EAP. DE INGENIERIA DE SISTEMAS E INFORMATICA

Módulo de

Tópicos I

II Unidad

Lic. Luis Ramirez Milla

Abril - 2014

Chimbote – Perú

I N D I C E

Página

PRESENTACION

El propósito del presente módulo es servir como guía a los estudiantes que cursan la asignatura Tópicos I. El material presentado es apropiado para el desarrollo de la segunda unidad, describiendo en forma detallada términos que forman parte de la formación integral del futuro ingeniero de sistemas.

El presente modulo está estructurado en cuatro capítulos, cada uno de los cuales se detallan a continuación:

Capitulo I, describe el marco de referencia para el tratamiento y manipulación de cadenas de texto.

Capitulo II, describe algoritmos para la compresión de datos.

Capitulo III, describe el marco de referencia para algoritmos criptográficos.

Capitulo IV, describe el marco de referencia para el criptoanálisis y esteganografia relacionada con la criptografía.

Finalmente se incluyen las referencias bibliográficas.

UNIDAD II

SEGURIDAD Y CRIPTOGRAFIA

CAPITULO I

Búsqueda de cadenas

1.1 Introducción

A menudo sucede que los datos a procesar no se descomponen lógicamente en registros independientes que representen pequeñas partes identificables. Este tipo de datos se caracteriza fácilmente por el hecho de que se pueden escribir en forma de cadenas: series lineales de caracteres.

Las cadenas son el centro de los sistemas de tratamiento de texto, que proporcionan una gran variedad de posibilidades para la manipulación de textos. Tales sistemas procesan cadenas alfanuméricas, que pueden definirse como serie de letras, números y caracteres especiales. Otro tipo de cadena es la cadena binaria, que es una simple serie de valores 0 y 1. Los dos tipos de cadena son equivalentes.

Una operación fundamental sobre las cadenas es el reconocimiento de patrones: dada una cadena alfanumérica de longitud N y un patrón de longitud M (N>M), encontrar una ocurrencia del patrón dentro del texto. El reconocimiento de patrones se puede caracterizar como un problema de búsqueda en el que el patrón seria la clave.

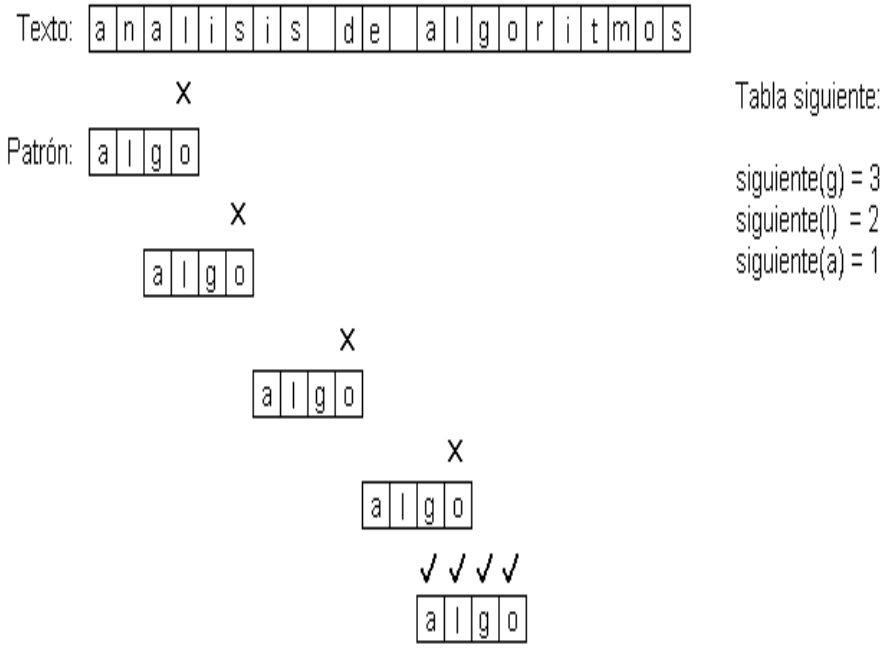
El objetivo de búsqueda en cadenas es hallar la localización de un patrón de texto específico dentro de un texto largo (ejm., una oración, un párrafo, un libro, etc.). En el cual los requisitos son velocidad y eficiencia

1.2 Algoritmo de fuerza bruta

Características

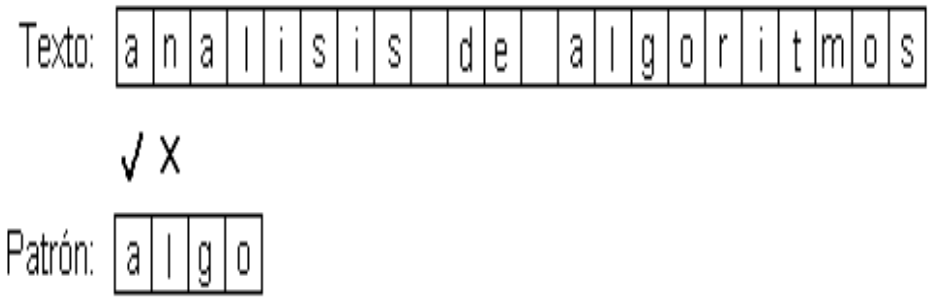
- Es el algoritmo más simple posible.
- Consiste en probar todas las posibles posiciones del patrón en el texto.
- Requiere espacio constante.

- Realiza siempre saltos de un carácter.
- Compara de izquierda a derecha.

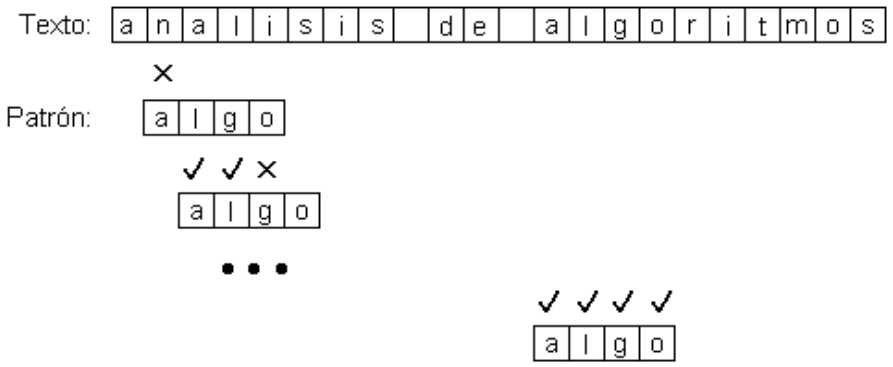


Lógica

- Se sitúa el patrón en la primera posición, y se compara carácter a carácter hasta encontrar una discrepancia o llegar al final del patrón.
- Se pasa a la siguiente posición y se repite el proceso.
- El proceso finaliza al alcanzar el final del texto
- No existe un preprocesamiento del patrón.



Si se detiene la búsqueda por una discrepancia, se desliza el patrón en una posición hacia la derecha y se intenta calzar el patrón nuevamente.



