Tema 2: Introducción a la Criptografía.

- Se hace uso de una clave y un algoritmo de cifrado. Si la misma clave sirve para cifrar y descifrar es un cifrado simétrico.
- **Criptografía:** Disciplina que estudia los principios, métodos y medio de transformar los datos para ocultar su significado, garantizando su integridad(que no hayan sido modificados), su autenticidad(asegurarnos quien ha sido el autor) y prevenir su repudio(que no se pueda negar la autoría.
  - Elementos básicos:
    - **Espacio de mensajes:** Mensajes en claro, que no han sigo transformados. M={m1,m2,...}
    - **Espacio de cifrados:** Mensajes que ya han sido transformados. C={c1,c2,...}.
    - **Espacio de claves**: Sirve para configurar el algoritmo de cifrado/descifrado. K={k1,k1,…}
  - Las transformaciones pueden ser:
    - **Cifrado M->C:** Entra un mensaje en claro, mediante un cifrador y una clave, pasa a ser un mensaje cifrado.
    - **Descifrado C->M**: Proceso inverso al cifrado, pasa de cifrado a en claro.
  - Características de los sistemas criptográficos:
    - Tipo de operaciones realizadas: Generalmente, sustitución y transposición, no puede perderse información.
    - **Numero de claves usadas:**
      - **Simétricos**: Hay una sola clave, tanto para cifrar como para descifrar.
      - **Asimétricos**: Se cifra con una clave publica, que es conocida, y se descifra con una clave privada, no conocida.
    - **Tipo de procesamiento del texto en claro:**
      - **Por bloques**: Va cifrando trozos de cadena.
      - **Flujo continuo:** Se hace sobre pequeños trozos como bytes o bits.
  - Codificador vs Cifrador:
    - **Codificador**: Se sigue una función conocida, que siempre es igual. Morse, ASCII,…
    - **Cifrador**: Se sigue un proceso que no es conocido por todos y que varia, y es necesaria una clave.
- **Criptoanálisis**: Ciencia que trata de frustrar las técnicas criptográficas, trata de obtener la clave de cifrado y si no es posible al menos descifrar el mensaje.
  - **Principio de Kerchoff:** La seguridad del cifrado debe residir en el secreto de la clave.
  - No debe ser por falta de claridad que es segura.
  - Los ataques se basan en el conocimiento sobre el algoritmo y en posible información adicional.
  - Métodos:
    - **Ataques al algoritmo:** Se basa en el conocimiento del algoritmo, y desarrolla métodos en función de como funciona. Además del algoritmo el atacante puede conocer el texto descifrado de otro dado cifrado con la misma clave, textos escogidos por el atacante pero cifrados. Conoce algoritmo y/o algún ejemplo de texto descifrado con su correspondiente cifrado. Cuanta menos información más difícil será de resolver.
    - **Fuerza bruta:** Consiste en probar todas las claves posibles, en media se probaran la mitad de las posibilidades para tener éxito. Este proceso requiere tiempo y aumenta exponencialmente al aumentar el tamaño de la clave. Lo primero que se prueban son combinaciones de palabra del diccionario, cuando esto tenga sentido.
  - **Algoritmo de cifrado incondicionalmente seguro:** No se filtra información por muy largo que sea el texto cifrado. Solo lo cumple el cifrador Vernam, y en ciertas condiciones(clave realmente aleatoria, usarla una sola vez y de longitud igual o superior al texto a cifrar). Pero no es practico, es muy difícil conseguir claves con esa condición.

- **Algoritmo de cifrado matemáticamente vulnerable:** Al aumentar la longitud del texto cifrado se filtra información, es lo que le ocurre al resto de algoritmos que no son Vernam.
- **Seguridad computacional:** Se requiere tantas operaciones que el tiempo en realizarlas es mayor que el tiempo útil de la información. Por lo que para cuando se ha resuelto no es útil ya. También puede ser que el coste de resolverlo sea superior al de la propia información.
- **Los cifradores simétricos** solo se pueden resolver mediante un ataque de fuerza bruta, no existe un algoritmo.

◦ **Teoría de la información:** Establece una medida para evaluar el secreto de un cifrador, que se basa en la incertidumbre que tiene el criptoanalista sobre el texto en claro al interpretar su cifrado.

- Sean m1, m2,… mensajes independientes con probabilidad de ocurrencia (de que se repita) p(m1), p(m2),…
- La **cantidad de información (ci)** de un mensaje mi es: $C_i = -\log_2 p(m_i)$ bits Un mensaje que se repite, alta probabilidad de ocurrencia aporta menos información que un mensaje único . Mayor p(mi), menor ci
- **Entropía**: Es el promedio de información transportada por un mensaje. Y la previsible tras la aparición de un mi. Mayor entropia, mayor incertidumbre.
  - Sea la fuente de mensajes M: $H(M) = -\sum_i p(m_i) \cdot \log_2 p(m_i)$ bits
  - H(M)=0 cuando p(mi)=1, ya que no aporta información si siempre aparece.
  - H(M)=máxima=log2 n cuando p(mi)=1/n.
- **Entropía condicionada:** Cuando existe alguna relación entre las apariciones de dos mensajes consecutivos de distinta fuente, la presencia del primero disminuye la incertidumbre del segundo. Conocer el primero me facilita encontrar el segundo. Los métodos tratan de maximizar H(M|N) para poder relacionar el conjunto de textos claros M y el de textos cifrados N.
  - $H(M/N) = -\sum_j p(n_j) \cdot \sum_i p(m_i/n_j) \log_2 p(m_i/n_j)$

◦ **Aleatoriedad:** Sea S un espacio muestral con distribución de probabilidad P, cada posible valor que X puede tomar en S tiene asociada una determinada probabilidad.

- **Usos**:
  - Distribución de claves.
  - Generación de claves de sesión y claves para RSA.
  - Generación de flujos de bits para algoritmos de cifrado simétrico de flujo.
- **Criterios**:
  - **Distribución uniforme**: La frecuencia de aparición de cada símbolo debe ser aproximadamente la misma(tantos 0's como 1's)
  - **Independencia**: Ninguna su secuencia puede ser inferida de otra.
- Se pueden hacer pruebas que comprueben que se cumple la uniformidad, pero no hay pruebas que comprueben la independencia, pero si que no es independiente.
- **Pseudoaleatoriedad(PRNG):** Pasan las pruebas de aleatoriedad, pero tienen un mecanismo determinista, algoritmo, que genera los números. Aunque un algoritmo no es verdaderamente aleatorio.
- **Aleatoriedad(TRNG):** Mecanismo no determinista, ciertos procesos naturales y eliminación del sesgo con funciones resumen.

◦ **Complejidad algorítmica:** Estudia los algoritmos bajo la dificultad de su resolución. Clasifica los algoritmos según su complejidad. Turing demostró que no todos los problemas tiene un algoritmo que lo resuelva.

- **Clasificación**:
  - **Indecidibles**: No hay algoritmo que lo resuelva.
  - **Decidible**: Existe al menos un algoritmo que lo resuelve.
  - **Intratable**: No se puede resolver en un tiempo razonable.
  - **Tratable**: Resuelve cualquier problema particular en un tiempo razonable.

- La complejidad se mide en **tiempo (t)** según el tamaño de la **entrada (n)**. Puede ser:
  - **Polinomica**: El tiempo es de orden polinómico o menor.
    - **Lineal**: O(n)
    - **Logarítmico**: O(log n)
    - **Polinómico**: O(n^c)
  - **Exponencial**: El tiempo t es de orden mayor que polinomio.
    - **Exponencial**: O(c^x)
    - **Factorial**: O(n!)
- Los problemas se clasifican en:
  - **Clase P**: Problemas decidibles, tratables, algoritmo determinista(cada paso se resuelve de manera única, una opción solo) y polinómico (buenos algoritmos).
  - **Clase NP**: Problemas tratables e intratables, algoritmo no polinómico (malos) y no
  - determinista, en cada paso hay que seleccionar una opción.

2.2 Métodos Criptograficos clásicos:

- **Transposición**: Consiste en alterar el orden de los caracteres, pero sin modificarlos.
  - **De riel**: Se hacen 2 columnas, un carácter va una columna y el siguiente a la otra. Después se concatenan las dos columnas, la segunda se añade al final de la primera.
  - **Por grupos, permutación**: Consiste en ir alterando el orden de los caracteres en grupos de caracteres. Se divide la cadena en grupos y se lleva a cabo el proceso. Indicando en que posición irá cada uno.
  - **Por series**: Ordenar mensajes como cadena de submensajes. Se hacen funciones o series que se concatenan y cada una coge ciertos caracteres (primos y no primos), y las suma de todas recoge todos los caracteres.
  - **Por columnas/filas**: Introducir la cadena original en una estructura o patrón, e ir alterando el orden de la filas o columnas. La cadena codificada será la originada de ese desorden de la estructura.
- **Sustitución**: Consiste en intercambiar los caracteres de la cadena original por otros de un alfabeto codificado.
  - **Monoalfabeto monográfica**(**simple)**: Ir sustituyendo cada carácter por su equivalente cifrado. El equivalente consiste en el numero de la letra multiplicada por un valor, sumándole otro y todo eso modulo del numero de letras del alfabeto. Se usa el modulo de ese valor para conocer que letra corresponde en la codificación. E(m)=(am+b) mod n, la a es la constante de decimación, la b la constante de desplazamiento (a y b forman la clave y deben ser coprimos) y n el numero de letras del alfabeto. Para descifrar consiste en invertir el proceso, se despeja la expresión modular. E(m)=7m+3 mod 27=n D(n)=(n-3)7^(-1) mod 27
    - Cifrador por desplazamiento puro, E(m)=(m+b) mod n
      - Cifrador Cesar, E(m)=(m+3) mod n
    - Cifrador por decimación pura, E(m)=(am) mod n
    - Cifrador por sustitución afín, E(m)=(am+b) mod n
  - **Monoalfabeto poligrafica**: Sustitución n (n>=2) caracteres texto-claro por n caracteres texto-cifrado.
    - **Playfair**: Se hace una matriz que comienza con las letras de una clave, sin que se repitan letras y el resto de la matriz se rellena con el resto de las letras en orden. Para todas la letras se usa una matriz 5x5, algunas como la **ñ y j** están en casilla con otra. Se separa la cadena en grupos de caracteres (2) y se consulta la posición de las letras en la matriz y según unos criterios(derecha y abajo, si están en la misma fila, se coge la inmediata a la derecha de cada una, si están en la misma columna es el inmediato de abajo, y si están en diagonal se hace espejo, perpendicular a la flecha que los conecta) se eligen como caracteres codificados unos desplazados en la matriz. Para descodificar se necesita la misma matriz, la cadena codificada y los criterios se hacen a la inversa, izquierda y arriba.

