Algoritmos de Criptografía

Angel Cutipa Samayani
Jose David Mamani Vilca
Luis Fernando Ttito Surco
Percy Maldonado Quispe
Kevin Jhomar Sanchez Sanchez

I. Introducción

Hoy en día la seguridad se a transformado en un elemento vital del sector tecnológico, debido a ello nos introduciremos al mundo de la criptografía para analizar todos los métodos y clases del mismo.

Seguiremos a la criptografía atraves de la historia y también veremos como este a repercutido en la vida social de las personas.

ÍNDICE

I. Introducción 1

II. ¿Qué es la criptografía? 3

III. Clases Criptografía 3
III-A. Transposición 3
III-B. Sustitución 3
III-C. Simétrica 3
III-C1. Cifrados de flujo 4
III-C2. Cifrados de Bloque . . 4
III-D. Asimétrica 4

IV. Limitaciones de la Criptografía 4

V. Resumen Histórico 4

VI. Criptografía Clásica 5
VI-A. Concepto 5
VI-B. Algoritmos 6
VI-B1. Transposición Inversa . 6
VI-B2. Transposición Simple . 6
VI-B3. Cesar 6
VI-B4. Escítala 6
VI-B5. Vigenere 6
VI-B6. Vernam 6
VI-B7. Polybios 7

VII. Criptografía Moderna 7
VII-A. Concepto 7
VII-B. Algoritmos 7
VII-B1. DES 7
VII-B2. Triple DES 8
VII-B3. MD5 8
VII-B4. Blowfish 9
VII-B5. RSA 10
VII-B6. DSA 10
VII-B7. ElGamal 11

VIII. Profundizando en un tema 12
VIII-A. **Diffie-Hellman** 12
VIII-A1. Concepto 12
VIII-A2. Funcionamiento 12
VIII-A3. Seguridad 12
VIII-B. **RC4** 12
VIII-B1. Concepto 12
VIII-B2. Cifrado 13
VIII-B3. Ejemplo de RC4 13
VIII-B4. Seguridad de RC4 . . . 13
VIII-B5. Ventajas 14
VIII-C. **Secure Hash Algorithm (SHA)** . 14
VIII-C1. Concepto 14
VIII-C2. Funciones Hash 14
VIII-C3. SHA-0 y SHA-1 14
VIII-C4. SHA-2 15
VIII-C5. SHA-3 15
VIII-D. **Criptografía Cuántica** 16
VIII-D1. Concepto 16
VIII-D2. Mecánica Cuántica en Criptografía 16
VIII-D3. Distribucion de Claves Cuanticas 16
VIII-D4. Protocolo BB84 16
VIII-E. **Advanced Encryption Standard (AES)** 17
VIII-E1. Concepto 17
VIII-E2. Proceso de creación del AES 18
VIII-E3. Seguridad 19

IX. Conclusión 19

Referencias 20

II. ¿Qué es la criptografía?

La Criptografía es una rama de las matemáticas que, al orientarse al mundo de los mensajes digitales, proporciona las herramientas idóneas para solucionar los problemas relacionados con la autenticidad y la confiabilidad. El problema de la confidencialidad se vincula comúnmente con técnicas denominadas de .[en]cripción‘ la autenticidad con técnicas denominadas de ”firma digital”, aunque la solución de ambos, en realidad, se reduce a la aplicación de procedimientos criptográficos de encripción y desencripción. [12]

El uso de técnicas criptográficas tiene como propósito prevenir algunas faltas de seguridad en un sistema computarizado.

La seguridad, en general, se considera como un aspecto de gran importancia en cualquier corporación que trabaje con sistemas computarizados. El hecho de que gran parte de actividades humanas sean cada vez más dependientes de los sistemas computarizados, hace que la seguridad desempeñe una función protagónica. [13]

Las principales características que un sistema de seguridad quiere obtener son:

1. **Confidencialidad**. Consiste en garantizar que sólo las personas autorizadas tienen acceso a la información.
2. **Integridad**. Consiste en garantizar que el documento original no ha sido modificado. El documento puede ser tanto público como confidencial.
3. **Autenticación**. Permite garantizar la identidad del autor de la información.

III. Clases Criptografía

III-A. Transposición

Consiste en crear el texto cifrado simplemente desordenando las unidades que forman el texto original; los algoritmos de transposición, reordenan las letras pero no las disfrazan.

Ejemplo: TU SECRETO ES TU PRISIONERO; SI LO SUELTAS, TU ERES SU PRISIONERO

Figura 1. Representación gráfica de Transposición

III-B. Sustitución

Consiste en sustituir las unidades del texto original por otras; Los algoritmos de sustitución y los códigos, preservan el orden de los símbolos en claro, pero los disfrazan.

Ejemplo de sustitución:
ENCONTREMONOS A MEDIANOCHE

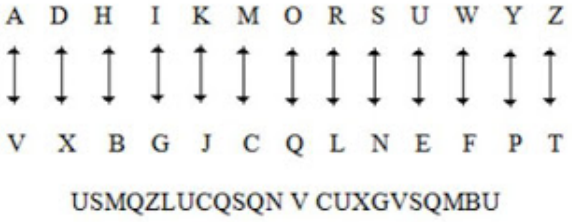

Figura 2. Representación gráfica de Sustitución

III-C. Simétrica

Es un método criptográfico en el cual se usa una misma clave para cifrar y descifrar mensajes en el emisor y el receptor. Las dos partes que se comunican han de ponerse de acuerdo de antemano sobre la clave a usar. Una vez que ambas partes tienen acceso a esta clave, el remitente cifra un mensaje usando la clave, lo envía al destinatario, y éste lo descifra con la misma clave. [14]

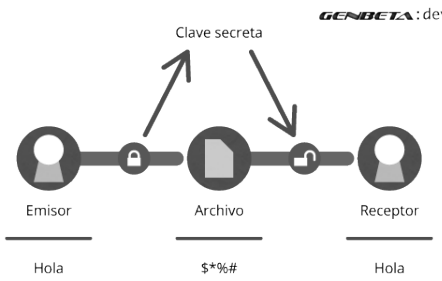

Figura 3. Representación gráfica de la criptografía Simétrica

III-C1. Cifrados de flujo: . cifran el mensaje con correspondencias bit a bit sobre el flujo (stream). Algunos cifrados de flujo son RC4 o RC6.

III-C2. Cifrados de Bloque: . cifran el mensaje dividiendo el flujo en bloques de k bits. Cada bloque se corresponde con otro diferente.

Por ejemplo, un bloque con k=3 ”010”se podría corresponder con ”110”. Un ejemplo de cifrado de bloque es el algoritmo AES (que veremos mas adelante).

III-D. Asimétrica

La criptografía asimétrica es el método criptográfico que usa un par de claves para el envío de mensajes. Las dos claves pertenecen a la misma persona a la que se ha enviado el mensaje. Una clave es pública y se puede entregar a cualquier persona, la otra clave es privada y el propietario debe guardarla de modo que nadie tenga acceso a ella. Además, los métodos criptográficos garantizan que esa pareja de claves sólo se puede generar una vez, de modo que se puede asumir que no es posible que dos personas hayan obtenido casualmente la misma pareja de claves.

Si el remitente usa la clave pública del destinatario para cifrar el mensaje, una vez cifrado, sólo la clave privada del destinatario podrá descifrar este mensaje, ya que es el único que la conoce. Por tanto se logra la confidencialidad del envío del mensaje, nadie salvo el destinatario puede descifrarlo. [15]

Si el propietario del par de claves usa su clave privada para cifrar el mensaje, cualquiera puede descifrarlo utilizando su clave pública. En este caso se consigue por tanto la identificación y autentificación del remitente, ya que se sabe que sólo pudo haber sido él quien empleó su clave privada (salvo que alguien se la hubiese podido robar). Esta idea es el fundamento de la firma electrónica.

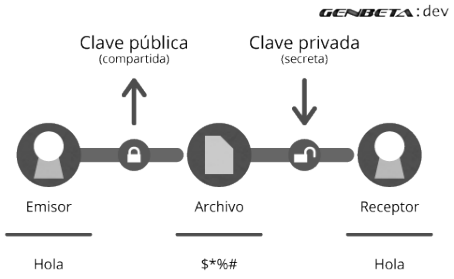

Figura 4. Representación gráfica de la criptografía asimétrica

Los sistemas de cifrado de clave pública o sistemas de cifrado asimétricos se inventaron con el fin de evitar por completo el problema del intercambio de claves de los sistemas de cifrado simétricos. Con las claves públicas no es necesario que el remitente y el destinatario se pongan de acuerdo en la clave a emplear. Todo lo que se requiere es que, antes de iniciar la comunicación secreta, el remitente consiga una copia de la clave pública del destinatario. Es más, esa misma clave pública puede ser usada por cualquiera que desee comunicarse con su propietario. Por tanto, se necesitarán sólo n pares de claves por cada n personas que deseen comunicarse entre sí.

IV. Limitaciones de la Criptografía

Los algoritmos criptográficos tienden a degradarse con el tiempo (esto hace que sean muy limitados). A medida que transcurre el tiempo, los algoritmos de encriptación se hacen más fáciles de quebrar debido al avance de la velocidad y potencia de los equipos de computación.

Todos los algoritmos criptográficos son vulnerables a los ataques de fuerza bruta (tratar sistemáticamente con cada posible clave de encripción, buscando colisiones para funciones hash, factorizando grandes números, etc.), la fuerza bruta es más fácil de aplicar en la medida que pasa el tiempo.