Descripción

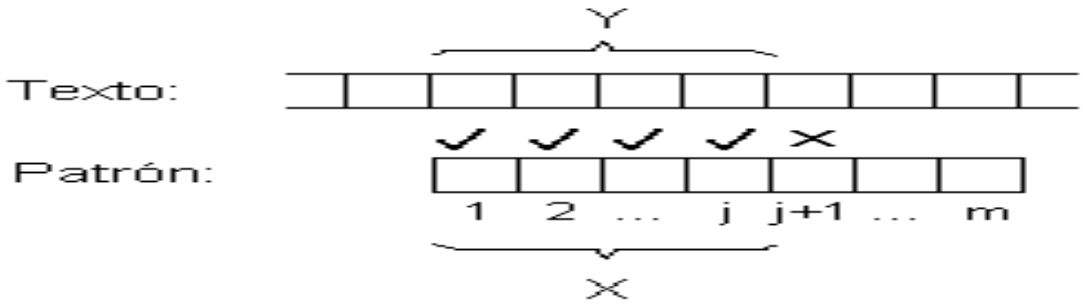
No requiere ninguna fase de preproceso previo, ni un espacio extra constante además del espacio asignado al patrón y al texto.

Para la búsqueda:

- Consiste en la comparación de todas las posiciones del texto entre 0 y el n-m, si una ocurrencia del patrón corresponde o no.
- Si encuentra una no ocurrencia, o una ocurrencia total del patrón, salta un carácter hacia la derecha.

1.3 Algoritmo de Knuth-Morris-Pratt

Suponga que se está comparando el patrón y el texto en una posición dada, cuando se encuentra una discrepancia

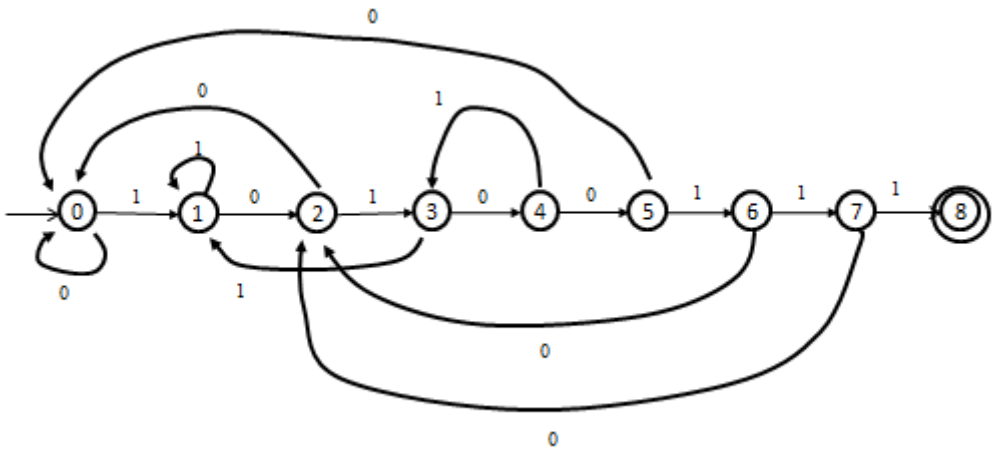


Sea **X** la parte del patrón que coincide con el texto, y **Y** la correspondiente parte del texto, y suponga que el largo de X es **j**. El algoritmo de fuerza bruta mueve el patrón una posición hacia la derecha, sin embargo, esto puede o no puede ser lo correcto en el sentido que los primeros **j-1** caracteres de X pueden o no pueden coincidir con los últimos **j-1** caracteres de Y.

La observación clave que realiza el algoritmo Knuth-Morris-Pratt (KMP) es que **X** es igual a **Y**, por lo que la pregunta planteada en el párrafo anterior puede ser respondida mirando solamente el patrón de búsqueda, lo cual permite precalcular la respuesta y almacenarla en una tabla. Por lo tanto, si deslizar el patrón en una posición no funciona, se puede intentar deslizarlo en 2, 3, ..., hasta **j** posiciones.

Por ejemplo:
Cuando se está buscando X=10100111 en Y=101010011100101, se comienza por detectar la discordancia en el quinto carácter, pero se debe retroceder al tercero para continuar la búsqueda, puesto que de lo contrario se perdería la concordancia.

Máquina de estados finitos: 10100111



Para determinar el desplazamiento (posiciones de reinicialización) en caso de no existir concordancia. Cada desplazamiento hacia atrás corresponde a

desplazar una posición hacia la derecha del patrón con respecto a la cadena evaluada, y se reinicia el proceso de reconocimiento.

1.4 Algoritmo de Rabin-Karp

El método se basa en calcular la función de dispersión (hash) para la posición i del texto conociendo su valor para la posición i-1. Se supone que se transforman los **M** caracteres en números agrupándolos en una palabra que podría tratarse como un número entero. Esto equivale a escribir los caracteres como números en un sistema de base **d**, donde **d** es el número de caracteres posibles.

Calcula un **valor hash** para el patrón, y para cada subsecuencia de M-caracteres de texto.

- Si los valores hash son diferentes, se calcula un valor para la siguiente secuencia.
- Si los valores hash son iguales se usa una comparación de **Fuerza Bruta** (la función de dispersión puede presentar colisiones).

Valor Hash de "AAAAA" es 37
Valor Hash de"AAAAH" es 100

1) AAAAAAAAAAAAAAAAAAH
AAAAH
37<>100
2) AAAAAAAAAAAAAAAAAAH
AAAAH
37<>100
3) AAAAAAAAAAAAAAAAAAH
AAAAH
37<>100
.
.
N) AAAAAAAAAAAAAAAAAAH
AAAAH
100=100

Función hash

La función hash tiene la forma $d(k)=x_k \bmod q$; donde **q** es un número primo grande que será el tamaño de la tabla de dispersión y x_k se calcula de la forma indicada más abajo.

Para transformar cada subcadena de **m** caracteres en un entero lo que hacemos es representar los caracteres en una base **B** que en el planteamiento original coincide con el tamaño del alfabeto. Por tanto el entero x_i correspondiente a la subcadena de texto $C_i...C_{i+m-1}$ sería:

$$X_i = C_i \times B^{m-1} + C_{i-1} \times B^{m-2} + ... + C_{i+m-1}$$

podemos calcular el valor de x_{i+1} en función de x_i

$$x_{i+1} = x_i \times B - C_i \times B^m + C^{i+m}$$

Es decir, si la cadena es un número en base B, el nuevo valor será el resultado de multiplicar por la base el valor anterior eliminado el dígito de mayor peso (ya que no está en la cadena) y añadiendo como componente de menor peso el valor del nuevo símbolo.

Siguiendo este planteamiento, y dependiendo de la longitud del patrón, el x_i podría superar el rango de enteros representable por el computador. Para que esto no suceda se usa la función módulo (resto de la división). Como la función módulo es asociativa, podemos calcular el x_{i+1} incrementalmente a partir de cada x_i.

CAPITULO II

Compresión de archivos

2.1 Fundamentos

En general, la mayor parte de los archivos tienen un gran nivel de redundancia. Las técnicas de compresión de archivos sirven a menudo para archivos de texto (en los que ciertos caracteres aparecen con mucha más frecuencia que otras), para archivos de exploración de imágenes codificadas (que presentan grandes zonas homogéneas) y para archivos de representación digital de sonido y de otras señales analógicas (que presentan un gran número de patrones repetidos).

¿En qué consiste?

Básicamente la compresión consiste en tomar una trama de símbolos y transformarlos en códigos/claves. Si la compresión es eficiente, las claves resultantes ocuparán menor espacio que los símbolos originales. La decisión de obtener una codificación a partir de ciertos símbolos (o conjunto de ellos) está basada en un modelo.

El modelo es simplemente una colección de datos y reglas usados para procesar la entrada de símbolos y determinar su correspondiente codificación a la salida.

