cuando el anterior ha dado una vuelta completa y así sucesivamente. El reflector no existía en los primeros modelos, se introdujo posteriormente para permitir que la misma máquina sirviera tanto para cifrar como para descifrar.

Figura 2. **Máquina Enigma**



Fuente: <http://www.temakel.com/temakel/files/images/menigma.jpg>.

Consulta: febrero de 2012.

Se observa que un rotor no es más que una permutación dentro del alfabeto de entrada. El cableado hace que cada una de las letras se haga corresponder con otra. Todas las letras tienen imagen y no hay dos letras con la misma imagen. Si se nota una permutación como π , se puede escribir que la permutación resultante de combinar todos los rotores en un instante dado es:

$$\pi_{total} = \langle \pi_0, \pi_1, \pi_2, \pi_3, \pi_2^{-1}, \pi_1^{-1}, \pi_0^{-1} \rangle$$

(Ecuación 3)

La permutación π_3 corresponde al reflector y debe cumplir que $\pi_3 = \pi_3^{-1}$, es decir, que aplicada dos veces dé lo mismo que tenía al principio. De esta forma se cumple la propiedad de que, para una misma posición de los rotores, la codificación y la decodificación son simétricas.

La fuerza de la máquina Enigma radica en que tras codificar cada letra se giran los rotores, lo cual hace que la permutación que se aplica a cada letra sea diferente. La máquina, por tanto, es un sistema de cifrado de sustitución polialfabética. Además, cada sustitución concreta no se repite hasta que los rotores recuperan su posición inicial, lo que da lugar a un tamaño de ciclo realmente grande.

Se debe tener en cuenta que hay 17 576 posiciones iniciales de los rotores y 60 combinaciones de tres rotores a partir de los cinco de entre los que se puede elegir. Puesto que el *stecker* presenta en torno a cien mil millones de combinaciones, cantidad enorme de posibles disposiciones iniciales de la máquina aproximadamente 10^{17} .

La potencia del método de criptoanálisis empleado radica en que se podía identificar un emparejamiento válido entre el criptograma y el texto claro e ignorar la posición del *stecker*, de forma que sólo bastaba con rastrear dentro del espacio de posibles configuraciones, para encontrar aquella que llevará a cabo la transformación esperada. No disponer de dicho emparejamiento hubiera complicado enormemente el criptoanálisis, tal vez hasta el punto de hacerlo fracasar.

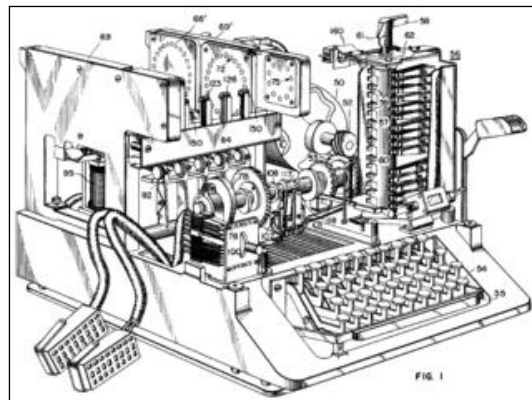
1.1.6. La máquina Sigaba

Esta máquina al igual que la Enigma basada en rotores, es también conocida como ECM Mark II, Converter M-134 y CSP-889, fue empleada por el ejército de los EE.UU. durante la Segunda Guerra Mundial. A diferencia de la máquina Enigma, en la que los rotores avanzan una posición cada vez que se pulsa una tecla, Sigaba incorpora un segundo juego de rotores, que se encarga de decidir qué rotores principales avanzan cada vez que se pulsa una tecla.

Esto aumenta considerablemente la longitud de ciclo de la máquina y complica la localización de posibles patrones en los textos cifrados. El principal inconveniente de esta máquina era su excesivo peso y tamaño, sin contar con su complejidad mecánica, dificultad de manejo y fragilidad. Esto supuso que, en la práctica, no pudiera ser utilizada en muchas situaciones a lo largo de la guerra, a diferencia de la máquina Enigma, mucho más ligera y resistente.

En su lugar, se usaba, entre otros, el famoso código que consistía en emplear indios navajos, que simplemente se comunicaban por radio en su propio idioma, demasiado desconocido y complejo como para ser comprendido por el enemigo.

Figura 3. **Máquina Sigaba**



Fuente: <http://upload.wikimedia.org/wikipedia/commons/thumb/f/fb/SIGABA-patent.png/320px-SIGABA-patent.png>. Consulta: febrero de 2012

1.1.7. Cifrado de Lorenz

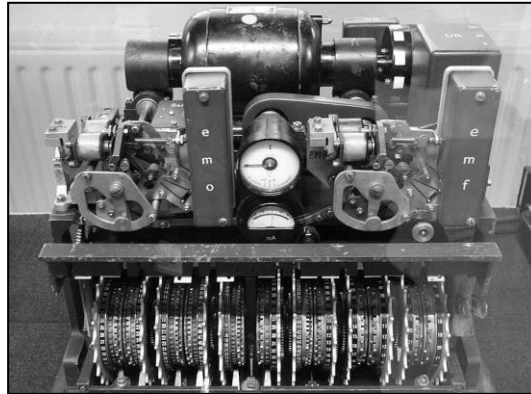
El cifrado de Lorenz se llevaba a cabo mediante una máquina, denominada SZ40, que tenía un diseño considerablemente más complejo que el de la Enigma. Para su descifrado se desarrolló una máquina descifradora, denominada Colossus, que supuso el inicio de las computadoras tal y como hoy se conocen.

Si bien la bomba que se utilizaba para descifrar Enigma efectuaba en esencia una búsqueda sistemática de la clave, el cifrado de Lorenz necesitaba de análisis estadísticos más complejos y para ello, el matemático Max Newman, inspirándose en el concepto de máquina universal de Turing, dirigió el desarrollo de Colossus, que podría muy bien ser considerada como la primera computadora moderna, aunque su existencia se mantuvo en secreto hasta mediados de los 70.

Su construcción fue llevada a cabo por el ingeniero Tommy Flowers, esta constaba de unas 1 500 válvulas de vacío, tecnología mucho más moderna que los relés que constituían el corazón de las bombas. En el cifrado de Lorenz se empleaba teletipos para codificar, en los teletipos los textos venían representados por una matriz formada por columnas de cinco puntos. Cada una de esas columnas correspondía a una letra y en cada punto podía haber (o no) un agujero.

En esencia, se tiene un sistema de codificación de cinco bits por letra. La máquina de Lorenz generaba una secuencia pseudoaleatoria binaria que era combinada con la matriz de puntos, mediante una operación or-exclusivo para producir el criptograma.

Figura 4. **Máquina SZ40**



Fuente: http://www.eyeintheskygroup.com/Azar-Ciencia/Tragamonedas-Maquinas-de-Casino/Ingenieria-Inversa-Capacidad-de-Computo-Descifrar-Randomizer-RNG_archivos/Lorenz-SZ42.jpg.
Consulta: febrero de 2012.

La máquina SZ40 pretendía emular un sistema seguro de Shannon, pero para ello, las secuencias generadas tendrían que ser totalmente aleatorias.

Sin embargo, si las secuencias producidas por la máquina fueran de este tipo, sería imposible reproducirlas en los dos extremos de la comunicación, por lo que el sistema en realidad es una técnica de cifrado de flujo.

Si uno dispone de dos mensajes con sentido en un idioma determinado, cifrados con la misma secuencia pseudoaleatoria, bastará con buscar cadenas de bits que permitan descifrar simultáneamente ambos mensajes. Por ejemplo, suponiendo la siguiente codificación binaria para cada letra:

Figura 5. **Codificación binaria cifrado de Lorenz**

a	b	c	d	e	f	g	h	i	j	k	l	m
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	1	1
0	0	0	0	1	1	1	1	0	0	0	0	1
0	0	1	1	0	0	1	1	0	0	1	1	0
0	1	0	1	0	1	0	1	0	1	0	1	0
n	o	p	q	r	s	t	u	v	w	x	y	z
0	0	0	1	1	1	1	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0	0	0	1	1
1	1	1	0	0	0	0	1	1	1	1	0	0
0	1	1	0	0	1	1	0	0	1	1	0	0
1	0	1	0	1	0	1	0	1	0	1	0	1

Si se encuentran los mensajes

10100 11000 11000 10101
10000 11000 11001 11011

Fuente: www.kryptopolis.org. Consulta: junio de 2009.

Se pueden generar las 1 024 combinaciones posibles de 10 bits y tratar de descifrar las dos primeras letras de cada mensaje. Se quedará únicamente con aquellas combinaciones de bits que den lugar a sílabas (o partes de sílabas) legales en castellano en ambos mensajes.

Por ejemplo, la cadena 1010101010 da lugar a las letras BS para el primer mensaje y FS para el segundo, ambas con muy poca probabilidad de aparecer al principio de un mensaje correcto en castellano.

Si, por el contrario, se contara con un único mensaje cifrado, este análisis resultaría imposible, ya que para todos y cada uno de los mensajes posibles en castellano existirá una secuencia de bits que lo genera. La clave está en que existirán muy pocas secuencias tal vez sólo una que den lugar en ambos mensajes cifrados a textos claros válidos. La tarea es tediosa, pero da resultado, incluso si la secuencia empleada para cifrar es totalmente aleatoria.

En consecuencia, los fallos sobre los que se cimentó el éxito del criptoanálisis del cifrado de Lorenz fueron dos: en primer lugar, el cifrado accidental de dos mensajes distintos con la misma secuencia y en segundo, el carácter poco aleatorio de la secuencia en cuestión.

1.2. Reglas básicas del criptoanálisis

A continuación se describirán algunas de las reglas que rigen el campo de la criptografía, así como los principios sobre los que se basan dichas leyes y algunos ejemplos sobre cómo y cuándo aplicarlas para efectuar un análisis de una mejor manera.

1.2.1. Reglas de Kerck-hoffs

La criptología se puede definir como la ciencia de esconder o codificar. Y está comprendida en el desarrollo de métodos para encriptar mensajes y señales, la criptología se puede dividir en 2 ramas muy relacionadas entre sí: la criptografía y el criptoanálisis, la criptografía se puede ver como una competencia entre el diseñador del sistema criptográfico y su contraparte que intenta descifrar los mensajes u obtener la clave para el sistema criptográfico, a el proceso de decrepitación se le llama criptoanálisis en el siglo XIX Kerck-hoffs en su libro “La criptografía Digital”, definió las reglas básicas para el criptoanálisis, siendo las siguientes:

- K1: el sistema debe ser, si no teóricamente irrompible, irrompible en la práctica.
- K2: el compromiso del sistema no debe incomodar a los correspondientes.

- K3: el método para elegir la llave particular del sistema criptográfico que se utilice debe ser fácil de memorizar y modificar.
- K4: el texto encriptado debe ser transmitido por telégrafo.
- K5: el aparato debe ser portátil.
- K6: el uso del sistema no debe requerir una larga lista de reglas o tensión mental.

La reglamentación de Kerck-hoffs fue formulada en una era de baja velocidad y en la cual la forma típica de comunicación era por medio del telégrafo, actualmente, los métodos criptográficos se llevan en la vanguardia de la tecnología.

Se aclaran los postulados:

- K1: hay algunos encriptamientos que se categorizan como irrompibles, diciendo que no existe una técnica de criptoanálisis que sea capaz de recuperar el texto encriptado o la clave de encriptación. Estos sistemas existen pero depende de la aplicación que se requiera y del tiempo que se quiera guardar el secreto, por ejemplo, una orden de ataque se requiere que sea secreta hasta después que sea iniciada la batalla o de otra manera un plan estratégico de mercadeo o planeación económica puede requerir protección por varios años.

Los sistemas criptográficos con K1 finita son utilizables dependiendo de cuánto tiempo se requiera para descifrarlos, algunos sistemas monoalfabéticos en los cuales se sustituían unas letras por otras que cumplieron su función en el siglo XIX ahora son considerados obsoletos, debido a la introducción del análisis digital en la era actual.

- K2: cualquier encriptamiento consistirá de 2 tipos de información; pública y privada. Por información pública se refiere a las reglas o algoritmos por medio de los cuales la información fue encriptada. Si el sistema fue encriptado por medio de hardware, las especificaciones serán publicadas y deberá tener manuales de servicio disponibles, al igual que en una implementación por medio de software, el código no puede ser mantenido en secreto indefinidamente.

Por compromiso del sistema Kerck-hoffs se refería al conocimiento de la información pública. La información privada es la que se presume que está inaccesible para el oponente (el criptoanalista), la llave o tabla es utilizada para encriptar el texto, los parámetros que se seleccionan de cualquier forma posible de encriptación.

- K3: no es poco usual que los usuarios escojan nombres como ALAN, LUIS, etcétera o que escojan frases como AHORA ES TIEMPO, para que sirvan como claves de acceso y de esta forma hacer fácil el recordarlas.

En varios de los sistemas criptográficos comerciales la clave será almacenada magnéticamente en tarjetas de crédito, pero en otros el usuario es requerido que memorice la clave o parte de ella. Este requerimiento es un importante factor de diseño ya que se debe tomar en consideración que se debe pedir algo corto de memorizar.

- K4: el cuarto requerimiento refleja la tecnología dominante del siglo XIX; se interpreta para significar o estar representado por cualquier formato de transmisión disponible así como formato de grabación. Se excluirán los métodos de *steganography* en los cuales se esconde la mera existencia del mensaje en un micropunto o con tinta invisible.
- K5: la portatibilidad del equipo es menos importante en el procesamiento de la información y aplicaciones que lo hacen.
- K6: son importantes la facilidad de uso, el costo del encriptado y su impacto en el desempeño del sistema donde se utiliza. El costo de proveer protección a los datos es generalmente transmitido a los usuarios, por lo que se necesita una clara razón para que los datos requieran ser encriptados y también el conocimiento de si el sistema de encriptación logrará o no el cumplimiento a totalidad de sus objetivos.

Criptoanálisis es el área de la criptología que abarca las diferentes técnicas para descifrar datos codificados sin conocimiento previo de qué tipo de clave ha sido utilizada en el proceso. Esto es comúnmente conocido como '*hacking*'.

Es evidente que tanto la criptografía como el criptoanálisis están relacionados de forma muy estrecha y alguien que desea diseñar un algoritmo criptográfico necesita tener conocimiento suficiente sobre las diferentes técnicas de criptoanálisis que un posible agresor utilizaría para descifrar lo que ha codificado.

1.2.2. Medida de la información de Shannon

En 1948 Claude E. Shannon estandarizó lo que actualmente se conoce como Teoría de la información. Un año después presentó un artículo conocido como teoría de comunicación de sistemas secretos, el cual es una contribución importante para la criptografía. La teoría de la información es la ciencia en la que se da el concepto de información y la medida de esta, la cual está relacionada con la medida de incertidumbre, hay varios métodos de representarlo.

Considerando la fuente de información X , que genera una serie de símbolos se asume que los símbolos fueron seleccionados de un alfabeto dado de la siguiente manera:

$$X = \{x_1, x_2, \dots, x_n\}$$

(Ecuación 4)

Para cada símbolo, la probabilidad de ocurrencia está definida por: la probabilidad correspondiente a un símbolo x_i es denotado como $p(x_i)$. De manera que la probabilidad de todos los elementos X , de la cual se genera el alfabeto sería:

$$H(X) = -\sum_{i=1}^n p_i \log_2 p_i$$

(Ecuación 5)

La cantidad de información o incertidumbre que compete a la fuente de información X, (la medida marginal de información), es definida por Shannon como:

$$H(X) = -\sum_{i=1}^n p_i \log_2 p_i$$

(Ecuación 6)

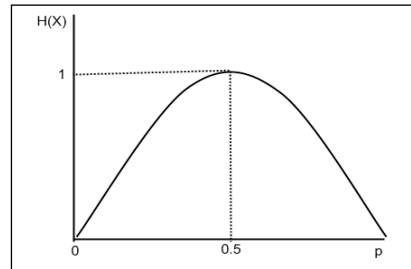
Si se considera que el alfabeto X sólo consiste de dos símbolos con $p(x_1)=p$ y $p(x_2)=1-p$, la fórmula anterior se reescribe como:

$$H(X) = -p \log_2 p - (1-p) \log_2 (1-p)$$

(Ecuación 7)

Graficando H(X) como función de p se tiene:

Figura 6. **Gráfica H(x) medida de la información de Shanon**



Fuente: G. KONHEIM, Alan. Cryptography: A Primer.

De la figura donde se grafica $H(x)$ como función de p se puede concluir que cuando la incertidumbre de un símbolo es cero, la probabilidad de ocurrencia es 1, por lo que la medida de información es cero. En el caso de $p=0$; ambos símbolos ocurren con probabilidad $=1$; y alcanza su valor máximo cuando $p=0,5$, $H(x)$ alcanza su valor máximo. En un alfabeto de 2 caracteres el logaritmo sería base 2, por lo que la probabilidad de ocurrencia de los símbolos sería la misma e igual a 0,5, de manera que la incertidumbre será igual siempre que ocurra uno de los 2 símbolos.

Para el caso general de un alfabeto de n símbolos la medida de información alcanza el máximo cuando la probabilidad de ocurrencia es igual para todos los símbolos y alcanza la mínima cuando la probabilidad de uno de los símbolos es 1 y se puede representar así: $0 \leq H(x) \leq \log_2 n$ donde los límites superior e inferior corresponden a los valores de medida de información.

La definición de medida de información relacionada a un par de símbolos (X_i, Y_j) , la información conjunta, corresponde a la medida marginal de información así:

$$H(X, Y) = - \sum_{i=1}^n \sum_{j=1}^m p_{(i, y_j)} \log p_{(i, y_j)}$$

(Ecuación 8)

La fórmula anterior describe el caso general, en el que debe haber cierto grado de dependencia entre ambas fuentes de información.

Ahora si no hay dependencia entre las fuentes o sea que la ocurrencia de un símbolo de la fuente x no afecte la ocurrencia de un símbolo de la fuente y se tendría que:

$$\text{Para toda } i \text{ \& } j: p_{(i, y_j)} = p_{(i)} \cdot q_{(j)}$$

Por lo que:

$$H(X, Y) = H(X) + H(Y)$$

Se define $H(X/Y)$ como una medida condicional de la información, la cual es una medida de la incertidumbre de los símbolos de X cuando se genero un símbolo de Y.

Y viene dada por:

$$H\left(\frac{X}{y_j}\right) = - \sum_{i=1}^n \sum_{j=1}^m p_{(i / y_j)} \log p_{(i / y_j)}$$

(Ecuación 9)

Y de la misma forma se representaría $H(X|Y)$.

Se tiene $H(X|Y) \leq H(X)$ lo que implica que la incertidumbre de los símbolos que provienen de la fuente X teniendo previo conocimiento de los símbolos que provienen de la fuente Y es menor o igual a la incertidumbre de esos símbolos sin previo conocimiento de los de la fuente Y.

La igualdad aplica cuando ambas fuentes son completamente independientes por lo que se puede definir ~~$H(X;Y)$~~ y esta cantidad es llamada medida de información mutua.

De la misma manera se puede definir la medida de dependencia entre las fuentes de información X Y quedando así ~~$H(X;Y) = H(X) + H(X|Y)$~~ . Expresando así el hecho que la información combinada es la suma de las informaciones marginales y la condicional de información.

Y esto permite definir ~~$H(X;Y) = H(X) + H(X|Y) \leq H(X) + H(Y)$~~ en el que la ecuación sostiene que X Y son independientes. Por lo que se puede decir que la medida combinada de información alcanza su máximo cuando ambas fuentes son independientes y decrece a medida que la dependencia crece. En el caso de dependencia completa se tiene ~~$H(X;Y) = 0$~~ por lo que:

~~$$H(X;Y) = H(X)$$~~

1.2.3. Ataque criptoanalítico

Es cuando un intruso trata de descubrir el contenido de un mensaje o la clave secreta para cualquier uso que este le quiera dar. El intruso encontrará más interés en intentar descubrir la clave por sí mismo, en lugar de ocasionalmente descifrar partes del texto, ya que si obtiene la clave otros textos encriptados pueden ser descifrados fácilmente.

Uno de los métodos para encontrar la clave secreta es el de probar todas las posibilidades, hasta que la correcta es encontrada, esto es llamado búsqueda exhaustiva de la llave, pero esto no es realmente un ataque criptoanalítico, generalmente se espera que los criptoanalistas se comporten de forma más inteligente.

Se puede clasificar los ataques criptoanalíticos en 3 tipos:

- Ataque basado únicamente en el texto encriptado (*ciphertext-only-attack*).
- Ataque basado en un texto dado y su texto encriptado correspondiente (*known-plaintext-attack*).
- Ataque basado en un texto elegido y su texto encriptado correspondiente (*chosen-plaintext-attack*).