El ejemplo mas claro de la limitación de la criptografía sería:

.[En] 1977 Martin Gardner escribió que los números de 129 dígitos nunca serían factorizados, en 1994 se factorizó uno de esos números”

Además de la fuerza bruta, avanzan las matemáticas fundamentales que proveen nuevos métodos y técnicas de criptoanálisis.

V. Resumen Histórico

La historia de la criptografía es larga y está llena de anécdotas. Ya las primeras civilizaciones desarrollaron técnicas para enviar mensajes durante las campañas militares de forma que si el mensajero era interceptado la información que portaba no corriera el peligro de caer en manos del enemigo. Posiblemente, el primer criptosistema que se conoce fuera documentado por el historiador griego Polibio: un sistema de sustitución basado en la posición de las letras en una tabla.

También los romanos utilizaron sistemas de sustitución, siendo el método actualmente conocido como César, porque supuestamente Julio César lo utilizó en sus campañas, uno de los más conocidos en la literatura – según algunos autores, en realidad Julio César no utilizaba este sistema de sustitución, pero la atribución tiene tanto arraigo que el nombre de éste método de sustitución ha quedado para los anales de la historia –.

Otro de los métodos criptográficos utilizados por los griegos fue la escitala espartana, un método de transposición basado en un cilindro que servía como clave en el que se enrollaba el mensaje para poder cifrar y descifrar. En 1465 el italiano Leon Battista Alberti inventó un nuevo sistema de sustitución polialfabética que supuso un gran avance de la época.

Otro de los criptógrafos más importantes del siglo XVI fue el francés Blaise de Vigenere que escribió un importante tratado sobre "la escritura secretaz que diseñó un algoritmo que ha llegado a nuestros días asociado a su nombre.

A Selenus se le debe la obra criptográfica Çryptomenytices et Cryptographiae"(Lüneburg, 1624). Durante los siglos XVII, XVIII y XIX, el interés de los monarcas por la criptografía fue notable. Las huestes de Felipe II utilizaron durante mucho tiempo un algoritmo con un alfabeto de más de 500 símbolos que los matemáticos del rey consideraban inexpugnable. Cuando el matemático francés François Viète consiguió cripto-analizar aquel sistema para el rey de Francia, a la sazón Enrique IV, el conocimiento mostrado por el rey francés impulsó una queja de la corte española ante del papa Pío V acusando a Enrique IV de utilizar magia negra para vencer a sus ejércitos. Por su parte, la reina María Estuardo, reina de los Escoceses, fue ejecutada por su prima Isabel I de Inglaterra al descubrirse un complot de aquella tras un criptoanálisis exitoso por parte de los matemáticos de Isabel.

Desde el siglo XIX y hasta la Segunda Guerra Mundial las figuras más importantes fueron la del holandés Auguste Kerckhoffs y la del prusiano Friedrich Kasiski. Pero es en el siglo XX cuando la historia de la criptografía vuelve a presentar importantes avances. En especial durante las dos contiendas bélicas que marcaron al siglo: la Gran Guerra y la Segunda Guerra Mundial. A partir del siglo XX, la criptografía usa una nueva herramienta que permitirá conseguir mejores y más seguras cifras: las máquinas de cálculo. La más conocida de las máquinas de cifrado, posiblemente sea la máquina alemana Enigma: una máquina de rotores que automatizaba considerablemente los cálculos que era necesario realizar para las operaciones de cifrado y descifrado de mensajes. Para vencer al ingenio alemán, fue necesario el concurso de los mejores matemáticos de la época y un gran esfuerzo computacional. No en vano, los mayores avances tanto en el campo de la criptografía como en el del cripto-análisis no empezaron hasta entonces.

Tras la conclusión de la Segunda Guerra Mundial, la criptografía tiene un desarrollo teórico importante; siendo Claude Shannon y sus investigaciones sobre teoría de la información esenciales hitos en dicho desarrollo. Además, los avances en computación automática suponen tanto una amenaza para los sistemas existentes como una oportunidad para el desarrollo de nuevos sistemas.

A mediados de los años 70 el Departamento de Normas y Estándares norteamericano publica el primer diseño lógico de un cifrador que estaría llamado a ser el principal sistema criptográfico de finales de siglo: el Estándar de Cifrado de Datos o DES. En esas mismas fechas ya se empezaba a gestar lo que sería la, hasta ahora, última revolución de la criptografía teórica y práctica: los sistemas asimétricos. Estos sistemas supusieron un salto cualitativo importante ya que permitieron introducir la criptografía en otros campos que hoy día son esenciales, como el de la firma digital.

VI. Criptografía Clásica

VI-A. Concepto

Como ya hemos visto, la criptografía es casi tan antigua como las primeras civilizaciones de nuestro planeta. Ya en el siglo V antes de J.C. se usaban técnicas de cifrado para proteger a la información. Se pretendía garantizar sólo la confidencialidad y la autenticidad de los mensajes.

La criptografía clásica abarca desde tiempos inmemoriales hasta la mitad del siglo XX. El punto de inflexión en esta clasificación la marcan tres hechos relevantes:

1. En el año 1948 se publica el estudio de Claude Shannon sobre la Teoría de la Información.
2. En 1974 aparece el estándar de cifra DES.
3. En el año 1976 se publica el estudio realizado por W. Diffie y M. Hellman sobre la aplicación de funciones matemáticas de un solo sentido a

un modelo de cifra, denominado cifrado con clave pública.

VI-B. Algoritmos

VI-B1. Transposición Inversa: Es nuestro algoritmo más simple. Lo requerido para poder ejecutar el algoritmo, se debe saber donde inicia y donde termina nuestro mensaje. Se trata de invertir el inicio y el final de nuestro mensaje, cabe destacar que el algoritmo de cifrado es igual al de descifrado. Ejemplo:
$Mensaje$: $holamundo$
$Cripto$: $odnumaloh$

Este texto tiene adjunto los algoritmos mencionados programados en JAVA, por lo que se sugiere ver los códigos fuente y probar su funcionamiento. Debido a eso, se procurará no añadir aquí dichos códigos y facilitar así la lectura del documento. Sin embargo se mostraran los diseños de estos programas, en este caso corresponden, diagramas de clase.

VI-B2. Transposición Simple: El algoritmo divide un mensaje en claro símbolo por símbolo, si el número de símbolos es impar, el primer grupo de símbolos tendrá un elemento más. Podemos ver el algoritmo como si numeráramos los elementos, en el primer bloque tendremos los elementos impares mientras en el segundo estarán los elementos pares. Para finalizar concatenamos los bloques y así tendremos el criptograma. Ejemplo:
$Mcla$: $Holamundo$
$Bloque1$: $hlmno$
$Bloque2$: $oaud$
$Cripto$: $hlmnooaud$

El proceso de descifrado, es similar, dividimos el criptograma en dos partes iguales, la primera mitad del criptograma será el primer bloque. Teniendo ambos bloques, se intercalan uno a uno los elementos de cada bloque, puede leerse el ejemplo de abajo hacia arriba, para ver la operación.

VI-B3. Cesar: En el siglo I a.d.C., Julio César usa este cifrado, cuyo algoritmo consiste en el desplazamiento de tres espacios hacia la derecha de los caracteres del texto en claro. Es un cifrado por sustitución monoalfabético en el que las operaciones se realizan módulo n, siendo n el número de elementos del alfabeto (en aquel entonces latín).

Cada letra se cifrará siempre igual. Es una gran debilidad y hace que este sistema sea muy vulnerable y fácil de atacar simplemente usando las estadísticas del lenguaje.

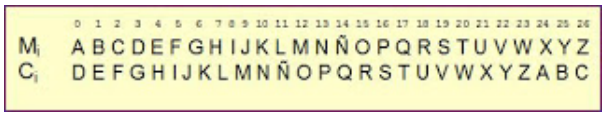

Figura 5. Representacion de Cesar

VI-B4. Escítala: La escítala era usada en el siglo V a.d.C. por el pueblo griego de los lacedemonios. Consistía en un bastón en el que se enrollaba una cinta de cuero y luego se escribía en ella el mensaje de forma longitudinal. Al desenrollar la cinta, las letras aparecen desordenadas. La única posibilidad de recuperar el texto en claro pasaba por enrollar dicha cinta en un bastón con el mismo diámetro que el usado en el extremo emisor y leer el mensaje de forma longitudinal. La clave del sistema está en el diámetro del bastón. Se trata de una cifra por transposición pues los caracteres del criptograma son los mismos que en el texto en claro distribuidos de otra forma.

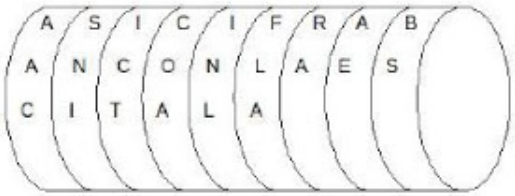

Figura 6. Representación gráfica de Escítala

VI-B5. Vigenere: Es un cifrado basado en diferentes series de caracteres o letras del cifrado César formando estos caracteres una tabla, llamada tabla de Vigenère, que se usa como clave. El cifrado de Vigenère es un cifrado de sustitución simple polialfabético.

El cifrado Vigenère se ha reinventado muchas veces. El método original fue descrito por Giovan Battista Belasso en su libro de 1553 La cifra del Sig. Giovan Battista Belasso. Sin embargo, fue incorrectamente atribuido más tarde a Blaise de Vigenère, concretamente en el siglo XIX, y por ello aún se le conoce como el çifrado Vigenère".