Por ejemplo un programa usa el modelo para definir aproximadamente las probabilidades para cada símbolo y el codificador para producir una codificación apropiada basada en esas probabilidades.

Modelo y Codificación

Los conceptos de modelo y codificación son cosas diferentes.

Usualmente se cae en el error de emplear el término de "codificación" para referirse a todo el proceso de compresión de datos en vez de considerarlo como un simple componente de ese proceso.

Compresión de datos basada en modelos

Cualquier compresor de datos puede describirse en términos de un modelo de la fuente de datos a comprimir más un codificador.
En el caso de la compresión de texto, frecuentemente se utiliza un modelo probabilístico (que en estas implementaciones es de orden 0 o sin memoria) y un codificador de Huffman.

El objetivo del modelo probabilístico es averiguar las probabilidades de ocurrencia de los símbolos codificados. Concretamente, un modelo de orden 0 calcula la probabilidad de un símbolo contando el número de veces que ese símbolo aparece en la secuencia de símbolos.

Por otro lado, la finalidad del codificador de longitud variable es utilizar las probabilidades calculadas por el modelo para asignar un código de compresión corto a los símbolos más frecuentes a costa de asignar un código de compresión largo a los símbolos más raros.

Figura 1: Compresión de datos utilizando un modelo probabilístico y un codificador de longitud variable.

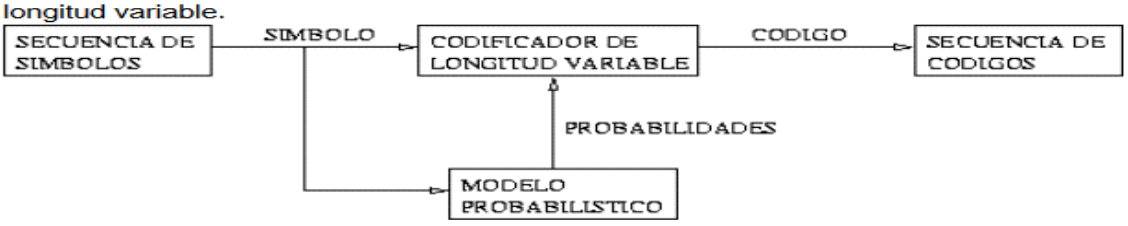
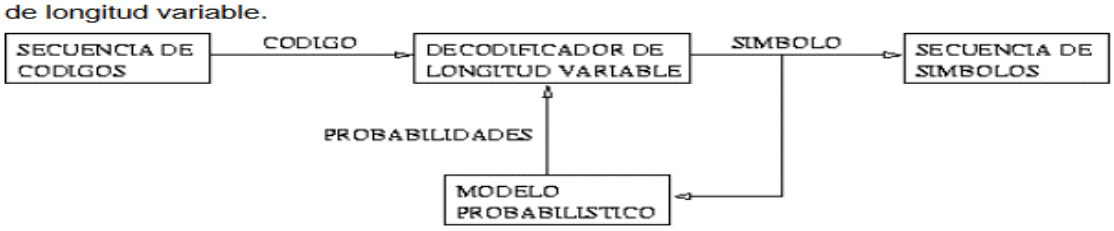


Figura 2: Expansión de datos utilizando un modelo probabilístico y un decodificador de longitud variable.

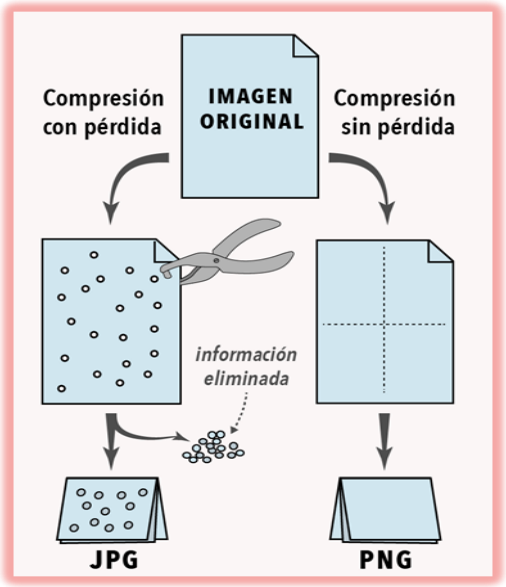


Técnicas de compresión

Dentro de las técnicas de compresión de datos, y atendiendo a la reversibilidad de la información original, hay dos grandes familias:

- **"lowless" ó sin perdida.** Para datos en los que es imprescindible que no se pierda nada de información, como por ejemplo registros de bases de datos, ficheros ejecutables, hojas de cálculo...etc.

- **"lossy" ó con perdida.** Para datos en los que se permite cierta pérdida de información "sin que se note demasiado", como por ejemplo en ficheros en MP3, imágenes en JPEG, PNG...etc. Aquí una pequeña disminución en la calidad final no se nota demasiado, pero influye muy positivamente en la reducción del peso del fichero.



Tipos de compresión lowless

- **Algoritmos estadísticos.** Utilizan las propiedades estadísticas de la fuente para mejorar la codificación (a cada mensaje de la fuente asigna una cadena de símbolos del alfabeto de salida). Se trata de aprovechar la redundancia de información de la fuente para conseguir esa compresión.

- Algoritmo huffman
- Algoritmo Shannon-Fano
- Algoritmos Aritméticos

- **Algoritmos basados en diccionarios.** Son las técnicas más utilizadas, generalmente se las implementa en conjunción con compresores estadísticos.

- Algoritmo Run-Length
- Algoritmo LZW
- Algoritmo LZ77

2.2 Codificación por longitud de series

El tipo más simple de redundancia que se puede encontrar en un archivo son las largas series de caracteres repetidos.

Por ejemplo, en la cadena siguiente:

AAAABBBAABBBBBCCCCCCCCDABCBAAABBBBCCCD

Esta cadena se puede codificar de forma más compacta reemplazando cada repetición por un solo ejemplar del carácter repetido seguido del número de veces que se repite.

Sería mejor decir que esta cadena consiste de 4 letras A, seguida de 3 B, seguidas de 2 A, seguidas de 5 B, etc. Esta forma de comprimir una cadena se denomina codificación por longitud de series.

4A3BAA5B8CDABCB3A4B3CD

Algunos inconvenientes:

El método de compresión de caracteres no funciona con cadenas que contienen dígitos.

Si se utilizan otros caracteres para codificar las longitudes de las series, no podría aplicarse el método a cadenas que contengan esos caracteres.

¿Cómo se puede lograr que algunas letras representen dígitos y otras formen parte de la cadena que se va a codificar?

Una solución consiste en utilizar un carácter con pocas probabilidades de aparecer en el texto, al que se denomina carácter de escape. Cada aparición de dicho carácter indica que las dos letras siguientes forman un par (longitud, carácter), en el que la i-ésima letra del alfabeto representa una longitud igual a i.

QDABBBAAQEBQHCDABCBAAAQDBCCCD

Tomando Q como carácter de escape

Nueva interrogante:

¿Pero qué pasa si el carácter de escape aparece también en la serie de entrada?

Una solución a este problema consiste en utilizar para representar al carácter de escape una secuencia de escape con una longitud de serie cero. De esta manera el espacio en blanco podría representar al cero, y la secuencia de escape **"Q<espacio>"** representaría a cualquier aparición de Q en la entrada.