En el primer caso el criptoanalista únicamente tiene acceso a la señal encriptada y debe basarse en técnicas estadísticas y análisis de patrones dentro de la señal para intentar obtener el mensaje encriptado y más importante aún la clave. Está claro que esta es la peor situación que puede esperar un criptoanalista.

Una situación más favorable se da cuando el criptoanalista obtiene información de parte del mensaje que quiere obtener y a su vez la parte de la señal encriptada que corresponde al fragmento que obtuvo, partiendo de esto si se encuentra alguna relación, entre las partes obtenidas se puede utilizar este conocimiento para decodificar otras secciones del mensaje.

La situación más favorable es cuando el criptoanalista puede obtener cualquier parte del mensaje que él quiera y generar su señal codificada, seleccionando las partes correspondientes de manera correcta, se puede decodificar partes del mensaje que continúen encriptadas o hasta encontrar la clave de encriptación utilizada.

Idealmente un sistema criptográfico debe ser capaz de abolir cualquiera de estos 3 tipos de ataque aunque en la práctica esto es difícil de realizar, pero si un algoritmo es capaz de soportar un ataque del tipo *chosen-plaintext* se espera que sea más resistente a los otros 2 tipos de ataque y presente un nivel de seguridad bastante más alto.

1.3. Herramientas del criptoanálisis

El criptoanálisis se consideró un arte hasta el siglo XX. Donde se convirtió evidente la importancia de la inteligencia militar y diplomática para mantener la conducta de las políticas exteriores de los países. Los conceptos básicos de criptoanálisis fueron desarrollados como un rubro dentro de la matemática aplicada. Este utiliza las herramientas de teoría de la probabilidad, álgebra lineal, álgebra abstracta y teoría de la complejidad.

1.3.1. Estadísticas en el criptoanálisis

El criptoanálisis tiene varias herramientas estadísticas a su disposición para encontrar la tabla de encriptación o decodificar el texto codificado, por ejemplo, para determinar si 2 columnas tienen la misma frecuencia de distribución como en el caso de la máquina de Hagelin. Entre las principales herramientas se tiene:

- Índice de Coincidencia (CI)

Desarrollado por William F. Friedman (1891-1969).

El índice de coincidencia I se define como la probabilidad que dos letras seleccionadas aleatoriamente del texto encriptado representen la misma en el texto original.

Y viene dado por:

$$I = \frac{\binom{n_0}{2} + \binom{n_1}{2} + \dots + \binom{n_j}{2}}{\binom{n}{2}} = \frac{1}{n-1} \sum_{i=0}^{i=j} n_i (n_i - 1)$$

(Ecuación 10)

Para aclarar el por qué este índice da información útil, se debe notar que la probabilidad de que seccionando 2 letras iguales aleatoriamente de un texto en inglés es donde p_0 es la probabilidad de seleccionar una A, p_1 es la probabilidad de seleccionar una B y así hasta la última letra y los valores de p_i son dados en la tabla 1.3.

Esto implica que una palabra en inglés encriptada teniendo un índice de coincidencia de $I \approx 0,065$ es probablemente asociada con un criptograma del tiempo monoalfabético, ya que la estadística no cambiará si las letras están simplemente en distinto orden.

Entre más largo y aleatorio sea una palabra encriptada por método de Vigenere, es más eventual la distribución de las letras en el texto. Con una clave bastante larga y aleatoria se esperaría que:

$$I = 26 \frac{1}{26}^2 = \frac{1}{26} \approx 0,03846$$

Por lo que si se tiene un índice de correlación de 0,03846 se esperaría que el texto encriptado que se tiene se haya codificado por algún método polialfabético. En el idioma inglés el índice de coeficiencia I debe satisfacer.

De esta manera se puede saber ante qué tipo de encriptación se está enfrentando y la mejor manera de iniciar el análisis.

Si se considera un texto completamente aleatorio, construido con base en un alfabeto de 26 letras, se tiene que cada una de las letras tiene la misma probabilidad de ocurrencia la cual sería de $1/26$ Suponiendo que se tiene un segundo alfabeto y que se implanta sobre el primero, se puede imaginar que tanta probabilidad se tiene de encontrar 2 caracteres iguales uno sobre el otro.

Sabiendo que cada carácter representa una letra aleatoriamente la probabilidad de encontrar por ejemplo 2 letras “e” sería $(1/26)^2$. Obviamente, lo mismo sería para cualesquiera otras 2 letras lo que daría que la probabilidad de encontrar 2 letras juntas sería $26 \cdot (1/26)^2$.

- Prueba de *Kaiski*

El método de *Kaiski* consiste en buscar repeticiones de cadenas de caracteres en el criptograma. Si estas cadenas son mayores o iguales a tres caracteres y se repiten más de una vez, lo más probable es que esto se deba a cadenas típicas del texto en claro (trigramas, tetragramas, etcétera, muy comunes) que se han cifrado con una misma porción de la clave. Si se detectan estas cadenas, la distancia entre las mismas será múltiplo de la longitud de la clave. Luego, el máximo común divisor entre esas cadenas es un candidato a ser la longitud de la clave, es decir L.

Se divide el criptograma en L subcriptogramas que entonces han sido cifrados por una misma letra de la clave y en cada subcriptograma se hace un ataque simple ahora de tipo estadístico monoalfabético. La idea es buscar a través de los tres caracteres más frecuentes en cada subcriptograma las posiciones relativas de las letras A, E y O que en español están separadas por 4 y 11 lugares. Si la posición relativa de la letra A es el valor 0, entonces la letra E está cuatro espacios a la derecha de la A ($m+4 \pmod{27}$) y la letra O está 15 espacios a la derecha de la letra A ($m+15 \pmod{27}$).

Entonces se buscará en cada subcriptograma C_i las tres letras más frecuentes y que cumplan además con esta distribución. Es suficiente contar con estas tres letras para que el ataque prospere.

No obstante, se puede afinar más el ataque si se toma en cuenta la siguiente letra frecuente en castellano (S) en posición $(m+19) \pmod{27}$, luego de esto se utiliza el índice de coincidencia para analizar cada subcriptograma por separado y así obtener la llave para descifrar el mensaje.

- Prueba Chi

Este es otro método que utiliza la relación entre 2 diferentes distribuciones estadísticas que representan la repitencia de cierto caracter en un texto, entre más parecidas sean las distribuciones, más alto será el número de correlación.

1.3.2. Aritmética modular

La aritmética modular es una parte de las Matemáticas extremadamente útil en Criptografía, ya que permite realizar cálculos complejos y plantear problemas interesantes, manteniendo siempre una representación numérica compacta y definida, ya que sólo maneja un conjunto finito de números enteros. También es conocida como aritmética del reloj, debido a su parecido con la forma que se cuenta el tiempo.

Por ejemplo, si son las 19:13:59 y pasa un segundo, se dice que son las 19:14:00 y no las 19:13:60. Como se ve, los segundos al igual que los minutos, se expresan empleando sesenta valores cíclicamente, de forma que tras el 59 viene de nuevo el 0. Desde el punto de vista matemático se diría que los segundos se expresan módulo 60.

Aplicando un punto de vista más formal y riguroso: dados tres números $a, b, n \in \mathbb{N}$, se dice que a es congruente con b módulo n y se escribe:

$$a \equiv b \pmod{n}$$

Si se cumple:

$$a = b + kn. \text{ Para algún } k \in \mathbb{N}$$

Ejemplo:

$37 \equiv 5 \pmod{8}$ Ya que $37 = 5 + 4 \cdot 8$. De hecho, los números 5, -3, 13, -11, 21, -19, 29: son todos equivalentes en la aritmética módulo 8, es decir, forman una clase de equivalencia.

Como se puede apreciar, cualquier número entero pertenecerá necesariamente a alguna de esas clases y en general, se tendrá n clases de equivalencia módulo n (números congruentes con 0, números congruentes con 1, . . . , números congruentes con $n - 1$).

En este ejemplo (módulo 8) se tendría el conjunto de clases de equivalencia $\{0, 1, 2, 3, 4, 5, 6, 7\}$, al que se denomina Z_8 . Se pueden definir ahora las operaciones suma y producto en este tipo de conjuntos:

$$a + b \equiv c \pmod{n} \iff a + b = c + kn \quad k \in Z$$

$$ab \equiv c \pmod{n} \iff ab = c + kn \quad k \in Z$$

Propiedades de la suma:

- Asociativa: $\forall a, b, c \in Z_n \quad (a + b) + c \equiv a + (b + c) \pmod{n}$
- Conmutativa: $\forall a, b \in Z_n \quad a + b \equiv b + a \pmod{n}$
- Elemento neutro: $\forall a \in Z_n \quad \exists 0$ tal que $a + 0 \equiv a \pmod{n}$
- Elemento simétrico (opuesto): $\forall a \in Z_n \quad \exists b$ tal que $a + b \equiv 0 \pmod{n}$

Propiedades del producto:

- Asociativa: $\forall a, b, c \in \mathbb{Z}_n \quad (a \cdot b) \cdot c \equiv a \cdot (b \cdot c) \pmod{n}$
- Conmutativa: $\forall a, b \in \mathbb{Z}_n \quad a \cdot b \equiv b \cdot a \pmod{n}$
- Elemento neutro: $\forall a \in \mathbb{Z}_n \quad \exists 1 \text{ tal que } a \cdot 1 \equiv a \pmod{n}$

Propiedades del producto con respecto de la suma:

- Distributiva: $\forall a, b, c \in \mathbb{Z}_n \quad (a + b) \cdot c \equiv a \cdot c + b \cdot c \pmod{n}$

La operación suma cumple las propiedades asociativa y conmutativa y posee elementos neutro y simétrico. Se puede decir, por tanto que el conjunto \mathbb{Z}_n , junto con esta operación, tiene estructura de grupo conmutativo.

Con la operación producto se cumplen las propiedades asociativa y conmutativa y tiene elemento neutro, pero no necesariamente simétrico. La estructura del conjunto con las operaciones suma y producto son de anillo conmutativo.

1.3.3. Algoritmo de Euclides

Permite obtener de forma eficiente el máximo común divisor de dos números. Sean a y b dos números enteros de los que se quiere calcular su máximo común divisor m . El algoritmo de Euclides explota la siguiente propiedad:

~~1.3.3. Algoritmo de Euclides~~ Con ~~1.3.3. Algoritmo de Euclides~~

a/b Quiere decir que a divide a b, o en otras palabras, que b es múltiplo de a, mientras que $(a \bmod b)$ representa el residuo de dividir a entre b. En esencia se esta diciendo, que, puesto que m divide tanto a a como a b, debe dividir a su diferencia. Entonces si se resta k veces b de a, llegará un momento en el que se obtiene el residuo de dividir a por b, o sea $a \bmod b$.

Si se llama c a $(a \bmod b)$, se puede aplicar de nuevo la propiedad anterior y se tendría: ~~$m \bmod d$~~ Esto implica que m tiene que dividir a todos los residuos que se vayan obteniendo. Es evidente que el último de ellos será cero, puesto que los residuos siempre son inferiores al divisor. El penúltimo valor obtenido es el mayor número que divide tanto a a como a b, o sea, el máximo común divisor de ambos.

1.3.4. Complejidad de las operaciones aritméticas en \mathbb{Z}_n

La complejidad algorítmica de las operaciones aritméticas modulares es la misma que la de las no modulares:

- Suma modular: $(a + b) \bmod n \leq O(\log_2 n + \log_2 n) = O(\log_2 n)$
- Resta modular: $(a - b) \bmod n \leq O(\log_2 n + \log_2 n) = O(\log_2 n)$
- Multiplicación modular: $(a \cdot b) \bmod n \leq O(\log_2 n \cdot \log_2 n) = O(\log_2^2 n)$

1.3.4.1. Cálculo de inversas en \mathbb{Z}_n

Existencia de la inversa: los elementos de un grupo finito no tienen por qué tener inversa (elemento simétrico para el producto). Se verá qué condiciones han de cumplirse para que exista la inversa de un número dentro de un grupo finito.

Definición: dos números enteros a y b se denominan primos entre sí (o coprimos), si

$$\text{mcd}(a, b) = 1$$

Teorema: dados $a, n \in \mathbb{N}$

$$\text{mcd}(a, n) = 1 \Rightarrow ai \equiv aj \pmod{n} \Rightarrow \forall i \neq j \quad 0 < i, j < n$$

Demostración: suponiendo que $\text{mcd}(a, n) = 1$ y que existen $i \neq j$ tales que:

$$ai \equiv aj \pmod{n}. \text{ Se cumple: } n \mid (ai - aj) \Rightarrow n \mid a(i - j).$$

Puesto que a y n son primos entre sí, n no puede dividir a a , luego

~~$$n \mid i - j = i - j + n$$~~

Con lo que se ha alcanzado una contradicción.

Ahora se puede hacer la siguiente reflexión: Si $a_i = a_j$ para cualesquiera $i = j$, multiplicar a por todos los elementos del grupo finito módulo n producirá una permutación de los elementos del grupo (exceptuando el cero), por lo que forzosamente ha de existir un valor tal que al multiplicarlo por a dé 1. Eso conduce al siguiente teorema:

Definición: un número entero $p \geq 2$ se dice primo si sus únicos divisores positivos son 1 y p . En caso contrario se denomina compuesto.

Teorema: si $\text{mcd}(a; n) = 1$, a tiene inversa módulo n .

Corolario: si n es primo, el grupo finito que genera tiene estructura de cuerpo (todos sus elementos tienen inversa para el producto excepto el cero). Estos cuerpos finitos tienen una gran importancia en Matemáticas, se denominan Cuerpos de Galois y su notación es $\text{GF}(n)$.

1.3.5. Función de Euler

Se llama conjunto reducido de residuos módulo n (y se denota Z_n) al conjunto de números primos relativos con n . En otras palabras, Z_n es el conjunto de todos los números que tienen inversa módulo n . Por ejemplo, si n fuera 12, su conjunto reducido de residuos sería:

$$\{1, 5, 7, 11\}$$

Existe una expresión que permite calcular el número de elementos (el cardinal) del conjunto reducido de residuos módulo n :

$$\phi(n) = \prod_{i=1}^n p_i^{e_i-1} (p_i - 1)$$

(Ecuación 11)

Siendo p_i los factores primos de n y e_i su multiplicidad. Por ejemplo, si n fuera el producto de dos números primos p y q , $\phi(n) = (p-1)(q-1)$.

Se define la función de Euler sobre n :

$$\phi(n) = |\mathbb{Z}_n^*|$$

Teorema: si $\text{mcd}(a, n) = 1$: $a^{\phi(n)} \equiv 1 \pmod{n}$

Demostración: puesto que a y n son primos entre sí, a multiplicado por cualquier elemento del conjunto reducido de residuos módulo n $\{r_1, \dots, r_{\phi(n)}\}$ ha de ser también primo con n , por lo tanto, el conjunto $\{ar_1, \dots, ar_{\phi(n)}\}$ no es más que una permutación del conjunto anterior, lo cual lleva a:

$$\prod_{i=1}^{\phi(n)} r_i = \prod_{i=1}^{\phi(n)} ar_i = a^{\phi(n)} \prod_{i=1}^{\phi(n)} r_i \Rightarrow a^{\phi(n)} \equiv 1 \pmod{n}$$

Teorema de Fermat: si p es primo entonces $a^{p-1} \equiv 1 \pmod{p}$

Como consecuencia de este último teorema se puede deducir que si p es un número primo y $r \equiv s \pmod{p-1}$ entonces: $a^r \equiv a^s \pmod{p}$ sea cual sea el valor de a . Por lo tanto, cuando se trabaja módulo p , siendo p primo, los exponentes pueden ser reducidos $\pmod{p-1}$.

Definición: sea $a \in \mathbb{Z}_n$. Se define el orden de a , denotado $\text{ord} | a |$, como el menor entero positivo t tal que $a^t \equiv 1 \pmod{n}$.

Existe una interesante propiedad de $\text{ord} | a |$. Si $a \equiv 1 \pmod{n}$, entonces $\text{ord} | a |$ divide a s . En particular, se tiene que $\text{ord} | a |$ siempre divide a $\phi(n)$.

Después de todo lo expuesto, queda claro que uno de los posibles métodos para calcular inversas \pmod{n} , es precisamente la Función de Euler, puesto que:

$$a^{\phi(n)} \equiv a^{-1} \pmod{n}$$

1.3.6. Teorema del chino resto

El teorema chino del resto es una potente herramienta matemática, que posee interesantes aplicaciones criptográficas.

Teorema: sea p_1, \dots, p_r una serie de números primos entre sí, y $n = p_1 \cdot p_2 \cdot \dots \cdot p_r$, entonces el sistema de ecuaciones en congruencias $x \equiv x_i \pmod{p_i}$ $i = 1, \dots, r$ tiene una única solución común en $[0, n-1]$, que viene dada por la expresión:

$$x \equiv \sum_{i=1}^r x_i \cdot \left[\frac{n}{p_i} \right] \cdot \left[\frac{p_i^{-1}}{n} \right] \pmod{n}$$

Demostración: para cada i , $\text{mcd}(p_i, n/p_i) = 1$. Por lo tanto, cada n/p_i debe tener una inversa x_i y tal que:

$$\left[\frac{n}{p_i} \right] y_i \equiv 1 \pmod{p_i}$$

También se cumple:

$$\left[\frac{n}{p_i} \right] y_i \equiv 0 \pmod{p_j} \quad \forall_i \neq j$$

Ya que $\frac{n}{p_i}$ es múltiplo de cada p_j . Sea $x = \sum_{k \neq i} \frac{n}{p_k} y_k x_k \pmod{\phi}$.

Se define la ecuación del chino resto:

$$x = \sum_{k \neq i} \frac{n}{p_k} y_k x_k + \frac{n}{p_i} y_i x_i = 0 + 1 \cdot x_i \equiv x_i \pmod{\phi_i}$$

(Ecuación 12)

1.3.7. Exponenciación

Muchos de los algoritmos de llave pública emplean exponenciaciones dentro de grupos finitos para codificar los mensajes. Tanto las bases como los exponentes en esos casos son números astronómicos, incluso de miles de bits de longitud. Efectuar las exponenciaciones mediante multiplicaciones reiterativas de la base sería inviable. En esta sección se verán mecanismos eficientes para llevar a cabo estas operaciones. También se comentará brevemente el problema inverso, el cálculo de los logaritmos discretos, puesto que en su dificultad intrínseca se apoyan muchos algoritmos criptográficos.

1.3.7.1. Algoritmo rápido de exponenciación

Suponiendo que se tienen dos números naturales a y b y se quiere calcular a^b . El mecanismo más sencillo sería multiplicar a por sí mismo b veces. Sin embargo, para valores muy grandes de b este algoritmo no sirve.

Al tomar la representación binaria de b :

$$b = b_{n-1}b_{n-2}\dots b_1b_0$$

Al expresar la potencia que se va a calcular en función de dicha representación:

$$a^b = a^{b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \dots + b_12^1 + b_02^0}$$

(Ecuación 13)

Se debe recordar que los b_i sólo pueden tomar valores 0 o 1, por tanto, para calcular a^b sólo se deben multiplicar los a^{2^i} correspondientes a los dígitos binarios de b que valgan 1. Se puede notar, además, que $a^{2^i} = (a^{2^{i-1}})^2$, por lo que, partiendo de a , se puede calcular el siguiente valor de esta serie elevando al cuadrado el anterior.

1.3.7.2. El problema de los logaritmos discretos

El problema inverso de la exponenciación es el cálculo de logaritmos discretos. Dados dos números a , b y el módulo n , se define el logaritmo discreto de a en base b módulo n como:

$$c = \log_b a \pmod{n} \iff a \equiv b^c \pmod{n}$$

En la actualidad no existen algoritmos eficientes que sean capaces de calcular en tiempo razonable logaritmos de esta naturaleza y muchos esquemas criptográficos basan su resistencia en esta circunstancia.

El problema de los logaritmos discretos está íntimamente relacionado con el de la factorización, de hecho está demostrado que si se puede calcular un logaritmo, entonces se puede factorizar fácilmente.

1.3.7.3. El problema de Diffie-Hellman

El problema de Diffie-Hellman está íntimamente relacionado con el problema de los logaritmos discretos y es la base de algunos sistemas criptográficos de clave pública, como el de *Diffie-Hellman* y el del Gamal.

El enunciado del problema es el siguiente: dado un número primo p , un número α que sea un generador de Z_p^* , y los elementos α^a y α^b , encontrar $\alpha^{ab} \pmod{p}$. Nótese que se conoce α^a y α^b , pero no el valor de a ni el de b .

De hecho, si se pudiera efectuar de forma eficiente logaritmos discretos, sería suficiente con calcular a y luego $\alpha^b \mid^a = \alpha^{ab}$.