Este cifrado es conocido porque es fácil de entender e implementar, además parece irresoluble; esto le hizo valedor del apodo el código indescifrable (le chiffre indéchiffrable, en francés).

VI-B6. Vernam: Se usa el mismo algoritmo para cifrar y para descifrar el mensaje. Usa una clave cons-

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

Figura 7. Cuadro de Vigenere

tituida por una sucesión de símbolos (bits o caracteres), operando XOR cada símbolo de ésta con el correspondiente del texto en claro.

Debido a la definición de la función XOR, el descifrado se realiza, igualmente, operando con dicha función cada bit de la misma serie cifrante con el correspondiente del texto cifrado. Si la serie cifrante no se repite, es aleatoria, y de longitud igual, al menos, al texto a cifrar éste cifrado alcanza el secreto perfecto. Además, es el único que verifica tal condición.

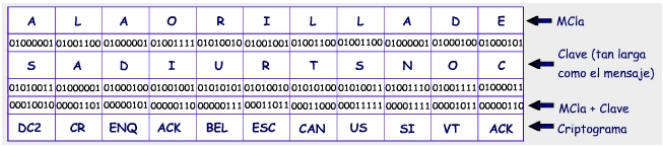

Figura 8. Cuadro de Vernam

VI-B7. Polybios: Es el cifrado por sustitución de caracteres más antiguo que se conoce (siglo II a.d.C.) pero como duplica el tamaño del texto en claro, con letras o números, resulta poco interesante.

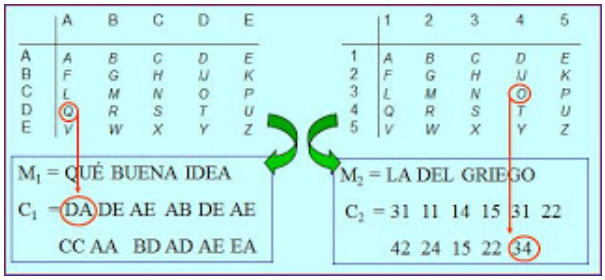

Figura 9. Representacion de Polybios

VII. Criptografía Moderna

VII-A. Concepto

Como hemos visto en el apartado anterior, los sistemas criptográficos clásicos presentaban una dificultad en cuanto a la relación complejidad-longitud de la clave / tiempo necesario para encriptar y desencriptar el mensaje.

En la era moderna esta barrera clásica se rompió, debido principalmente a los siguientes factores:

1. **Velocidad de cálculo**. con la aparición de los computadores se dispuso de una potencia de cálculo muy superior a la de los métodos clásicos.
2. **Avance de las matemáticas** que permitieron encontrar y definir con claridad sistemas criptográficos estables y seguros.
3. **Necesidades de seguridad** surgieron muchas actividades nuevas que precisaban la ocultación de datos, con lo que la Criptología experimentó un fuerte avance.

A partir de estas bases surgieron nuevos y complejos sistemas criptográficos, que se clasificaron en dos tipos o familias principales, los de clave simétrica y los de clave asimétricos. Los modernos algoritmos de encriptación simétricos mezclan la trasposición y la permutación, mientras que los de clave pública se basan más en complejas operaciones matemáticas.

VII-B. Algoritmos

VII-B1. DES: Data Encryption Standard (DES) es un algoritmo de cifrado, es decir, un método para cifrar información, escogido como FIPS en los Estados Unidos en 1976, y cuyo uso se ha propagado ampliamente por todo el mundo. El algoritmo fue controvertido al principio, con algunos elementos de diseño clasificados, una longitud de clave relativamente corta. [16]

Hoy en día, DES se considera inseguro para muchas aplicaciones. Esto se debe principalmente a que el tamaño de clave de 56 bits es corto; las claves de DES se han roto en menos de 24 horas. Existen también resultados analíticos que demuestran debilidades teóricas en su cifrado, aunque son inviables en la práctica. Se cree que el algoritmo es seguro en la práctica en su variante de Triple DES, aunque existan ataques teóricos.

Desde hace algunos años, el algoritmo ha sido sustituido por el nuevo AES (Advanced Encryption Standard).

La estructura básica del algoritmo aparece representada en la Figura 10: hay 16 fases idénticas de proceso, denominadas rondas. También hay una permutación inicial y final denominadas PI y PF, que son funciones inversas entre sí (PI "deshace"la acción de PF, y viceversa). PI y PF no son criptográficamente significativas, pero se incluyeron presuntamente para facilitar la carga y descarga de bloques sobre el hardware de mediados de los 70. Antes de las rondas, el bloque es dividido en dos mitades de 32 bits y procesadas alternativamente. Este entrecruzamiento se conoce como esquema Feistel.

La estructura de Feistel asegura que el cifrado y el descifrado sean procesos muy similares — la única diferencia es que las subclaves se aplican en orden inverso cuando desciframos. El resto del algoritmo es idéntico. Esto simplifica enormemente la implementación, en especial sobre hardware, al no haber necesidad de algoritmos distintos para el cifrado y el descifrado.

El símbolo rojo representa la operación OR exclusivo (XOR). La función-F mezcla la mitad del bloque con parte de la clave. La salida de la función-F se combina entonces con la otra mitad del bloque, y los bloques son intercambiados antes de la siguiente ronda. Tras la última ronda, las mitades no se intercambian; ésta es una característica de la estructura de Feistel que hace que el cifrado y el descifrado sean procesos parecidos.

VII-B2. Triple DES: Se le llama al algoritmo que hace triple cifrado del DES. También es conocido como TDES o 3DES, fue desarrollado por IBM en 1998.

Este método de cifrado es inmune al ataque por encuentro a medio camino, doblando la longitud efectiva de la clave (112 bits), pero en cambio es preciso triplicar el número de operaciones de cifrado, haciendo este método de cifrado muchísimo más seguro que el DES. Por tanto,

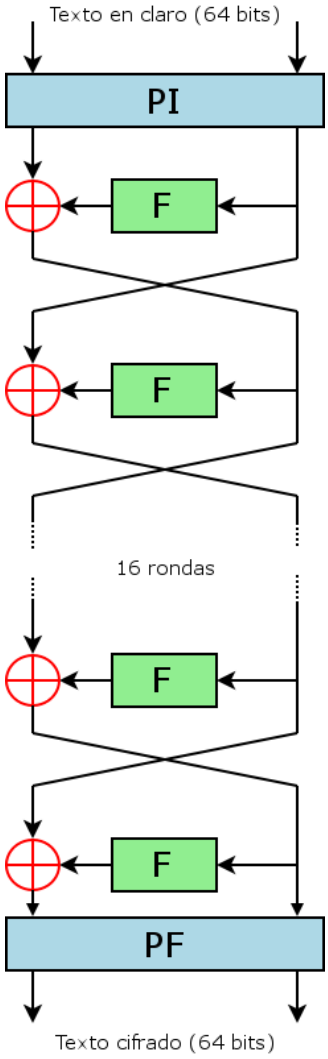

Figura 10. La estructura general de Feistel en DES

la longitud de la clave usada será de 168 bits (3x56 bits), aunque como se ha dicho su eficacia solo sea de 112 bits. Se continúa cifrando bloques de 64 bits.

El Triple DES está desapareciendo lentamente, siendo reemplazado por el algoritmo AES. Sin embargo, la mayoría de las tarjetas de crédito y otros medios de pago electrónicos tienen como estándar el algoritmo Triple DES (anteriormente usaban el DES). Por su diseño, el DES y por lo tanto el TDES son algoritmos lentos. AES puede llegar a ser hasta 6 veces más rápido y a la fecha no se ha encontrado ninguna vulnerabilidad.

VII-B3. MD5: (Message-Digest Algorithm 5) Es un algoritmo de reducción criptográfico de 128 bits ampliamente usado. uno de sus usos es el de comprobar que

algún archivo no haya sido modificado.

La codificación del MD5 de 128 bits es representada típicamente como un número de 32 dígitos hexadecimal. El siguiente código de 28 bytes ASCII será tratado con MD5 y veremos su correspondiente hash de salida:

$$MD5(\text{"}Generando\ un\ MD5\ de\ un\ texto\text{"})$$
$$5df9f63916ebf8528697b629022993e8$$

Un pequeño cambio en el texto (cambiar '5' por 'S') produce una salida completamente diferente.

$$MD5(\text{"}Generando\ un\ MDS\ de\ un\ texto\text{"})$$
$$e14a3ff5b5e67ede599cac94358e1028$$

Otro ejemplo sería la codificación de un campo vacío:

$$MD5(\text{""})$$
$$d41d8cd98f00b204e9800998ecf8427e$$

VII-B4. Blowfish: Es un codificador de bloques simétricos, diseñado por Bruce Schneier en 1993 e incluido en un gran número de conjuntos de codificadores y productos de cifrado. No se han encontrado técnicas de criptoanálisis efectivas contra el Blowfish.

Blowfish usa bloques de 64 bits y claves que van desde los 32 bits hasta 448 bits. Es un codificador de 16 rondas Feistel y usa llaves que dependen de las Cajas-S. Tiene una estructura similar a CAST-128, el cual usa Cajas-S fijas.