AAAAQBBBAABBBBBQQQQQCCCCCCCCCDABCBAAABBBBCCCD

QDAQ BBBAAQEBQEQQHCDABCBAAAQDBCCCD

Otro ejemplo:

Una secuencia de 51 A, debería codificarse como: QZAQYA

2.3 Codificación de longitud variable

La idea es abandonar la forma como se almacenan habitualmente los archivos de texto; en lugar de emplear siete u ocho bits por carácter, se utilizaran solamente unos pocos bits para los caracteres más frecuentes y algunos más para los que aparecen raramente.

Ejemplo:

Palabra a codificar: **ABRACADABRA**

Empleando el código binario compacto estándar, en el que la representación con cinco bits de **i** reproduce a la i-ésima letra del alfabeto (0 para los espacios en blanco), proporcionan la siguiente serie de bits.

00001000101001000001000110000100100000010001010010000001

Con el código estándar visto anteriormente, la D que aparece una sola vez, necesita el mismo número de bits que la A, que aparece cinco veces.

Con un código de longitud variable se puede alcanzar ahorros de espacio codificando los caracteres más frecuentemente utilizados con el menor número de bits posible, de forma que se minimice el número total de bits.

Codificando A por 0, B por 1, R por 01, C por 10 y D por 11 se tendría la siguiente codificación:

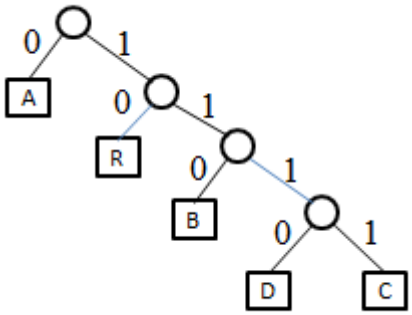
0 1 01 0 10 0 11 0 1 01 0

Esta cadena utiliza solo 15 bits en lugar de los 55 anteriores, pero depende de los espacios en blanco como delimitador, aun con estos (10 delimitadores) que sumarian 25 bits, aún sigue siendo mucho más compacto.

Los delimitadores no son necesarios si el código de un carácter no es el prefijo de otro. Por ejemplo si se codifica A por 0, B por 110, C por 1111, D por 1110 y R por 10, no hay más que una sola forma de codificar la cadena que emplea 23 bits.

01101001111011100110100

Una forma fácil de representar el código es a través de un trie (ordenación por residuos).



El código de cada carácter se determina por el camino desde la raíz al carácter con 0 para “ir a la izquierda” y 1 para “ir a la derecha”, como es habitual en un trie. Cada vez que se encuentra un nodo externo, se da salida al carácter del nodo y se comienza de nuevo en la raíz.

¿Pero qué trie es el mejor para utilizar?

2.4 Algoritmo de Huffman

El algoritmo de Huffman es un algoritmo para la construcción de códigos de Huffman, desarrollado por David A. Huffman en 1952 y descrito en “A Method for the Construction of Minimum-Redundancy Codes”.

Este algoritmo toma un alfabeto de n símbolos, junto con sus frecuencias (cantidad ó porcentajes) de aparición asociadas, y produce un código de Huffman para ese alfabeto y esas frecuencias.

Descripción

El algoritmo consiste en la creación de un árbol binario (trie) que tiene cada uno de los símbolos por hoja, y construido de tal forma que siguiéndolo desde la raíz a cada una de sus hojas se obtiene el código Huffman asociado.

1. Se crean varios árboles, uno por cada uno de los símbolos del alfabeto, consistiendo cada uno de los árboles en un nodo sin hijos, y etiquetado cada uno con su símbolo asociado y su frecuencia de aparición.
2. Se toman los dos árboles de menor frecuencia, y se unen creando un nuevo árbol. La etiqueta de la raíz será la suma de las frecuencias de las raíces de los dos árboles que se unen, y cada uno de estos árboles será un hijo del nuevo árbol. También se etiquetan las dos ramas del nuevo árbol: con un **0 la de la izquierda**, y con un **1 la de la derecha**.
3. Se repite el paso 2 hasta que sólo quede un árbol.

Con este árbol se puede conocer el código asociado a un símbolo, así como obtener el símbolo asociado a un determinado código.

Obtención del código asociado a un símbolo

Procedimiento:

1. Comenzar con un código vacío
2. Iniciar el recorrido del árbol en la hoja asociada al símbolo
3. Comenzar un recorrido del árbol hacia arriba
4. Cada vez que se suba un nivel, añadir al código la etiqueta de la rama que se ha recorrido
5. Tras llegar a la raíz, invertir el código
6. El resultado es el código Huffman deseado

Obtención de un símbolo a partir del código

Procedimiento:

1. Comenzar el recorrido del árbol en la raíz de éste
2. Extraer el primer símbolo del código a descodificar
3. Descender por la rama etiquetada con ese símbolo
4. Volver al paso 2 hasta que se llegue a una hoja, que será el símbolo asociado al código

En la práctica, casi siempre se utiliza el árbol para obtener todos los códigos de una sola vez; luego se guardan en tablas y se descarta el árbol.

Ejemplo de uso

La tabla describe el alfabeto a codificar, junto con las frecuencias de sus símbolos. En el gráfico se muestra el árbol construido a partir de este alfabeto siguiendo el algoritmo descrito.

Símbolo	Frecuencia
A	0,15
B	0,30
C	0,20
D	0,05
E	0,15
F	0,05
G	0,10

Árbol para construir el código Huffman del ejemplo

Se puede ver con facilidad cuál es el código del símbolo E: subiendo por el árbol se recorren ramas etiquetadas con 1, 1 y 0; por lo tanto, el código es 011. Para obtener el código de D se recorren las ramas 0, 1, 1 y 1, por lo que el código es 1110.

Limitaciones

Para poder utilizar el algoritmo de Huffman es necesario conocer de antemano las frecuencias de aparición de cada símbolo, y su eficiencia depende de lo próximas a las frecuencias reales que sean las estimadas. Algunas implementaciones del algoritmo de Huffman son adaptativas, actualizando las frecuencias de cada símbolo conforme recorre el texto.

La eficiencia de la codificación de Huffman también depende del balance que exista entre los hijos de cada nodo del árbol, siendo más eficiente conforme menor sea la diferencia de frecuencias entre los dos hijos de cada nodo.

Ejemplo

Se tiene la cadena de longitud 64:

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAB

El algoritmo de Huffman aplicado únicamente a los símbolos devuelve el código:

11111111111111111111111111111111111111111111111111111111111110

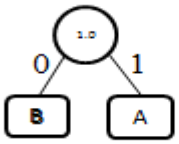
También de longitud 64.

Sin embargo, si antes de utilizar el algoritmo, se agrupan los símbolos en las palabras "AA", "AB" (que se codifican como 1, 0), el algoritmo devuelve la siguiente cadena:

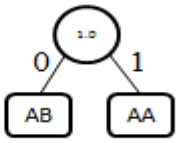
11111111111111111111111111111110

Que tiene longitud 32, la mitad que si no se hubiera agrupado.

Si observa el árbol de Huffman, se puede comprobar que la diferencia de frecuencias entre las ramas del árbol es menor cuando esta se agrupa.



Símbolo	Frecuencia
A	0,9844
B	0,0156



Símbolo	Frecuencia
AA	0,9688
AB	0,0312

¿Y para la decodificación?