        - **Hill:** Dividimos la cadena en vectores de tamaño n, es capaz de cifrar “n”caracteres a un tiempo. La clave es una matriz nxn que multiplicaremos por los vectores formados por los grupos de la cadena, lo realizamos para toda la cadena y la cadena cifrada es el resultado de concatenar todos. Para descifrar hallamos el inverso de la matriz((adj(nxn)^t/det(nxn)) y lo multiplicamos por los vectores cifrados. Los valores deben estar en mod numero de letras.
    - **Polialfabeto periodica: Vigenére**
        - Hay 27 alfabetos, uno para cada letra, que comienza desde esa letra. Ejem: Alfabeto de C E(mj) = (mj+k^sub(j mód m)) mód. 27
        - **Con clave:** Lo que equivale a colocar bajo la cadena original la clave tantas veces como para cubrirla entera. Y lo que se hace es buscar en el alfabeto de la letra de la clave(fila) la posición de la letra de la cadena original(parte alta de la tabla). Para descifrar colocamos también la clave bajo la cadena codificada y mirando la fila de la letra de la clave buscamos dentro de la misma la letra de la cadena codificada y la letra correspondiente en la cabecera será el carácter descodificado.
        - **Con autoclave:** Consiste en colocar una vez la clave bajo la cadena original y a continuación la cadena a codificar también como clave, hasta que cada letra de la cadena original le corresponda uno de clave. Para codificar se miran la fila de la letra de la clave y buscamos en la cabecera de las columnas el carácter que le corresponde de la original, así hasta completar la cadena. Para descodificar colocamos la clave una vez, y comenzamos a descodificar los caracteres de la cadena codificada que tienen otro de la clave, para ello mirando la fila de la clave buscamos dentro de la fila la letra codificada y el valor que le corresponda en la cabecera será el descodificado, y cuando necesitamos más caracteres de clave vamos poniendo la cadena descodificada.
    - **Polialfabeto no periódica: Vernam**
        - La longitud de la clave es igual o mayor que la longitud del texto sin codificar, los valores son aleatorios, lo que la hace perfecta. Los problemas son el tamaño de la clave y que no se puede reutilizar. XOR del valor de cada letra de la cadena con su correspondiente en la clave, y si el valor supera el numero posible de caracteres, hacemos mod numCaracteres.
        - Para codificar: ci= E(mi) = (mi XOR ki)
        - Para descodificar: mi= E(ci) = (ci XOR ki)
    - **Máquina Enigma:**
        - Cifrado/descifrado rotatorio.
        - Funciona con rotores que leen una placa en la que están creadas las conexiones entre letras...
- **Criptoanálisis de Criptografía clásica:**
    - **Romper el Cifrado de Desplazamiento:** Consiste en fuerza bruta, ir probando todos los posibles valores de la constante de desplazamiento(n) que pueden ir desde 0 hasta el tamañoAlfabeto-1(ya que si supera ese tamaño al hacer mod equivale a uno menor)
        - Codificar: E(x)=(x+n) mod m. Descodificar: D(x)=(x-n) mod m
    - **Romper el Cifrado de Sustitución Monoalfabeto:** Para no hacerlo por pura fuerza bruta, se analiza la frecuencia de aparición de ciertos caracteres y vamos relacionándolos con los caracteres mas frecuentes de nuestro alfabeto, ya que la frecuencia se transmite al codificar. Suponiendo dos parejas en los que uno de los mas frecuentes codificado es otro de los mas frecuentes propios del alfabeto, creamos un sistema de ecuaciones con sus valores y buscamos resolverlo y probar si tienen sentido ese resultado. Y mediante la prueba y error de los mas frecuentes hallaremos los valores de las constantes.
        - E(m)=(am+b) mod n
        - Algunas técnicas para dificultarlo son: Cadenas suficientemente cortas, evitar emplear las letras mas frecuentes o evitar terminaciones que se repiten con frecuencia en las palabras.

- **Romper el Cifrado de Sustitución Polialfabeto:** Método de Kasiski e Índice de Coincidencia. Consiste en ir calculando el IC de subcadenas de la cifrada, estas subcadenas se originan de ir saltándose x letras, pero crear tantas cadenas como se pueda con ese tipo de salto. Aquellas subcadenas que tengan un IC similar al de un lenguaje indicaran el tamaño de la clave de Vegenére. Conociendo el tamaño de la clave aplicamos Kasiski (este obtendría el tamaño de la clave observa ando repeticiones en la cadena y midiendo su separación y el tamaño seria el mcd de esas separaciones), que consiste en dividir la cadena en tantos grupos como tamaño tiene la clave, así tenemos agrupadas aquellas letras que perteneces al mismo alfabeto. Dentro de esos grupos observamos la frecuencia de aparición de cada carácter y ~~haremos combinaciones~~ cogiendo un carácter de cada grupo y probaremos si esa es la clave, lo hacemos descifrando normal mirando la fila de la letra de la clave y dentro de ella el caracter codificado.

Cogemos las n letras + frecuentes del lenguaje y hallamos la separación entre ellas. Después con las + frecuentes del criptograma miramos las que mejor encajan con esas separaciones, y esas son la clave. Ejem: A, F, O → 0 +4 +15

A 4 E 15 O
B 5 F 15 P
E 15 S
R 4 V 15 G

Indice de Coincidencia: Medida estadística sobre un texto. Es la probabilidad de que dos letras seleccionadas aleatoriamente en un texto sean la misma.

- Se calcula con:

Cálculo del IC

$$\frac{\sum(f_i * (f_i - 1))}{N(N-1)}$$

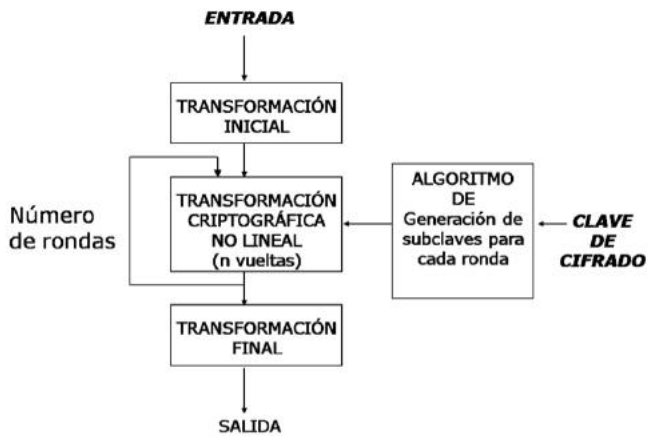
$$\frac{\sum \text{vecesAparece} \cdot (\text{vecesAparece} - 1)}{\text{numletras}(\text{numletras} - 1)}$$

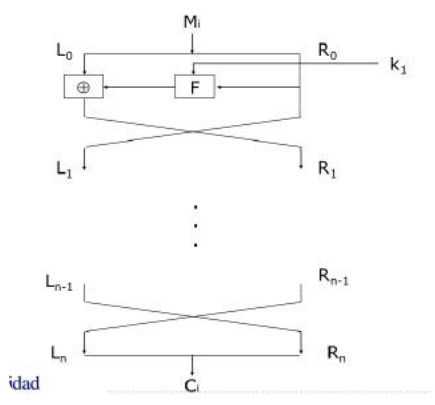
para $0 \geq i \geq 26$,

f_i es la frecuencia (nº de apariciones) de la letra i-ésima del alfabeto, en el texto analizado

y N es el nº de letras del texto analizado

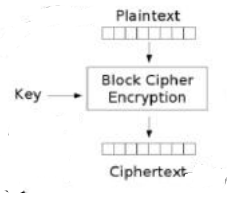
- **2.3 Criptosistemas simétricos:** El mensaje se descompone en bloques de símbolos de igual longitud, y todos los bloques se cifran con la misma clave, y el proceso inverso descifra.
  - Clasificación:
    - **Tipo de operaciones realizadas:** En general, sustitución y permutación. No puede perderse información.
    - **Número de claves usadas**:
      - **Simétricos**: Con una clave, que es secreta y sirve para cifrar y descifrar.
      - **Asimétrico**: Con dos claves, una publica para cifrar que es conocida y otra secreta que no es conocida para descifrar. Para transmitir la clave privada se cifra simétricamente, cuando se ha descifrado la simétrica ya tenemos la clave secreta para descifrar el mensaje publico.
    - **Tipo de procesamiento:** Por bloques o Flujo continuo.
  - **Esquema de Feistel:** Es un cifrador de bloque, con 2^n! posibles claves, siendo n el tamaño del bloque de texto y k el tamaño de la clave, k<=n.
    Hay alta difusión, cada valor de la cadena cifrada depende de muchos de la cadena original, se consigue permutando, cifrado y permutando de nuevo y una alta confusión, busca complicar la relación estadística entre C y k, se logra con sustituciones complejas.