1.3.7.4. Importancia de los números primos

Para explotar la dificultad de cálculo de logaritmos discretos, muchos algoritmos criptográficos de clave pública se basan en operaciones de exponenciación en grupos finitos. Dichos conjuntos deben cumplir la propiedad de que su módulo n sea un número muy grande con pocos factores (usualmente dos).

Estos algoritmos funcionan si se conoce n y sus factores se mantienen en secreto. Habitualmente para obtener n se calculan primero dos números primos muy grandes, que posteriormente se multiplican.

Se necesitan pues mecanismos para calcular esos números primos grandes. La factorización es el problema inverso a la multiplicación: dado n , se trata de buscar un conjunto de números tales que su producto valga n . Normalmente, para que la solución sea única, se impone la condición de que los factores de n que se obtienen sean todos primos elevados a alguna potencia.

Al igual que para el problema de los logaritmos discretos, no existen algoritmos eficientes para efectuar este tipo de cálculos. Esto permite confiar en que en la práctica, será imposible calcular los factores de n , incluso disponiendo de elevados recursos computacionales. En cuanto al cálculo de primos grandes, bastaría con aplicar un algoritmo de factorización para saber si un número es primo o no.

Este mecanismo es inviable, puesto que se acaba de decir que no hay algoritmos eficientes de factorización. Por suerte, sí existen algoritmos probabilísticos que permiten decir con un grado de certeza bastante elevado si un número cualquiera es primo o compuesto.

Cabría preguntarse, dado que para los algoritmos asimétricos de cifrado se necesitarán generar muchos números primos, si realmente hay suficientes.

De hecho se puede pensar que, a fuerza de generar números, llegará un momento en el que se repite un primo generado con anterioridad. Se puede estar tranquilo, porque si a cada átomo del universo se le asigna mil millones de números primos cada microsegundo desde su origen hasta hoy, harían falta un total de 10^{109} números primos diferentes, mientras que el total estimado de números primos de 512 bits o menos es aproximadamente de 10^{151} .

También se podría pensar en calcular indiscriminadamente números primos para luego emplearlos en algún algoritmo de factorización rápida. Por desgracia, si se quisiera construir un disco duro que albergue diez mil GBytes por cada gramo de masa y milímetro cúbico para almacenar todos los primos de 512 bits o menos, el artilugio pesaría más de 10^{135} kilogramos y ocuparía casi 10^{130} metros cúbicos, es decir, sería miles de billones de veces más grande y pesado que la vía láctea.

1.4. Criptografía y números aleatorios

Los algoritmos de llave pública, debido a su mayor orden de complejidad, suelen ser empleados en conjunción con algoritmos de llave privada de la siguiente forma: el mensaje primero se codifica empleando un algoritmo simétrico y la llamada clave de sesión, que será diferente cada vez. Es únicamente la clave de sesión la que se cifra empleando criptografía asimétrica, produciendo un importante ahorro de coste computacional.

Sin embargo, si el proceso de generación de estas claves fuera predecible o, al menos, reproducible, un atacante malicioso podría llegar a adivinar la siguiente clave de sesión a partir de una o varias claves, lo cual tendría resultados catastróficos.

Un famoso ejemplo de este problema tuvo lugar en una de las primeras versiones del navegador Netscape, que resultaba insegura debido al uso de un generador de claves demasiado previsible. La única manera de protegerse frente a estos ataques es asegurarse de que no exista ningún tipo de dependencia entre una clave y la siguiente, esto es, que sean aleatorias. De aquí surge el interés por los números aleatorios en criptografía.

1.4.1. Tipos de secuencias aleatorias

En realidad es casi del todo imposible generar secuencias auténticamente aleatorias en una computadora, puesto que estas máquinas son (al menos en teoría) completamente deterministas. De hecho, cualquier generador que emplee únicamente métodos algorítmicos en su propósito producirá secuencias reproducibles, por lo que se estará hablando en realidad de secuencias pseudoaleatorias.

En general, todos los generadores pseudoaleatorios producen secuencias finitas y periódicas de números empleando exclusivamente operaciones aritméticas o lógicas. No obstante, si se emplean elementos externos a la computadora, se pueden generar también secuencias realmente aleatorias. En esta sección se describirán dos tipos de secuencias pseudoaleatorias, en función de sus propiedades, además de las secuencias auténticamente aleatorias.

1.4.1.1. Secuencias estadísticamente aleatorias

En principio, es relativamente fácil conseguir que una secuencia pseudoaleatoria sea lo más larga posible antes de comenzar a repetirse y que supere los *tests* estadísticos de aleatoriedad. En este sentido se puede hablar de:

Secuencias estadísticamente aleatorias: secuencias pseudoaleatorias que superan las pruebas estadísticas de aleatoriedad. Los generadores congruenciales lineales cumplen esta propiedad y de hecho son muy utilizados en informática, especialmente en entornos de simulación, pero en Criptografía resultan del todo inútiles, debido a que cada valor de la secuencia se emplea como semilla para calcular el siguiente, lo cual permite conocer toda la serie a partir de un único valor.

Suponiendo que se tiene un sistema que se basa en emplear claves aleatorias para cada sesión y se usa un generador de este tipo. Bastaría con que una de las claves quedará comprometida para que todas las comunicaciones pasadas y futuras pudieran ser descifradas sin problemas. Incluso se ha demostrado que conociendo únicamente un bit de cada valor de la secuencia, esta puede ser recuperada completamente con una cantidad relativamente pequeña de valores.

1.4.1.2. Secuencias criptográficamente aleatorias

El problema de las secuencias estadísticamente aleatorias y lo que las hace poco útiles en Criptografía, es que son completamente predecibles.

Secuencias criptográficamente aleatorias: para que una secuencia pseudoaleatoria sea criptográficamente aleatoria, ha de cumplir la propiedad de ser impredecible.

Esto quiere decir, que debe ser computacionalmente intratable el problema de averiguar el siguiente número de la secuencia, teniendo total conocimiento acerca de todos los valores anteriores y del algoritmo de generación empleado.

Existen generadores pseudoaleatorios capaces de generar secuencias criptográficamente aleatorias, generalmente a través del uso en el algoritmo de información de inicialización denominada semilla o de estado que ha de mantenerse en secreto. Sin embargo, habrá situaciones en las que esto no sea suficiente para los propósitos y en las que se desee tener valores realmente impredecibles, de forma que el adversario no pueda averiguarlos ni tratar de simular el proceso de generación que se ha llevado a cabo.

1.4.1.3. Secuencias totalmente aleatorias

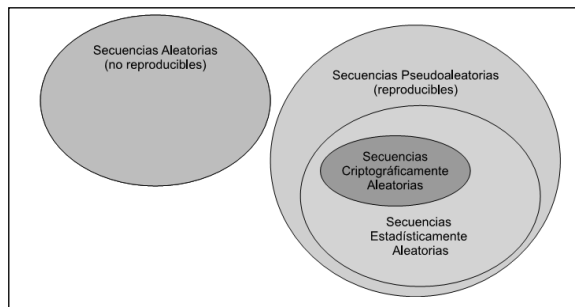
Como ya se ha dicho antes, no existe la aleatoriedad cuando se habla de computadoras. Sin embargo, se puede hacer que el ordenador, a través de sus dispositivos de entrada/salida, obtenga de su entorno sucesos que pueden considerarse impredecibles. Por lo que se considera un tercer tipo de secuencias:

Secuencias aleatorias: una secuencia es totalmente aleatoria (o simplemente aleatoria) si no puede ser reproducida de manera fiable.

1.4.1.4. Utilidad de las secuencias aleatorias en criptografía

En este punto parece claro que el objetivo en la mayor parte de las ocasiones no va a consistir en generar secuencias aleatorias puras, sino secuencias impredecibles e irreproducibles para un atacante. De hecho, habrá dos escenarios típicos en los que se van a encontrar:

Figura 7. **Secuencias aleatorias en criptografía**



Fuente: LUCEMA LÓPEZ, Manuel José. Criptografía y seguridad en computadores. P. 118.

Para generar una secuencia de números impredecible e irreproducible, por ejemplo, para generar claves de sesión. Para ello, se puede utilizar indistintamente un generador totalmente aleatorio o un generador pseudoaleatorio criptográficamente aleatorio.

En este último caso, se empleará como semilla la salida producida por un generador totalmente aleatorio.

Se quiere generar una secuencia de números que luego pueda reproducirse, por ejemplo, para construir un cifrado de flujo. En ese caso se empleará un generador criptográficamente aleatorio, cuya semilla hará las veces de clave, ya que permitirá al emisor generar una secuencia pseudoaleatoria impredecible para un atacante y combinarla con el mensaje para obtener el criptograma.

El receptor usará la misma semilla para generar una secuencia idéntica y recuperar así el mensaje original.

1.4.2. Generadores aleatorios criptográficamente seguros

Se va a ver a continuación un par de generadores pseudoaleatorios que permiten obtener secuencias lo suficientemente seguras como para ser empleadas en aplicaciones criptográficas. Ambos emplean una semilla que puede ser obtenida a partir de un generador totalmente aleatorio, e incluso uno de ellos, emplea internamente información de gran variabilidad, como es el reloj del sistema, para hacer más difícil de reproducir la secuencia resultante.

1.4.2.1. Generador X9.17

Propuesto por el Instituto Nacional de Estándares Norteamericano. Permite, a partir de una semilla inicial s_0 de 64 bits obtener secuencias de valores también de 64 bits. En sus cálculos emplea valores difíciles de adivinar desde el exterior, como el tiempo del sistema en el momento de obtener cada elemento de la secuencia, para de esta forma aproximar más su comportamiento al de un generador totalmente aleatorio.

El algoritmo para obtener cada uno de los valores g_n de la secuencia es el siguiente:

$$g_n = \text{DES}(s_{n-1} \oplus t) \oplus t$$
$$s_n = \text{DES}(g_n \oplus k)$$

(Ecuación 14)

Donde k es una clave aleatoria reservada para la generación de cada secuencia y t es el tiempo en el que cada valor es generado (cuanta más resolución tenga (hasta 64 bits), mejor). $\text{DES}_{K,M}$ Representa la codificación de M mediante el algoritmo DES, empleando la clave K y \oplus representa la función or-exclusivo. El valor K ha de ser mantenido en secreto para que la seguridad de este generador sea máxima.

1.4.2.2. Generador Blum Blum Shub

Si bien se trata en realidad de un generador pseudoaleatorio, es uno de los algoritmos que más pruebas de resistencia ha superado, con la ventaja adicional de su gran simplicidad (aunque es computacionalmente mucho más costoso que el algoritmo X9,17). Consiste en escoger dos números primos grandes, p y q , que cumplan la siguiente propiedad:

$$p \pmod{4} = q \pmod{4}$$

Sea entonces $n = pq$. Se escoge un número x aleatorio primo relativo con n , que será la semilla inicial. Al contrario que x , que debe ser mantenido en secreto, n puede ser público.

Se calculan los valores s_i de la serie de la siguiente forma:

$$s_0 = (x^2) \pmod{n}$$

$$s_{i+1} = (s_i^2) \pmod{n}$$

Hay que tener cuidado de emplear únicamente como salida unos pocos de los bits menos significativos de cada s_i . De hecho, si se toman no más que $\log_2(\log_2(s_i))$ bits en cada caso se puede asegurar que predecir el siguiente valor de la serie es al menos tan difícil como factorizar n .

2. SISTEMAS SECRETOS

2.1. Alfabetos

El encriptamiento es la transformación del texto base (*plaintext*) a texto encriptado (*ciphertext*) con la intención de esconder la información contenida en el texto base. Ambas partes tanto el *plaintext* como el *ciphertext* están compuestos por letras de una gama finita de símbolos, a esto se le conoce como alfabetos. Se pueden citar como ejemplos:

- El caso de las letras mayúsculas:
ABCDEFGHIJKLMNOPQRSTUVWXYZ
- Letras mayúsculas y números
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
- Letras mayúsculas, minúsculas y números
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
- Letras mayúsculas, minúsculas, números y símbolos de puntuación
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789.,:;’¿?’
- La secuencia binaria de 0 a 256 de 8 bit de longitud
00000000, 00000001, ..., ..., 11111110, 11111111

Una forma de generar alfabetos es concatenando las letras de 2 o más diferentes alfabetos, por ejemplo, el alfabeto ~~A B C D E F G H I J K L M N O P Q R S T U V W X Y Z~~ se podría formar el alfabeto A2 el cual sería un alfabeto de 2 caracteres así:

$$a_0a_0, a_0a_1 \dots a_{m-1}a_m$$

Por ejemplo, si se tiene un alfabeto:

$$A = \{A, B, C, D, E, F, G, H, I, J, K, L, M, N, \tilde{N}, O, P, Q, R, S, T, U, V, W, X, Y, Z\}$$

Entonces $A^2 = \{AA, AB, AC, \dots, BA, BB, \dots, ZA, ZB, \dots, ZX, ZY, ZZ\}$ el cual sería un alfabeto de 676 símbolos de 2 caracteres. De igual forma se pueden crear alfabetos más grandes agregando más caracteres como A3, el cual sería un alfabeto de 17 576 símbolos de 3 caracteres.

También puede resultar útil reemplazar los elementos del alfabeto por caracteres numéricos lo cual permitiría una manipulación algebraica fácil, la manera más normal de hacerlo sería:

Tabla III. **Alfabetos**

A ↓◇ 0	J ↓◇ 9	R ↓◇ 18
B ↓◇ 1	K ↓◇ 10	S ↓◇ 19
C ↓◇ 2	L ↓◇ 11	T ↓◇ 20
D ↓◇ 3	M ↓◇ 12	U ↓◇ 21
E ↓◇ 4	N ↓◇ 13	V ↓◇ 22
F ↓◇ 5	Ñ ↓◇ 14	W ↓◇ 23
G ↓◇ 6	O ↓◇ 15	X ↓◇ 24
H ↓◇ 7	P ↓◇ 16	Y ↓◇ 25
I ↓◇ 8	Q ↓◇ 17	Z ↓◇ 26

Fuente: www.kryptopolis.org. Consulta: junio de 2009

El utilizar números para reemplazar un alfabeto cualquiera permite concentrarse en formas que no se podría enfocar si se trabaja con letras, también es más difícil encontrar fragmentos de palabras en un alfabeto numérico y se permite formular conceptos con base a esta sustitución.

2.2. ***Plaintext source*** (fuente de texto plano)

El encriptamiento es un proceso que se aplica en cualquier texto en cualquier lenguaje, como ejemplos de lenguajes se podría incluir, texto en inglés, programación de computadoras y comunicación de datos.

Las reglas de un lenguaje están definidas por cuales son sus formas admisibles, por ejemplo:

- En el texto en inglés: la letra Q es generalmente seguida de la letra U.
- En el lenguaje ensamblador: las instrucciones dadas por números generalmente aparecen en columnas de 73-80.
- En Fortan: las instrucciones normalmente inician en la columna 7.
- En el lenguaje utilizado por una aerolínea para hacer reservaciones: la fecha y el número de vuelo deberán aparecer en segmentos predeterminados del texto.

Un lenguaje es definido, especificando la cantidad de formas que este puede admitir. Al contrario de los lenguajes de programación, la mayoría de lenguajes no tienen la consistencia suficiente para que sea posible efectuar un análisis de descripción. Se puede representar un lenguaje como un tramo probabilístico. Un *plaintext source* finito S para textos desde Z^m es un proceso estocástico, una secuencia finita o infinita de variables aleatorias:

$$X_0, X_1, \dots, X_{n-1}, \dots \text{ (resp. } X_0, X_1, \dots, X_{N-1} \text{)}$$

Una fuente modela texto plano experimental, cuya salida es la secuencia de letras: $X_0, X_1, \dots, X_{n-1}, \dots$ una fuente S es definida especificando la probabilidad

$\Pr_{\text{plain}} \{ X_j = x_0, X_{j+1} = x_1, \dots, X_{j+n+1} = x_{n-1} \}$ de cada uno de los eventos.

$\{ X_j = x_0, X_{j+1} = x_1, \dots, X_{j+n+1} = x_{n-1} \}$ para cada forma $x = \{ x_0, x_1, \dots, x_{n-1} \}$, $j=0, 1, \dots$ y $n=1, 2, \dots$ se representa la probabilidad del evento por $\Pr_{\text{plain}} = \{ x_0, x_1, \dots, x_{n-1} \}$.

$$\Pr_{plain}(x_0, x_1, \dots, x_{n-1})$$

La probabilidad de la figura $\Pr_{plain}(x_0, x_1, \dots, x_{n-1})$ debe satisfacer 3 condiciones:

$$\Pr_{plain}(x_0, x_1, \dots, x_{n-1}) \geq 0$$

$$1 = \sum_{(x_0, x_1, \dots, x_{n-1})} \Pr_{plain}(x_0, x_1, \dots, x_{n-1})$$

$$\Pr_{plain}(x_0, x_1, \dots, x_{n-1}) = \sum_{(x_0, x_1, \dots, x_{n-1})} \Pr_{plain}(x_0, x_1, \dots, x_{n-1}) \quad \text{si } s > n.$$

(Ecuación 15)

La ecuación es versión de la condición de consistencia de Kolmogorov. Conecta la probabilidad asignada por la fuente de las figuras con una figura dada prefijo $(x_0, x_1, \dots, x_{n-1})$.

$$\sum_{(x_0, x_1, \dots, x_{n-1})} \Pr_{plain}(x_0, x_1, \dots, x_{n-1})$$

Para la probabilidad asignada de la fuente al prefijo $\Pr_{plain}(x_0, x_1, \dots, x_{n-1})$.

Ahora la probabilidad $\Pr_{\text{plain}}\{\text{HELP}\}$ de la palabra de cuatro símbolos HELP es igual a la suma de probabilidades asignadas a las 263 palabras de 7 símbolos HELP***, las cuales contienen el prefijo HELP. Un proceso estocástico provee un modelo matemático para la generación de texto plano, el cual es sólo una aproximación al lenguaje. Un modelo probabilístico, elimina la necesidad de especificar el grupo de palabras admisibles. Para que esta aproximación tenga validez, el modelo debe contener las principales características del lenguaje, mientras se mantiene lo suficientemente simple para permitir cálculos numéricos.

En principio los modelos se pueden definir de manera que reflejen la estructura de un lenguaje a cualquier grado deseado de exactitud. Generalmente, el precio que se paga es un detalle matemático muy alto lo cual lleva al modelo al límite de su utilidad.

Definición 2.2.1: una fuente de texto plano S genera palabras de 1 letra por pruebas independientes si:

$$\Pr_{\text{plain}}\{x_0, x_1, \dots, x_{n-1}\} = p(x_0) p(x_1) \dots p(x_{n-1})$$

Para cada n entero $n=1,2,\dots$ y para cada símbolo $(x_0, x_1, \dots, x_{n-1})$ la probabilidad del símbolo de un carácter satisface $p(x_t) \geq 0; 0 \leq t \leq m$

$$p(t) \geq 0 \quad 0 \leq t \leq m$$

$$1 = \sum_{0 \leq t \leq m} p(t)$$

La ecuación anterior implica que para $n \geq 2$,

$$p(x_0) p(x_1 | x_0) \dots p(x_{n-2} | x_{n-3}) = \sum_{x_{n-1}} p(x_0) p(x_1 | x_0) \dots p(x_{n-1} | x_{n-2})$$

O lo que es equivalente

$$\Pr_{\text{plain}}(x_0, x_1, \dots, x_{n-2}) = \sum_{x_{n-1}} \Pr_{\text{plain}}(x_0, x_1, \dots, x_{n-1})$$

Lo que satisface la consistencia de Kolmogorov.

Ejemplo 1.

Un texto plano de compuestos de 1 caracter generado por pruebas independientes para el alfabeto {A, B,...Z} las probabilidades de cada letra p (A), p (B),...,p (Z) son listadas en la siguiente tabla.

Tabla IV. Probabilidades ejemplo 1

Letra	P	Letra	P	Letra	P
A	0.0856	J	0.0013	S	0.0607
B	0.0139	K	0.0042	T	0.1045
C	0.0279	L	0.0339	U	0.0249
D	0.0378	M	0.0249	V	0.0092
E	0.1304	N	0.0707	W	0.0149
F	0.0289	O	0.0797	X	0.0017
G	0.0199	P	0.0199	Y	0.0199
H	0.0528	Q	0.0012	Z	0.0008
I	0.0627	R	0.0677		

Fuente: G. KONHEIM, Alan. Cryptography: A Primer.