El árbol se debe almacenar o bien enviarlo junto con el mensaje para decodificarlo. Así el ahorro de espacio antes mencionado no es totalmente exacto, porque el mensaje no se puede decodificar sin el trie y se debe en tener en cuenta el coste que significa almacenar el árbol (array código) junto con el mensaje.

Por lo tanto, la codificación Huffman solo es efectiva para archivos largos donde el ahorro de espacio en el mensaje es suficiente como para compensar el coste.

CAPITULO III

Criptología I: Criptografía

3.1 Fundamentos

Información y sus estados

- La información actual es digital
- Su manejo implica considerar estados:
  - Adquisición
  - Creación
  - Almacenamiento
  - Proceso (transformación, análisis, etc.)
  - Transmisión

Problemas en manejo de Información

- Confiabilidad de las fuentes y de la información misma
- Integridad (verificación)
- Confidencialidad
- Disponibilidad
- Control de Acceso
- Autenticación de las partes
- No repudio

Problemática

- El canal es público
- Uso de canales seguros nada fácil, práctico, ni barato
- Las fuentes pueden no ser compatibles ni confiables
- Los sistemas de análisis y toma de decisiones asumen lo contrario
- Los resultados y reportes pueden ser equivocados y las decisiones se toman a partir de esos resultados

- Expuesta a ataques de todo tipo
- Nada fácil es crear un modelo para estudiar la seguridad
  - Redes heterogéneas de todos los tipos
  - Plataformas, medios, SO's, arquitecturas distintas
  - La pesadilla: Internet

Seguridad de la Información

- Técnicas, procedimientos, políticas y herramientas para proteger y resguardar información en medios y dispositivos electrónicos
- Proteger la información almacenada en los equipos y la que se transfiere e intercambia por canales públicos
- Proteger información de amenazas, ataques y vulnerabilidades reales y potenciales
- Garantizar propiedades de información en todos sus estados: creación, modificación, transmisión y almacenamiento
- Implementar servicios de seguridad usando mecanismos útiles y eficientes

Servicios y Mecanismos de Seguridad

• Servicios	Mecanismos
— Confidencialidad	Cifrado
— Integridad	Funciones Hash
— Autenticación	Protocolos Criptográfico
— Control de Acceso (CA)	Esquemas de CA
— No Repudio	Firma Digital
— Disponibilidad	No se puede !!

Seguridad Informática

- Se deben implementar los 5 primeros servicios para disminuir el riesgo de sufrir indisponibilidad
- El Triángulo de Oro de la Seguridad incluye:
  - Confidencialidad
  - Integridad
  - Disponibilidad

3.2 Criptologia

La criptología se divide en:

- Criptografía.- Diseño de sistemas de comunicaciones secretas

- Criptoanálisis.- Estudio de las formas de transgredir los sistemas de comunicaciones secretas

- Esteganografía.- Estudio y aplicación de técnicas que permiten ocultar mensajes u objetos dentro de otros llamados portadores, de modo que no se perciba su existencia.

Criptografía

Oculta información aplicando al mensaje **M**, una transformación (algoritmo) **F,** usando una llave **K** y produce el texto cifrado **C.**

$$C = F_k(M)$$

El algoritmo **F** debe ser público.
La seguridad debe basarse en:

- Diseño y fortaleza de **F**

– Mantener **K** en secreto

Algoritmo **F** integra 2 procesos:

Cifrado: $F_k(M) = C$
Descifrado: $F'_k(C) = M$

Criptografía y Seguridad

Cuatro (4) de los cinco (5) servicios estandarizados se implementan usando Criptografía:

– Confidencialidad
– Integridad (Verificación)
– Autenticación
– No Repudio
– No se puede hablar de seguridad sin hablar de Criptografía

Algoritmos

- **Algoritmos Simétricos o de Llave Secreta**

  Basan su diseño en operaciones elementales y buscan eficiencia. La seguridad depende del tiempo y los recursos de cómputo

  – DES (Data Encription Standard) . Anterior estándar mundial
  – AES (Advanced Encription Standard). Nuevo estándar mundial
  – Triple-DES

- **Algoritmos Asimétricos o de Llave Pública**

  Diseño basado en problemas matemáticos, seguros computacionalmente. La seguridad no depende de tiempo ni de recursos de cómputo. Pero...son ineficientes

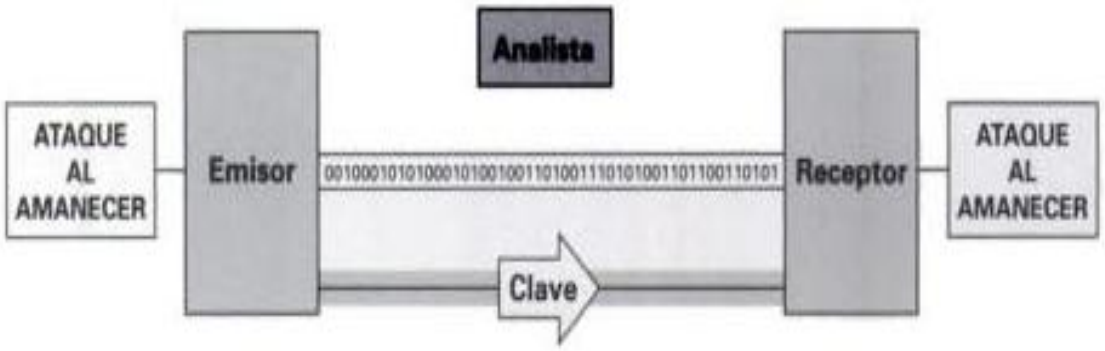
- RSA (Estándar de facto)
- ElGamal
- DSS (Digital Signature Standard)

- **Algoritmos Hash**

  Diseño basado en operaciones elementales. Son eficientes. La seguridad depende del tiempo y recursos de cómputo y proporcionan una huella digital del mensaje.

  - MD5
  - SHA-1

Reglas del juego



El emisor envía un mensaje (denominado el texto en claro) al receptor, transformándolo en una forma secreta propicia para la transmisión (denominada el texto cifrado) por medio de un algoritmo de criptografía (el método de cifrado) y algunos parámetros (claves).

Para leer el mensaje, el receptor debe tener un algoritmo criptográfico equivalente (el método de descifrado) y los mismos parámetros clave que transformaran el texto cifrado en el texto original.

Implantación del sistema

En la mayoría de las aplicaciones, el objetivo del criptógrafo es desarrollar sistemas de bajo coste con las características de que el criptoanalista debe invertir para leer los mensajes mucho más de lo que está dispuesto a pagar.

En el diseño de algoritmos se intenta seguir la pista a los costes de seleccionar el mejor algoritmo; en criptología, los costes desempeñan un papel fundamental en el proceso de diseño.

3.3 Métodos elementales

Entre los métodos de cifrado más simples y de los más antiguos se encuentra la cifra de cesar. Si una letra del texto en claro es la N-esima del alfabeto se reemplaza por la (N+K)-esima letra del alfabeto, siendo K un numero entero fijo.

Ejemplo.

Texto en claro: ATAQUE AL AMANECER

Considerando **K=1**;

Texto cifrado : BUBRVFABMABNBOFOFS

En este caso el método es débil porque el criptoanalista solo tiene que adivinar el valor de K: intentando con cada una de las 26 opciones, podrá estar seguro de leer el mensaje.

Un método mucho mejor consiste en utilizar una tabla general para definir la sustitución a efectuar: para cada letra del alfabeto en claro la tabla dice que letra poner en el texto cifrado.