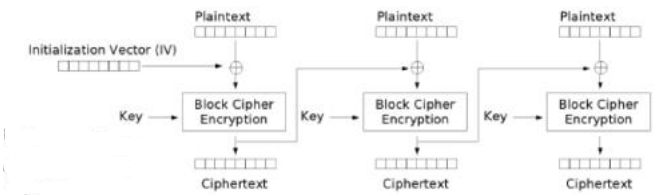


- La cadena tras la transformación inicial, se parte en dos, la parte derecha pasa a ser la parte izquierda. La parte izquierda tras hacer un XOR con la parte derecha tras pasar por una función de ronda F, se pone como parte derecha. Y este proceso se repite una serie de veces, para dar mas seguridad. Se cifra y descifra de la misma manera, pero cambiando el orden de las subclaves de la función de ronda. Lo mas importante de este método es un buen método de expansión de claves y una buena función de ronda.
- Tamaño bloque 64, clave 128, numero de rondas 16. Cuanto mayor sea el tamaño mas seguridad, pero también será mas lento

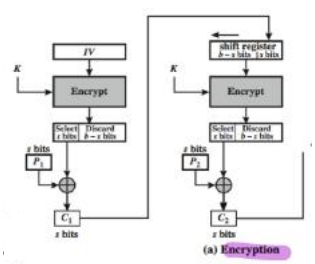
- **Modos de operación:** Técnicas para mejorar el efecto de un algoritmo criptográfico, se puede aplicar para cualquier cifrador de bloque.
  - **Electric CodeBook (ECB):** El mismo bloque a la entrada genera el mismo bloque a la salida. Consiste en que cada bloque del texto en claro pase por el cifrador de bloque, de esta manera obtenemos el bloque cifrado.



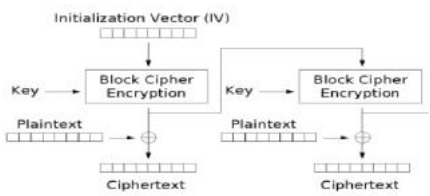
    - **Ventajas**: El cifrado y descifrado se puede hacer en paralelo, los errores no se propagan y transmisión segura de un bloque.
    - **Desventajas**: Los bloques repetidos dan como resultado criptogramas repetidos, es posible alterar el orden/modificar/repetir/eliminar los bloques y necesita relleno el ultimo bloque, este indica cierta información sobre el mensaje.
  - **Cypher Block Chaining(CBC):** Se realiza en cadena, cada cifrador de bloque recibe el texto claro XOR vector, y da como salida el bloque cifrado, que también será el vector del siguiente bloque. El primer vector es dado, el IV. ~ bloque anterior cifrado
    - Bloque cifrado=Cifrar con clave K(Bloque en claro XOR Vector)
    - Bloque descifrado= Descifrar(Bloque cifrado m) XOR Bloque cifrado m-1
    - Requiere relleno y un error se propaga en dos bloques.



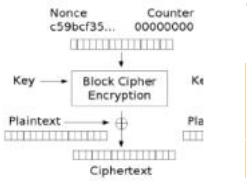
  - **Cypher FeedBack(CFB):** Usa un registro de desplazamiento y opera sobre segmentos mas pequeño que el bloque, por lo que se puede pasar de cifrado en bloque a flujo. Un error en el cifrado se propaga dos bloques.
    - El vector(texto cifrado anterior o inicial) pasa por el cifrador de bloque, de su resultado se escogen los n primeros bits y se hace un XOR con el bloque de texto en claro de tamaño n, eso da lugar a ese bloque cifrado y al vector del siguiente bloque que desplazará hacia la izquierda para hacer le hueco.



  - **Outer FeedBack(OFB):** El vector inicial es un nonce(valor aleatorio no utilizado antes), un error en el cifrado afecta a un solo bloque y se descartan los bits sobrantes para el ultimo bloque. Se puede pasar de cifrado en bloque a uno de flujo.
    - El vector(el primero el inicial y después el vector anterior cifrado), se pasa por el bloque de cifrado y se bifurca, por un lado será el vector del siguiente bloque, y por el otro se hace XOR con el texto en claro y es el bloque cifrado.



- **Counter(CTR):** Utiliza un contador del tamaño de bloque(n), se inicializa con un nonce y se le suma 1 mod 2^n en cada bloque, se descartan lo bits sobrantes. Se puede pasar de cifrado en bloque a uno de flujo. Es muy usado por su simplicidad.

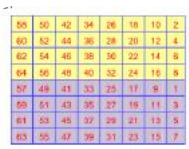


  - El nonce + contador se pasan por el cifrador de bloque, la salida XOR bloque en claro da lugar al bloque cifrado.

- **Cifradores de bloque: Ventajas y desventajas:**
  - **Ventajas:** Alta difusión y confusión en el criptograma, y fácil implementación. Son simétricos, el cifrado y descifrado son casi idénticos, y son eficientes, muy rápidos.
  - **Desventajas**: El canal para distribuir las claves, el gran numero de claves y que un error se propaga a otros bloques. Vulnerable a ataques si se repiten bloques.
- **Data Encryption Standard (DES):** Clave de 64 bits, bloques de 64 y se hacen 16 rondas.
  - Se hace una permutación inicial, se hacen 16 rondas y por ultimo la permutación final.
    - Los bloques de texto en claro se presentan como matriz de 8x8 números escritos por filas.

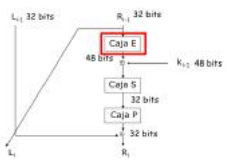
1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

    - La **permutación inicial** separa los que ocupan posiciones pares de los que ocupan una impar, en dos bloques, y se escriben de derecha a izquierda y de arriba a abajo. El bloque de los pares es el bloque izquierdo y el otro el derecho.



L_0 = 58 50 42 34 26 18 10 02 60 52 44 36 28 20 12 04 62 54 46 38 30 22 14 06 64 56 48 40 32 24 16 08

R_0 = 57 49 41 33 25 17 09 01 59 51 43 35 27 19 11 03 61 53 45 37 29 21 13 05 63 55 47 39 31 23 15 07



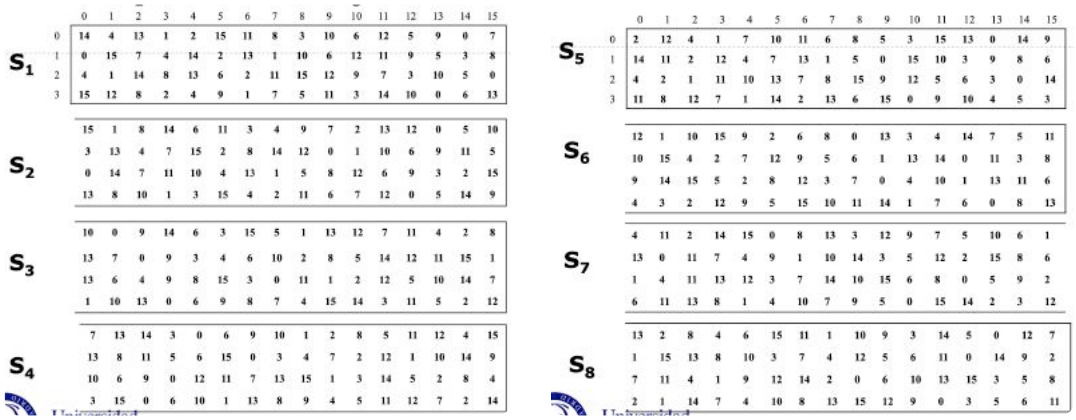
    - En cada **ronda** se hace Feistel, el bloque derecho pasa a ser el bloque izquierdo, pero el bloque izquierdo se hace XOR con el resultado de que el bloque derecho recorra: Caja de Expansión, XOR con una clave, Caja de Sustitución y Caja de permutación. El proceso se repite 16 veces,
      - **Caja de expansión**, el bloque pasa de 32 a 48 bits, se añade uno a cada lado de las filas de 4 bits. El de la derecha es el ultimo de la fila anterior, y el de la izquierda es el primero de la siguiente fila.

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

      - **XOR** con una clave de 48 bits.
      - **Caja de Sustitución**, pasa de 48 a 32 bits, cada fila se pasa por su propia tabla(hay 8 tablas diferentes), el primer y ultimo bit indican la fila en la tabla, y el resto indica la columna de la tabla. El valor que corresponda se escribe con 4 bits sustituyendo al anterior que ocupa 6 bits.

S_1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

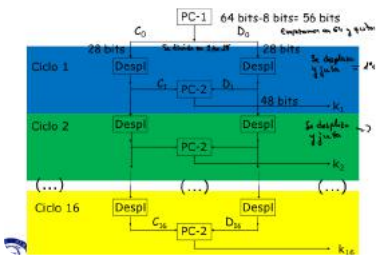


- **Caja de Permutación**, según un tabla cambiamos las posiciones de los elementos del bloque.

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

- Se hace **XOR con el bloque izquierdo** y pasa a ser el bloque derecho.

◦ El **generado de claves del segundo paso**, genera 16 claves, una para cada ronda:

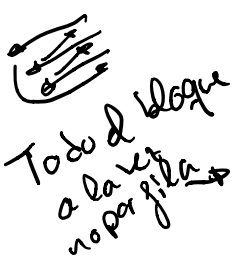


- Se parte de una clave de 64 bits de la que se quitan 8.
- Con los 56 bits hacemos una permutación y los dividimos en dos bloques de 28 bits.

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Arriba izq y abajo derecha

Ciclo nº	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits a desplazar	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1



- Cada bloque de 28 bits de desplaza 1 o 2 posiciones a la izquierda, según el ciclo en el que se esté. Para hallar la propia solución de juntaran, pero para hallar una de las siguientes claves seguimos haciendo desplazamientos.