El diagrama muestra la acción de Blowfish. Cada línea representa 32 bits. El algoritmo guarda 2 arrays de subclaves: El array P de 18 entradas y 4 cajas-S de 256 entradas. Una entrada del array P es usada cada ronda, después de la ronda final, a cada mitad del bloque de datos se le aplica un XOR con uno de las 2 entradas del array P que no han sido utilizadas.

La función divide las entrada de 32 bits en 4 bloques de 8 bits, y usa los bloques como entradas para las cajas-S. Las salidas deben estar en módulo 232 y se les aplica un ok XOR para producir la salida final de 32 bits.

Debido a que Blowfish está en la red Feistel, puede ser invertido aplicando un XOR entre P17 y P18 al bloque texto codificado, y así sucesivamente se usan las P-entradas en orden reversivo.

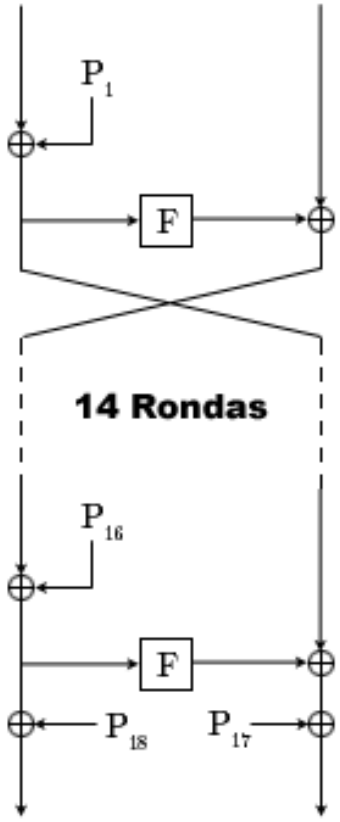

Figura 11. Diagrama de Blowfish

La generación de claves comienza inicializando los P-arrays y las cajas-S con los valores derivados de los dígitos hexadecimales de pi, los cuales no contienen patrones obvios. A la clave secreta se le aplica un XOR con las P-entradas en orden (ciclando la clave si es necesario). Un bloque de 64 bits de puros ceros es cifrado con el algoritmo como se indica. El texto codificado resultante reemplaza a P1 y P2. Entonces el texto codificado es cifrado de nuevo con las nuevas subclaves, P3 y P4 son reemplazados por el nuevo texto codificado. Esto continúa, reemplazando todas las entradas del P-array y todas las entradas de las cajas-S. En total, el algoritmo de cifrado Blowfish correrá 521 veces para generar todas las subclaves, cerca de 4KB de datos son procesados.

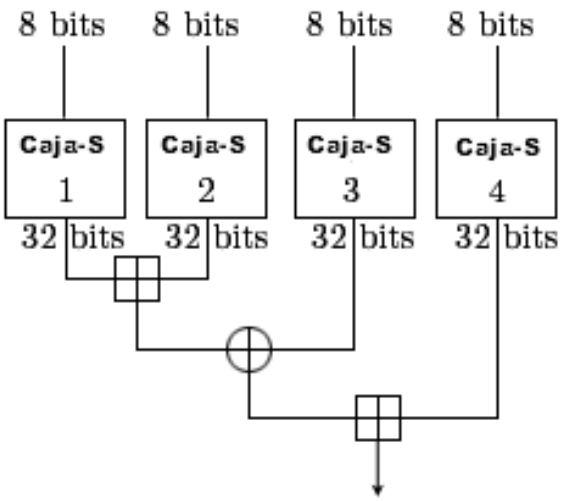

Figura 12. Diagrama de Blowfish a nivel de bits

VII-B5. RSA: Es un sistema criptográfico de clave pública desarrollado en 1977 por ingenieros estadounidenses. En la actualidad, RSA es el primer y más utilizado algoritmo de este tipo y es válido tanto para cifrar como para firmar digitalmente.

La seguridad de este algoritmo radica en el problema de la factorización de números enteros. Los mensajes enviados se representan mediante números, y el funcionamiento se basa en el producto, conocido, de dos números primos grandes elegidos al azar y mantenidos en secreto. Actualmente estos primos son del orden de 10200, y se prevé que su tamaño aumente con el aumento de la capacidad de cálculo de los ordenadores.

Como en todo sistema de clave pública, cada usuario posee dos claves de cifrado: una pública y otra privada. Cuando se quiere enviar un mensaje, el emisor busca la clave pública del receptor, cifra su mensaje con esa clave, y una vez que el mensaje cifrado llega al receptor, este se ocupa de descifrarlo usando su clave privada.

La seguridad del criptosistema RSA está basado en dos problemas matemáticos: el problema de factorizar números grandes y el problema RSA. El descifrado completo de un texto cifrado con RSA es computacionalmente intratable, no se ha encontrado un algoritmo eficiente todavía para ambos problemas. Proveyendo la seguridad contra el descifrado parcial podría requerir la adición de una seguridad padding scheme.

El problema del RSA se define como la tarea de tomar raíces eth módulo a componer n: recuperando un valor m tal que $me = c \ mod \ n$, donde (e, n) es una clave pública RSA y c es el texto cifrado con RSA. Actualmente la aproximación para solventar el problema del RSA es el factor del módulo n. Con la capacidad para recuperar factores primos, un atacante puede computar el exponente secreto d desde una clave pública (e, n), entonces descifra c usando el procedimiento standard. Para conseguir esto, un atacante factoriza n en p y q, y computa $(p-1)(q-1)$ con lo que le permite determinar d y e. No se ha encontrado ningún método en tiempo polinómico para la factorización de enteros largos. Ver factorización de enteros para la discusión de este problema.

La factorización de números grandes por lo general proponen métodos teniendo 663 bits de longitud usando métodos distribuidos avanzados. Las claves RSA son normalmente entre 1024-2048 bits de longitud. Algunos expertos creen que las claves de 1024 bits podrían comenzar a ser débiles en poco tiempo; con claves de 4096 bits podrían ser rotas en un futuro. Por lo tanto, si n es suficientemente grande el algoritmo RSA es seguro. Si n tiene 256 bits o menos, puede ser factorizado en pocas horas con un computador personal, usando software libre. Si n tiene 512 bits o menos, puede ser factorizado por varios cientos de computadoras como en 1999. Un dispositivo hardware teórico llamado TWIRL descrito por Shamir y Tromer en el 2003 cuestionó a la seguridad de claves de 1024 bits. Es actualmente recomendado que n sea como mínimo de 2048 bits de longitud.

En 1993, Peter Shor publicó su algoritmo, mostrando que una computadora cuántica podría en principio mejorar la factorización en tiempo polinomial, mostrando RSA como un algoritmo obsoleto. Sin embargo, las computadoras cuánticas no se esperan que acaben su desarrollo hasta dentro de muchos años.

VII-B6. DSA: (Digital Signature Algorithm) Es un estándar del Gobierno Federal de los Estados Unidos de América o FIPS para firmas digitales. Fue un Algoritmo propuesto por el Instituto Nacional de Normas y Tecnología de los Estados Unidos para su uso en su Estándar de Firma Digital (DSS), especificado en el FIPS 186. DSA se hizo público el 30 de agosto de 1991, este algoritmo como su nombre lo indica, sirve para firmar y no para cifrar información. Una desventaja de este

algoritmo es que requiere mucho más tiempo de cómputo que RSA.

Generación de claves

1. Elegir un número primo p de L bits, donde $512 \leq L \leq 1024$ y L es divisible por 64.
2. Elegir un número primo q de 160 bits, tal que $p-1 = qz$, donde z es algún número natural.
3. Elegir h, donde $1 < h < p-1$ tal que $g = hz(mod\ p) > 1$.
4. Elegir x de forma aleatoria, donde $1 < x < q-1$.
5. Calcular $y = gx(mod\ p)$.

Los datos públicos son p, q, g e y. x es la clave privada.

Firma

1. Elegir un número aleatorio k, donde $1 < k < q$.
2. Calcular $r = (g^k\ mod\ p) mod\ q$.
3. Calcular $s = k^-1(H(m) + r * x) modq$, donde $H(m)$ es la función hash SHA-1 aplicada al mensaje m.
4. La firma es el par (r, s).

Si r ó s es cero, se vuelve a repetir el procedimiento.

Verificación

1. Calcular $w = (s)^-1(mod\ q)$.
2. Calcular $u1 = H(m) * w(mod\ q)$.
3. Calcular $u2 = r * w(mod\ q)$.
4. Calcular $v = [g^(u1) *^(yu2) mod\ p]\ mod\ q$.
5. La firma es válida si $v = r$.

VII-B7. ElGamal: Se refiere a un esquema de cifrado basado en el problema matemático del logaritmo discreto. Es un algoritmo de criptografía asimétrica basado en la idea de Diffie-Hellman y que funciona de una forma parecida a este algoritmo discreto.

Hasta el momento el algoritmo ElGamal de cifrado/descifrado puede ser considerado un algoritmo efectivo.

Un adversario con la habilidad de calcular logaritmos discretos podría ser capaz de romper un cifrado ElGamal. Sin embargo, en la actualidad, el algoritmo de computación de logaritmos discretos (cuando trabajamos módulo un primo) es subexponencial con una complejidad de $\lambda = 1/3$, la misma que la de factorizar dos números primos, y por tanto, no es accesible de realizar tal tarea en números grandes en un tiempo razonable. El logaritmo discreto es aún más difícil si trabajamos en otros grupos (como por ejemplo, curvas elípticas).

Mostremos el criptosistema propuesto inicialmente por Tahar ElGamal en su artículo.