Se utiliza la fuente S para generar los símbolos de 4 caracteres SEND Y SEDN y el símbolo de 2 caracteres QU, se tendrían las probabilidades:

~~SEND~~

$$Pr_{PLAIN} = 0,067 \times 0,1304 \times 0,0707 \times 0,0378 \cong 2,119 \times 10^{-5}$$

~~SEDN~~

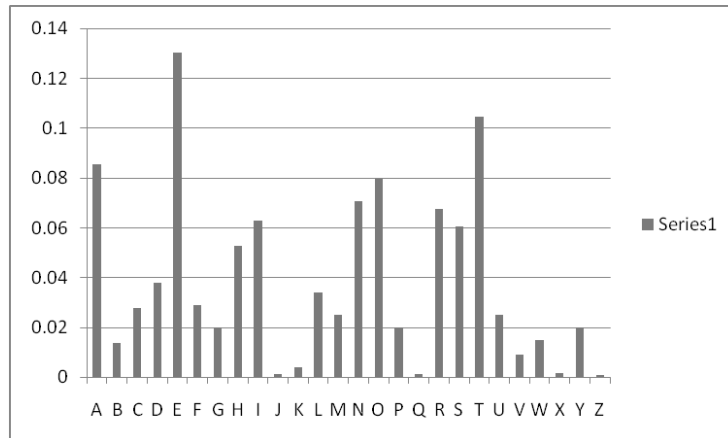
$$Pr_{PLAIN} = 0,067 \times 0,1304 \times 0,0378 \times 0,0707 \cong 2,119 \times 10^{-5}$$

~~QU~~

P_{PLAIN}^{SEN} y P_{PLAIN}^{QU} son las frecuencias de ocurrencia de SEND y QU en un muestreo típico de símbolos de 4 y 2 caracteres respectivamente. La ley de números largos implica que en textos generados por la fuente S analizada anteriormente sería aproximadamente:

- 21 ocurrencias cada una SEND y SEDN en una muestra de 106 de símbolos de 4 caracteres.
- 25 ocurrencias de un símbolo de cuatro letras QU** empezando con las letras QU en una muestra de 106 símbolos de 4 caracteres.

Figura 8. **Gráfica de ocurrencias en ejemplo plain text source**



Fuente: G. KONHEIM, Alan. Cryptography: A Primer.

La gráfica muestra las probabilidades de cada letra, la característica importante de esta distribución es la alternación de vaves y picos, se puede dividir las letras en rangos, alto, medio, bajo y muy bajo. Respecto a la frecuencia de uso.

Tabla V. Clasificación de las letras por probabilidad de ocurrencia

Alto	Medio	Bajo	Muy bajo
A	O	D	J
E	N	L	K
T	I	F	X
	R	C	Z
	S	M	
	H	U	
		G	
		P	
		Y	
		W	
		B	
		V	

Fuente: G. KONHEIM, Alan. Cryptography: A Primer. p. 17.

La probabilidad asignada a \Pr_{PLAIN^x} en el ejemplo anterior describe las características de un texto en idioma inglés en al menos 2 características:

- La probabilidad diferente de cero asignada a la figura de 2 caracteres Q^* con $* \in U$.
- La probabilidad igual asignada a la figura de 4 caracteres SEND y SEDN.

Para ello, sería necesario agregar una condicionante de posición de la letra de manera que las figuras SEND y SEDN tengan probabilidades diferentes.

Definición 2.2.2:

Una fuente de texto plano S genera símbolos de 2 caracteres de Z_m por pruebas independientes e iguales si.



Para cada entero $n=1,2,\dots$ y símbolos de 2 caracteres $(x_0, x_1, \dots, x_{2n-1})$ donde las probabilidades de los símbolos $(t_0, t_1, \dots, t_{2n-1})$ satisfacen 2 condiciones:

$$p(s,t) \geq 0 \quad 0 \leq t, s \leq m$$

$$1 = \sum_{0 \leq t, s \leq m} P(t, s)$$

En la ecuación anterior se muestra que:

$$p(t_0, t_1) p(t_2, t_3) \dots p(t_{2n-4}, t_{2n-3}) = \sum_{(t_{2n-2}, t_{2n-1})} p(t_0, t_1) p(t_2, t_3) \dots p(t_{2n-2}, t_{2n-1})$$

O equivalentemente

$$\Pr_{plain} \langle x_0, x_1, \dots, x_{2n-3} \rangle = \sum_{x_{n-1}} \Pr_{plain} \langle x_0, x_1, \dots, x_{2n-1} \rangle$$

Lo cual implica que las probabilidades asignadas satisfacen la consistencia de Kolmogorov y por lo tanto, esto define una fuente de texto plano.

Definición 2.2.3:

Una fuente de texto plano S genera símbolos de 1 carácter de Z_m por medio de una cadena de markov con matriz de transición P:

$$P = (p_{s/t}) \quad 0 \leq t, s < m$$

Y una distribución de equilibrio

$$\pi = (\pi_t) \quad 0 \leq t < m$$

Si se cumple

$$\pi P = \pi$$

Para cada n entero: $n=1,2,\dots$ y símbolos de n caracteres $(X_0, X_1, \dots, X_{n-1})$.

La probabilidad de transición $(p_{s/t})$ y la distribución de equilibrio satisfacen las condiciones:

$$p_{s/t} \geq 0 \quad 0 \leq t, s < m$$

$$\mathbf{1} = \sum_{s=0}^{m-1} p_{s/t} \quad 0 \leq t < m$$

$$\pi_t \geq 0 \quad 0 \leq t < m$$

$$\mathbf{1} = \sum_{t=0}^{m-1} \pi_t$$

La ecuación anterior muestra que:

$$P(X_1 = x_1, \dots, X_n = x_n) = \prod_{i=1}^n P(X_i = x_i)$$

O equivalentemente

$$P(X_1 = x_1, \dots, X_n = x_n) = \prod_{i=1}^n P(X_i = x_i)$$

Lo cual implica que la consistencia de Kolmogorov se satisface por lo que

$\{X_n\}_{n \geq 1}$ define una fuente de texto plano.

La ecuación $P(X_1 = x_1, \dots, X_n = x_n) = \prod_{i=1}^n P(X_i = x_i)$, $0 \leq s < m$ muestra la relación:

$$P(X_1 = x_1, \dots, X_n = x_n) = \prod_{i=1}^n P(X_i = x_i)$$

equivalentemente:

$$P(X_1 = x_1, \dots, X_n = x_n) = \prod_{i=1}^n P(X_i = x_i)$$

El lado izquierdo de la ecuación anterior es la probabilidad de la fuente de texto plano que generará texto satisfaciendo:

$$P(X_1 = x_1, \dots, X_n = x_n)$$

Y el lado derecho es la probabilidad de que la fuente genere textos satisfaciendo:

$$P(X_1, X_2, \dots, X_n)$$

Un proceso de Markov es un ejemplo de un proceso estocástico, con dependencia intersímbolos o memoria; el valor de la letra X_i es dependiente del valor previo X_{i-1} , la dependencia en pruebas consecutivas es expresada por la probabilidad de transición p_{ij} . La distribución de equilibrio es la solución lineal del sistema de ecuaciones.

$$\pi_1 = \pi_1 p_{11} + \pi_2 p_{21} + \dots + \pi_n p_{n1}$$

$$\pi_2 = \pi_1 p_{12} + \pi_2 p_{22} + \dots + \pi_n p_{n2}$$

...

...

$$\pi_n = \pi_1 p_{1n} + \pi_2 p_{2n} + \dots + \pi_n p_{nn}$$

Lo cual puede ser escrito en forma compacta en notación de matriz como:

$$\pi = \pi P$$

Donde $\pi = [\pi_1 \ \pi_2 \ \dots \ \pi_n]$

Definición 2.2.4:

La fuente de texto plano finita S:

$$X_0 X_1 \dots X_{m-1} \quad (\text{re } X_0 \dots X_{m-1})$$

genera figuras de 1 carácter por medio de pruebas independientes si:

$$P_j(t) = p_j^t \quad (0 \leq t < m)$$

Donde la probabilidad de distribución para cada una de las letras Z_m $(j: 0 \leq j < m)$ para cada j viene dada por:

$$p_j^t \geq 0 \quad 0 \leq t < m$$

$$1 = \sum_{0 \leq t < m} p_j^t \quad (0 \leq j < m) \quad (0 \leq j < N)$$

Definición 2.2.5:

La fuente de texto plano finita S:

$$X_0 X_1 \dots X_{m-1} \quad (\text{re } X_0 \dots X_{m-1})$$

genera figuras de 1 carácter por medio de pruebas independientes si:

$$P_j(t) = p_j^t \quad (0 \leq t < m)$$

$$P_j = \sum_{0 \leq t < m} \left\lfloor \frac{t}{s} \right\rfloor \quad 0 \leq t < m \quad 0 \leq j < \infty \quad | \quad 0 \leq j < N$$

2.3. Sistemas criptográficos

Un sistema criptográfico puede definirse como una quintupla (M;C;K;E;D), donde:

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

Todo sistema criptográfico ha de cumplir la siguiente condición:

$D_k \circ E_k m = m$ es decir, que si se tiene un mensaje m, se cifra empleando la clave k y luego se descifra empleando la misma clave, se obtiene de nuevo el mensaje original m.

Existen dos tipos fundamentales de sistemas criptográficos:

- Simétricos o de clave privada: son aquellos que emplean la misma clave k tanto para cifrar como para descifrar. Presentan el inconveniente de que para ser empleados en comunicaciones la clave k debe estar tanto en el emisor como en el receptor, lo cual lleva a preguntar cómo transmitir la clave de forma segura.
- Asimétricos o de llave pública: emplean una doble clave (k_p ; k_P). k_p se conoce como clave privada y k_P se conoce como clave pública. Una de ellas sirve para la transformación E de cifrado y la otra para la transformación D de descifrado. En muchos casos son intercambiables, esto es, si empleando una para cifrar la otra sirve para descifrar y viceversa. Estos sistemas criptográficos deben cumplir además que el conocimiento de la clave pública k_P , que no permita calcular la clave privada k_p .

Ofrecen un abanico superior de posibilidades, pudiendo emplearse para establecer comunicaciones seguras por canales inseguros o para llevar a cabo autenticaciones. En la práctica se emplea una combinación de estos dos tipos de sistemas criptográficos, puesto que los segundos presentan el inconveniente de ser computacionalmente mucho más costosos que los primeros. En el mundo real se codifican los mensajes (largos) mediante algoritmos simétricos, que suelen ser muy eficientes y luego se hace uso de la criptografía asimétrica para codificar las claves simétricas (cortas).

2.4. Claves débiles

En la inmensa mayoría de los casos los conjuntos M y C definidos anteriormente son iguales. Esto quiere decir, que tanto los textos claros como los textos cifrados se representan empleando el mismo alfabeto, cuando se usa el algoritmo DES, ambos son cadenas de 64 bits. Por esta razón puede darse la posibilidad de que exista algún $k \in K$ tal que $E_k M = M$, lo cual sería catastrófico para los propósitos, debido a que el empleo de esas claves dejaría todos los mensajes sin codificar.

También puede darse el caso de que ciertas claves generen textos cifrados de poca calidad. Una posibilidad bastante común en ciertos algoritmos es que algunas claves tengan la siguiente propiedad: $E_k E_k M = M$, con lo cual se entiende que basta con volver a codificar el criptograma para recuperar el texto claro original. Esto simplificaría enormemente un intento de violar el sistema, por lo que también habrá que evitar la utilización de estas claves a toda costa.

La existencia de claves con estas características, depende de las peculiaridades de cada algoritmo en concreto, así como de los parámetros escogidos a la hora de aplicarlo. Se llamarán en general a las claves que no codifican correctamente los mensajes claves débiles. En un buen sistema criptográfico, la cantidad de claves débiles es cero o muy pequeña en comparación con el número total de claves posibles.

Definición 2.4.1:

Una transformación criptográfica T es una secuencia de transformaciones $T = \{T(n) : 1 \leq n < \infty\}$ donde $T(n) : Z_{m,n} \rightarrow Z_{m,n}$ especifica como cada símbolo de la fuente de texto $x \in Z_{m,n}$ es remplazado por un símbolo criptográfico $T(n)(x) = y \in Z_{m,n}$ para cada n , el mapeo en la ecuación $T(n) : Z_{m,n} \rightarrow Z_{m,n}$ es requerido para ser una transformación de uno a uno en $Z_{m,n}$ de forma que la fuente de texto plano pueda ser recuperada del texto encriptado de una forma no ambigua.

En la definición anterior no se requiere ninguna relación entre los símbolos de n caracteres y los de s caracteres cuando $s \neq n$. Por ejemplo, los símbolos $T(4)(\text{SEND})$ y $T(8)(\text{SENDHELP})$ no necesitan tener ninguna relación especial entre ellos.

Definición 2.4.2:

Un sistema criptográfico T es una familia de transformaciones criptográficas, indexadas o etiquetadas por un parámetro k que se conoce como clave. Los posibles valores de la llave en el espacio de la llave K . Claves distintas $k_1 \neq k_2$ generarán transformaciones criptográficas distintas $T_{k_1} \neq T_{k_2}$.

La definición 2.4.2 tiene 4 componentes importantes:

- El texto plano y el texto encriptado se generan utilizando el mismo alfabeto.
- El encriptamiento es definido para cada símbolo n del texto plano y $n = 1, 2, \dots, n-1$.
- Cada texto plano de símbolo n es encriptado precisamente en un texto criptográfico específico.
- La longitud del texto no es afectada por la encriptación.

Pero no es necesario imponer ninguna de estas condiciones a la definición de una transformación criptográfica.

- Se pueden utilizar alfabetos diferentes para un texto plano y un texto encriptado.
- El método de encriptamiento no necesita ser definido para todos los símbolos de n caracteres del texto plano.
- Más de un texto criptográfico podría ser asociado a un texto plano.
- La longitud del texto puede cambiar bajo la encriptación.

Los requerimientos impuestos en la definición 1 son naturales en una aplicación de criptografía para sistemas de procesamiento de información.

Para este entorno:

- El texto plano y el texto encriptado son ambos escritos utilizando el mismo alfabeto $\{1,0\}$.
- Cualquier símbolo que pueda representar el texto plano, de manera que $T(x)$ debe estar definida para todo el texto plano.
- Si el encriptamiento T no es de valor único, que es, si el texto plano puede ser reemplazado por varios textos encriptados diferentes llamado homophones, o la longitud del texto cambia bajo el encriptamiento, se requeriría que algo del texto plano sea expandido en longitud.

Expandir la longitud de un texto durante la encriptación en un sistema de procesamiento de información puede tornarse en una fuente de dificultad, como se ilustra a continuación:

- Una opción útil en seguridad de archivos es la habilidad de sobrescribir el texto plano con el texto encriptado. Si las longitudes pueden cambiar una forma impredecible, un programador debería estimar el factor de extensión y reservar suficiente espacio para almacenar el tamaño de 2 textos. Un cambio de longitud de texto bajo durante la encriptación complica innecesariamente el ambiente computacional al requerir cambios en los atributos de los datos fijados con anterioridad, en aplicaciones de programa, arreglos, tamaños y longitud de los textos.

- En algunas aplicaciones, solamente algunos campos del archivo grabado son encriptadas. Por ejemplo, un archivo médico, debe organizar los datos en varios campos:

Paciente	Síntomas	Diagnóstico	Tratamiento	Condición
16 bytes	64 bytes	64 bytes	64 bytes	64 bytes

En ocasiones hay razones para compartir la información en archivos con otras organizaciones para obtener datos estadísticos; por ejemplo, la eficacia de un medicamento en particular o el tiempo de recuperación después de cierto tipo de intervención quirúrgica.

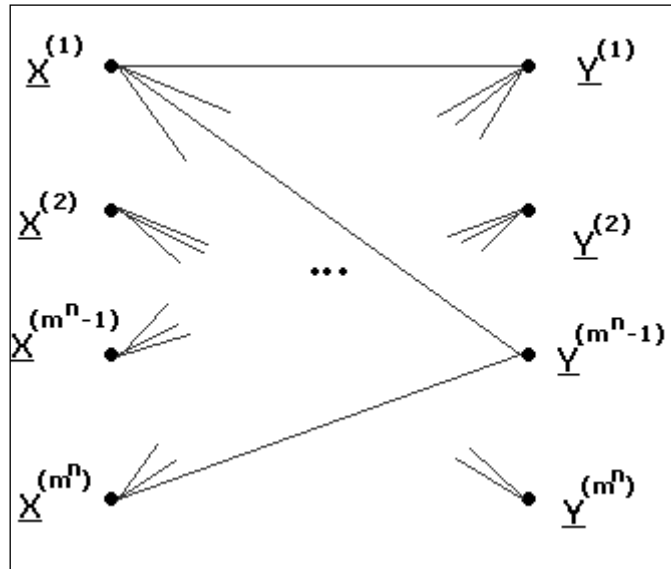
La pregunta planteada sería ¿pueden los campos ser compartidos respetando la privacidad del paciente? Si el primer campo de 16 bytes de cada archivo es encriptado, el archivo puede ser diseminado sin comprometer la privacidad del paciente.

El texto encriptado en el campo Paciente, se utiliza como etiqueta para identificar los archivos. Si se permite que el encriptamiento tenga la variación de texto, los archivos deberían de plantearse en otro formato.

- Extensión de la longitud del texto por encriptamiento cuando se utilizan líneas de telecomunicaciones seguras, reduce la efectividad de la tasa de transmisión de bits, lo cual incrementa el costo de la transmisión.

Un sistema criptográfico $T = \{T_k : k \in K\}$ puede ser descrito como una secuencia de gráficas orientadas bipartitas $(G, T : n \rightarrow 2)$, como se muestra en la siguiente figura.

Figura 9. Diagrama de transmisión en claves débiles



Fuente: www.kryptopolis.org. Consulta: junio de 2009.

Una gráfica orientada $G = (E, V)$ es una colección de vértices $v \in V$ y ejes $e \in E$. Los ejes son pares ordenados de vértices (v, v') . Se dice que el eje cuyos vértices están divididos en las disyunciones L y R ; los ejes se unen únicamente en pares de vértices (v, v') con $v \in L$ y $v' \in R$. Los $2m^n$ vértices de la gráfica $G_n | T$ corresponden a los mn textos planos y las mn figuras de texto encriptado. Un eje etiquetado por la clave k une el texto plano y el texto encriptado en los vértices $x(i)$, $y^i = T_k(x^i)$ respectivamente. Lo que indica que cada clave k aparecerá como una etiqueta en mn ejes.

2.5. Propiedades de los sistemas de información

Hay ciertas cosas que un sistema que maneja información no puede permitirse. Por ejemplo, resultaría inaceptable que los datos que se introducen en el mismo sufrieran alteraciones cuando se van a recuperar o que fuera posible para terceros consultar la información. En general, se definen tres propiedades fundamentales que deben proporcionar los sistemas informáticos: confidencialidad, integridad y disponibilidad.

- **Confidencialidad:** se dice que una información posee la característica de confidencialidad si sólo pueden tener acceso a ella las entidades que están autorizadas para ello. Gran parte de la información que hoy día manejan los sistemas informáticos perdería su valor si se hiciera pública por ejemplo: estrategias de empresas, secretos industriales, etcétera. Mientras que en otros casos su revelación sería sencillamente inaceptable tal es el caso de: expedientes médicos, información acerca de orientaciones políticas, religiosas o sexuales, etcétera.

Está claro que no es lo mismo salvaguardar la confidencialidad en un ordenador personal, con un acceso limitado al exterior, que en un sistema compuesto por múltiples computadoras interconectadas.

- **Integridad:** la integridad garantiza que, si se almacenan datos en el sistema o se colocan en un extremo de un canal de comunicaciones, van a permanecer inalterados al recuperarlos o al llegar al otro extremo. Es fácil imaginar situaciones en las que una información corrompida o manipulada puede llegar a ser altamente dañina. Para garantizar la integridad se emplean desde mecanismos de bajo nivel, que garantizan que los datos no se pierdan o corrompan accidentalmente, hasta mecanismos basados en funciones que garantizan que se puede saber si la data fue corrompida.
- **Disponibilidad:** la información suele alcanzar su verdadero valor cuando se utiliza. En muchos casos, es crucial emplearla en el momento oportuno, por lo que un buen sistema de información tendrá que proporcionar una adecuada flexibilidad a la hora de acceder a los datos. A la capacidad de tener acceso a la información en todo momento se denominará disponibilidad. En cierto modo, facilitar la disponibilidad puede dificultar mantener la confidencialidad y viceversa, por lo que, las tres características que se acaban de describir han de ser tenidas en cuenta de forma global a la hora de diseñar un sistema de información seguro.

2.6. Problemas de los sistemas de información

Todos los sistemas de información están sujetos a la posibilidad de experimentar un funcionamiento anómalo, ya sea de manera fortuita o provocada. El objetivo de la seguridad informática consistirá en evitar que se produzcan esas situaciones. Para ello, se deben tomar medidas a todos los niveles: diseño, implementación, políticas de uso, monitorización, análisis, etcétera. Para así, poder enfrentarlas más eficazmente, los principales fallos que puede sufrir un sistema criptográfico se describen a continuación.