- Se juntan ambos bloques según un el orden determinado por la tabla.

◦ Para descifrar se sigue el proceso inverso, y las claves también en orden inverso, en vez de desplazamiento a la izquierda es a la derecha.

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

- **Cifrado múltiple**, para DES sí tiene sentido realizar cifrado múltiple, el sistema no s un grupo. Por los un aumenta la seguridad.
  - **3 DES con 2 claves**: La primera vez con la clave 1, el segundo con la segunda y el tercero de nuevo con la primera. Este es suficiente seguro.
  - **3 DES con 3 claves**: Una clave para cada ciclo de cifrado, apena se usa ya que con 2 claves es suficiente.

- **Ataques al DES:**
  - **Por fuerza bruta**: se puede romper en menos de un día con el HW adecuado.
  - **Con criptografía diferencial**, se necesitan 2^{49} textos en claro escogidos y sus correspondientes cifrados.
  - **Con criptografía lineal**, se necesitan 2^{49} textos en claro conocidos y sus correspondientes cifrados.
- **Ataques al Triple DES:** "Meet-in-the-middle"(meterse en el proceso y alterar las claves desde dentro) y los ataque con textos conocidos y escogidos.

- **Advanced Encryption Standart (AES):** Se utiliza para comunicaciones gubernamentales, transferencia de fondos bancarios y comunícanos civiles.
  - Criptografía simétrica, es un cifrador de bloques de 16 bytes.
  - Hay tres longitudes de clave: 128, 192 y 256 bits.
  - Red de sustitución-permutación, pero no es esquema de Feistel, el bloque no se divide.
  - Es rápido tanto en software como en hardware, se basa en 4 funciones invertibles, que para descifrar se hacen su inverso.
    - El bloque de texto y la clave están compuestos por 16 bytes, que se colocan de arriba a abajo y de izquierda a derecha.
    - **Proceso de cifrado:**
      - **Ronda inicial:** AddRoundKey.
      - **Ronda:** ByteSub, ShiftRow, Mixcolumns y AddRoundKey. Se repite este paso.
      - **Ronda final:** ByteSub, ShiftRow y AddRoundKey.
    - **Proceso de descifrado:** Se recorre el esquema de abajo de arriba, haciendo las funciones inversas.

Algoritmo RIJNDAEL

```
Rijndael(State, Key) {
    KeyExpansion( Key, ExpandedKey );
    AddRoundKey( State, ExpandedKey );
    for (i=1; i<10; i++)
            Round(State, ExpandedKey+4Xi);
    FinalRound(State,ExpandedKey+4X10);
}
```

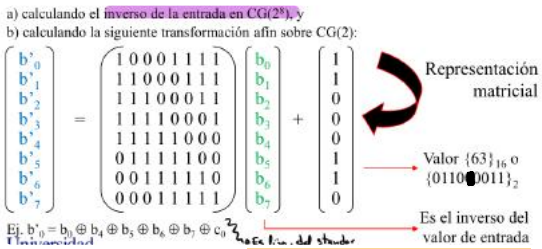
```
Round(State, RoundKey) {
    ByteSub(State);
    ShiftRow(State);
    MixColumn(State);
    AddRoundKey(State, RoundKey);
}
```

```
State          -- array de 4 words (de 32 bits)
No. of Rounds  -- 10 rondas para la combinación de 128-128 bits
               -- XOR de las keywords (de 32 bits),
                  S-box lookups, rotación de bytes intra-word
AddRoundKey    -- bitwise-XOR con las keywords
FinalRound     -- similar a Round pero sin MixColumn
```

- Como la unidad es el byte, se opera en Cuerpos de Galois GF(2^8) para la suma y multiplicaciones. Para la reducción se hace con el polinomio $p(x)=x^8+x^4+x^3+x+1$.

Combinaciones posibles de estados en AES	Longitud del bloque (Nb palabras)	Longitud de la clave (Nk palabras)	Número de Rondas (Nr)
AES – 128	4	4	10
AES – 192	4	6	12
AES – 256	4	8	14

- **Las transformaciones:**
  - **ByteSub**(sustitución de un byte):
    - Consiste en hallar el inverso de cada byte y multiplicarlo por una matriz fija y el 63_{16} (representados como 1's y 0's). Las sumas y multiplicaciones sobre el cuerpo de Galois.

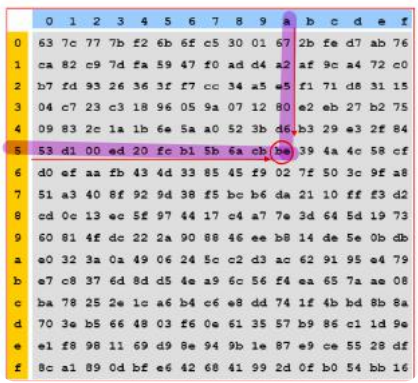


El inverso con Euclides

$x^3 + x = x^2 \cdot \_ + \_$

⋮ = ⋮

    - Otro método es consultar una tabla en la que ya se ha realizado el proceso y coger el correspondiente, los primeros 4 bits fila y los otros 4 columna, o en hexadecimal. Ej: 5a



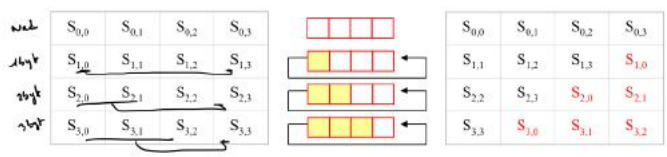
	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

$5a = 01011010 = x^6 + x^4 + x^3 + x$

$\text{inv}(5A) = 22 = 00100010$

$$\begin{pmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

  - **ShiftRow**(desplazamiento de filas): Desplazar bloques de un byte hacia la izquierda modulo columna, comenzado por 0 cada vez fila desplaza uno mas.
    - Primera fila 0, la segunda 1, la tercera 2 y la cuarta desplaza 3 bytes.



  - **MixColumn**(mezcla de datos dentro de cada columna): Opera sobre columnas, que se consideran como polinomios de GF(2^8), se multiplica cada columna con una matriz fija de números en hexadecimal. Se ponen todos en binario y se opera en Galois, teniendo como modulo la $p(x)=x^8+x^4+x^3+x+1$. Se hace para todas la columnas.

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$$

Recuerde que $\{03\} = x + 1$, $\{02\} = x$, $\{01\} = 1$.

Representación matricial de la función MixColumns

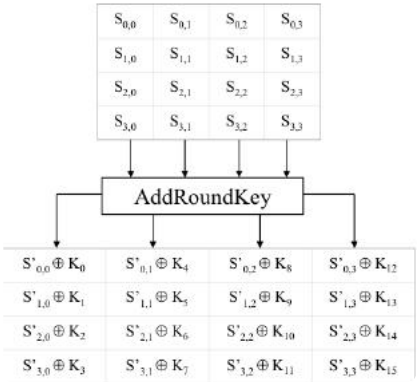
$$\begin{pmatrix} S'_{0,C} \\ S'_{1,C} \\ S'_{2,C} \\ S'_{3,C} \end{pmatrix} = \begin{pmatrix} 02&03&01&01 \\ 01&02&03&01 \\ 01&01&02&03 \\ 03&01&01&02 \end{pmatrix} \begin{pmatrix} S_{0,C} \\ S_{1,C} \\ S_{2,C} \\ S_{3,C} \end{pmatrix}$$

Para $0 \le C < Nb$

Luego, las operaciones sobre columnas se expresan como:

$$S'_{0,C} = (\{02\} \bullet S_{0,C}) \oplus (\{03\} \bullet S_{1,C}) \oplus S_{2,C} \oplus S_{3,C}$$
$$S'_{1,C} = S_{0,C} \oplus (\{02\} \bullet S_{1,C}) \oplus (\{03\} \bullet S_{2,C}) \oplus S_{3,C}$$
$$S'_{2,C} = S_{0,C} \oplus S_{1,C} \oplus (\{02\} \bullet S_{2,C}) \oplus (\{03\} \bullet S_{3,C})$$
$$S'_{3,C} = (\{03\} \bullet S_{0,C}) \oplus S_{1,C} \oplus S_{2,C} \oplus (\{02\} \bullet S_{3,C})$$

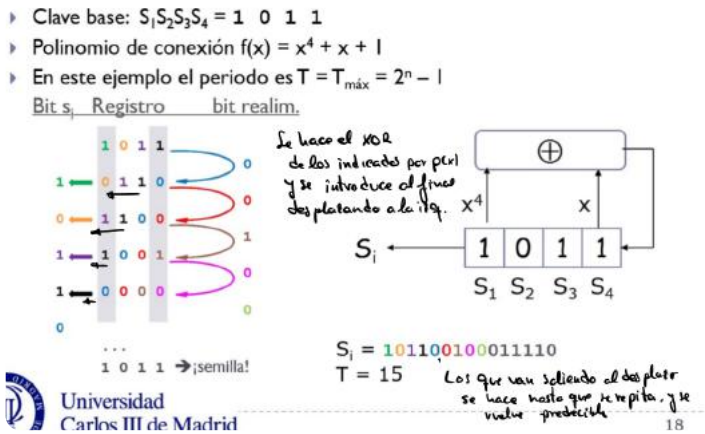
    - **AddRoundKey**(añade un clave de vuelta al estado): Se hace XOR del bloque tras los pasos anteriores y una clave de ronda, a cada byte del bloque le corresponde otro byte de la clave.



- La expansión generará los bytes de la subclaves a partir de la clave K principal. Revisar diapo 61, 2.3.2.