Generación de la clave

Para generar la clave, Alicia escoge un número primo p, cualquiera tal que $p-1$, tenga un factor primo grande (esto se pide para que el problema del logaritmo discreto sea difícil). Además elige dos números aleatorios g, (el generador) y a, (que actuará como clave privada) tal que $a \in \{0, \ldots, p-1\}$ $a \in \{0, \ldots, p-1\}$.

Alicia calcula el valor de $K = g^a \pmod p$ $K = g^a \pmod p$. La clave pública será (g, p, K) (g, p, K), mientras que el valor de a lo mantendrá en secreto.

Cifrado

Supongamos que Bruno tiene un texto claro que quiere enviar cifrado a Alicia. Lo primero por hacer es convertir este texto en un entero entre 1 y $p-1$, obteniendo un m, (esto no es parte del cifrado, sino que es una manera de codificar estándar, conocida por todos). Luego Bruno escoge arbitrariamente un número $b \in \{2, \ldots, p-1\}$ $b \in \{2, \ldots, p-1\}$ (que mantendrá secreto) para finalmente calcular:

$y_1 = g^b \pmod p$ $y_1 = g^b \pmod p$ $y_2 = K^b\, m \pmod p$ $y_2 = K^b\, m \pmod p$ El mensaje cifrado final corresponde a la tupla $C_b(m, b) = (y_1, y_2)$ $C_b(m, b) = (y_1, y_2)$

Decifrado

Para descifrar se tiene que realizar el siguiente cálculo:

$y_1^{-a} y_2 \pmod p$ $y_1^{-a} y_2 \pmod p$ donde y_1^{-a} y_1^{-a} representa el inverso de y_1^a y_1^a módulo p (que, utilizando el Pequeño teorema de Fermat puede calcularse como y_1^{p-1-a} y_1^{p-1-a}).

Veamos por qué esto da como resultado m:

$y_1^{-a} y_2 y_1^{-a} y_2 = (g^b)^{-a} K^b m = g^{-ab} (g^a)^b m = (g^a)^{-b} (g^a)^b m = (g^b)^{-a} K^b m = g^{-ab} (g^a)^b m = (g^a)^{-b} (g^a)^b m = m \pmod p = m \pmod p$

Vale observar que para calcular el descifrado, es necesario conocer a, que es justamente la clave privada de Alicia.

VIII. Profundizando en un tema

En esta sección empezaremos a desarrollar un poco más lo que ya hemos visto anteriormente y pondremos mas énfasis en el uso y seguridad de los algoritmos.

VIII-A. **Diffie-Hellman**

VIII-A1. Concepto: El protocolo de cifrado Diffie-Hellman (recibe el nombre de sus creadores) es un sistema de intercambio de claves entre partes, que no han contactado previamente, a través de un canal inseguro y sin autenticación.

Este protocolo se utiliza principalmente para intercambiar claves simétricas de forma segura para posteriormente pasar a utilizar un cifrado simétrico, menos costoso que el asimétrico.

Se parte de la idea de que dos interlocutores pueden generar de forma conjunta una clave sin que esta sea comprometida.

VIII-A2. Funcionamiento:

1. Se escoge un número primo p y un generador g, siendo este último coprimo de p. Ambos números son públicos.
2. Escogemos un número a menor que p, en este caso a, y calculamos:

   $$A = g^a modp$$

   Enviamos A, p y g al otro interlocutor.
3. El otro interlocutor escoge un numero b menor que p y calcula:

   $$B = g^b modp$$

   Retornandonos B.
4. Ahora, ambos podemos calcular:

   $$K = g^(a - b) modp$$

   Siendo para nosotros $B^a modp = K$ y para nuestro interlocutor $A^b modp = K$. Usamos K como clave.
5. Al ser p y g públicos cualquier atacante puede llegar a conocerlos, esto no supone una vulnerabilidad. Aunque capturase A y B, le resultaría computacionalmente imposible obtener a y b con la consecuencia de tampoco acceder a K.

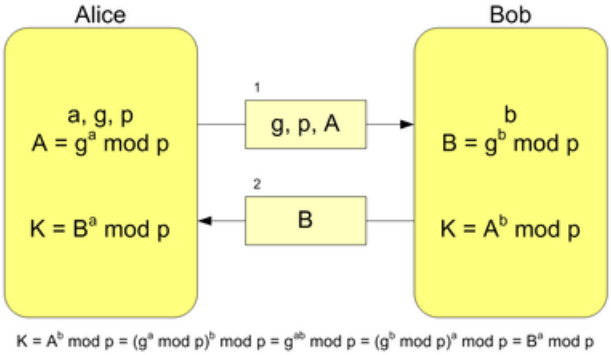

Figura 13. Funcionamiento del sistema Diffie-Hellman

VIII-A3. Seguridad: Este protocolo es vulnerable al ataque ”man in the middle”; el cual consiste en que un tercero se coloca en el medio del canal y hace creer a ambos que es el otro. De esta forma se podría acordar una clave con cada parte y servir de .[en]lace.[en]tre los dos participantes. Para que este ataque sea funcional se necesita saber que método de cifrado simétrico se va a emplear. Ocultar el algoritmo de cifrado no cumple con el principio de Kerckhoffs de que la efectividad de un sistema no debe depender de que su diseño permanezca en secreto por lo que conocer el sistema que se va a emplear se hace trivial.

Para evitar esto se puede emplear un protocolo de autenticación de las partes mediante por ejemplo TLS(Transport Layer Security).

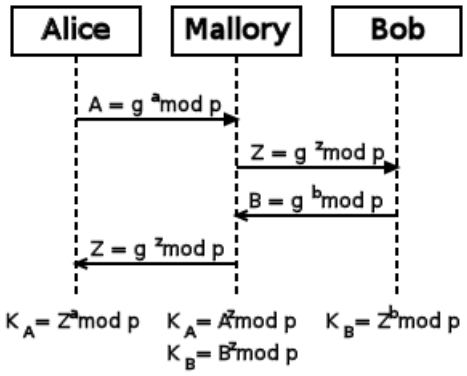

Figura 14. Esquema de la intrusión Man-in-the-middle

VIII-B. **RC4**

VIII-B1. Concepto: El algoritmo RC4 fue disenado por Ron Rivest en 1987 para la compañía RSA Data

Security. Su implementación es extremadamente sencilla y rápida, y está orientado a generar secuencias en unidades de un byte, ademas de permitir claves de diferentes longitudes. Por desgracia es un algoritmo propietario, lo cual implica que no puede ser incluido en aplicaciones de tipo comercial sin pagar los royalties correspondientes.Su nombre completo es Rivest Cipher 4, teniendo el acrónimo RC un significado alternativo al de Ron's Code utilizado para los algoritmos de cifrado RC2, RC5 y RC6.

Inicialmente el algoritmo era un secreto registrado, pero en septiembre de 1994 una descripción del algoritmo fue posteada anónimanente en una lista de correo de Cypherpunks. Enseguida pasó al grupo de correo sci.crypt y de ahí fue publicado en numerosos sitios de Internet. Debido al conocimiento del algoritmo, éste dejó de ser un secreto registrado.

RC4 se ha convertido en parte de algunos protocolos y normas de cifrado de uso común, como WEP y WPA para tarjetas inalámbricas y TLS. Los principales factores en el éxito de RC4 sobre una amplia gama de aplicaciones tales son su velocidad y simplicidad, implementaciones eficientes en software y hardware son muy fáciles de desarrollar.

VIII-B2. Cifrado: El cifrado RC4 esta basado en dos algoritmos. El primero es Key Scheduling Algorithm (KSA) y el segundo es Pseudo-Random Generation Algorithm (PRGA).

El algoritmo consta de una S-Caja de 8*8. Primero, el S-box array es llenado con valores secuenciales desde 0-255. Este array será llamado simplemente S. Entonces, el otro array de 256-bits es llenado con el valor de la "semilla"(clave), repitiendo como sea necesario hasta que todo el array es llenado. Este array es llamado K, y se mezcla el array K de la siguiente manera:

- **Key Scheduling Algorithm (KSA)**

  Inicialización:
  $j = 0;$
  $for\ i\ =\ 0\ to\ 255$
  {
  $j\ =\ (j + S[i]\ +\ K[i])\ mod\ 256;$
  $intercambia\ S[i]\ y\ S[j];$
  }

  Cuando terminamos con esto, la S-box es intercambiada basándose en el valor de la "semilla". Esa es la "Key"programada para el uso en el algoritmo.

- **Pseudo-Random Generation Algorithm (PRGA).**

  Generación de secuencia cifrante:
  Ahora cuando se necesita el keystream data, que nos ayudara a cifrar y descifrar el mensaje, se usa el Pseudo-Random Generation Algorithm (PRGA). Este algoritmo tiene 2 contadores, el i y la j, en el cual ambos son inicializados en 0 para comenzar. Después de eso, cada bit de keystream data es usado en el siguiente Pseudo-Code:

  $i\ =\ (i\ +\ 1)\ mod\ 256;$
  $j\ =\ (j\ +\ S[i])\ mod\ 256;$
  $intercambia\ S[i]\ and\ S[j];$
  $t\ =\ (S[i]\ +\ S[j])\ mod\ 256;$
  $Exponer\ valor\ de\ S[t];$

  El algoritmo RC4, genera una clave de flujo la cual es simplemente pasada por la función XOR para producir el flujo cifrado. Para descifrar se utiliza exactamente la misma función de cifrado. Una de las razones a las que debe su popularidad es su sencillez.