- Daño: el perjuicio que se produce cuando un sistema informático falla. Dicho perjuicio debe ser cuantificable: el coste económico de los datos perdidos, el tiempo y esfuerzo necesarios para volver al estado normal, etcétera. El daño puede ser provocado o producirse de manera accidental.
- Ataque: un ataque es el acto deliberado de intentar provocar un daño concreto en un sistema. Los ataques constituyen una de las mayores amenazas a la seguridad informática, ya que pueden llegar a ser extremadamente sofisticados. Aunque pueden darse por fallas del equipo utilizado para el proceso de información.
- Riesgo: una vez definido el concepto de daño, el objetivo será proteger el sistema frente al mayor número posible de situaciones de este tipo. Sin embargo, la cantidad de recursos que se pueden asignar a esta tarea estará siempre limitada, por lo que a la hora de priorizar el trabajo se tendrá que medir de alguna manera el peligro asociado a cada situación de daño.

El riesgo (R) puede definirse como el producto entre la magnitud de un daño (d) y la probabilidad de que este tenga lugar (pd):

$$R=d \cdot pd$$

Es fácil observar que una situación con un elevado nivel de daño, pero muy poco probable, puede suponer menor riesgo que una situación con un nivel de daño moderado, pero mucho más probable. Siempre y cuando se haga una buena estimación de los daños y las probabilidades de que se produzcan, se obtendrán unos valores de riesgo útiles.

- Amenaza: se entiende por amenaza aquella situación de daño cuyo riesgo de producirse es significativo. Puesto que el valor de riesgo R tiene un carácter relativo, será la tarea definir un umbral a partir del cual se considerará que un riesgo merece la pena ser tenido en cuenta o lo que es lo mismo, constituye una amenaza.
- Vulnerabilidad: una vulnerabilidad es una deficiencia en un sistema susceptible de producir un fallo en el mismo. Como puede observarse, existen vulnerabilidades de todo tipo, ya que ningún sistema es perfecto. No obstante y a efectos prácticos, solo se tendría en consideración aquellas vulnerabilidades que constituyen una amenaza. El objetivo será, por tanto, conocer y eliminar (o al menos mitigar) el mayor número posible de vulnerabilidades en los sistemas.

- **Exploit:** se llamará exploit a cualquier técnica que permita aprovechar una vulnerabilidad de un sistema para producir un daño en el mismo.

Usualmente, los *exploits* son publicados como prueba de concepto, para demostrar que una vulnerabilidad es capaz de dar lugar a un daño determinado. De hecho, es muy habitual encontrar vulnerabilidades potenciales en los sistemas, que aunque pueden suponer un riesgo, en realidad se desconoce si son realmente útiles para perpetrar un ataque.

2.7. Tipos de vulnerabilidades

Los tipos básicos de vulnerabilidades son:

- **Debidas al diseño:** suelen ser las más delicadas, ya que su corrección puede tener impacto a todos los niveles, complicando el sistema y aumentando los costes de manera inaceptable o incluso pueden resultar imposibles de eliminar. Por esta razón, es esencial que los temas de seguridad sean tenidos en cuenta a la hora de diseñar un sistema informático. Por ejemplo, en un Sistema Operativo diseñado sin tener en cuenta la posibilidad de existencia de múltiples usuarios.

Esto da en la práctica unos privilegios excesivos de uso en el trabajo diario, lo cual representa una clara vulnerabilidad. Lo ideal sería modificar el diseño para incorporar esta característica, pero eso probablemente hará que muchos programas diseñados para el sistema antiguo dejen de funcionar.

- Debidas a la implementación: todos los programas tienen fallos, debido a un código fuente incorrecto, a una mala utilización de las características del compilador, a una comprensión deficiente de las especificaciones, etcétera. Este tipo de errores es, con mucho, la mayor fuente de vulnerabilidades en los productos informáticos actuales, por lo que un administrador deberá conocerlos en la medida de lo posible. Aunque suele tenerse menos en cuenta, la configuración de *hardware* escogida para un sistema también forma parte de su implementación y puede asimismo, dar lugar a la aparición de vulnerabilidades.
- Debidas al uso: de nada sirve tener el mejor sistema de contraseñas del mundo si luego se anota la palabra clave en un papelito y se pega al monitor con cinta adhesiva. Aunque suele dejarse de lado en demasiados casos, las condiciones concretas y los protocolos específicos de utilización de cualquier sistema informático deben ser diseñados cuidadosamente por un lado y seguidos escrupulosamente por otro.

En caso contrario, un sistema bien diseñado e implementado puede volverse del todo inseguro. Otra situación muy común, especialmente en el *software* comercial, es la inclusión de opciones por defecto inseguras. Una instalación apresurada o poco cuidadosa del sistema puede, por lo tanto, dar lugar a una configuración claramente vulnerable.

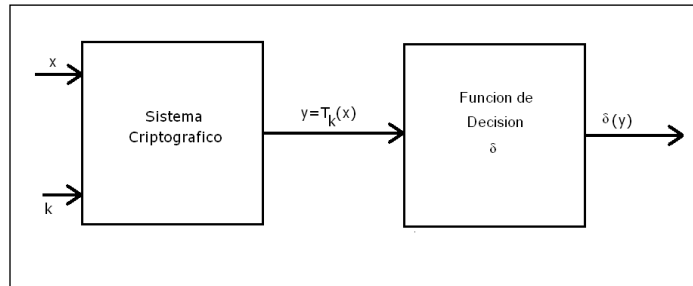
2.8. Criptoanálisis por el oponente bayesiano

La formulación bayesiana de criptoanálisis, se basa en que el analista tiene conocimiento de las claves por medio de su distribución de probabilidad; $\Pr_{KEY} | k$ es la primera estimación del analista de que el usuario seleccione la clave k . Se asuma que la selección de la clave y la generación de texto plano son procesos estadísticos independientes.

Se dice que $T = \{T_k : k \in K\}$ es un sistema criptográfico. Los usuarios acuerdan que seleccionarán la clave utilizando una transformación particular proveniente de la familia T . Se asume que el analista observa texto encriptado y encriptado por alguna transformación en T el sistema T es conocido por el analista, lo que significa que el tiene una descripción del espacio de la clave K y la forma en que la clave K en K determina T_k , pero no una clave en particular que algún usuario haya seleccionado.

La relación del analista con el sistema criptográfico se muestra en la siguiente figura:

Figura 10. Diagrama de flujo en criptoanálisis por oponente bayesiano



Fuente: G. KONHEIM, Alan. Cryptography: A Primer. p. 31.

El problema del analista es estimar el texto plano, así como la clave basada en la información proporcionada por:

- El texto encriptado (y)
- Conocimiento de la familia de transformaciones criptográficas $\{T_k : K \in \mathcal{K}\}$.
- La distribución de probabilidad del texto plano \Pr_{PLAIN} .
- La distribución de probabilidad de las claves \Pr_{KEY} .

El encriptamiento transforma un par (x,k) de clave y texto plano en texto encriptado $y = T_k(x)$.

La probabilidad a priori de la distribución \Pr_{PLAIN} y \Pr_{KEY} presentan las siguientes distribuciones conjuntas y condicionadas.

- $\Pr_{PLAIN,KEY} X, K$: distribución conjunta de texto plano y clave.
- $\Pr_{PLAIN,CRIPHER} X, Y$: distribución conjunta de texto plano y encriptado.
- $\Pr_{KEY,CRIPHER} X, K$: distribución conjunta de texto encriptado y clave.
- $\Pr_{CRIPHER} Y$: distribución marginal del texto encriptado Y.
- $\Pr_{PLAIN / CRIPHER} X / Y$: distribución condicional del texto plano X dado el texto encriptado Y.
- $\Pr_{KEY / CRIPHER} K / Y$: distribución condicional de la clave K dado el texto encriptado Y.

Las cuales están definidas por:

- $\Pr_{PLAIN,KEY} X, K = \Pr_{PLAIN} X \Pr_{KEY} K$
- $\Pr_{PLAIN,CRIPHER} X, Y = \sum_{K \in \mathcal{K}} \Pr_{PLAIN,KEY} X, K \Pr_{CRIPHER} Y, K$
- $\Pr_{KEY,CRIPHER} X, K = \sum_{Y \in \mathcal{Y}} \Pr_{PLAIN,CRIPHER} X, Y \Pr_{KEY,CRIPHER} Y, K$
- $\Pr_{CRIPHER} Y = \sum_{X \in \mathcal{X}} \Pr_{PLAIN,CRIPHER} X, Y$

- $\Pr_{PLAIN/CRYPTER} X/Y =$

- $\Pr_{KEY/CRYPTER} K/Y =$

Las distribuciones condicionales están definidas únicamente cuando

$$\Pr_{CRYPTER} Y/C$$

La distribución de probabilidad \Pr_{PLAIN} y \Pr_{KEY} y el texto encriptado interceptado $Y = T_K(x)$ constituyen la información total por medio de la cual el oponente basará la estimación del texto plano x y la clave K . la acción tomada por el oponente es representada en la figura por medio de la función de decisión, la cual hace la decisión:

“Texto codificado y originado del encriptamiento de texto plano $\delta(y)$ ”

Definición 2.9.1:

Una decisión determinántica de la función $\delta = \{\delta^n\}$ es una secuencia de transformaciones

$$\delta^n : Z_{rn} \rightarrow Z_{rn} \quad n = 1, 2, \dots$$

$$\delta : Y \rightarrow \delta(Y)$$

Se denota la familia de funciones determinánticas por Δ_D .

La decisión del oponente $\delta|y|$ es la misma para todos los pares (x,k) que satisfacen $T_k|x|=Y$. Aunque la función de Δ_D producirá un estimado incorrecto del texto plano para el oponente para cada par (x,k) que satisfaga $\delta T_k|x| \neq x$. Para comparar la función de decisión, se introduce la función de pérdida L_δ definida por:

$$L_\delta|x,y| = \begin{cases} 1 & \text{si } \delta|y| \neq x \\ 0 & \text{si } \delta|y| = x \end{cases}$$

$L_\delta|X,T_k|x|$ asigna el valor de pérdida cero a la decisión correcta y una pérdida de 1 a la decisión incorrecta. La pérdida promedio de la función determinística de decisión d en los símbolos de expectación de la variable aleatoria $L_\delta|x,y|$ que respecto a la función de probabilidad conjunta $P_{FLAIDRIPE}$ en el set de símbolos de pares (X,Y) es representada por :

(Ecuación16)

La pérdida promedio de la función es la secuencia de valores:

$$l_{\delta,n} = \int_{\Omega} L_\delta|x,y| dP$$

La estrategia bayesiana es usar una función en que minimice la pérdida $l_{\delta,n}$ para cada n, $1 \leq n < \infty$.

Definición 2.9.2:

Una función de decisión determinística δ^* es óptima sí:

$$l_{n,\delta^*} \leq l_{n,\delta} \text{ para cada } \delta \in \Delta_D \text{ con } n=1,2,\dots.$$

Definición 2.9.3:

Una función de decisión δ_n es bayesiana sí:

~~PRIPPER~~

$$\delta_B|y = x \text{ sólo sí } \text{PRIPPER}$$

La función de decisión δ_B está definida por el texto encriptado y que satisface PRIPPER .

Hay que notar que puede haber más de un símbolo en el texto plano x que pueda maximizar la probabilidad condicional posterior PRIPPER por lo cual la posibilidad de que exista más de una función de decisión bayesiana δ_B . La función de decisión óptima funciona cuando es fácil de describir Δ_D .

Teorema 2.9.1:

Una función de decisión determinística δ es óptima sí y sólo sí es bayesiana. Se puede comprobar el anterior enunciado de la siguiente manera:



$l_{n,\delta}$ Será mínimo solamente si $\delta^n | y$ hace un mínimo para cada figura y de probabilidad positiva.

Por lo tanto $l_{n,\delta}$ será un mínima sí y sólo sí $\delta^n | y$ satisface:



De esta manera se prueba que la función bayesiana de decisión δ_B es óptima en la clase de función determinística de decisión.

Definición 2.9.4:

Una función de decisión estocástica δ es la secuencia de matrices m^n por m^n y es representada de la siguiente manera:



Donde las entradas de $\delta^n | x, y$ son números reales no negativos. La suma de entradas en δ^n sobre todas las filas es igual a uno sí:

$$\delta^n | x, y \geq 0$$

$$\mathbf{1} = \sum_x \delta^n(x|y) \quad \text{Para todo } y \in Z_{m,n}$$

La familia de funciones de decisión estocásticas es representada por el símbolo Δ_s .

La decisión se hace calculando la probabilidad de ocurrencia de un símbolo en el texto plano con probabilidad $\delta^n(x|y)$, una vez se ha interceptado el símbolo correspondiente en el texto encriptado.

Una función de decisión determinística δ es un caso especial de una función de decisión determinística $\delta^n(x|y)$ en el que la entrada en la fila "y" en la columna $x = \delta^n(x|y)$ es igual a 1 y cero para todas las demás entradas.

Una matriz de r por s $S = [s_{ij}]$ es llamada matriz estocástica si cumple con:

$$s_{ij} \geq 0 \quad 0 \leq i < r, \quad 0 \leq j < s$$

$$\mathbf{1} = \sum_{0 \leq j < s} s_{ij} \quad 0 \leq i < r$$

Una matriz estocástica P en la que cada columna contiene una entrada simple que es igual a 1, es llamada matriz de permutación, cada matriz estocástica S puede ser expresada como la combinación convexa de un número finito de matrices de permutación de la siguiente manera:

$$S = \sum_i \lambda_i P_i$$

$$\lambda_i \geq 0 \quad 1 = \sum_i \lambda_i$$

Una función estocástica δ es una combinación combexa de funciones de decisión determinísticas en el sentido que:

$$\delta(x,y) = \sum_i \lambda_i \delta_i(x,y) \quad n=1,2,\dots$$

$$\lambda_{i,n} \geq 0 \quad 1 = \sum_i \lambda_{i,n}$$

La decisión tomada por el oponente ulizando la función anterior es un proceso de 2 pasos:

Paso 1: se realiza un experimento de ocurrencia; de lo que se obtiene δ_i^n con probabilidad $\lambda_{i,n}$.

Paso 2: se hace la decisión asociada con la función determinística δ_i^n .

Teorema 2.9.2:

Una función de decisión estocástica δ es óptima sí y sólo sí cumple con:

$$\delta^n(x,y) \geq 0$$

Lo cual implica ~~para~~ para pares de figuras (x,y) y cada $n=1,2,\dots$

Si hay más de un texto plano x que maximice $\Pr_{PLAIN|x}$ para algún y o equivalente, si existe más de una función determinística de decisión bayesiana δ_B . Las probabilidades $\lambda_{i,n}$ pueden ser asignadas de cualquier forma de la familia de decisiones bayesianas determinísticas δ_i^n generando las funciones óptimas de decisión. Cada función óptima de decisión tiene la misma pérdida promedio.

La decisión bayesiana:

Una función de decisión bayesiana provee un texto plano acorde a una muestra lo suficientemente grande de texto encriptado.

2.9. Secretividad perfecta

La información de un texto plano x que un oponente tiene disponible se obtiene por medio de la distribución de probabilidad \Pr_{PLAIN} . Suponiendo que un oponente quiere predecir qué texto será transmitido a continuación antes de observar el texto encriptado que fue transmitido, ¿cuál es la mejor estimación bayesiana antes de interceptar texto encriptado? La mejor aproximación a *priori* es cualquier figura que maximice la probabilidad $\Pr_{PLAIN|x}$.

Después de interceptar el texto encriptado, la estimación del oponente puede ser revisada, utilizando la información contenida en el texto interceptado. Es posible que el texto encriptado que se intercepto no de información adicional, esto se comprueba si la distribución de probabilidad $\Pr_{PLAIN|x}$ después de haber interceptado el texto encriptado y sea igual a la probabilidad calculada a *priori* \Pr_{PLAIN} .

$$Pr_{\text{PLAIN} \rightarrow \text{PLAIN}}^{\text{PLAIN}} \text{ Para toda y con } Pr_{\text{CRIPER} \rightarrow \text{PLAIN}}^{\text{PLAIN}}$$

Si se cumple lo anterior el conocimiento del texto encriptado no mejora la cantidad de información que conoce el oponente respecto al texto plano x que fue transmitido. En este caso, el sistema criptográfico T tiene secretividad perfecta según Shannon:

$$Pr_{\text{PLAIN} \rightarrow \text{PLAIN}}^{\text{PLAIN}} = Pr_{\text{PLAIN} \rightarrow \text{PLAIN}}^{\text{PLAIN}}$$

Siempre que:

$$Pr_{\text{PLAIN} \rightarrow \text{PLAIN}}^{\text{PLAIN}} = Pr_{\text{CRIPER} \rightarrow \text{PLAIN}}^{\text{PLAIN}}$$

La secretividad perfecta es también equivalente en la siguiente condición:

$$Pr_{\text{CRIPER} \rightarrow \text{PLAIN}}^{\text{PLAIN}} \text{ Para todo } x \text{ con } Pr_{\text{PLAIN} \rightarrow \text{PLAIN}}^{\text{PLAIN}} > 0$$

Teorema 2.10.1: para que la ecuación $Pr_{\text{CRIPER} \rightarrow \text{PLAIN}}^{\text{PLAIN}} = Pr_{\text{PLAIN} \rightarrow \text{PLAIN}}^{\text{PLAIN}}$ sea válida para todas las figuras "x" e "y" para los cuales $Pr_{\text{PLAIN} \rightarrow \text{PLAIN}}^{\text{PLAIN}} > 0$ y $Pr_{\text{CRIPER} \rightarrow \text{PLAIN}}^{\text{PLAIN}} > 0$, es necesario que haya al menos tantas claves en K como figuras en x de probabilidad positiva.

Entropía

Las probabilidades condicionales

$$P_{\text{PLAIN}}(x_j | x_1, \dots, x_{j-1})$$

$$P_{\text{PLAIN}}(x_j | x_1, \dots, x_{j-1})$$

$$(x_j - k) \pmod{m}$$

Serán maximizadas por $(x_j - k) \pmod{m}$ y $j = k$ para algunas fuentes de texto plano y sistemas encriptados. Si estas probabilidades condicionales varían monótonamente con n , si se incrementa la cantidad de texto encriptado mejorará la probabilidad de estimar la clave correcta.

Desafortunadamente la variación respecto a n no es monótona.

La equivocación de texto plano dado el texto encriptado será definida cortamente como un promedio de las probabilidades condicionales

$$P_{\text{PLAIN}}(x_j | y)$$

Esto muestra la variación monótona deseada con n la palabra equivocación sugiere indeterminación, lo cual es un sinónimo de entropía, un concepto que fue originado en la termodinámica y se ha hecho ampliamente aplicable en Matemáticas. La entropía de una distribución de probabilidad p es:

$$H(P) = -\sum_{i=1}^m p_i \log p_i$$

$$p_i \geq 0 \quad 0 \leq i < m$$

$$H(P) = -\sum_{i=1}^m p_i \log p_i$$

Es el número $H(P)$ definido por

$$H(P) = -\sum_{i=1}^m p_i \log p_i$$

(Ecuación 17)

Resaltando:

Si $p_i = 0$, la indeterminación de $0 \log 0$ es evaluada en 0.

La selección de la base del logaritmo, es cuestión de gusto; las diferentes bases solamente cambian en $H(P)$ por una constante multiplicativa. Por convención en la teoría de información se utiliza logaritmo base 2.

La entropía ha sido definida como un número en $H(P)$ asociado con la distribución de probabilidad P , también se habla de entropías en variables aleatorias, en donde significa lo siguiente:

Si X es una variable aleatoria con una distribución de probabilidad

$$p_i = P\{X=i\} \quad 0 \leq i < m$$

La entropía de X es la entropía de la distribución probabilidad de X

$$H(X) = -\sum_{i=1}^n p_i \log p_i$$

Si X_0, X_1, \dots, X_{n-1} son variables aleatorias con distribución de probabilidad conjunta:

$$P(X_0, X_1, \dots, X_{n-1})$$

La entropía de la variable aleatoria $(X_0, X_1, \dots, X_{n-1})$ es:

$$H(X_0, X_1, \dots, X_{n-1}) = -\sum_{i_0, i_1, \dots, i_{n-1}} P(i_0, i_1, \dots, i_{n-1}) \log P(i_0, i_1, \dots, i_{n-1})$$

La entropía condicional de X dado Y=j es la entropía condicional de la distribución de X dado Y=j:

$$H(X|Y=j) = -\sum_{i=1}^n p_{X|Y=j}(i) \log p_{X|Y=j}(i)$$

La entropía de X dado Y es definida por:

$$H(X|Y) = -\sum_{i=1}^n \sum_{j=1}^m p_{X,Y}(i,j) \log p_{X,Y}(i,j)$$

Donde $p_{X|Y=j} = \sum_{i=1}^n p_{X,Y}(i,j)$ ya que $p_{X|Y} = \sum_{j=1}^m p_{X,Y}(i,j)$ donde $p_Y(j) = P_r \{Y=j\} = \sum_{i=1}^n p_{X,Y}(i,j)$, y se tiene:

~~$$H(X) = \sum_i p_i \log \frac{1}{p_i}$$~~

Teorema 2.10.2:

$$H(X, Y) = H(X) + H(Y|X)$$

$$H(X, Y) = H(Y) + H(X|Y)$$

El teorema anterior describe los relacionamientos básicos de entropía.