- **Cifradores de flujo:** Descomponen el mensaje en bytes (o en bits), cifran cada mi con el correspondiente ki.
  - **Sincrono**: Emisor y receptor se sincronizan externamente. Cifran independiente del texto en claro y del criptograma.
  - **Autosincrono**: Emisor y receptor se sincronizan automáticamente. La serie cifrante es una función de símbolos previamente cifrados.
- **Serie cifrante:** Aproximación para generar de la serie cifrante en emisor y receptor.
  - Mediante un generador de números pseudoaleatorios, de forma determinista.
  - A partir de una clave base, secreta e impredecible, de centenas de bits para evitar ataques.
- Propiedades deseables: **Postulados de Golomb.**
  - **Postulado G1:** Igual numero de ceros que de unos.
  - **Postulado G2:** La mitad de las rachas tiene longitud 1, la cuarta parte longitud 2 y la octava parte de longitud 3, etc. Que no haya demasiadas repeticiones.
  - **Postulado G3:** Para todo k, la Autocorrelación fuera de fase AC(k) es igual a una constante. Si desplazamos la cadena y contamos cuantos valores se mantienen en su posición, debe ser constante en todo momento.
  - **Función de Autocorrelación**: Desplazamiento de la secuencia S de periodo T de k bits hacia la izquierda: AC(k)= (A-F)/T
  - Periodos grandes.
  - **Aleatoriedad**: Distribución uniforme, independiente.
  - **Impredecible**.
    - Se puede medir por su complejidad lineal LC: numero de bits necesarios para predecir el resto de la secuencia, viene dado por la longitud mínima.
    - Se busca obtener la complejidad lineal mas alta posible.

- **PRNGs criptograficos:**
  - Basado en algoritmos existentes.
  - **LFSR**: Registró de desplazamiento con retroalimentación lineal.
    - Se da una ecuación que indica los grados de los términos de los que hacer XOR, que se introducirán al final de clave, desplazando hacia la izquierda y el bit que sale es uno de los bits generado por este método. Se vuelve a coger los términos de esos grados y se repite el proceso de desplazamiento, etc. El proceso se repite hasta que vuelva ha aparecer la clave.



  - Periodos altos pero complejidad lineal muy baja.
  - Aumentar la complejidad lineal, se puede hacer llevando el paralelos dos LFSR y haciendo XOR de ambos.
  - **Ventajas**: Transformación byte a byte, o bit a bit. Alta velocidad de cifrado y no se propagan los errores.
  - **Desventajas**: Escasa difusión, cada símbolo de M se corresponde con uno de C. No son realmente aleatorias, es generación determinista.
  - Problemas de reutilización de la clave:
    - Ataque con texto original conocido, se puede obtener K teniendo el texto M y correspondiente cifrado C. M XOR C=K
    - Ataque solo al criptograma, se puede obtener Mi XOR Mj si conocemos sus correspondiente cifrados, Ci XOR Cj. Mi XOR Mj=Ci XOR Cj.
  - **RC4**: Algoritmo muy simple y rápido, se usa mucho.
    - **Fase de inicialización**: Vector de estados S={S[0], ...,S[255]}, se usa la clave para permutar el vector S.

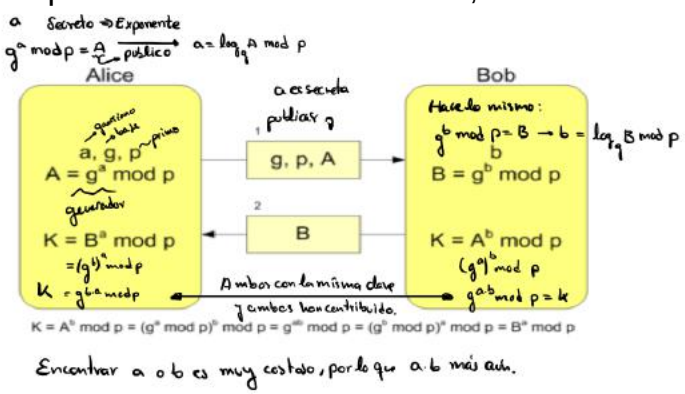
```
for i = 0 to 255 do
    S[i] = i
j = 0
for i = 0 to 255 do
    j = (j + S[i] + k[i mod l]) (mod 256)
    swap (S[i], S[j])
```

    - **Serie cifrante y cifrado:** Cada paso de cifrado se modifica S, en cada paso se vuelve a desordenar/permutar. Cogemos las posiciones i y j hacemos una modificación en los indices y sumamos sus valores y al clave es el elemento de la posición de las sumas. Cambiamos las 2 posiciones, i y j, y la suma sera el indice t del que sacamos el valor con el que hacer XOR con el mensaje para cifrar.

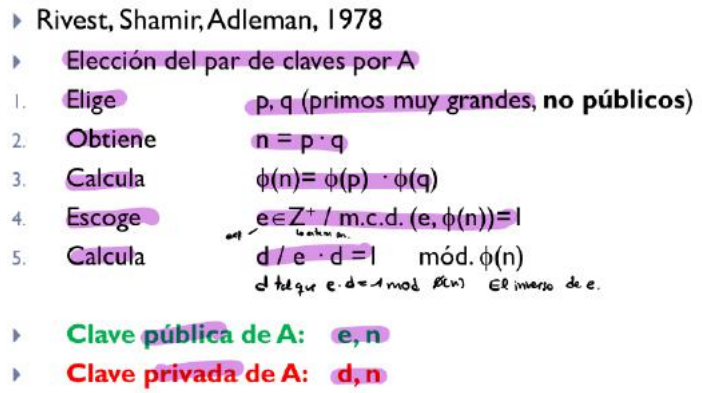
```
i = j = 0
for each message byte Mi
    i = (i + 1) (mod 256)  // contador simple
    j = (j + S[i]) (mod 256) // simula un random-walk
    swap(S[i], S[j])
    t = (S[i] + S[j]) (mod 256)
    Ci = Mi ⊕ S[t]
```

Cogemos 2 (i,j) posiciones, sumamos sus valores y la clave es el elemento de la posición de la suma. Y cambiamos las 2 pos, i y j

        - **Seguridad**: El resultado es no-lineal, ningún ataque práctico había dado resultados con claves de tamaños razonables (128 bits o mas) hasta 2015. Y desde ese momento se ha prohibido su uso, para usar otros mas seguros.
- **2.5 Cifradores Asimétricos:** Hay dos claves, una publica (cifrar) y otra secreta (descifrar). El que envía el mensaje lo cifra con la clave publica de receptor, para que solo el receptor pueda descifrarlo, que es el que conoce la clave privada. Surge del problema de acordar una clave sobre un canal inseguro, ya que de esta manera tu clave secreta no es conocida y lo que te cifran con la privada solo lo puedes descifrar tu.
    - La seguridad es del tipo seguridad computacional, que es posible en la teoría, pero en la práctica llevaría tanto tiempo o coste que no merece la pena.
    - Mas lento en comparación con los criptosistemas simétricos.
    - **Distribución de claves secreta** mediante criptografía simétrica: Hay muchas posibilidades
        - Entrega física de la clave.
        - Una tercera persona crea la contraseña y se la entrega a ambos.
        - Si se ha comunicado previamente pasarse la nueva por ese medio.
    - **Jerarquias de KDC (Key Distribution Center)**: Distribuye claves de sesión para cada par de usuarios, son temporales y se usan para una sesión y se descartan. Cada usuario tiene una clave maestra para poder recibir las claves de sesión, y si hay n participantes debe haber n*(n-1)/2 claves temporales.
    - **Intercambio de claves DH (Diffie-Hellman):** Hay dos sujetos A y B.
        - A necesita escoger un numero primo p y otro generador g que es la base, ambos públicos, y por ultimo un numero aleatorio a, que no se compartirá. Además genera g^a mod p=A. **A envía a B: g, p, A**
            - **p primo de 300 dígitos decimales y g de entre 2 y 5. a es secreto.**
        - B recibe lo de A, también escoge un numero aleatorio b y genera $B=g^b$ mod p que se lo enviará a A. **B envía a A: B**
            - Hallar a desde A o b desde B es inviable o determinar la K con A o B.
        - Ahora pueden generar la clave: Ambos con la letra mayúscula que reciben, el resultado de la base elevada a su exponente secreto, elevan la letra recibida a su exponente secreto con modulo p. **Clave en A: $K=B^a$ mod p**. **Clave en B: $K= A^b$ mod p.**
            - **No se garantiza la autenticación.**
            - Permite ataque de hombre interpuesto (man in the middle), una persona en medio que hace como intermediario, hace la función de B para A y la de A para B.



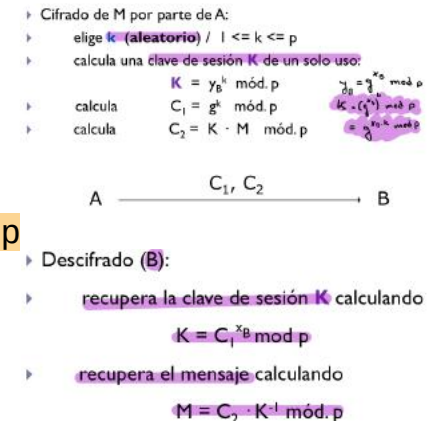
    - **RSA**: Conocido algoritmo efectivo de clave publica. Funcionamiento para una parte:
        - Se eligen dos números primos muy grandes, no públicos. p y q.
        - Se obtiene n=p*q, la n si será publica.
        - Se halla la identidad de Euler de n, que para el creador es sencillo conoce los dos primos. Será la multiplicación de p-1 y q-1. φ(n)=φ(p)*φ(q)
        - Se escoge un numero e, tal que sea positivo y sea coprimo con φ(n). Mcd(e, φ(n))=1
        - Y la d será el inverso de e modulo identidad de Euler de n. e*d=1 mod φ(n)
            - **Clave publica de A: e, n**
            - **Clave privada de A: d, n**



Cifrar: C=M^e mod n
Descifrar: M=C^d mod n

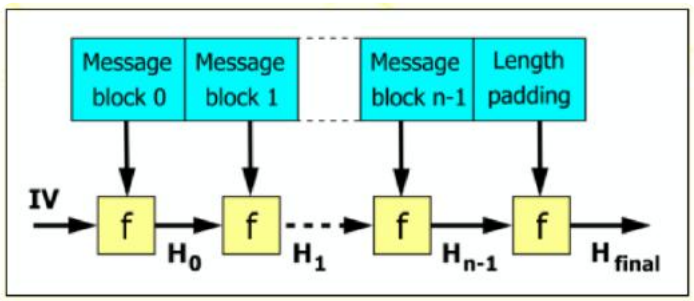
- Su seguridad se basa en la factorización de números grandes, la dificultad de hallar g y p. Y es de complejidad muy elevada.