VIII-B3. Ejemplo de RC4: Para este ejemplo utilizaremos el texto original "HI" y lo veremos en binario.

H
0100 1000
XOR 0000 0011
———————
0100 1011

I
0100 1001
XOR 0000 0000
———————-
0100 1001

Texto original: 0100 1000 0100 1001
Texto cifrado: 0100 1011 0100 1001

VIII-B4. Seguridad de RC4:

- Algunos estudios indican que pueden existir claves débiles y que es sensible a estudios analíticos del contenido de la S- Caja. De hecho, algunos afirman

que en una de cada 256 claves posibles, los bytes que se generan tienen una fuerte correlación con un subconjunto de los bytes de la clave.
- Se puede recuperar la clave empleada si la inicialización del algoritmo cumple determinadas premisas muy comunes, y se interceptan el suficiente número de mensajes.

VIII-B5. Ventajas:

1. Genera secuencias con periodos bastante grandes.
2. Es inmune a los criptoanálisis diferencial y lineal.
3. RC4 es uno de los cifradores más rápidos utilizados en aplicaciones serias y comerciales, siendo un cifrador de flujo, y como tal es básicamente un generador de números pseudo aleatorios inicializado desde una llave secreta por arriba de los 256 bytes (para el caso de ssl y para evitar el ataque Fluhrer, Martin,Shamir).

VIII-C. Secure Hash Algorithm (SHA)

VIII-C1. Concepto: Los algoritmos hash se utilizan como componentes de otros algoritmos y procesos criptográficos para proporcionar servicios de seguridad de la información. Las funciones hash se utilizan a menudo con la fotografía digital Algoritmos de firma, códigos de autenticación de mensajes-clave hash, funciones de derivación de claves, y generadores de números aleatorios. Un algoritmo de hash convierte un mensaje de longitud variable en una representación condensada de los datos electrónicos en el mensaje. Esta representación, o resumen del mensaje, se puede utilizar para las firmas digitales, la autenticación de mensajes, y otras aplicaciones seguras. Cuando se emplea en una aplicación de firma digital, el valor de hash del mensaje se firma en lugar del mensaje en sí mismo; el receptor puede usar la firma a verificar el firmante del mensaje y para autenticar la integridad del mensaje firmado. [15]

Los algoritmos hash seguras son componentes del Kit de herramientas de cifrado del NIST que ayuda organizaciones federales y del sector privado seleccionar los componentes de seguridad criptográfica y procesos para proteger sus datos, comunicaciones y operaciones.

Una función de hash seguro es una función resistente a colisiones, de una sola vía. Resistencia a la colisión significa que es muy difícil encontrar dos mensajes diferentes que producirán el mismo valor hash. Una forma significa que es fácil de calcular el valor hash de la entrada, pero es extremadamente difícil de reproducir la entrada desde el valor de hash, o encontrar otro de entrada que va a producir el mismo valor hash. Las funciones hash se utilizan a menudo para determinar si los datos han cambiado.

VIII-C2. Funciones Hash: En una función HASH dicho valor será generado por una función H de la forma: $h = H(M)$

Donde:
M: Es una longitud variable del mensaje.
H(M): Es un valor HASH de longitud finita.

El valor HASH es aparentemente para el mensaje de la fuente de un tiempo cuando el mensaje es asumido o conocido que está correcto, donde el Receptor autentifica dicho mensaje pero recalcula el valor porque dicha función si no considera que el valor sea secreto.

Mensaje = M (Función Resumen = h (M)
Firma (rúbrica): r = EdE h(M)
dE : Es la clave privada del emisor que firmará h(M)

VIII-C3. SHA-0 y SHA-1: SHA-1 forma parte de varias aplicaciones y protocolos de seguridad ampliamente utilizados, incluyendo TLS y SSL, PGP, SSH, S / MIME , y IPsec. Esas aplicaciones también pueden utilizar MD5 ; tanto MD5 y SHA-1 son descendientes de la MD4 . Hash SHA-1 también se utiliza en control de versiones distribuidos sistemas como Git, Mercurial, y monótono para identificar las revisiones, y para detectar la corrupción de datos o la manipulación. El algoritmo también se ha utilizado en la de Nintendo de Wii consola de juegos para la verificación de la firma cuando el arranque, pero un defecto significativo en las primeras implementaciones del firmware permite a un atacante eludir esquema de seguridad del sistema.

A, B, C, D y E son de 32 bits palabras de estado; F es una función no lineal que varía; n denota una rotación de la broca izquierda por n lugares; n varía para cada operación; W t es el mensaje de la palabra ampliada de ronda t; K t es la constante ronda de ronda t; denota adición módulo 2^23.

Ejemplo: Estos ejemplos son de SHA-1 resúmenes de mensajes en formato hexadecimal y en base 64 binario a ASCII codificación de texto.

SHA1 (El rápido zorro marrón salta sobre el perezoso dog) da hexadecimal:

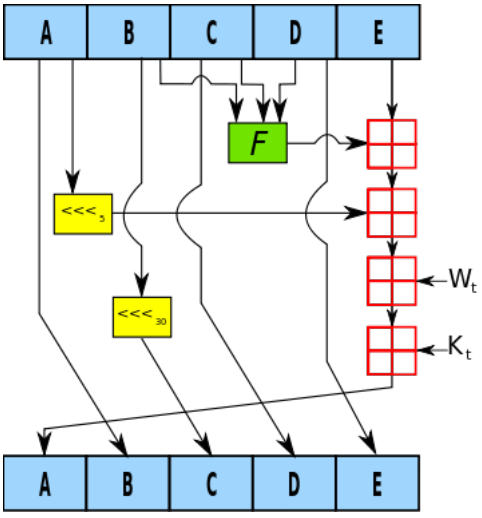

Figura 15. Representación gráfica de del SHA Básico

2fd4e1c67a2d28fced849ee1bb76e7391b93eb12
en base 64 binario a ASCII de codificación de texto:
L9ThxnotKPzthJ7hu3bnORuT6xI

VIII-C4. SHA-2: SHA-2 incluye cambios significativos respecto a su predecesor, SHA-1 . La familia SHA-2 consta de seis funciones hash con digestiones (valores hash) que son 224, 256, 384 o 512 bits: SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA -512 / 256 .

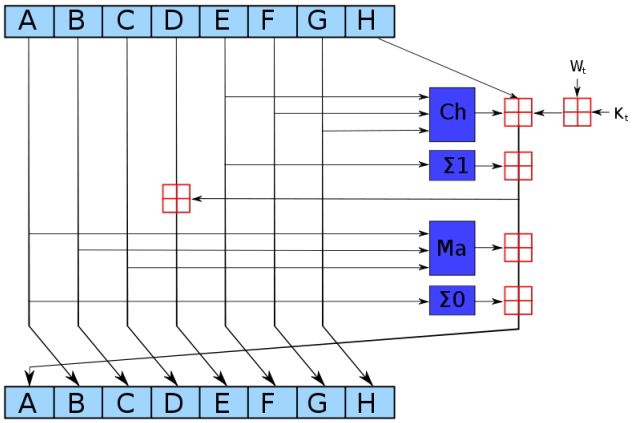

Figura 16. Representación gráfica de del SHA-2

Aplicaciones La función hash SHA-2 se lleva a cabo en algunas aplicaciones y protocolos de seguridad ampliamente utilizados, incluyendo TLS y SSL , PGP , SSH , S / MIME , y IPsec.

VIII-C5. SHA-3: La construcción de la esponja para las funciones de hash. P i son de entrada, z i son ordenadas de salida. El inusitado çapacidadç debe ser el doble de la resistencia deseada a la colisión o ataques imagen inversa.

SHA-3 utiliza la construcción de esponja, en el que se .ªbsorbe"los datos en la esponja, entonces el resultado es .exprimido"hacia fuera. En la fase de absorción, bloques de mensaje se XOR en un subconjunto del estado, que luego se transforma en su conjunto. En la fase de "squeeze", los bloques de salida se leen desde el mismo subconjunto del estado, alternadas con las transformaciones del Estado. El tamaño, r, de la parte del estado que se escribe y se lee que se llama la "tasa", y el sitio, c de la parte que está al margen de entrada / salida se denomina la çapacidad". La capacidad determina la seguridad del esquema. El nivel de seguridad máximo es la mitad de la capacidad.

En SHA-3, el estado consiste en un 5 x 5 matriz de palabras de 64 bits, 1600 bits en total. Keccak también se define para pequeños tamaños de palabra potencias de 2 W por debajo de 1 bit (25 bits en total estado). Tamaños pequeño estado pueden ser utilizados para probar los ataques criptoanalíticas y tamaños estado intermedio $dew = 8,200bits, paraw = 32,800bits$ se puede utilizar en aplicaciones prácticas, de peso ligero.

La transformación de un bloque es una permutación que utiliza xor, y no operaciones, y diseñado para una fácil implementación en software y hardware. Los autores afirman 12.5 ciclos por byte en un 2 Intel Core CPU. Sin embargo, en implementaciones de hardware, es notablemente más rápido que todos los demás finalistas.