La ecuación ~~$H(X, Y) = H(X) + H(Y|X)$~~ es un miembro de la familia de relaciones de entropía. La definición de $H(X)$ es independiente de la dimensión de una variable aleatoria, lo que significa que $H(X)$ es definida en la misma forma si:

- X es una variable aleatoria tomando valores de Z_m .
- X es una variable aleatoria de vector evaluado $(X_0, X_1, \dots, X_{n-1})$ donde cada uno de sus componentes toman valores de Z_m .

Teorema 2.10.3:

Para todos los vectores (m) ~~$H(X) \leq H(X, Y)$~~

El máximo valor de $H(p)$ es $\log m$, y es alcanzado sí y sólo sí $p_t = 1/m$ para $0 \leq t < m$

Teorema 2.10.4:

$$H(X|Y) \leq H(X)$$

$$H(Y|X) \leq H(Y)$$

En ambas ecuaciones la condición de igualdad se cumple únicamente si las variables aleatorias son independientes.

2.10. Sistemas criptográficos aleatorios

Aún para modelos sencillos de sistemas criptográficos y de texto plano, la calculación numérica de equivocación que puede ser hecha es únicamente para valores pequeños de n , debido al crecimiento exponencial del número de símbolos respecto de n , para obtener resultados para un n grande, Shannon utilizó una técnica llamada randomización. La cual se empleó con éxito en su escrito en 1948 para probar el teorema de codificación, en la teoría de la información.

En randomización, un simple sistema criptográfico T es reemplazado por una acoplación de sistemas $\{T^n\}$ con una distribución de probabilidad. Las expectativas son calculadas con respecto al estable, si el valor esperado relativo al ensamble es alguna variable que satisface ciertas condiciones, se puede deducir que al menos uno de los sistemas criptográficos que componen el ensamblaje satisface la condición. Y todos los sistemas que componen el ensamblaje se acercan en algún sentido a satisfacer la condición. La ventaja técnica en randomización es la habilidad para evaluar una expectativa que es difícil de calcular para miembros individuales del ensamblaje.

Se hacen las siguientes suposiciones acerca del texto plano y el espacio de claves.

- Hay N_k claves en el espacio de claves K cada una con probabilidad $\frac{1}{N_k}$.
- El arreglo de símbolos $Z_{m,n}$ está dividido en 2 partes $Z_{m,n,H}$ y $Z_{m,n,L}$.
- Los símbolos en $Z_{m,n,H}$ son todos de probabilidad $\frac{1}{|Z_{m,n,H}|}$, donde $|Z_{m,n,H}|$ es igual al número de símbolos en $Z_{m,n,H}$.
- Los símbolos en $Z_{m,n,L}$ son probabilidad total \in .

Por ejemplo. $Z_{26,n}$ contiene $26^n = 2^{470n}$ símbolos. Es decir, que $R_n = \log_2 |Z_{m,n}|$.

Para ϵ cercano a 0, la entropía $H(X_1, \dots, X_n)$ en la primera letra n del texto plano es aproximadamente R_n , y R_n/n es la tasa aproximada a la que la fuente S produce las letras. La redundancia en la fuente S es la diferencia $D_n = H(X_1, \dots, X_n) - R_n$ y D_n/n es la redundancia promedio para cada letra del texto plano.

La randomización se introduce de la siguiente manera:

Considere un experimento con espacio de muestra $\Omega' = \{\omega'\}$ y una distribución de probabilidad Pr' . Si T es una variable aleatoria definida en Ω' , cuyo valor $T(\omega')$ en el punto de muestra ω' es un sistema criptográfico en el sentido de la primera definición de sistemas criptográficos. Lo que significa:

$$T(\omega') = K$$

(Ecuación 18)

El compuesto de sistemas criptográficos aleatorios es la colección de sistemas criptográficos:

$$T(\omega') = K$$

Donde ω' sirve como etiqueta para identificar los miembros del compuesto y varía respecto al espacio de muestras Ω' . Se utiliza el símbolo (') para distinguir entre Ω' que etiqueta los sistemas criptográficos aleatorios de Ω que etiqueta la generación de texto plano y texto criptográfico. Si se asume que el mecanismo de prueba que genera el compuesto de sistemas criptográficos satisface la condición siguiente $T|\omega'$ $\omega' \in \Omega'$ si:

$$P_{E_k|y} = 2^D \quad \text{Y si } |k, y \neq j, z \text{ se tiene:}$$

Donde k es una clave e y es una figura del texto plano entonces:

$$P_{E_k|y} = 2^D \quad \text{Y si } |k, y \neq j, z \text{ se tiene:}$$

$$P_{E_k|y} = 2^D$$

Un sistema criptográfico puede ser representado por una secuencia de gráficas bipartitas. Cada miembro del ensamblaje $T|\omega' \in \Omega'$ corresponde a una secuencia de gráficas. La condición $P_{E_k|y} = 2^D$ establece el subconjunto $E_k|y$ del espacio de muestras Ω' , que consiste de sistemas en los cuales el límite etiquetado por la clave K crea un vínculo entre los símbolos de texto plano en $Z_{m,n,H}$ y los símbolos del texto encriptado.

Teorema 2.11.1:

La equivocación promedio de la clave dado el texto encriptado en el arreglo es:



(Ecuación 19)

3. APLICACIONES

3.1. Seguridad en archivos y comunicación

La aplicación de sistemas criptográficos en la protección de sistemas que procesan información tipificada de la manera en que la criptografía ha respondido a los cambios tecnológicos. Se examinarán las dos principales aplicaciones de los métodos criptográficos en el proceso de información.

- Protección de sistemas de comunicación entre terminales y servidores.
- Protección de archivos en línea.

El uso de métodos criptográficos en sistemas de información difiere del uso clásico que la criptografía tiene para aplicaciones militares o diplomáticas en un aspecto crucial; el cómo. En ambas situaciones la clave se distribuye por un canal o método seguro. Por ejemplo, para enviar la clave de algún mensaje hacia una embajada en un país enemigo por parte del departamento de estado se utiliza un diplomático como mensajero.

En el caso de la informática, la clave debe residir en el sistema y este está expuesto a ser penetrado por un intruso, por lo que se debe recurrir a métodos criptográficos para proteger los archivos que guardan las claves que dan acceso a diferentes niveles dentro de un sistema para evitar que alguien externo o no deseado las utilice y pueda dañar el sistema.

Algunas de las principales aplicaciones de criptografía en los sistemas de proceso de información son:

Terminales bancarias: algunos bancos permiten en varios puntos la ejecución de algunas transacciones, como pago de préstamos, transferencias entre cuentas de cheques y ahorros, entrega de efectivo, etcétera. En terminales no atendidas del banco, para esto el banco le da a cada cuentahabiente una tarjeta en la cual la clave de acceso o número de cuenta está almacenado magnéticamente, esto se conoce como ACCT y una clave secundaria conocida como PIN (Número Personal de Identificación).

Una transacción funciona así:

- ACCT es leída de la tarjeta en el terminal.
- Se pide que el PIN sea ingresado por el teclado numérico.
- Se hace la requisición de transferencia.

El sistema autentica la transacción de la siguiente manera:

- ACCT es un número válido de cuenta.
- El PIN que proporcionó el usuario es válido para el ACCT presentado.

El PIN es una clave secundaria la cual permite evitar que parte de la información privada pueda ser accedida por una tercera persona que ha robado una tarjeta, también juega un papel de firma durante una transacción de pago por medio de tarjeta, el banco indica a sus clientes que se memoricen el PIN y que no lo escriban en la tarjeta para mantener el nivel de seguridad.

Por lo que el PIN debe consistir de una pequeña cantidad de números que es usualmente de 4 a 6. Si el PIN es lo suficientemente corto para facilitar la memorización, se haría posible utilizar un ataque de fuerza bruta para acceder a la información de la cuenta, por otra parte, si el PIN es suficientemente largo para evitar este método, los usuarios terminarían escribiendo el PIN en la tarjeta, volviendo este método inseguro y haciendo inútil el uso del PIN ya que no habría una segunda clave separada.

Cualquier sistema de transacción es vulnerable a ataques si la comunicación se hace por medio de texto explícito.

Un posible atacante podría hacer lo siguiente:

- Grabar el mensaje de respuesta, que autoriza la terminal despachar efectivo.
- Cortar la comunicación entre el terminal y el servidor y conectar un servidor falso a la terminal.
- Enviar el mensaje a la terminal, con lo cual se le despacharía efectivo.

La transmisión encriptada no protege del todo de este tipo de ataque, ya que si el atacante obtiene el mensaje encriptado que autoriza la transacción y ahora lo sustituye por el inciso 3 anterior.

- Retransmitir el texto encriptado a la terminal, esta despacharía efectivo.

Este tipo de ataque únicamente es deshabilitado si el atacante no es capaz de construir una respuesta válida. El mensaje que le permite a la terminal despachar efectivo contiene algunas variables y previene el reuso de mensajes anteriores.

Una solución es incorporar alguna combinación del número de transacción o número de recibo en el mensaje, lo cual cambia con cada transacción y causa una impredecibilidad en el cambio que tiene el texto encriptado. En este caso la información obtenida de un primer mensaje de requisición y una respuesta válida no daría suficiente información para recrear una respuesta válida a una nueva requisición.

Claves por tarjetas personales: un método estándar para limitar el acceso a un edificio es el de utilizar una tarjeta como llave, la cual contiene almacenada magnéticamente el código que permite identificar a cada empleado y le da acceso a los lugares que este está permitido. La pérdida de una sola tarjeta deja comprometida la seguridad.

Estas tarjetas son equivalentes a una llave ordinaria que se le da a cada empleado y pueden ser implementadas criptográficamente como sigue: un número es asignado a cada habitación del edificio, (FACNUM), un número elegido aleatoriamente decimal es seleccionado para cada empleado. Se encripta la unión del número aleatorio (RAN) y el número de las habitaciones a las que este está permitido ingresar FACNUM-RAN1 por lo que el código sería ~~RAN-GR~~ el cual sería guardado en la tarjeta para cualquier empleado.

Para ingresar a cualquier lugar el empleado inserta la tarjeta en el lector, se desencripta el mensaje y se revisa si es un código de autorización válido para la habitación donde se está pidiendo ingreso y también se revisa y compara con una lista de perdidos/robados, si la tarjeta fue robada se le asigna un nuevo RAN al empleado y el antiguo se adjunta a la lista de perdidos/robados.

Palabras clave: un sistema de control de acceso convencional está basado en palabras clave (*passwords*), a las cuales se hace referencia en el ingreso. Pero estos *passwords* se almacenan en la memoria del sistema, por lo que son vulnerables a ser robados, borrados o modificados, una forma alterna de hacerlo es encriptar los *passwords* con una llave K y encriptar el *password* que inserta cada usuario y compararlo con este código, de esta forma se deberá proteger únicamente la llave K y el sistema chequea la valides encontrando la llave K de cada *password* recibido y de esa forma identifica a los diferentes usuarios.

Encriptación de los archivos en línea:

Incluye las aplicaciones siguientes:

- APL aéreas de trabajo guardado como archivos CMS.
- Bases de datos que contienen información valiosa; por ejemplo, los datos de diseño de un circuito electrónico.
- Archivos que contienen claves criptográficas generadas por la modificación repetitiva y encriptación aleatoria de una cadena de texto.

3.2. Administración de la clave

En esta sección se estudiarán algunas formas de administrar de manera eficiente la clave para mantener un sistema seguro, así como los conceptos básicos necesarios para llevarlo a cabo.

3.2.1. Aspectos generales

Se debe tomar en cuenta la importancia que tiene la clave para el desarrollo de un sistema criptográfico confiable, por lo que la clave debe mantenerse en secreto ya que la seguridad de la mayoría de los algoritmos criptográficos dependen en gran parte, a qué tan secreta es su clave generada, sin importar qué tan ingeniosos y complicados sean los algoritmos cualquiera que tenga acceso a la clave es capaz de acceder a la información o acceder al sistema utilizando la identidad de alguien más.

Esto no es sólo aplicable a sistemas simétricos, que requieren que la clave se mantenga secreta, sino también, para sistemas asimétricos, que están basados en claves públicas y privadas. En general, se puede declarar que el manejo que concierne a la clave va desde el punto en el que esta es creada hasta el punto donde esta es destruida.

Por lo que se tratará de cubrir en este tema:

- Generación
- Distribución
- Almacenamiento
- Reemplazo/cambio
- Uso
- Destrucción final

Un sistema no sólo debe protegerse de que intrusos obtengan una llave, sino además debe evitar usos no autorizados de las llaves, modificaciones deliberadas y otras formas de manipulación a las que las claves puedan estar expuestas, también es deseable detectar cualquier situación en la que esto pueda ocurrir, una vez que la clave fue modificada o accesada de una forma no autorizada se debe dar por terminado su uso de forma inmediata.

Analizando los aspectos asociados al manejo de claves se tiene:

- Generación de clave

La clave generada debe ser tan impredecible como sea posible, si algunas claves tienen mayor probabilidad de ocurrencia que otras en práctica un intruso potencial podría descubrir fácilmente cual fue la clave que se utilizó. La clave impredecible puede ser generada utilizando un proceso pseudoaleatorio como el DES, el cual genera un patrón de 64-bit.

En la práctica por lo general un grupo de n personas generan cada uno su proceso aleatorio y con la unión de estos se crea la clave. Por ejemplo, se dice que $Z1$ es el número generado por el algoritmo de la persona 1 y $Z2$ el número generado por el algoritmo de la persona 2, la clave sería $K=Z1+Z2$, y ninguna de las 2 personas sabría cual es en realidad la clave, a esto se le conoce comúnmente como control dual.

Hay situaciones en las que la clave generada debe ser probada para verificar si tiene debilidades o semidebilidades, como en el caso de DES. Por ejemplo, en el algoritmo RSA los números p y q deben ser probados de manera que se pueda verificar que tan fuertemente son números primos.

- Distribución de la clave

El método más obvio de distribución de la clave es mano a mano. Este método era frecuentemente usado en los días de criptogramas hechos a mano, hoy en día es utilizado en muy raras ocasiones, ya que la distribución de las claves se hace en la mayoría de las veces en forma automática, esta forma no es sólo más conveniente sino que en ocasiones es esencial para que el criptograma funcione correctamente.

En un sistema de red de computadoras, con conexiones cliente-servidor, etcétera. Usualmente 2 tipos de claves son empleadas, claves que son utilizadas para la seguridad de los datos en el sistema y autenticación, llamadas de sesión y claves que son usadas para la seguridad de la sesión, mientras se transmite del terminal al servidor, estas son llamadas claves de transportación.

- Almacenamiento de claves

Se aplica el mismo tipo de situación en almacenamiento y en transportación, lo que significa que la clave de sesión se debe almacenar utilizando una clave de almacenaje, ambas claves la de transportación y la de almacenaje son también llamadas *meta keys*.

- Cambio de clave

Es evidente que un cambio regular de clave especialmente la de sesión es muy importante para la seguridad de un sistema. Un criptoanalista puede ser desanimado en su búsqueda por descifrar la clave si esta cambia de manera frecuente. Al igual que frustraría una búsqueda exhaustiva de clave, ya que para cuando se logra encontrar la clave por este método ya se tendrá otra clave en uso.

La frecuencia con la que se ha de cambiar la clave depende de:

- La sensibilidad de los datos encriptados: lo más importante es garantizar un nivel de seguridad alto, entre más frecuentemente se cambie la clave más alto es dicho nivel de seguridad.

- El período de validez de los datos protegidos: por ejemplo los mensajes que son transmitidos por un teléfono políaco tiene un tiempo limitado de validez; al tiempo de la transmisión puede que haya información valiosa para un intruso. Pero unos pocos minutos después ya se habrá perdido el contenido de este mensaje.
- La fuerza que tenga el sistema criptográfico: la clave de un sistema ordinario de aplicación del DES debe ser cambiada más frecuentemente que la clave de un sistema triple de DES.
- Destrucción de la clave

Esto debe ser efectuado simplemente borrando la clave de la memoria. La eliminación de información puede ser pasiva, removiendo la fuente de alimentación del chip de memoria o activa, sobrescribiendo datos sobre las casillas de memoria que ocupaba la clave.

- Manejo de llave en un sistema de procesamiento de información:

Cuando un sistema protegido por clave encriptada es utilizado para procesar información, debe provisionarse la distribución de la clave.

Para esto se requiere:

- Cargar las claves al sistema cuando sea necesaria una operación de encriptar/desencriptar.
- La clave o la manera de construirla debe estar en el servidor.

Si solamente es una comunicación terminal-terminal, que pasa por el servidor pero no se decripta ahí, no se necesita almacenar una clave en el servidor. En la actualidad, no existe un Sistema Operativo que sea capaz de almacenar de forma segura una lista extensa de claves de usuario y la implementación de un sistema cuya efectividad depende de que tan secreta se puede almacenar esta lista que ha sido muy cuestionada.

3.2.2. Claves de sesión

En 1977 IBM lanzó una gran variedad de productos criptográficos [IBM 1, 2], y para comunicaciones punto a punto utilizaba el algoritmo DES. El subsistema criptográfico de IBM está basado en el concepto de una clave de sesión (KS por sus siglas en inglés), compartida por el servidor y el terminal y utilizada únicamente durante el tiempo que dure la sesión.

Las características básicas del subsistema criptográfico de IBM son:

- Dos clases de claves: claves operacionales (equipo y sesión), las cuales sirven de encriptamiento y claves de sistema, utilizadas para encriptar las claves operacionales que están almacenadas en el servidor.
- No se almacena la clave de operación en el servidor.
- Una clave única de administrador de servidor, KMH (por sus siglas en inglés), almacenada en el servidor en un dispositivo seguro capaz de ejecutar un pequeño set de instrucciones. Hay dos variantes de clave maestra, KMH0 y KMH1, estas variantes se obtienen complementando ciertos bits en la KMH.

- Clave maestra de dispositivo o terminal: $KMT_1, KMT_2, \dots, KMT_N$, almacenadas de manera segura en los dispositivos o terminales. Una tabla de *host* que contiene $\{DES\{KMH1, KMT_i\}; i=1, 2, \dots, N\}$.
- Una clave de sesión KS (por sus siglas en inglés), es una clave operacional utilizada para encriptar la transmisión entre el servidor y el terminal. La clave de sesión es generada por el servidor al inicio de la sesión
- Cuatro instrucciones no privilegiadas para encriptar/desencriptar datos.
 - (i) ECPH: data encriptada.

$$[DES\{KMH0, KS\}, PLAIN] = DES\{KS, PLAIN\}$$

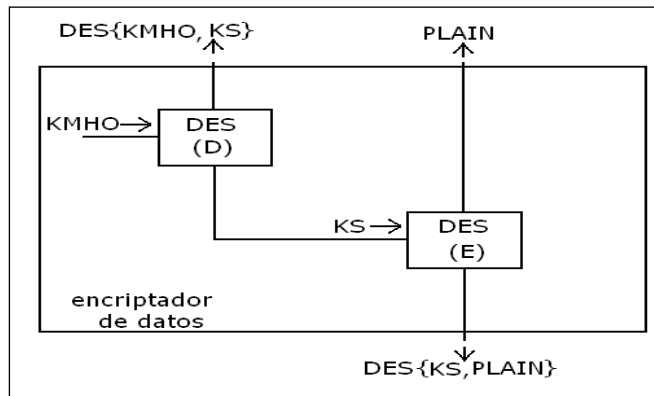
(Ecuación 20)

Los argumentos de ECPH son (1) el encriptamiento de la clave de sesión bajo KMH0, y (2) el texto plano PLAIN.

ECPH devuelve el encriptamiento de PLAIN bajo KS,

$$CRIPPER = DES\{KS, PLAIN\}$$

Figura 11. Diagrama encriptador DES



Fuente: G. KONHEIM, Alan. Cryptography: A Primer. p. 289.