- **ElGamal**: El algoritmo de cifrado es distinto que el de firma.
  - **Algoritmo de cifrado**: x es privada e y es publica.
    - **Cifrar** el mensaje M por parte de A:
      - Elegir una k entre 1<=k<=p
      - La clave de sesión será: K=y^k mod p Es privada para A. p primo y g generador.
      - Al B se le envía C_1=g^k mod p y C_2=M*K mod p
    - **Descifrar** el criptograma C por parte de B:
      - La clave de sesión será: K=C_1^x mod p Se halla la misma que A
      - El mensaje recuperado: M=C_2*K^(-1) mod p Para revertir su efecto se hace el inverso.



- **2.6 Funciones Resumen y MAC**
  - **Funciones Resumen:** Mecanismo para asegurar la integridad del mensaje. Es una función que acepta un bloque de datos (M) de longitud variable y genera un resumen (hash) de longitud fija. El resumen que se genera es siempre de una longitud fija y es único para cada mensaje, con independencia de la longitud del mensaje de entrada.
    - H(M)= hash
    - **Colisión**: Dado que el tamaño del resumen es fijo y menor que el tamaño del mensaje, las posibilidades de resumen es menor que las de mensajes, lo que quiere decir que habrá mensajes que comparten resumen.
      - H(M)=H(M') Colisión
    - **Funciones Resumen Criptograficas:** Son las que se aplican para generar el resumen y deben cumplir los siguientes requisitos:
      - Ser aplicable para mensajes de entrada de cualquier longitud.
      - Producir resúmenes de salida de una longitud fija.
      - La salida generada por la función resumen debe satisfacer los requisitos de pseudo-aleatoriedad.
        - **Difusión**: Si se modifica un solo bit del mensaje, en el resumen deben varias al menos la mitad de los bits.
        - **Determinista**: La aplicación de la función sobre los mismos datos debe producir el mismo resumen.
        - **Eficiente**: El calculo del resumen de un mensaje debe ser rápido.
      - **Resistente a preimagenes:** Dado un resumen h, es computacionalmente imposible encontrar un mensaje M' cuyo resumen coincidan con el primero.
        - NO debe ser reversible la función.
      - **Resistente a segunda imagen:** Dado un mensaje M, es computacionalmente imposible encontrar un M' tal que el resumen de ambos coincidan.
      - **Resistente a colisiones**: Es computacionalmente imposible encontrar dos mensajes M y M' tales que sus resúmenes coincidan.
      - No debe permitir hallar colisiones con complejidad menor que la fuerza bruta.

- **Computacionalmente imposible:** Si no hay un algoritmo para la búsqueda de colisiones mas eficiente que el de fuerza bruta. Si el espacio de resúmenes es suficientemente grande, no será posible encontrar una colisión en un tiempo razonable.
  - Se considera que el algoritmo se ha roto si la complejidad es menor que la fuerza bruta. La barrera de 2^{64} estable el mínimo aceptable para una complejidad algorítmica.
- **Probabilidad de encontrar una colisión:** En definitiva, la complejidad algorítmica.
  - **Ataque de preimagen:** 1/2^n
  - **Ataque de segunda preimagen:** 1/2^n
  - **Ataque de colisión:** 1/2^(n/2) (Ataque del cumpleaños, p>=50%)
- **Estructura Merkle-Damgard**: Algoritmo con iteraciones encadenadas.
  - **Entrada**: El mensaje se divide en bloques de tamaño b, y en el ultimo bloque se añade al final la longitud total del mensaje. Si es necesario se añade relleno al bloque. Dificulta la búsqueda de colisiones.
  - **Función**: Recibe dos entradas, una es el bloque de mensaje y otra es la salida del bloque anterior, que en caso de que sea el primero es un vector. Cada etapa produce un resumen de n bits fijos.
  - **Salida**: El resumen final es la salida del ultimo bloque.
  - Si la función de compresión es resistente a colisiones, también lo es la función resumen (lo contrario no tiene porque ser cierto)



$CV_0 = IV$ = valor inicial del resumen
$CV_i = f(CV_{i-1}, B_{i-1})$ $1 \le i \le L$
$H(M) = CV_L$

- **MD5**: Es una función de resumen.
  - **Entrada**: El mensaje se divide en bloques de 512 bits, sobre el ultimo bloque se hace rellenado. Esos 512 bits se subdividen 16 bloques de 32 bits.
  - **Función**: Se realizan 4 rondas de 16 operaciones basadas en:
    - Función no lineales.
    - Suma modulo 2^{32}.
    - Rotación de bits.
  - **Salida**: Genera un resumen de 128 bits.
- SHA-0, SHA-1, SHA-256 y SHA-512.

Algorithm	Output size	Internal state size	Block size	Collision
HAVAL	256/224/192/160/128	256	1024	Yes
MD2	128	384	128	Almost
MD4	128	128	512	Yes
MD5	128	128	512	Yes
RIPEMD	128	128	512	Yes
RIPEMD-128/256	128/256	128/256	512	No
RIPEMD-160/320	160/320	160/320	512	No
SHA-0	160	160	512	Yes
SHA-1	160	160	512	With flaws
SHA-256/224	256/224	256	512	No
SHA-512/384	512/384	512	1024	No
WHIRLPOOL	512	512	512	No

- **Message Authentication Code (MAC):** Es un algoritmo que emplea una clave secreta para producir un valor de longitud fija sobre un mensaje de longitud variable.
  - Todos los que conocen la clave pueden comprobar la integridad.
  - Los receptores que conocen la clave, son capaces de autenticar el origen del mensaje. Ya que la clave la conocen unos pocos.
  - En caso que el mensaje incluya un numero de secuencia, se evitan ataques por replicación.
  - **NO tiene que ser invertible.**
  - Como pasa en las funciones resumen, el numero de posibles valores MAC es menor que el numero de posibles mensajes, por lo que se producen **colisiones**.
  - **Requisitos**:
    - Dado un mensaje M y el valor MAC(K,M), es computacionalmente imposible encontrar un mensaje M' cuyo MAC(K, M') coincidan.
    - MAC(K,M) debe estar uniformemente distribuido, de forma que la probabilidad de encontrar dos mensajes M y M' cuyos valores MAC coincidan es 1/2^n
    - Sea M' un mensaje resultante de aplicar una transformación a M[M'= f(M)]. En ese caso la probabilidad es 1/2^n
  - **Ataques a funciones MAC:**
    - Dado un conjunto de Mi, MAC(K, Mi), el atacante desea generar M', MAC(K,M'), con M'!=Mi para cualquier i = 0...n
      - Por fuerza bruta la complejidad viene dada por que es más fácil, si hallar la clave 1/2^k o el valor MAC 1/2^n.
      - **Min(1/2^k, 1/2^n)**
      - Por criptoanalisis, deben existir vulnerabilidades en el diseño.
  - **MAC Basado en funciones resumen (HMAC)**: Emplea funciones resumen existentes. Se aplica la función sobre una versión del mensaje al que añaden un conjunto de bits calculados a partir de la clave.

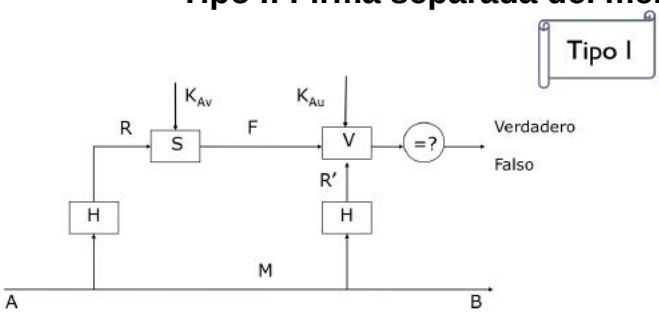
    $$HMAC(K, M) = H[(K' \oplus opad) \,||\, H[(K' \oplus ipad) \,||\, M]]$$

    K': K *padded* con 0's a la izquierda hasta tener longitud b
    b: Longitud de cada bloque procesado por la función resumen
    ipad: 00110110 (0x36) repetido b/8 veces
    opad: 01011100 (0x5C) repetido b/8 veces
    ||: operación concatenación
  - **MAC Basado en cifrado e bloque:** Cifran el mensaje mediante un algoritmo de cifrado simétrico en bloque en modo CBC. El valor del MAC es el resultado del cifrado del ÚLTIMO BLOQUE. De esta manera el MAC dependerá de todos los bit anteriores.