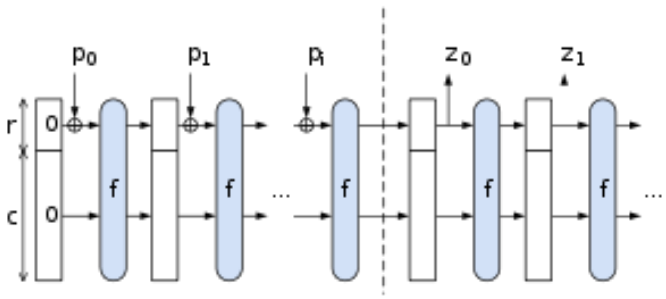

Figura 17. La construcción de la esponja para el hash.

Pi son de entrada, Zi son ordenadas de salida. El inusitado çapacidadç debe ser el doble de la resistencia deseada a la colisión o ataques imagen inversa.

VIII-D. Criptografía Cuántica

VIII-D1. Concepto:

VIII-D2. Mecánica Cuántica en Criptografía:

- Principio de Incertidumbre.
- Polarizacion de un Foton.
- Qubits. [1]
- Verschrankung
- Teorema de no-clonacion.

[5] [6] [4]

VIII-D3. Distribucion de Claves Cuanticas: [7]Se deben cumplir las siguientes condiciones:

- Ningun intruso puede obtener la clave transmitida
- Cualquier intento de intromisión para obtener la clave transmitida puede ser detectado con alta probabilidad
- Los usuarios pueden estar seguros de que están compartiendo la misma clave

VIII-D4. Protocolo BB84: El esquema propuesto en 1984 por Brassard y Bennett implica el envío de fotones preparados en diferentes estados de polarización. [3] [9]

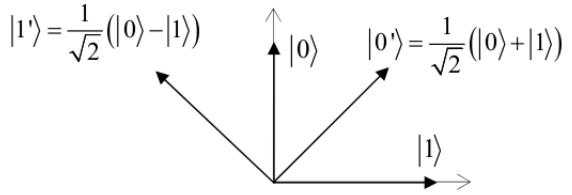

Figura 18. Polarizacion de fotones en el protocolo BB84.

Usando un filtro de polarización, se selecciona el ángulo de polarización con respecto a la horizontal. También se puede usar un método más sofisticado, en el que se usa un aparato conocido como Pockels cell que hace posible que el campo eléctrico del fotón oscile en el plano deseado. Como se vio anteriormente, se pueden elegir cuatro ángulos en particular: 0, 45, 90, y 135 grados, $\leftrightarrow \nearrow \updownarrow \searrow$. Los fotones polarizados en ángulos de 0 y 45 representan el valor binario 0, y los fotones polarizados en ángulos de 90 y 135 representan el valor binario 1; una vez hecha esta correspondencia, una secuencia de bits puede ser convertida en una secuencia de fotones polarizados.

1. Sin la Presencia de Eve.-

- Alicia codifica bits como fotones polarizados. La primera fila indica la secuencia de bits. La segunda indica la orientación usada por el filtro. La tercera indica el resultado de la polarización. [8]

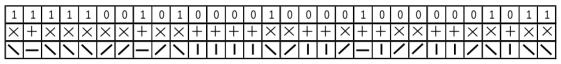

Figura 19. Alice codifica bits como fotones polarizados

Alicia genera una secuencia de fotones. Cuando Bob recibe los fotones, decide aleatoriamente si medir las polarizaciones a lo largo de las direcciones rectilíneas o a lo largo de las diagonales. De esta forma Bob podría extraer un bit de información por cada fotón.

- Bob decodifica fotones polarizados como bits. La primera fila indica la secuencia de fotones recibida. La segunda fila indica la configuración del cristal de calcita de Bob. La tercera fila indica el resultado de la medición.

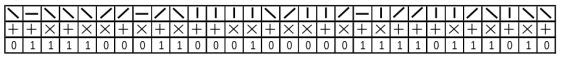

Figura 20. Bob decodifica fotones polarizados como bits

Este comportamiento se explica por el principio de incertidumbre de Heisenberg.

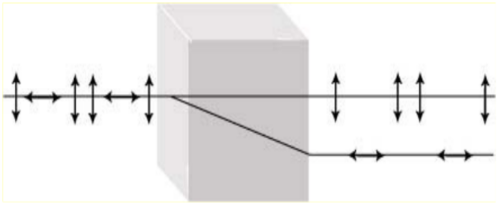

Figura 21. Cristal de Calcita que separa fotones.

Alicia le dice a Bob (canal clasico) el valor de los bits que debió haber medido, y la verificación debe asegurar que los bits de Bob concuerdan al 100 % con los de Alicia.

La probabilidad de detectar un espía que esté presente es $1 - \left(\frac{3}{4}\right)^N$.

- Alicia y Bob comparan un subgrupo de bits para probar, y verificar la presencia de un espía. En todos los bits se usó la misma orientación para polarizar y medir, y el valor del bit es igual, mostrando que no hay un espía presente.

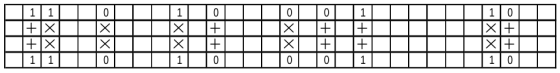

Figura 22. Alicia y Bob comparan un subgrupo de bits

- Clave generada. Los casos en que la orientación del polarizador es igual a la orientación del cristal, se representan por :), y esto significa que el bit se tomó como parte de la clave.

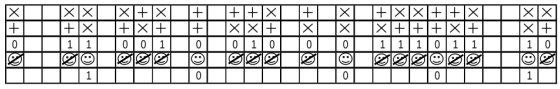

Figura 23. Clave Generada

2. Eve Presente.- En este caso, para que Eva pueda medir los fotones interceptados, debe haber escogido una orientación de polarización. Si Eva quisiera tener certeza de no ser detectada, necesitaría correr la suerte de escoger para cada bit transmitido la misma orientación de polarización que Alicia, lo cual, si el tamaño de la clave es lo suficientemente largo, sería prácticamente imposible. Si Eva elige la orientación incorrecta, modificará la polarización del fotón y su presencia podrá ser detectada en la fase de prueba. [2]

- Codificación de los bits de Alicia a estados de polarización:

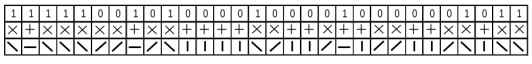

Figura 24. Codificación de los bits de Alicia

- Intercepción y medición de Eva (Si las polarizaciones no son iguales, Eva modifica irreparablemente el estado original):

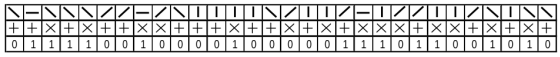

Figura 25. Intercepción y medición de Eva

- Bob, aún sin estar consciente de la presencia de Eva, realiza sus mediciones:

Figura 26. Bob realiza mediciones

- Pero en la fase de prueba, Alicia y Bob detectan la presencia de Eva y deciden desechar toda la secuencia de fotones para empezar de nuevo el procedimiento.

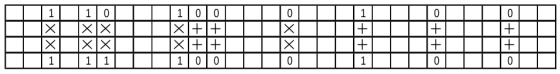

Figura 27. Detectan la presencia de Eva

VIII-E. Advanced Encryption Standard (AES)

VIII-E1. Concepto: Este algoritmo (también llamado, Rijndael por sus creadores Joan Daemen y Vincent Rijmen) fue publicado por el NIST en 2001, el cual presenta diferencias notable con respecto al resto de los cifradores simétricos, entre ellos destacan de manera sobresaliente: el tamaño de los bloques de 128 bits, manejo de claves de longitudes diferentes y uso de matemáticas polinomiales en estructuras de campos finitos. [11]

AES presenta dos características importantes:

1. **Linealidad:** El algoritmo debe probar que no existe linealidad o correlación alguna entre las entradas y las salidas, dicho de otra manera, que las salidas no sean una función directa de las entradas, y no haya posibilidad de determinar alguna dependencia entre los datos de entrada y los que salen. Para ello el algoritmo debe considerar la alternancia de claves, de ahí la importancia de utilizar una clave distinta en cada iteración.

2. **Propagación:** El algoritmo debe realizar una adecuada propagación de las diferencias de los bits, de manera que haya una alta dispersión de los bits en el mensaje de cifrado y no sean identificados en un ataque criptoanalista. Así, una buena propagación será capaz de bloquear un criptoanálisis diferencial ya que posibilitará derivar información de la clave a partir de conocer las probabilidades de las diferencias de la propagaciórn, por lo tanto, debe buscarse que estas probabilidades sean muy pequeñas y para ello, entre otros aspectos resulta relevante la cantidad de iteraciones que se deben realizar.