(11) DCPH: desencriptador de datos.

$$DCPH[DES\{KMHO,KS\},DES\{KS,PLAIN\}]=PLAIN$$

(Ecuación 21)

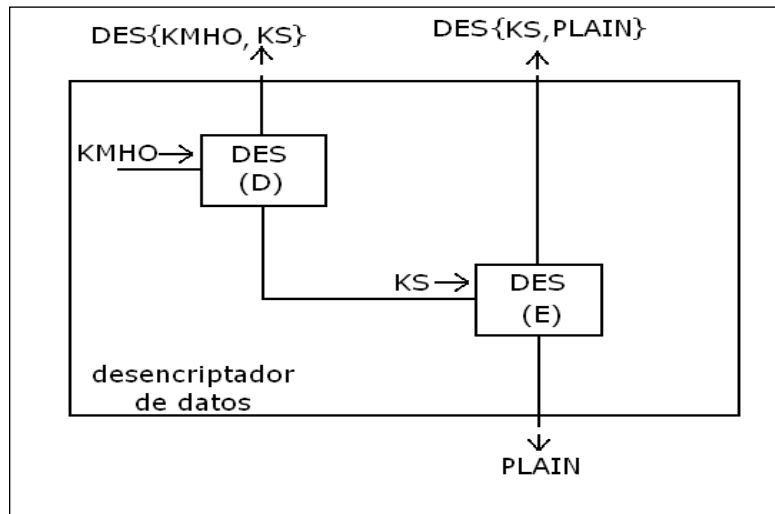
Los argumentos de DCPH son el encriptamiento de la clave de sesión KS bajo KMHO, y el texto encriptado.

$$CRIPPER=DES\{KS,PLAIN\}$$

DCPH devuelve el valor del texto plano PLAIN:

$$PLAIN=DES^{-1}\{KS;CRIPPER\}$$

Figura 12. Diagrama descriptador DES



Fuente: G. KONHEIM, Alan. Cryptography: A Primer. p. 289.

3.3. Sistemas de clave pública

Los sistemas criptográficos de clave pública basan su seguridad en que no haya un método eficiente de calcular la clave de descifrado, aún conociendo la clave de encriptación. Estos sistemas tienen la ventaja de que como la clave de encriptación no ayuda a calcular la clave de descifrado, pueden almacenarse las claves de encriptación de muchos usuarios en una guía pública a la cual todos tienen acceso, evitando así que cada vez que dos usuarios quieran comunicarse tengan que ponerse de acuerdo en una clave común.

Entre estos sistemas criptográficos se pueden mencionar:

- Sistema criptográfico RSA

Este sistema creado por Rivest, Shamir y Adle-man (RSA) es uno de los más famosos. La idea atrás de este sistema es la de construir una función que sea fácil de calcular (en este caso multiplicar dos primos), pero que su inversa sea difícil de calcular (en este caso dado un número que es producto de 2 primos, hallar esos primos).

Se elijen dos números primos grandes p y q y se calcula $n = pq$.

Se calcula $\phi(n) = \phi(p)\phi(q) = (p-1)(q-1)$.

Se elije un número aleatorio e con: $1 < e < \phi(n)$ y $\text{mcd}(e, \phi(n)) = 1$.

Con la ayuda del algoritmo de Euclides se calcula $d \in \mathbb{Z}^+$ tal que $de \equiv 1 \pmod{\phi(n)}$.

Al final se tiene una función definida por:

$$E: \mathbb{Z}_n \rightarrow \mathbb{Z}_n : E(x) = x^e \pmod{n}$$

La clave pública viene dada por el par (n, e) que puede ser publicada en una guía de claves. Se observa que el par brinda toda la información necesaria para calcular la función de encriptado E .

Si alguien desea enviar un mensaje confidencial x , busca la clave pública en la guía de claves y de esa forma puede enviar un mensaje encriptado $E(x)$.

La función de desencriptación se define de la siguiente manera:

$$D: Z_n \rightarrow Z_m : D(E(x)) = x^d \pmod{n}$$

Para calcular la función de desencriptado es necesario conocer d . Lo que significaría conocer los 2 números primos p y q que se utilizaron para generar la función de encriptación. De esta forma si se eligieron bien los parámetros la forma de encontrar los primos es factorizar n y esto tomaría demasiado tiempo, aún utilizando los mejores algoritmos de factorización conocidos hasta el momento y las computadoras más rápidas de la actualidad.

Entonces la seguridad de este sistema criptográfico se basa en la dificultad que existe al factorizar números grandes. Como el receptor conoce d , no tiene problema al momento de recibir el mensaje x para calcular la función de desencriptación $D(x)$, ya que cuenta con algoritmos eficientes para calcular potencias módulo n con el método de exponenciación rápida.

- Sistema criptográfico basado en el problema del logaritmo

Se asume que para una p suficientemente larga, el problema del logaritmo es inatacable computacionalmente. Dado $q^k \pmod{p}$, q y p , el número de operaciones requeridas para recuperar k es muy grande para ser considerado computacionalmente práctico. Teniendo esta afirmación se puede construir un sistema de llave pública fuerte [DI]. Este directorio se prepara con entradas para cada usuario del sistema.

El usuario selecciona una llave secreta y computa la entrada al directorio. Cuando el usuario i y el usuario j se desea comunicar siguen los siguientes pasos:

El usuario i utiliza la clave k_i y la entrada al directorio q^{kj} del usuario j para calcular.

$$k_{ij} = (q^{kj})^{k_i} \pmod{p} = q^{kj k_i} \pmod{p}$$

El usuario j utiliza la clave k_j y la entrada al directorio q^{ki} del usuario i para calcular.

$$k_{ji} = (q^{ki})^{k_j} \pmod{p} = q^{ki k_j} \pmod{p}$$

Como $k_{ij} = k_{ji}$ ambas partes han establecido una clave común.

Si hay alguna duda respecto a que la utilización repetitiva por los usuarios j e i pueda comprometer la clave, el protocolo de encriptamiento puede ser modificado; y $k_{ij} = k_{ji}$ serviría como una clave intermedia utilizada para encriptar alguna información conocida pero variable, de manera que se obtenga una clave común.

Los usuarios pueden comunicarse por medio de una clave compartida sin necesidad de revelar sus claves privadas al sistema. Sin embargo, el problema de la distribución de clave de usuario a usuario no ha sido resuelto por completo todavía. Debido a que el directorio debe ser accesible para todos los usuarios del sistema y por esa causa es susceptible a alteraciones durante algún ataque.

No se puede proteger realmente un directorio con 1 000 entradas, ya que demasiados usuarios tienen acceso total al directorio. Hay varias posibles soluciones para esta exposición, como proveer a cada usuario con un directorio encriptado en su clave privada o un directorio impreso con cada entrada.

- Sistema criptográfico basado en el problema de Knapsack.

Dado un entero N y un vector $a = (a_0, a_1, \dots, a_{n-1})$, el problema de knapsack es encontrar una solución $x = (x_0, x_1, \dots, x_{n-1})$ $x_i = 0$ o $x_i = 1$, para $0 \leq i < n$. De la ecuación
$$N = \sum_{0 \leq i < n} a_i x_i$$
.

Se refiere a x como la representación $\{a_i\}$ de n y a $\{a_i\}$ como la longitud de knapsack. Como x tiene el arreglo $Z_{2,n}$, la suma $\sum_{0 \leq i < n} a_i x_i$ toma valores en un arreglo denotado por $KNAPV[\{a_i\}]$. El cual contiene al menos 2^n entradas.

Un algoritmo logarítmico temporal para el problema de knapsack $ALG(\{a_i\}, N)$ es un algoritmo que:

Dada la longitud de knapsack $a = (a_0, a_1, \dots, a_{n-1})$ y N .

Encuentra una solución $x = (x_0, x_1, \dots, x_{n-1})$ para $N = \sum a_i x_i$.

O demuestra que no hay solución en el polinomio.

El problema Knapsack, pertenece a NP-completo y es difícil en el sentido de que no hay un algoritmo polinómico temporal conocido para resolver el problema general.

La afirmación anterior significa que existe al menos un problema knapsack difícil $\{a_i\}$. Para longitudes especiales de knapsack, encontrando una solución para el problema de knapsack no representa dificultad.

- Sistema de encriptamiento de River-Shamir-Adelman

Si se dice que $n = pq$ siendo p y q dos números primos distintos. Y si tanto e como d satisfacen: $ed \equiv 1 \pmod{\phi(n)}$, entonces los mapeos $E_{e,n}$ y $E_{d,n}$, son inversos en el arreglo Z_n . Ambas $E_{e,n}$ y $E_{d,n}$ son fáciles de computar, cuando e, d, p y q son conocidos.

Si e y n son conocidos pero p y q son secretos, se cree que $E_{e,n}$ es una función de un sentido; para encontrar $E_{d,n}$ a partir de n, es equivalente a la factorización de n. Cuando p y q son primos suficientemente grandes, la factorización de n aparenta ser poco práctica. Esta es la base del sistema RIVEST-SHAMIR-ADELMAN. La fuerza del sistema descrito anteriormente depende completamente de cuan complejo es encontrar los factores del entero n, el cual es producto de dos números primos grandes.

3.4. Firmas digitales y autenticación

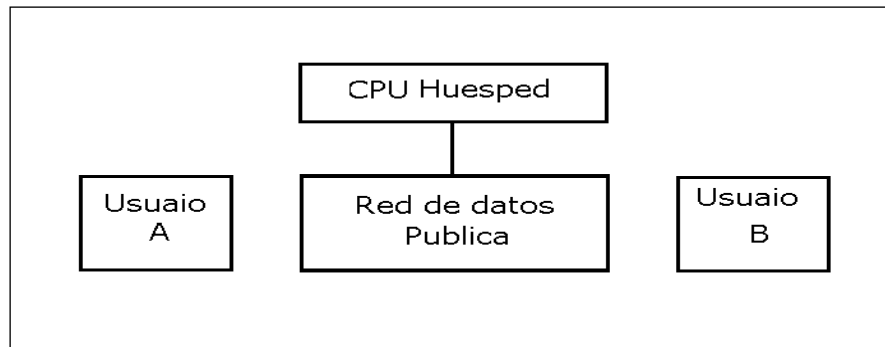
El por qué:

Se considera una transacción entre pares de usuarios de la siguiente manera. El cliente A hacia el corredor B: compra para mi cuenta... comparte XYZ al precio cooperativo no excediendo \$... El banco A hacia el banco B: cuenta de crédito 853 134 la suma de \$1 000.

En transacciones comerciales normales, se utilizan protocolos para proteger a todas las partes de ataques dañinos. Procedimientos arbitrarios son utilizados para resolver las disputas si no se acercan. Actualmente, el mundo gira rápidamente hacia una era digital donde las comunicaciones en las cuales se hacen transacciones por bienes y servicios deben ser manejadas por sistemas de procesamiento de datos que conectan usuarios por medio de líneas teleprocesadoras.

El elemento usual de contacto personal, físico y visual o vocal, puede ser ausente. Mensajes electrónicos son fácilmente creados o alterados. Por ejemplo, una secuencia de 0s y 1s en un mensaje que representa la transacción banco A hacia banco B; puede ser fácilmente cambiado para que \$1 000 se convierta en \$1 000 000 o reemplazar el número de cuenta 853 134 por 765 541. El protocolo en transacciones comerciales, el cual de alguna manera previene o complica el uso inadecuado detectando y reprimiendo, según sea necesitado en la definición para la transacción manejada por un sistema de procesamiento de datos.

Figura 13. **Diagrama de flujo firmas digitales**



Fuente: G. KONHEIM, Alan. Cryptography: A Primer. p. 331.

Amenazas:

Una transacción del originador al receptor, envuelve la transmisión de datos, comprometiendo a los usuarios en algún curso de la acción. Los datos pueden representar una transferencia de fondos interbanco, la venta de algunos productos en algún mercado o correo electrónico conteniendo material sensible. Los participantes requieren ser protegidos contra una variedad de actos dañinos, incluyendo:

- Renunciar: el originador subsecuentemente abandona una transacción
- Falsificación: el receptor fabrica una transacción
- Alteración: el receptor altera una transacción previamente válida
- Enmascarado: un usuario intenta hacerse pasar por otro

Para la verificación de una transacción se requiere:

Verificar datos de usuario A hacia usuario B, el protocolo debe incluir los siguientes elementos:

- El originador (A) debe incorporar con los datos una firma; información adicional que depende de los datos, el receptor de la transacción y la información secreta K_A , conocida únicamente por el originador. $SIG\{k_A, DATA, UsuarioB\}$ denota la firma de datos hacia usuario B transmitida por el Usuario A.
- Una firma correcta de datos, hacia usuario B debe ser efectivamente imposible de construir sin k_A .
- El usuario B debe ser capaz de verificar que $SIG\{k_A, DATA, UsuarioB\}$ es una firma correcta del usuario A.
- El proceso de verificación debe envolver dependencia de tiempo para prevenir la reutilización de mensajes viejos.

Autenticación:

Es un proceso por medio del cual cada una de las partes de una comunicación verifica la identidad del otro. Cualquier proceso de autenticación incluye algo de secreto implícitamente, alguna información por medio de la cual algún participante de la comunicación se identifica al mismo en el otro punto.

Para que el originador pueda ser autenticado en el receptor en un sistema de proceso de información se muestran los siguientes pasos:

- El originador debe proveer al receptor información de autenticación $AUTH\{k_A, UsuarioB\}$ dependiendo del secreto de la información k_A debe ser conocida únicamente por el usuario A.
- Una autenticación válida $AUTH\{k_A, UsuarioB\}$ por el usuario A para el usuario B debe ser prácticamente imposible de construir sin k_A .
- El usuario B debe tener un procedimiento para comprobar que $AUTH\{k_A, UsuarioB\}$ autentica de manera correcta al usuario A.
- El proceso de autenticación debe tener alguna dependencia temporal para evitar que se utilice información de autenticación previa.

Se puede notar que la autenticación y la verificación de transacción llevan consigo elementos similares. Una firma digital es una información de autenticación con el requerimiento agregado que es dependiente del contenido de la transacción.

Presentación:

Para firmar una transacción es necesario que alguna información sea transmitida del que la origina al que la recibe para que esta sea validada por el que la recibe, un método de llevar esto a cabo requiere un intercambio secuencial de información y este se conoce como *handshaking* (presentación); un contrato como se muestra en la figura, es construido para cada par de usuarios que participa en el sistema de transacción, el contrato graba firmas $2t$ de un mensaje estándar, las llaves t son seleccionadas por cada par de participantes.

El contrato de usuario A a usuario B es firmado y notariado por el arbitro, que es un tercero de confianza, la firma de datos para el usuario B por el usuario A con la llave KA es algún encriptamiento de (DATA, UserB) por un sistema criptográfico con llave Ka.

Figura 14. Diagrama mensaje estándar utilizando autenticación

SM : Standard Message	
User A	User B
$SIG\{k_{A,0}, SM, User B\}$	$SIG\{k_{B,0}, SM, User A\}$
$SIG\{k_{A,1}, SM, User B\}$	$SIG\{k_{B,1}, SM, User A\}$
...	...
$SIG\{k_{A,T-1}, SM, User B\}$	$SIG\{k_{B,T-1}, SM, User A\}$
<i>Alan G. Konheim</i>	<i>Carol A. Konheim</i>

Fuente: G. KONHEIM, Alan. Cryptography: A Primer. p. 335.

La transacción:

Una transacción entre las partes A y B se compone de los siguientes pasos:

- En la transacción i th el originador envía datos al receptor con un agregado de 21 firmas.

$$DATA, SIG\{k_{A,2li}, DATA, UserB\}, \dots, SIG\{k_{A,2li+20}, DATA, UserB\}$$

- El usuario A es ahora exigido por el usuario B a revelar 10 de las 21 claves.

$$k_{A,21+r_0}, k_{A,21+r_1}, \dots, k_{A,21+r_9}$$

$$0 \leq r_0 \leq r_1 \leq \dots \leq r_9 \leq 21$$

- El usuario B es ahora capaz de verificar estas claves.

$$k_{A,21+r_0}, \dots, k_{A,21+r_9}$$

- Las claves son consistentes con los bloques de entradas i th en el contrato.
- Las claves son consistentes con las 10 siguientes firmas.

$$SIG \{K_{A,21+r_s}, DATA, UserB\}$$

$$0 \leq s \leq 9$$

- El protocolo requiere un conocimiento de parte de los usuarios.

Una clave se utiliza únicamente en una transacción, el sistema de transacciones es utilizado para firmar el contrato como suplemento y contiene T nuevas claves generadas.

3.5. Curvas elípticas

Una curva elíptica es un par ordenado, donde E es una curva suave de género 1 y $O \in E$. La curva elíptica E es definida sobre K y se escribe $E(K)$ si E es definida sobre K como una curva y además $O \in E(\overline{K})$.

Aplicaciones en la criptografía

¿Cómo presentar el mensaje como un punto sobre una curva elíptica?

Varios de los métodos para encriptar mensajes de uso más común tienen su versión análoga para curvas elípticas. En este caso lo que cambia es básicamente que la seguridad de los algoritmos se basa en la dificultad del problema del logaritmo discreto para curvas elípticas. Este es simplemente el problema de que dados los puntos A, B sobre una curva elíptica E , halla un entero k tal que $kA=B$.

Para solucionar este problema Neal Koblitz propuso el siguiente método, para representar el mensaje como un punto sobre una curva elíptica. Sea E una curva elíptica sobre F_p dada por $y^2 = x^3 + Ax + B$.

Sea m un mensaje expresado como un número $0 \leq m \leq \frac{p}{100}$. Sea $x_j = 100m + j$ para $0 \leq j \leq 99$. Se calcula $s_j = x_j^3 + Ax_j + B$.

Ahora se verifica si s_j es un cuadrado módulo p observando si $\frac{p-1}{2} s_j^{-1} \equiv 1 \pmod{p}$.

En caso que s_j sea un cuadrado, no deben probar más valores de j y se detiene, si no se cambia el valor de j la idea es ir variando j entre 0 y 99 hasta hallar un valor que haga que s_j sea un cuadrado perfecto, ahora se calcula una raíz cuadrada por uno de los métodos conocidos y se toma y_j como una de las raíces cuadradas. Esto da un punto (x_j, y_j) en la curva elíptica E .

Ahora para recuperar el mensaje m de este punto simplemente se calcula:

$$\frac{x_j}{100}$$

No es garantizado que este método dará un punto para la elección de una curva elíptica de inicio, sin embargo, la probabilidad de que no funcione la curva escogida es aproximadamente de 2^{-100} .

Algoritmo de curvas elípticas de Lenstra:

Sea $n \geq 2$ un entero compuesto el cual se quiere factorizar.

- Verificar si $\text{mcd}(n, 6) = 1$ y que no tiene la forma m^r para algún $r \geq 2$
- Escoger enteros aleatorios b, x_1, y_1 entre 1 y n
- Sea $c = y_1^2 - x_1^3 - bx_1$ (módulo n), sea C la curva elíptica dada por la ecuación: $C: y^2 = x^3 + bx + c \pmod n$ y sea $P = (x_1, y_1) \in C$

- Verificar que $\text{mcd}(4b^3 + 27c^2, n) \neq 1$. Si es igual a n , volver al paso 2 y escoger un nuevo B si está estrictamente entre 1 y n , entonces es un factor no trivial de n y se habrá terminado.
- Escoger un número N y sea $k = \text{mcd}(1, 2, \dots, N)$
- Calcular $kP = \left(\frac{a_k}{d_k^2}, \frac{b_k}{d_k^3} \right)$
- Calcular $D = \text{mcd}(a_k, n)$. Si $1 < D < n$, entonces D es un factor no trivial de n y se tiene terminado. Si $D=1$, volver al paso 5 e incrementar el valor de N o volver al paso 2 y escoger una nueva curva. Si $D=n$, entonces volver al paso 5 y disminuir el valor de N .

Hay algunas cosas que se deben observar. Primero, en el algoritmo se debe calcular kP , este cálculo se puede hacer de manera eficiente doblando el punto de manera sucesiva, justo como se hace en la exponenciación rápida. Segundo, la forma particular del punto P dada por:

$$kP = \left(\frac{a_k}{d_k^2}, \frac{b_k}{d_k^3} \right)$$

Se obtiene de las fórmulas explícitas para la suma de puntos. Se debe recordar que las fórmulas (adaptadas al tipo partícula de curva que se está considerando) están dadas por:

$$x_3 = \lambda^2 - x_1 - x_2$$

$$y_3 = \lambda x_3 - v$$

Donde en particular λ es dado por una fracción y entonces es claro que al escribir todo sobre el mismo denominador, se obtiene la forma partícula del punto kP .

3.6. Funciones Hash

La criptografía asimétrica permite identificar al emisor y al receptor del mensaje. Para identificar el mensaje propiamente dicho se utilizan las llamadas funciones resumen (en inglés, *Hash*). El resultado de aplicar una función resumen a un texto es un número grande, el número resumen, que tiene las siguientes características:

- Todos los números resumen generados con un mismo método tienen el mismo tamaño sea cual sea el texto utilizado como base.
- Dado un texto base, es fácil y rápido (para un ordenador) calcular su número resumen.

- Es imposible reconstruir el texto base a partir del número resumen.
- Es imposible que dos textos base diferentes tengan el mismo número resumen.