2.7 Firma Digital y PKI (Interfaz de Clave Publica)

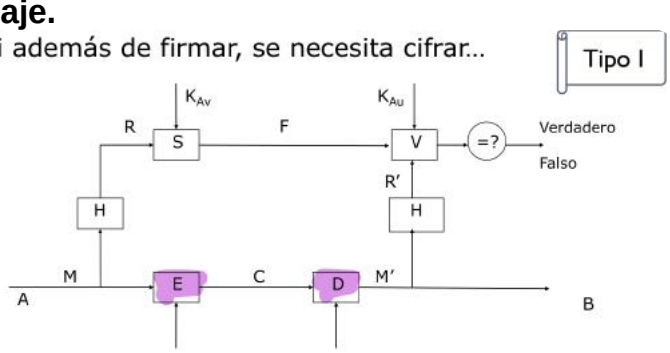
- **Firma Digital**: Valor calculado con un algoritmo criptografíco y que se asocia con un objeto de datos de tal manera que cualquier destinario de los datos pueda utilizar la firma para verificar el origen de los datos y la integridad. Va ligado a un mensaje y solo puede ser escrita por la persona a la que corresponde.
  - **Debe proporcionar:**
    - **Autenticación** del origen.
    - **Integridad** de los datos.
    - **No repudio** del firmante.
    - Si se usa cifrado, también confidencialidad.
  - **Propiedades de una firma manual:**
    - Fácil y barata de producir.
    - Fácil de reconocer.
    - Imposible de rechazar.
    - Infalsificable.

- La firma digital cumple las mismas propiedades, pero ademas no puede ser siempre la misma, ya que sera muy fácilmente falsificable.
- **Propiedades de seguridad**:
  - Auténtica indudablemente al signatario.
  - Garantia de la integridad del mensaje recibido.
  - Garantía de no repudio, medio de prueba en la resolución de disputas.
  - De base **NO** asegura la confidencialidad.
- **Componentes**:
  - **Algoritmo de firma**.
    - **Determinista**: Dos firmas del mismo mensaje producen el mismo resultado. RSA.
    - **Aleatoria**: Depende de un conjunto de indices. ElGamal.
    - **Tipos**:
      - **Tipo I**: Firma separada del mensaje o con apendice. M y F separados.
        - Se envía el mensaje original y la firma de su hash
      - **Tipo II**: Firma con recuperacion del mensaje. Todo dentro de F.
        - Se envía el mensaje original firmado.
      - **Tipo III**: Esquema de firma con recuperacion del mensaje transformado en esquema de firma separada con ayuda de una función resumen.
  - **Algoritmo de verificación de la firma.**
- **Tipo I: Firma separada del mensaje.**



$R=H(M)$ $F=S(K_{Av}, R)$

Si además de firmar, se necesita cifrar...



$C=E(K_{Bu}, M)$ $R=H(M)$ $F=S(K_{Av}, R)$

1. Obtener el comprimido resumen $R=H(M)$ y su firma $F=S(R)$ (obtenida con el algoritmo de firma S y la clave privada del remitente K_{Av})
2. Enviar el par (M, F)
3. El receptor calcula $R' = H(M)$ a partir del primer elemento del par (el mensaje M)
4. El receptor evalúa la validez de la firma recibida ejecutando el algoritmo de verificación de firma V con la clave pública del remitente K_{Au} y a partir del resumen calculado R' y la firma recibida F. Se acepta el mensaje si el resultado del algoritmo es verdadero y se rechaza en caso contrario

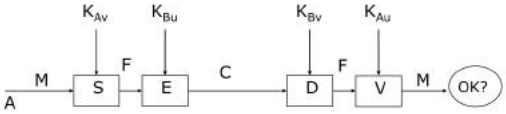
- **Tipo II: Con recuperacion de mensaje.**
  - El mensaje original se recupera durante el proceso de verificacion.

Tipo II

$F = S(K_{Av}, M)$

Si además de firmar, se necesita cifrar...

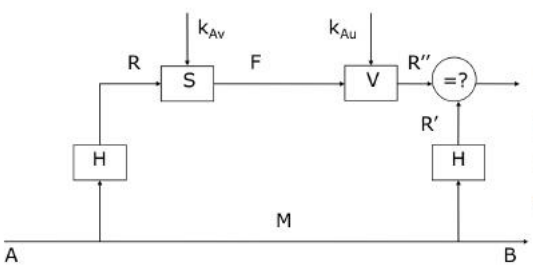
Tipo II



$C = E(K_{Bu}, F) = E(K_{Bu}, S(K_{Av}, M))$

1. Obtener la firma F=S(M) (obtenida con la clave privada del remitente K_{Av})
2. Enviar F
3. El receptor calcula M a partir de F utilizando el algoritmo de verificación de firma V y la clave pública del remitente K_{Au} . Se acepta el mensaje si el resultado del algoritmo es correcto (el mensaje obtenido pertenece al espacio de mensajes permitidos)

  - **Desventajas**:
    - Hay que cifrar dos veces si queremos garantizar la autenticidad y el secreto, lo que lo hace mas lento. En otros se hace en paralelo, no se cifra el que se firma.
    - Hay que dividir el mensaje en bloques y firmar cada uno, se concatenan.
    - No hay conexión entre los fragmentos, no se sabe si han llegado todos los fragmentos o si lo han hecho en orden.
- **Tipo III: Con recuperacion del mensaje combinadla con funcion resumen.**



$R=H(M)$ $F=S(k_{Av}, R)$

Tipo III

1. Obtener el resumen R = H(M) del mensaje M
2. Firmar R con la clave privada del emisor $F = S(R, K_{Av})$
3. A envía a B: (M, F)
4. El receptor calcula el resumen del mensaje por dos caminos distintos:
   1. R' = H(M)
   2. $R'' = V(K_{Au}, F)$
5. El receptor compara R' con R'', aceptando el mensaje si coinciden y rechazándolo en caso contrario

- **Firma opaca**: Permite a una entidad A conseguir que otra B firme un mensaje M sin que en el proceso B conozca el contenido del mensaje. B firma un mensaje de A sin conocerlo.
  - Pasos
    - A envía el mensaje M, encubierto, al Notario N, quien lo firma y remite a A
    - A invierte el encubrimiento y dispone del mensaje M firmado
    - El notario no ha conocido M
    - El encubrimiento debe ser compatible con el algoritmo de firma

- **Tipos de ataques:** El objetivo del atacante es crear firmas que sean aceptadas como validad.
  - **Rotura total:** El atacante tiene un algoritmo de firma funcionalmente equivalente al autentico.
  - **Rotura selectiva**: El atacante es capaz de forjar firmas para un tipo de mensaje particular.
  - **Rotura existencial:** El atancante es capaz de forjar una firma para al menos un mensaje.
- **Firma digital ElGamal:** Es un esquema de firma aleatorio y con apéndice (no se deshace la firma, solo verifica).

Inicialización:

- Se ha elegido un primo adecuado p (con p ~ 200 bits) y un elemento primitivo g (generador de CG(p))
- El firmante elige una clave secreta x_A, $1 < x_A < p - 1$ y hace pública $y_A \equiv g^{X_A}$ (mód. p)
- **Creación de la firma por A**
  - A, con claves x_A e $y_A = g^{X_A}$ mód. p, para firmar M (ver Nota) elige un entero k (coprimo con p-1) y calcula la firma, el par (r,s):
    - $r = g^k$ (mód. p)
    - $s = (M - x_A \cdot r) \cdot k^{-1}$ mód. (p -1)

$M = x_A \cdot r + k \cdot s$ (mód p−1)

- **A envía a B: (M, r, s)**
- **Verificación de la firma por B**
  - B, recibidos M y (r, s), acepta la firma si coinciden las dos expresiones: Debe conocer y_A, g, p
    - $V_1 = y_A^{\,r} \cdot r^s$ (mód. p)
    - $V_2 = g^M$ (mód. p)

Universidad

Nota: Trabajaremos directamente sobre M, aunque en realidad el algoritmo se aplicaría sobre H(M)

- **Firma digital con RSA (Tipo II):** Es un esquema determinista y con recuperacion de clave, permite por medio de la firma obtener el mensaje original.

1. Alicia crea una clave pública e (o K_{Au}) y una privada d (o K_{Av})
2. **Generación** de la firma :
   1. Alicia cifra el mensaje con su clave privada $F = D_{RSA}(M, d)$
3. Alicia envía F a Benito
4. **Verificación** de la firma:
   1. Benito obtiene el mensaje utilizando la clave pública de Alicia

$M = E_{RSA}(M, e)$

- **Firma digital con RSA (Tipo III)**: Se usa siempre transformado en firma digital con apéndice mediante la aplicación de una funcion resumen inicial al mensaje.

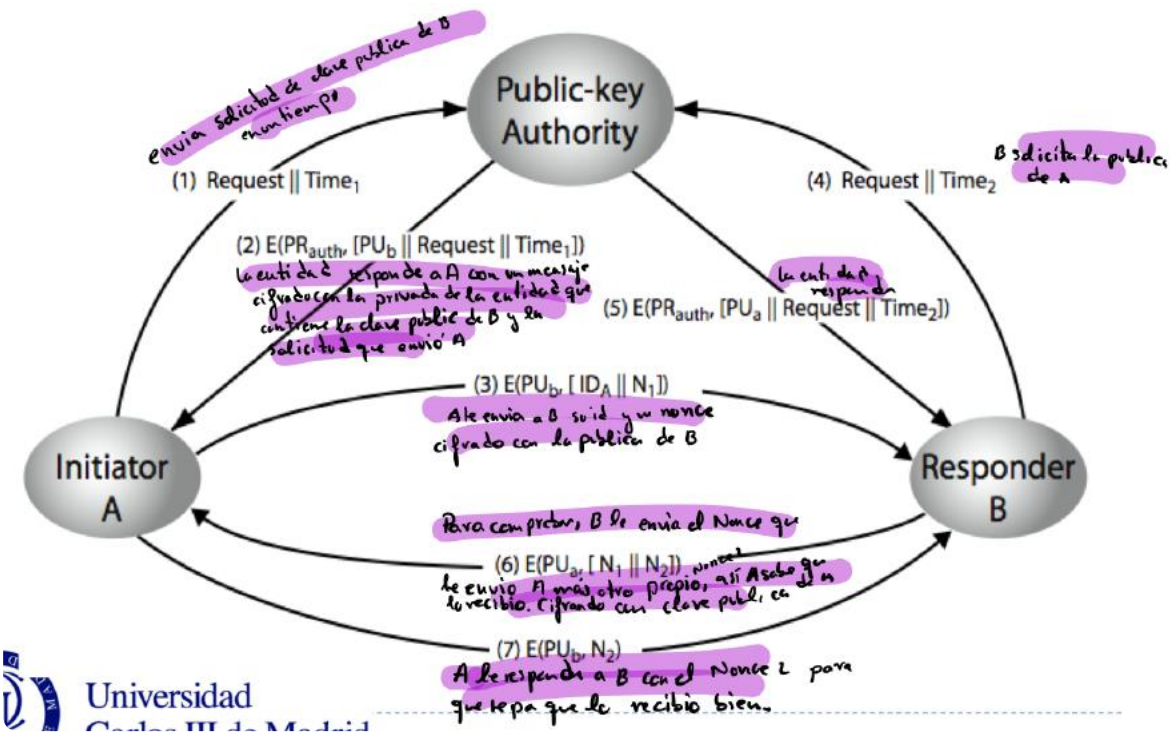
1. A genera la firma
   1. A obtiene el resumen R = H(M) del mensaje M
   2. A firma con su clave privada el resumen: $F = D_{RSA}(R, K_{vA})$
2. A envía a B: (M, F)
3. B verifica la firma
   1. Obtiene R'' aplicando el algoritmo de cifrado E_{RSA} sobre F con la clave pública K_{Au}: $R'' = E_{RSA}(K_{Au}, F)$
   2. A partir del M recibido, calcula el resumen de nuevo R'
   3. Compara R' con R''