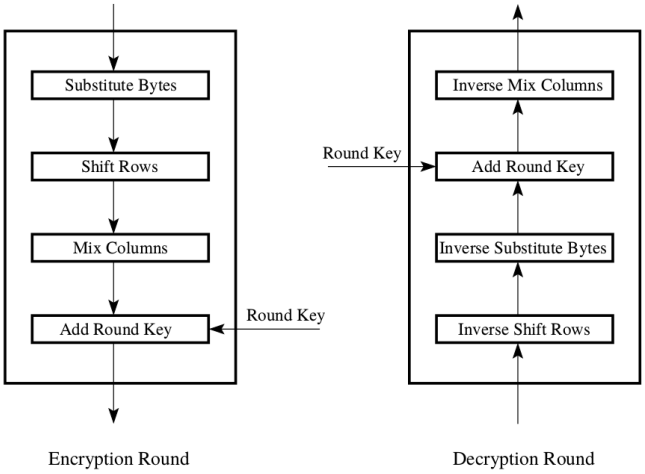

Figura 28. Gráfico de la encriptacion y desencriptacion con AES

VIII-E2. Proceso de creación del AES:

1. **SubBytes** es una sustitución de byte por byte que se realiza durante todo el proceso.
   - Este paso consiste de usar una tabla de 16 x 16 para encontrar un byte de reemplazo para un byte dado en la matriz de entrada.
   - Las entradas de la tabla de búsqueda se crean mediante el uso de las nociones de inversos multiplicativos en GF (2^8) y la aleatorización de bits de destruir las correlaciones a nivel de bits dentro de cada byte .
2. **ShiftRows** se usa para desplazar las filas de una matriz durante el proceso hacia adelante.
   - El objetivo de esta transformación es que revolver el orden de bytes dentro de cada bloque de 128 bits

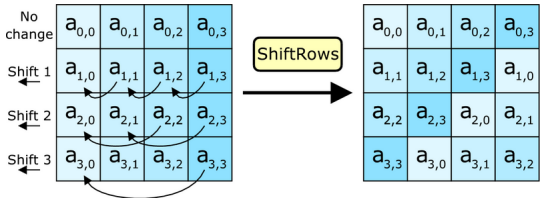

Figura 29. ShiftRows haciendo una que las filas se desplacen.

3. **MixColumns** hace la mezcla de los bytes en cada columna por separado durante el proceso hacia adelante.

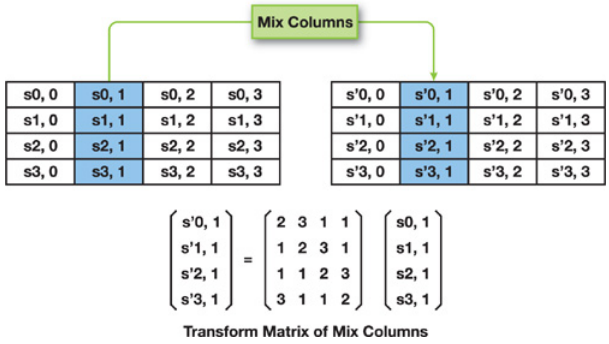

Figura 30. MixColumns haciendo una Multiplicacion de columnas.

4. **AddRoundKey** cada byte del «state» es combinado con la clave «round»; cada clave «round» se deriva de la clave de cifrado usando una iteración de la clave.

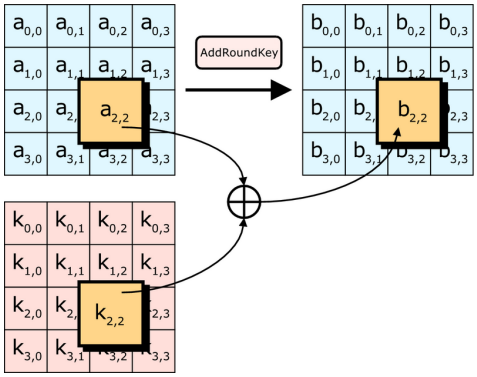

Figura 31. Ejemplo de AddRoundKey

En el paso AddRoundKey, cada byte del state se combina con un byte de la subclave usando la operación XOR. [10]

VIII-E3. Seguridad: El método más común de ataque hacia un cifrador por bloques consiste en intentar varios ataques sobre versiones del cifrador con un número menor de rondas. El AES tiene 10 rondas para llaves de 128 bits, 12 rondas para llaves de 192 bits, y 14 rondas para llaves de 256 bits.

En 2002, un ataque teórico, denominado .ªtaque XSL", fue anunciado por Nicolas Courtois y Josef Pieprzyk, mostrando una potencial debilidad en el algoritmo AES. Varios expertos criptográficos han encontrado problemas en las matemáticas que hay por debajo del ataque propuesto, sugiriendo que los autores quizá hayan cometido un error en sus estimaciones. Si esta línea de ataque puede ser tomada contra AES, es una cuestión todavía abierta. Hasta el momento, el ataque XSL contra AES parece especulativo; es improbable que nadie pudiera llevar a cabo en la práctica este ataque.

Hasta 2005, no se ha encontrado ningún ataque exitoso contra el AES. La Agencia de Seguridad Nacional de los Estados Unidos (NSA) revisó todos los finalistas candidatos al AES, incluyendo el Rijndael, y declaró que todos ellos eran suficientemente seguros para su empleo en información no clasificada del gobierno de los Estados Unidos. En junio de 2003, el gobierno de los Estados Unidos anunció que el AES podía ser usado para información clasificada.

IX. Conclusión

Esta investigación nos a demostrado que desde tiempos antiguos hemos vivido en una sociedad donde la criptografía a estado presente pero se ocultaban de nuestra vista.

La criptografía es y sera el método más utilizado para ocultar mensajes dentro de la guerra como en la zona computacional

El análisis de todas las clases de criptografía nos a hecho ver que no son eternas y que siempre se va a estar en constante creación de algoritmos de encriptación ya que tarde o temprano terminan rompiendo la encriptación.

Con respecto a nuestras investigaciones donde profundizamos en un tema en específico hemos podido ver que las matemáticas son la base indiscutible de la encriptación.

REFERENCIAS

[1] CASLAV BRUKNER. *Quantum Entanglement: Information-theoretical Foundations, Bell's Theorems and Quantum Communication Complexity*. PhD thesis, Universität Wien, 2002.

[2] JESUS NAVERRO FAUS. *HEISENBERG-¿Existe el Mundo cuando no lo Miras?* PhD thesis, Instituto de Fisica Corpuscular-Universidad de Valencia, 2010.

[3] ALBERTO T. PEREZ I. *MAX PLANCK-La Revolucion de lo muy Pequeño*. PhD thesis, Universidad de Sevilla, 2012.

[4] RAFAEL LAHOZ-BELTRA. *TURING-Pensando en Maquinas que Piensan*. PhD thesis, Universidad Complutence de Madrid, 2012.

[5] Vicente Martín. Criptografía cuántica aplicada. *Criptografía Cuántica Aplicada*, 2007.

[6] Jesús Martínez Mateo. Criptografía cuántica aplicada. *Criptografía Cuántica Aplicada*, 2008.

[7] HERNAN ORTIZ ROJAS. *Fundamentos de Criptografia Cuantica*. PhD thesis, UNIVERSIDAD EAFIT, 2007.

[8] MIGUEL A. SABADELI. *FEYNMAN-Cuando un Foton conoce un Electron*. PhD thesis, INTA-CSIC, 2012.

[9] MATTHIAS SCHOLZ. *Quantum Key Distribution via BB84 An Advanced Lab Experiment*. PhD thesis, U—, 2007.

[10] William Stallings. Cryptography and network security. *Cryptography and Network Security*, 4to Edicion:134–165, 2005.

[11] William Stallings. Cryptography and network security: Principles and practice. *Advanced Encryption Standar*, 5to Edicion:147–191, 2005.

[12] Alfred J.Menezes Paul C.van Oorschot Scott A. Vanstone. Handbook of applied cryptography. *Introduction, Information security and cryptography*, 4to Edicion:1–2, 1996.

[13] Alfred J.Menezes Paul C.van Oorschot Scott A. Vanstone. Handbook of applied cryptography. *Basic terminology and concepts*, 4to Edicion:11–15, 1996.

[14] Alfred J.Menezes Paul C.van Oorschot Scott A. Vanstone. Handbook of applied cryptography. *Symmetric-key encryption*, 4to Edicion:15–21, 1996.

[15] Alfred J.Menezes Paul C.van Oorschot Scott A. Vanstone. Handbook of applied cryptography. *Public-key cryptography*, 4to Edicion:25–31, 1996.

[16] Alfred J.Menezes Paul C.van Oorschot Scott A. Vanstone. Handbook of applied cryptography. *DES*, 4to Edicion:250–256, 1996.

ÍNDICE DE FIGURAS

1. Representación gráfica de Transposición . . . 3
2. Representación gráfica de Sustitución . . . 3
3. Representación gráfica de la criptografía Simétrica . . . 3
4. Representación gráfica de la criptografía asimétrica . . . 4
5. Representacion de Cesar . . . 6
6. Representación gráfica de Escítala . . . 6
7. Cuadro de Vigenere . . . 7
8. Cuadro de Vernam . . . 7
9. Representacion de Polybios . . . 7
10. La estructura general de Feistel en DES . . . 8
11. Diagrama de Blowfish . . . 9
12. Diagrama de Blowfish a nivel de bits . . . 10
13. Funcionamiento del sistema Diffie-Hellman . . . 12
14. Esquema de la intrusión Man-in-the-middle . . . 12
15. Representación gráfica de del SHA Básico . . . 15
16. Representación gráfica de del SHA-2 . . . 15
17. La construcción de la esponja para el hash. . . . 15
18. Polarizacion de fotones en el protocolo BB84. . . . 16
19. Alice codifica bits como fotones polarizados . . . 16
20. Bob decodifica fotones polarizados como bits . . . 16
21. Cristal de Calcita que separa fotones. . . . 16
22. Alicia y Bob comparan un subgrupo de bits . . . 17
23. Clave Generada . . . 17
24. Codificación de los bits de Alicia . . . 17
25. Intercepción y medición de Eva . . . 17
26. Bob realiza mediciones . . . 17
27. Detectan la presencia de Eva . . . 17
28. Gráfico de la encriptacion y desencriptacion con AES . . . 18
29. ShiftRows haciendo una que las filas se desplacen. . . . 18
30. MixColumns haciendo una Multiplicacion de columnas. . . . 18
31. Ejemplo de AddRoundKey . . . 18