Ejemplo de correspondencia:

ABCDEFGHIJKLMNOPQRSTUVWXYZ

NUEVOS BRILYHAMTWZYGQPFJVKC

Texto en claro: ATAQUE AL AMANECER

Texto cifrado : UQUZPSNUHNUAUMSVSY

Esta técnica es más poderosa que la simple cifra de cesar porque el criptoanalista tendría que ensayar con muchas tablas (alrededor de $27! > 10^{28}$) para estar seguro de leer el mensaje.

Se puede complicar más, utilizando varias tablas. Un ejemplo simple de esto es una extensión de la cifra de cesar denominada la cifra de vigenere. Se utiliza una pequeña clave repetida para determinar el valor de K para cada letra.

En cada paso, el índice de la letra de la clave se añade al de la letra del texto en claro para determinar el índice de la letra del texto cifrado.

Ejemplo:

Clave: ABCABCABCABCABC

Texto en claro: ATAQUE AL AMANECER
Texto cifrado: BVDRWHACOACPBPHDGU

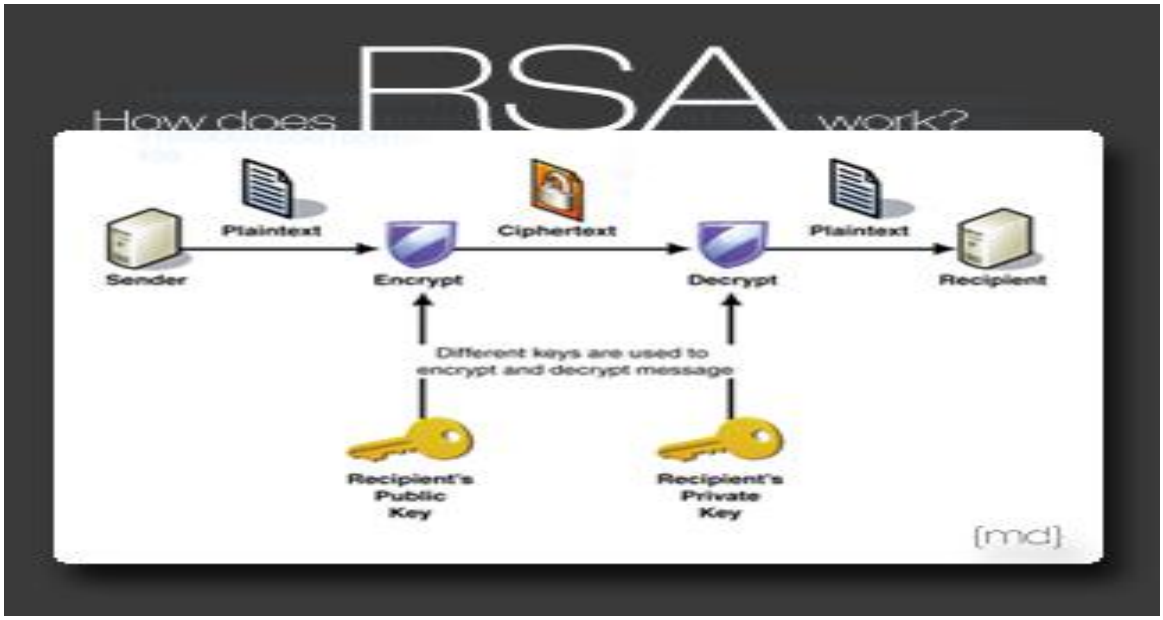
Por ejemplo, la última letra del texto cifrado es U, la vigésima primera letra del alfabeto, porque la letra correspondiente en el texto en claro es R (décimo octava letra) y la letra correspondiente en la clave es la C (la tercera letra).

3.4 Sistemas de cripto de claves públicas

¿En qué consiste?

La idea de los sistemas de cripto claves públicas es utilizar una “guía telefónica” de claves de cifra. La clave cifra de cada uno (**P**) es de dominio público: la clave de una persona puede figurar, por ejemplo, en la guía junto a su número de teléfono. Cada uno tiene también una clave secreta para descifrar: esta clave secreta (**S**) no la conoce nadie más.

Para transmitir un mensaje **M**, el emisor utiliza para cifrarlo la clave pública del receptor y luego lo transmite. El mensaje cifrado será **C=P(M)**. El receptor utiliza su clave privada para descifrar y leer el mensaje.



Condiciones del sistema

i. S(P(M))=M , para todo mensaje de M
ii. Todos los pares (S,P) son diferentes
iii. Obtener S a partir de P es tan difícil como leer M
iv. Tanto S como P son fáciles de calcular

La primera es una propiedad fundamental de la criptografía, la segunda y tercera son para dar seguridad y la cuarta hace que los sistemas sean factibles de utilizar.

SISTEMA DE CLAVE PÚBLICA (R. Rivest, A. Shamir, L. Adlemar)

- Es un sistema criptográfico que cumple con las condiciones de Diffie – Hellman (1976), pero no proponía método alguno que cumpliera estas condiciones.
- protocolo de establecimiento de claves entre partes que no han tenido contacto previo, utilizando un canal inseguro, y de manera anónima (no autentificada)
- Su seguridad se basa en la factorización de números compuestos como producto de primos.

Además, permite el intercambio de claves secretas y firmar matemáticamente

Criptosistema RSA

1. Cada usuario **i** elige dos números primos **p**, **q** lo suficientemente grandes que mantiene secretos.
2. La determinación de los números primos puede hacerse utilizando los tests de primalidad.
3. Se calcula: **n = p * q**, y **ϕ(n) = (p−1)(q−1)** donde ϕ es la función de Euler.
4. A continuación el usuario elige un entero ***e*** primo relativo con **ϕ(n)** tal que:
5. En la práctica se elige **e** primo directamente y mayores que p y q.
6. Calcular un entero d, tal que y $1 < d < \phi(n)$

$$d \equiv e^{-1} \bmod(\phi(n))$$

Luego, con lo anterior las claves serán:

- Pública:
  P (e, n), conocida por todos los usuarios.

- Privada:

  S (d), conocida sólo por quien desea desencriptar el mensaje

Obtención del criptograma y proceso de desencriptación

Para encriptar un texto plano M, se utiliza la función

$$C = M^{e} \pmod{n}$$

mientras que para desencriptar se utiliza la función

$$M = C^{d} \pmod{n}$$

Identificación de mensajes

Cada usuario posee un entero *n* tal que:

$$N^{k} < n < N^{l}$$

donde *N* representa el tamaño del alfabeto y *k, l* representan el número de letras del bloque de entrada y salida respectivamente.

Así, todo mensaje *M* se puede representar numéricamente de la siguiente forma:

$$M = m_1 N^{k-1} + m_2 N^{k-2} \ldots + m_{k-1} N + m_k$$

De la misma forma *C* puede considerarse como

$$C = c_1 N^{l-1} + c_2 N^{l-2} + \ldots + c_{l-1} N + c_l$$

Funcionamiento algoritmo RSA

1. Inicialmente es necesario generar aleatoriamente dos números primos grandes, a los que llamaremos p y q.
2. A continuación calcularemos n como producto de p y q: n = p * q
3. Se calcula fi: fi(n)=(p-1)(q-1)
4. Se calcula un número natural e de manera que MCD(e, fi(n))=1 , es decir e debe ser primo relativo de fi(n).