Firmas y funciones de *Hash*

Hay varias formas de poner firmas digitales que permiten mensajes pequeños ser firmados, por ejemplo, un mensaje de 160 bit es firmado con una firma de 320. Por lo general, se quiere firmar mensajes mucho más largos, una forma de hacerlo sería partiendo el mensaje largo en varios bloques de 160 pero esto haría un resultado casi igual que encriptar de nuevo el mensaje ahora por bloques utilizando la misma clave.

Pero hay varios problemas con estas aproximaciones al crear firmas digitales. Primero que nada para un mensaje largo se necesitaría terminarlo con una firma tan grande como 2 veces el mensaje original. Otra desventaja es que la mayoría de firmas seguras son lentos desde que utilizan aritmética compleja para ser originados. Pero aún mayor problema representa el que varios tramos de un mensaje firmado podrían ser reacomodados o se podrían perder algunas partes y el mensaje resultante aún necesitaría ser verificado.

Se necesita proteger la integridad del mensaje completo y esto no puede ser completado firmando independientemente partes del mensaje. La solución a este problema es utilizar una función criptográfica de Hash, la cual tomará un mensaje de longitud arbitraria para producir un resumen del mensaje en un tamaño específico, lo cual permite utilizar cualquier método de firma digital.

Por ejemplo: Bob desea firmar un mensaje x , primero construye el resumen $z = h(x)$, y luego elabora la firma $y = \text{sig } K(z)$. Él transmite el par ordenado (x, y) por el canal. Ahora la verificación del mensaje es realizada primero reconstruyendo el resumen del mensaje $z = h(x)$, utilizando la función pública de *Hash* h y luego verifica que $\text{ver } K(z, y) = \text{verdadero}$.

Funciones *Hash* libres de colisión

Se debe ser cuidadoso en cuanto que al utilizar una función de *Hash* h no vulnere la seguridad dentro del esquema de la firma. Esto significa que el mensaje resumido está firmado, no el mensaje. Será necesario para h satisfacer con algunas propiedades para prevenir algunas falsificaciones.

El tipo de ataque más obvio es el ataque efectuado por un oponente, Oscar, para iniciar con un mensaje firmado válidamente (x, y) , donde $y = \text{sig}K(h(x))$. El par (x, y) puede ser cualquier mensaje previamente firmado por Bob. Luego el computa $z=h(x)$ e intenta encontrar $x' \neq x$ de tal manera que $h(x') = h(x)$. Si Oscar puede hacer lo siguiente, (x', y) sería un mensaje válidamente firmado, por ejemplo, un resumen. Para prevenir este tipo de ataque se debe requerir que h satisfaga las siguientes propiedades de que es libre de colisión:

Definición

Sea x un mensaje. Una función de h es débilmente libre de colisión si computacionalmente no es viable encontrar un mensaje $x' \neq x$ que haga $h(x') = h(x)$. Oscar le da x a Bob y lo persuade de firmar el resumen del mensaje $h(x)$, obteniendo y entonces (x', y) es una variación (*forgery*) válida.

El caso anterior motiva una propiedad diferente sobre libre de colisión.

Definición 3.6.1.

Una función Hash h es fuertemente libre de colisión si es computacionalmente imposible encontrar el mensaje x y x' que satisfaga $x' \neq x$ y $h(x') = h(x)$.

Se debe observar que la función Hash h es fuertemente libre de colisión sí y sólo sí computacionalmente es imposible encontrar un mensaje x que logre debilitar la función h para x .

Definición 3.6.2.

Una función Hash h es de una vía si dado un resumen de mensaje z , es computacionalmente imposible encontrar un mensaje x que haga $h(x) = z$.

Ahora se va a probar que la propiedad de ser fuertemente libre de colisión implica la propiedad de una vía. Esta se hará proporcionando una sentencia contrapositiva. Más específicamente se probará que una inversión arbitraria del algoritmo para una función de Hash puede ser usada como un Oracle en el algoritmo probabilístico Las Vegas para encontrar la colisión.

Esta reducción puede ir acompañada por una asunción de debilidad justa sobre el tamaño relativo del rango de dominio de la función de Hash. Se asume temporalmente que la función de Hash $h: X \rightarrow Z$ es finita y $|X| \geq 2|Z|$. Esta asunción es razonable: si se piensa en un elemento x siendo codificado como una cadena de bits de longitud $\log_2 |X|$ y un elemento de Z es codificado como una cadena de bits de longitud $\log_2 |Z|$. De ahí el resumen del mensaje $z=h(x)$ es al menos un bit más corto que el mensaje x .

Se asume que se tiene una inversión del algoritmo para h . Que es teniendo el algoritmo A que acepta un resumen del mensaje como entrada $z \in Z$, y encuentra un elemento $A(z) \in X$ de forma que $h(A(z))=z$. Con esto se prueba el siguiente teorema.

Teorema

Se supone $h: X \rightarrow Z$ es una función Hash donde $|X|$ y $|Z|$ son valores finitos y $|X| \geq 2|Z|$. Suponiendo que A es una inversión del algoritmo para h . Entonces existe un algoritmo probabilístico "las Vegas" que encuentra una colisión para h con probabilidad al menos de $1/2$.

Prueba: se considera el siguiente algoritmo

- Se selecciona un $x \in X$ aleatorio
- Se computa $z = h(x)$

- Se computa $x_1 = A \leftarrow \mathbb{Z}$
- Si $x_1 \neq x$ entonces

x_1 y x colisionan bajo h (éxito)

Sino

Salir (falla)

Este es claramente un algoritmo probabilístico tipo las Vegas, desde que encuentra una colisión o no devuelve nada. Pero la tarea principal es computar la probabilidad de éxito. Para cualquier $x \in X$, se define $x \approx x_1$ si $h(x) = h(x_1)$.

Definiendo

$$[x] = \{x_1 \in X : x \approx x_1\}$$

Cada clase equivalente $[x]$ consiste de la imagen inversa de un elemento de Z , de tal manera que el número de clases equivalentes es al menos $|Z|$.

Suponiendo que x es un elemento de X elegido en el paso 1. Para este x hay un x_1 posible dentro de $[x]$ que podría ser retornado en el paso 3.

Como este valor de x_1 es diferente de x retornaría éxito en el paso 4. Así que dada una selección $x \in X$, la probabilidad de éxito es $\frac{|[x]| - 1}{|X|}$.

La probabilidad de éxito del algoritmo es calculada promediando todos los posibles valores de x así:

$$\begin{aligned}
 p(\text{success}) &= \frac{1}{|X|} \sum_{x \in X} \frac{||x|| - 1}{|x|} \\
 &= \frac{1}{|X|} \sum_{c \in C} \sum_{x \in c} \frac{|c| - 1}{|c|} \\
 &= \frac{1}{|X|} \sum_{c \in C} (|c| - 1) \\
 &= \frac{1}{|X|} \left(\sum_{c \in C} |c| - \sum_{c \in C} 1 \right) \\
 &\geq \frac{|X| - |Z|}{|X|} \\
 &\geq \frac{|X| - |X|/2}{|X|} \\
 &= \frac{1}{2}.
 \end{aligned}$$

Para una función Hash es suficiente satisfacer la propiedad fuertemente libre de colisión, desde que implica las otras 2 propiedades.

Funciones de Hash para criptosistemas

Otra forma de crear una función de Hash es utilizando una aproximación de un sistema criptográfico de llave privada. Suponiendo que (P, C, K, E, D) es un sistema criptográfico computacionalmente seguro. Por conveniencia se asume también que $P = C = K = \mathbb{C}_2^n$. Aca se debe hacer $n \geq 128$, para prevenir un ataque de cumpleaños. Esto se impide utilizando DES.

Suponiendo que se tiene una cadena de bits dada por:

$$x = x_1 \| x_2 \| \dots \| x_k,$$

Donde $x_i \in \mathbb{C}^n$, $1 \leq i \leq k$ la idea básica es iniciar con un valor inicial $g_0 = IV$ y de ahí construir $g_1 \dots g_k$ de manera que se cumpla con $g_i = f(x_i, g_{i-1})$.

Donde f es una función que incorpora la función de encriptación del criptosistema. Finalmente se define el resumen del mensaje como $h(x) = g_k$. Las funciones Hash creadas por este método han sido probadas y consideradas inseguras.

Estas aparentan hacerse seguras si cumplen con:

$$g_i = e_{g_{i-1}}(x_i) \oplus x_i$$

$$g_i = e_{g_{i-1}}(x_i) \oplus x_i \oplus g_{i-1}$$

$$g_i = e_{g_{i-1}}(x_i \oplus g_{i-1}) \oplus x_i$$

$$g_i = e_{g_{i-1}}(x_i \oplus g_{i-1}) \oplus x_i \oplus g_{i-1}.$$

3.7. Timestamping

Una dificultad que puede presentar la firma es que el algoritmo puede ser comprometido. Suponiendo que un usuario A determina el secreto del usuario B exponiendo a en el DSS. Luego A puede crear la firma de B en cualquier mensaje que el quiera. Pero otro problema es que al comprometerse el algoritmo de firma se generan dudas sobre la autenticidad de todos los mensajes firmados por el usuario B, aún aquellos firmados antes que robarán el algoritmo de firma.

Otra situación indeseable que podría aparecer: es que el usuario b firma un mensaje y luego desea (*disavow it*). Usuario b publicaría el algoritmo y luego clamaría que la firma en el mensaje es un fraude. La razón por la cual pueden darse estos eventos es porque no hay forma de saber cuándo fue firmado el mensaje.

Esto sugiere que se deben considerar varias formas de marcar el tiempo en el mensaje. Un (timestamping) debe proporcionar pruebas que el mensaje fue firmado en tiempo en particular. Así si el algoritmo de firma del usuario es comprometido no se invalidarían las firmas efectuadas antes de que el algoritmo fuera robado. Este concepto es similar al cual actúan las tarjetas de crédito. Si alguien pierde una tarjeta de crédito y lo notifica al banco, se vuelve inválida pero las compras efectuadas antes de que se reportará siguen siendo válidas.

A continuación se describirán algunos métodos para (timestamping). Un usuario puede producir un *timestamp* convincente por si mismo. Si el usuario obtiene alguna información actual que no podría ser predicha antes que esta pasara.

Por ejemplo, esta información puede consistir de los marcadores del día anterior de las ligas mayores de baseball. O los valores de los inventarios listados en el intercambio de valores de Nueva York.

Suponiendo que el usuario B quiere datar (timestamp) un mensaje x . Se asume que h es una función de Hash pública. El usuario B procederá acorde al algoritmo:

- El usuario computa $z = h(x)$
- El usuario computa $z' = h(x || pub)$
- El usuario computa $m = sig_K(z')$
- El usuario publica (z', pub, m) en el periódico al día siguiente

Lo anterior se describe así:

La presencia de la información significa que el usuario B pudo no haber producido el mensaje m , antes de la fecha en cuestión, la realidad es que m es publicado el día siguiente, el periódico prueba que el usuario b no computó m después de la fecha en cuestión. Así que la firma del usuario b en m está datada con un día de anticipación. También se puede observar que el usuario B no revela el mensaje x en su esquema solamente z es publicado. De ser necesario el usuario b puede probar que x era el mensaje que firmó y dató simplemente revelándolo.

Esta es una forma directa de producir una datación, si hay un servicio de datación confiable disponible. El usuario puede computar $z=h(x)$ y $y=sig_K(z)$ y luego enviar (z,y) al servicio de datación. Este servicio lo que hace es agregar una fecha D y firmar el mensaje (z,y,D) . Esto funciona perfectamente bien permitiendo que el algoritmo de datación se mantenga seguro y que el servicio de datación no pueda firmar nada con fechas anteriores.

Si no se desea confiar incondicionalmente en el servicio de datación, la seguridad puede ser incrementada secuencialmente la conexión de los mensajes que están datados.

En este esquema, el usuario debería ordenar una trivía $(z,y,ID(Usuario))$ al servicio de datación. Ahora el mensaje z es un resumen del mensaje x; y el mensaje y es la firma del usuario sobre z. Y ID (Usuario) es la información de identificación del usuario. El servicio de datación hará una secuencia de trivias de esta forma. Se denota la tripleta a ser datada por (z_n,y_n,ID_n) , y el tiempo en que es datada por t_n .

El servicio de datación procederá según el siguiente algoritmo.

- Se computa $L_n = \langle ID_{n-1}, z_{n-1}, y_{n-1}, h(C_{n-1}) \rangle$
- Se computa $C_n = \langle t_n, z_n, y_n, ID_n, L_n \rangle$
- Se computa $s_n = sig_{TSS}(C_n)$
- Se envía $\langle C_n, s_n, ID_{n+1} \rangle$ a ID_n

La cantidad L_n es la información de enlace. Que ata la enésima requisición a la anterior. Ahora si se reta al usuario a revelar su mensaje x y luego y puede ser verificado. Luego la firma s del sistema puede ser verificada y si se desea luego se puede requerir ID para producir sus dataciones (timestamps). Las firmas del sistema de datación pueden ser verificadas con estas impresiones de fechas, este proceso puede continuarse hacia adelante y hacia atrás.

CONCLUSIONES

1. Se han descrito los métodos básicos para encriptación de datos.
2. Para la criptografía digital se prefieren métodos que utilizan funciones matemáticas complejas para que una computadora sea la encargada de efectuar las operaciones y así mismo métodos donde pueda incluirse *timestamping* para incrementar la seguridad.
3. En el desarrollo del trabajo de graduación se han descrito diversas metodologías para encriptar datos, los algoritmos de codificación clásicos, como el Vignere y César debido a la facilidad de análisis que representa la computadora, actualmente demuestran ser poco viables al momento de crear un sistema criptográficamente seguro.
4. El desarrollo de una aplicación criptográficamente segura requiere la utilización de métodos que incluyan tanto llaves privadas como públicas y que puedan ser firmados digitalmente por medio de funciones de Hash y *timestamping* lo cual permite un nivel de seguridad muy alto.
5. Aún se haya aplicado correctamente un algoritmo criptográfico, si la clave de descryptación es fácilmente alcanzable por un posible interceptor el mensaje será fácilmente decodificado.

6. Los sistemas que requieren que se almacene la clave en un medio físico, requieren de seguridad extra para evitar posibles ataques.
7. El uso de las computadoras hace más vulnerables los sistemas que tienen un número limitado de posibles combinaciones.

RECOMENDACIONES

1. Antes de seleccionar un método criptográfico para una aplicación es necesario tomar en cuenta todas las partes involucradas para saber que nivel de seguridad se necesita.
2. Es importante proteger el acceso a la clave para evitar que se haga vulnerable el sistema.
3. En sistemas que requieran una seguridad bastante elevada, la utilización del *timestamping* ayuda a asegurar que los mensajes fueron enviados de manera correcta.
4. Es necesario conocer técnicas de decriptación para tomar en cuenta las formas posibles en que puede ser atacado un sistema que se está diseñando.
5. Se debe profundizar el estudio de sistemas criptográficos de llave pública para ser capaces de desarrollar un sistema criptográfico seguro.
6. Es necesario profundizar en el estudio de las estadísticas aplicadas a criptografía para diseñar sistemas más seguros.
7. Se debe tomar en cuenta la elaboración de un protocolo que tanto el emisor como el receptor utilicen a la hora de enviar y recibir mensajes para un sistema criptográfico seguro.

BIBLIOGRAFÍA

1. CABALLERO GIL, Pino. *Introducción a la criptografía*. Madrid: RA-MA, 2002. 133 p. ISBN 84-7897-520-9.
2. *Curvas elípticas y criptografía*. [en línea]. <<http://claroline.emate.ucr.ac.cr/claroline/backends/download.php/VHJhYmFqb3MvQ3VydmFzRWxpcHRpY2FzeUNyaXB0b2dyYWZpYS1BZHJpYW5CYXJxdWVyby0ucGRm?cidReset=true&cidReq=MA609>>. [Consulta: julio de 2011].
3. *Funciones Resumen (Hash)*. [en línea]. <<http://www.eumed.net/coursecon/ecoinet/seguridad/resumenes.html>>. [Consulta: julio de 2011].
4. *Introducción a los Criptosistemas de Curva Elíptica*. [en línea]. <<http://www.publispain.com/supertutoriales/matematica/criptografia/cursos/2/curva.pdf>>. [Consulta: julio de 2011].
5. KONHEIM, Alan. G. *Cryptography: A Primer*. New Jersey: John Wiley & Sons, 1981. 432 p. ISBN 0-471-08132-9.
6. LUCEMA LÓPEZ, Manuel José. *Criptografía y seguridad en computadores*. España: Universidad de Jaén, 2008. 303 p.
7. MUNUERA, Carlos. *Codificación de la información*. España: Universidad de Valladolid, 1997. 285 p. ISBN 8477627649.

8. SGARRO, Andrea. *Códigos secretos*. Madrid: Piramides, 1990. 128 p. ISBN 8436805259.
9. STAMP, Mark, M; LOW, Richard. *Applied cryptanalysis: breaking ciphers in the real world*. New Jersey: John Wiley & Sons, 2007. 424 p. ISBN 047011486X.
10. STINSON, Douglas. *Cryptography: Theory and Practice*. Danvers: CRC Press, 1995. 434 p. ISBN 0849385210.
11. VAN DER LUBBE, Jan C. A. *Basic methods of cryptography*. Cambridge: Cambridge University Press, 1998. 229 p. ISBN 0-521-55559-0.

APÉNDICE

Apéndice 1. Herramientas criptográficas propias

Codificador/decodificador de funciones rotativas.

```
//funcion principal de programa aca se encripta y desencripta
//programa completo en visual c#
private void button1_Click(object sender, EventArgs e)
{
    string Temp = "ABCDEFGHJKLMNÑOPQRSTUVWXYZ";
    //      012345678901234567890123456
    int valor = 0;//valor temporal de M.
    int valor2 = 0;//valor temporal para la suma total.
    string t1="";
    if (checkBox1.Checked == true)
    {
        //codificar se suma 7+11+M
        valor = Convert.ToInt32(textBox2.Text);//obtiene M
        for (int i = 0; i < textBox1.Text.Length; i++)
        {
            for (int j=0;j<27;j++)
            {
                if (textBox1.Text[i]==Temp[j])//busca letra por letra
                {
                    valor2=j+7+11+valor;
                    while (valor2>=27)
                    {
                        if (valor2>=27)
                            valor2=valor2-27;
                    }
                }
            }
        }
    }
}
```

Continuación del apéndice 1.

```
    if (t1 == "")
    {
        t1 = char.ToString(Temp[valor2]);
    }
    else
        t1=t1+char.ToString(Temp[valor2]);
    label4.Text = t1;
}
}
else
    if (checkBox2.Checked == true)
        { //decodificar
            valor = Convert.ToInt32(textBox2.Text); //obtiene M
            for (int i = 0; i < textBox1.Text.Length; i++)
            {
                for (int j=0;j<27;j++)
                {
                    if (textBox1.Text[i]==Temp[j]) //busca letra por letra
                    {
                        valor2=j-7-11-valor;
                        while (valor2<0)
                        {
                            if (valor2<0)
                                valor2=valor2+27;
                        }
                    }
                }
            }
            if (t1 == "")
```

Continuación del apéndice 1.

```
        {
            t1 = char.ToString(Temp[valor2]);
        }
        else
            t1=t1+char.ToString(Temp[valor2]);
        label4.Text = t1;
    }
}
else
    MessageBox.Show("Debe seleccionar una Operacion");//mensaje
de error
}
private void checkBox2_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox1.Checked == true)
        checkBox1.Checked = false;
    else
        checkBox2.Checked = true;
}
private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox2.Checked == true)
        checkBox2.Checked = false;
    else
        checkBox1.Checked = true;
}
```


ANEXO

Anexo 1. Herramientas criptográficas existentes

Codificador de Vigenere en c

```
#include <stdio.h>
#include <string.h>

void uso(){
    printf("Uso:\n\tvin <clave> [d|D|x|X]\nDonde\n d|D imprime decimal\n y x|X
Imprime hex\n");
}

int main(int argn, char **argv){
    int c,in=0,cot;
    if (argn>1){

        cot = strlen(argv[1]);

        while ((c=getchar())!=EOF){
            c=c^argv[1][in];
            if (argn>2){
                if ((argv[2][0]=='x')||(argv[2][0]=='X')){
                    printf("0x%x ",c);
                }else if ((argv[2][0]=='d')||(argv[2][0]=='D')){
                    printf("%i ",c);
                }else{
                    printf("Comando no reconocido\n");
                    uso();
                    exit(-1);
                }
            }
            in++;
        }
    }
}
```


Continuación del anexo 1.

```
        }

        }else{
            putchar(c);
        }

        in++;
        if (in >= cot) in =0;
    }

}

}else{
    printf("Epecificar una clave\n");
    uso();
}
return(0);
}
```

Fuente:

DUARTE, Oscar Medina.

<http://www.udlap.mx/~is111936/oto02/programas/>