   Es lo mismo que buscar un número impar por el que dividir fi(n) que de cero como resto.
5. Mediante el algoritmo extendido de Euclides se calcula d: e.d mod fi(n)=1
   Puede calcularse d=((Y*fi(n))+1)/e para Y=1,2,3,… hasta encontrar un d entero.
6. El par de números (e,n) son la clave pública.
7. El par de números (d,n) son la clave privada.
8. Cifrado: La función de cifrado es C = M^e mod n
9. Descifrado: La función de descifrado es M = C^d mod n

Ejemplo con números pequeños

1. Escogemos dos números primos, por ejemplo p=3 y q=11.
2. n = 3 * 11 = 33
3. fi(n) = (3-1) * (11-1) = 20
4. Buscamos e: 20/1=0, 20/3=6.67. e=3
5. Calculamos d como el inverso multiplicativo módulo z de e, por ejemplo, sustituyendo Y por 1,2,3,… hasta que se obtenga un valor entero en la expresión: d = ((Y * fi(n)) + 1) / e = (Y * 20 + 1) / 3 = 21 / 3 = 7
6. e=3 y n=33 son la clave pública
7. d=7 y n=33 son la clave privada

8. Cifrado: Mensaje = 5, C = M^e mod n = 5^3 mod 33 = 26
9. Descifrado: M = C^d mod n = 26^7 mod 33 = 8031810176 mod 33 = 5

Sea Σ un alfabeto con N=27 letras, donde se ha identificado A=00, …, Z=26, []=27.

1. Sean p = 29 y q = 31, de ahí que n = 899
2. z = ϕ (n)=(p-1)(q-1)=840
3. Buscamos 1< e < 840 primo relativo con 840, sea e = 37.
4. Buscamos **d** tal que **e.d = 1 mod z**, esto es d = 613
5. Así tenemos, la clave pública *P*(37,899)
   la clave privada *S*(613)
6. $27^2 < 899 < 27^3$, de ahí que se va a encriptar bloques de dos letras en bloques de tres letras.
7. Sea *m* : **"CONGRESO"** el texto plano.
   Utilizando el alfabeto Σ se tiene la siguiente codificación

C	O	N	G	R	E	S	O
02	15	13	06	18	04	19	15

Los bloques a cifrar son:

(02,15) (13,06) (18,04) (19,15)

Expresemos cada bloque como un número en base N=27

$$(02,15) = 2\times27^0 + 15\times27^1 = 407$$
$$(13,06) = 13\times27^0 + 6\times27^1 = 175$$
$$(18,04) = 18\times27^0 + 4\times27^1 = 126$$
$$(19,15) = 19\times27^0 + 15\times27^1 = 424$$

Obtenemos el criptograma *C* con ayuda de la igualdad

$$C = M^e \pmod n$$

Esto es:

$$\begin{aligned} 407^{37} \bmod 899 &= 233 =: c_1 \\ 175^{37} \bmod 899 &= 204 =: c_2 \\ 126^{37} \bmod 899 &= 221 =: c_3 \\ 424^{37} \bmod 899 &= 259 =: c_4 \end{aligned}$$

Expresemos c_i en base 27, teniendo en cuenta que se van a tener tres componentes

$$\begin{aligned} 233 &= 17\times27^0 + 8\times27^1 + 0\times27^2 \Leftrightarrow (17,08,00) \Leftrightarrow (QIA) \\ 204 &= 15\times27^0 + 7\times27^1 + 0\times27^2 \Leftrightarrow (15,07,00) \Leftrightarrow (OHA) \\ 221 &= 5\times27^0 + 8\times27^1 + 0\times27^2 \Leftrightarrow (05,08,00) \Leftrightarrow (FIA) \\ 259 &= 16\times27^0 + 9\times27^1 + 0\times27^2 \Leftrightarrow (16,09,00) \Leftrightarrow (PJA) \end{aligned}$$

Luego el criptograma es QIAHOAFIAPJA

Para desencriptar el mensaje utilizamos la igualdad

$$M = C^d \pmod n$$

Haciendo el proceso inverso eligiendo bloques de tres letras se tiene que

$$\begin{aligned} 233^{613} \bmod 899 &= 407 =: m_1 \\ 204^{613} \bmod 899 &= 175 =: m_2 \\ 221^{613} \bmod 899 &= 126 =: m_3 \\ 259^{613} \bmod 899 &= 424 =: m_4 \end{aligned}$$

$$407 = 2 \times 27^0 + 15 \times 27^1 \Leftrightarrow (02,15) \Leftrightarrow (CO)$$
$$175 = 13 \times 27^0 + 6 \times 27^1 \Leftrightarrow (13,06) \Leftrightarrow (NG)$$
$$126 = 18 \times 27^0 + 4 \times 27^1 \Leftrightarrow (18,4) \Leftrightarrow (RE)$$
$$424 = 19 \times 27^0 + 15 \times 27^1 \Leftrightarrow (19,15) \Leftrightarrow (SO)$$

Obteniendo así el texto plano *m*:

"CONGRESO"

CAPITULO IV

Criptología II: Criptoanálisis

4.1 Criptoanálisis

4.1.1 Fundamentos

"El criptoanálisis es el arte de descodificar comunicaciones cifradas sin conocer las claves de las mismas"

4.1.2 Técnicas

Ataque a partir del cifrado:
Esta es la situación en la cual el atacante desconoce el contenido del mensaje y trabaja únicamente sobre el texto cifrado.

Ataque a partir del texto en claro:
El atacante conoce o puede conocer el texto en claro correspondiente o algunas partes del texto cifrado. El objetivo es descifrar el resto de los bloques del texto usando esta información, lo cual suele hacerse averiguando la clave usada para cifrar los datos.

Ataque a partir del texto en claro elegido:
El atacante puede elegir un texto en claro y obtener su cifrado correspondiente.

Ataque a partir de la clave:
El atacante intenta determinar la clave actual a partir de claves que conoce y han sido utilizadas en cifrados previos.

Suplantación de identidad:
El atacante asume la identidad de uno de los participantes en la comunicación.

Ataque mediante intromisión:

El atacante se introduce en medio de la línea de comunicación, de modo que mientras ambas partes piensan que están manteniendo una comunicación segura él está interceptando todo.

Ataques adaptativos basados en texto claro conocido:

En este caso, además de poder seleccionar varios textos en claro y obtener sus correspondientes textos cifrados, el criptoanalista puede modificar su elección de los mensajes a encriptar teniendo en cuenta los resultados generados por encriptaciones previas.

Búsqueda exhaustiva:

El atacante genera aleatoriamente todos los valores posibles de las claves de acceso y las transforma hasta que encontrar aquella que coincida con la previamente interceptada.

¿Qué ataques son excluidos?

El criptoanálisis suele excluir ataques que no tengan como objetivo primario los puntos débiles de la criptografía utilizada; por ejemplo:

Ataques a la seguridad que se basan en el soborno,

- La coerción física,
- El robo,
- El keylogging*,

Estos tipos de ataques son un riesgo creciente para la seguridad informática, y se están haciendo gradualmente más efectivos que el criptoanálisis tradicional.

4.1.3 Niveles de seguridad

Aspectos a considerar

Un buen diseñador no debe descuidar ningún aspecto de su sistema, pues la seguridad de un sistema criptográfico está directamente relacionada con su elemento más débil.

Dentro de esta seguridad podemos hablar de dos tipos:

- La de los operadores
- La de los protocolos

La de los operadores

Que se viola cuando se descubre la clave del sistema.

En función de la dificultad de conseguir esa clave tenemos distintos niveles de seguridad:

Incondicional.

Cuando el cifrado no aporta ninguna información acerca de la clave,

Computacional

Cuando no existe capacidad suficiente de cálculo para obtener la clave,

Probable

Cuando no han sido violados a pesar de no estar basados en principios matemáticos de seguridad demostrable

Condicional

Cuando la dificultad de vulneración es mucho mayor que la capacidad de cálculo del posible atacante.

La de los protocolos

Que provoca vulneraciones del sistema debidas a posibles debilidades del mismo, sin necesidad de conocer su clave.

4.1.4 Técnicas de criptoanálisis

Criptoanálisis diferencial

Trata de encontrar correlaciones entre el texto claro y el texto cifrado obtenido a la salida del sistema criptográfico, partiendo del conocimiento de la existencia de ciertas diferencias entre varios textos claros que se han introducido en el sistema.

Técnicas de análisis estadístico de frecuencias

Estudia apariciones de algunas letras, diagramas, trigramas o de determinadas palabras (las más frecuentes en ese idioma), para determinar cuál ha sido el sistema de sustitución empleado en el cifrado, con la inestimable ayuda hoy en día de un equipo informático.

Interceptación de claves

Ataques de intermediación "man-in-the-middle", mediante el cual se pueden interceptar directamente las claves sin despertar sospechas de los usuarios del sistema criptográfico y sin que sea necesario estudiar los textos cifrados.

4.2 Esteganografía

4.2.1 Fundamentos

Esteganografía: escritura secreta mediante la ocultación del mensaje

steganos $\rightarrow$ encubierto grafo $\rightarrow$ escritura

Ejemplos de esteganografía

1. Herodoto: guerra entre Grecia y Prusia, siglo:

    - Ocultan en tablas de madera
        - Retiran la cera de un par de tablillas de madera.
        - Escriben el mensaje en la madera.
        - Cubren el mensaje con cera.
        -
    - Afeitan la cabeza del mensajero
        - Afeitan y tatúan el mensaje en la cabeza del mensajero
        - Esperan a que crezca el pelo
        - Envían al mensajero.

2. Eneas el Estratega (siglo V AC): comunica mensajes pinchando agujeros diminutos bajo las letras de un texto aparentemente inocuo

3.- China: escribían los mensajes sobre seda, se cubrían con cera y se hacían una pequeña bola que tragaba el mensajero.

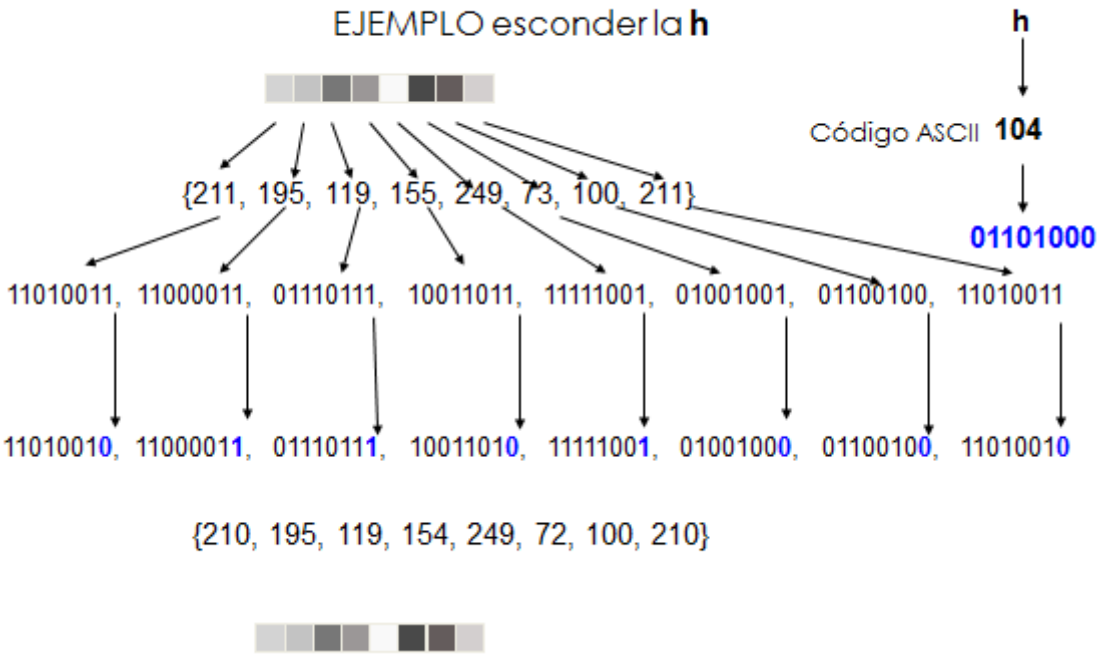
4.- Giovanni Porta (s XV) esconde un mensaje en un huevo cocido. Pinta sobre la cáscara, al cocer el huevo la tinta penetra por ella y al pelar el huevo duro se ve el mensaje.

5.- Escribir con tinta invisible

4.2.2 Esteganografía en una imagen

- La imagen se guarda en una matriz de 356 filas y 291 columnas.
- Cada elemento de la matriz es un pixel (un tono de gris).
- Hay 256 tonos de gris. Se guardan como números de 0 a 255.
- Trabajando con 8 bits:
  00000000 corresponde al negro

11111111 corresponde al blanco



En cada pixel de la imagen se cambia el bit menos significativo por el correspondiente del mensaje ¿por qué?

- {238, 238, 238, 238, 238, 238, 238, 238, 238, 238, 238, 238, 238, 238, 238, 236, 236, 237, 238, 238, 239, 240, 240, 240}

- {11101110, 11101110, 11101110, 11101110, 11101110, 11101110, 11101110, 11101110, 11101110, 11101110, 11101110, 11101110, 11101110, 11101110, 11101111, 11101100, 11101100, 11101101, 11101111, 11101111,11101111, 11110000, 11110000, 11110001}

- Para enviar el mensaje hoy → {01101000, 01101111, 01111001}

- Cambio el bit menos significativo de cada píxel
  {11101110, 11101111, 11101111, 11101110, 11101111, 11101110, 11101110, 11101110, 11101110, 11101111, 11101111, 11101110, 11101111, 11101111, 11101111, 11101101, 11101100, 11101101, 11101111, 11101111,11101111, 11110000, 11110000, 11110001}

- En la imagen sustituyo los 24 bits por estos nuevos

  {238, 239, 239, 238, 239, 238, 238, 238, 238, 239, 239, 238, 239, 239, 239, 239, 236, 237, 239, 239, 239, 240, 240, 241}

REFERENCIAS BIBLIOGRAFICAS

1. Ormen, Thomas H. “Introduction to Algorithms”. Ed. Mc Graw-Hill. 2001. 2da. Edición. EEUU.

2. William Stalings. “Cryptography and Network Security: Principles and Practice”. Ed. Prentice Hall. 2002. 3ra. Edición. EEUU.

3. Wang Paul S. “Java: con Programación Orientada a Objetos y Aplicaciones en la WWW”. Ed. Internacional Thomson. 2000. España.

4. Stuart McClure, Joel Sambay and George Kurtz. “Hacking Exposed: Network Security Secrets and Solution”. Ed. Osborne Mc Graw-Hill. 3ra. Edición. 2002. EEUU.

5. Sergio M. Fernandez S. “Fundamentos del diseño y la programación orientado a objetos”. Ed. Mc Graw Hill. 1995. España.