▸ Para mantener la confidencialidad (envío del mensaje cifrado):

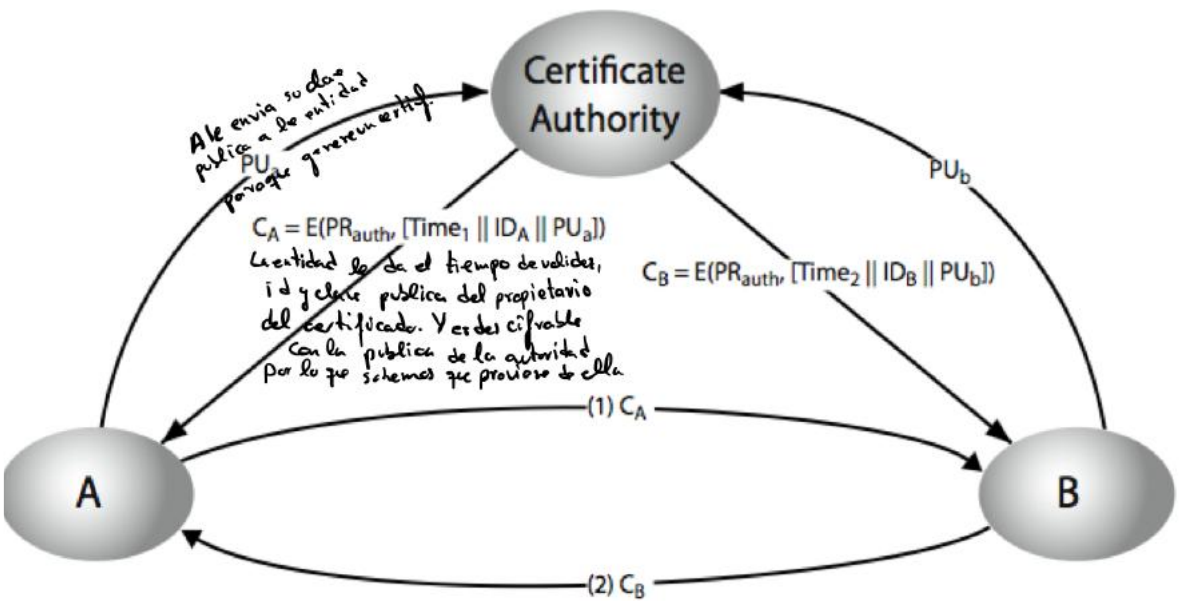
1. A obtiene el resumen R = H(M) del mensaje M
2. A firma con su clave privada el resumen: $F = D_{RSA}(K_{Av}, R)$
3. A cifra M con la clave pública del receptor B: $C = E_{RSA}(K_{uB}, M)$
4. A envía a B: $(C, F) = (E_{RSA}(K_{uB}, M), D_{RSA}(K_{Av}, R))$
5. B descifra el mensaje y verifica la firma
   1. Descifrando C con su clave ~~pública~~ privada, por lo que obtiene M
   2. Aplica el algoritmo de cifrado de E_{RSA} sobre F con la clave pública del emisor, por lo que obtiene R' (verificav)
   3. A partir del M obtenido al descifrar C, calcula de nuevo el resumen R''
   4. Compara R' con R''

- **Formato de firmas digitales**: Existen varios formatos, pero la mayoría encapsula en un sobre los datos, la identidad del firmante y la firma. Un ejemplo es el PKCS#7.
- **Formato de Sobre PKCS#7:** Estándar desarrollado por RSA Laboratories, define varios formatos: Data, EnvelopedData, SignedData, etc.
- **Firma Digital XML:** Permite la firma completa o parcial utilizando el lenguaje XML.
  - **Métodos**:
    - **Wrapped**, el formato incluye el contenido, los propios datos.
    - **Detached**, la firma esta separada del contenido, separada del XML.
    - **Embedded**, la firma es parte del contenido firmado.
  - Fno requiere infraestructura de certificados, se incluye la información de la clave y reserva espacio para dar informacion del certificado.

- **2.5 Distribucion de claves:**
  - **Distribucion de claves secretas mediante criptografía de clave publica.**
    - **Criptosistemas híbridos:**
      - **Ventajas y desventajas:**
        - **Simétricos**:
          - **Ventajas**: Simetría y rapidez.
          - **Desventajas**: Existen un canal seguro y difícil gestión de un gran numero de claves.
        - **Asimétricos**:
          - **Ventajas**: No exige un canal seguro y más fácil gestión de un gran número de claves.
          - **Desventajas**: Asimetría y lentitud.
      - **Metodo**: Cifra el mensaje simétricamente y la clave asimétricamente.
        - El texto en claro se cifra simétricamente con un clave de sesion que genera advocates de forma aleatoria.
        - La clave de sesion se cifra asimétricamente con la clave publica del destinatario.
  - **Distribucion de claves publicas:**
    - **Anuncios publico:** Cualquiera puede poner su clave publica y no hay ningún método que controle a quien corresponde cada una, por lo que se puede hacer pasar por la de otra e interceptar mensajes.
    - **Directorio publico con acceso universal:** Se almacena el nombre y clave publica correspondiente, de esta manera se puede asegurar la pertenencia de la clave.
      - Es esencial la CONFIANZA en el directorio.
      - Asegura la inviolabilidad, pero para pasar la clave y nombre se debe hacer por un canal seguro.
      - El autor puede remplazar en cualquier momento los datos, por si son obtenido por terceros y se siga pensado que es la correcta. Por ello también se actualizan periódicamente.
      - Se puede acceder electrónicamente.
    - **Autoridad de clave publica:** Parecido al directorio publico, pero mejora la seguridad mediante mecanismos de control sobre las claves y cada usuario tiene su clave publica del directorio para acceder. Requiere acceso en tiempo real.
      Se sigue un protocolo como:



- **Certificados de clave publica:** Permiten el intercambio de claves estando la autoridad offline. Un certificado de clave publica se asocia de forma segura (**autenticidad, integridad**). Se hace cadena de certificados, la autoridad superior certifica a la inferior, así sucesivamente hasta que el emisor y receptor confíen en esa autoridad, de esa manera se asegura que la clave publica es la correcta.
  - **Necesita**:
    - Una identidad.
    - Periodo de validez.
    - Derechos de uso, etc.
  - **Requisitos para la eficiencia:**
    - Cualquiera pueda leer el certificado para obtener el nombre y la clave publica del propietario del mismo.
    - Cualquiera pueda verificar que se origino de la entidad de certificación, no sea falso.
    - Solo la entidad pueda actualizar y modificar los certificados.
    - Cualquiera pueda verificar la validez del certificado, en cuanto a fecha.
  - Los contenidos son firmados por la Autoridad de Certificación.
  - La validez de los certificados puede ser comprobada por cualquiera que conozca la clave publica de la AC. (verificación de la firma)
  - Si confiamos en la AC, confiamos en los certificados que ella haya firmado.
  - Se asocia la clave publica con la identidad.



- **2.6 Infraestrucutra de Clave Pública**
  - Combinación de Hardware, Software y métodos de seguridad.
  - **Son necesarios:**
    - Usuario, inicia la operación.
    - Autoridad, permite verificar la pertenencia y garantiza la validez de los certificados
    - Destinatario, recibe los datos cifrados o firmados.
  - Vincula una identidad a una clave publica.
  - Vinculación (clave publica - ID), ambos datos se firman digitalmente. Aunque pueden aparecer mas datos. Es verificable si se conoce la clave publica del Fichero Publico.
  - Se basa en la confianza mutua en la Autoridad de certificado.

- **Que contiene:**
  - ID del emisor A
  - Clave publica de A
  - ID de la autoridad de certificado
  - Periodos de validez
  - Numero de serie
- Todos estos datos son firmados por la Autoridad de Certificado con la privada. Se puede verificar desfirmandolo con la publica.
- **Funcionamiento:** Intercambio mensaje entre A y B
  - A quiere cifrar un mensaje para B, B le envía su certificado de clave publica Cb.
    - A debe validar el certificado Cb.
  - A le envía a B, el mensaje, la firma y ademas su certicado, para que B pueda verificar su clave publica.
    - B debe validar el certicado Ca, y después validar la firma de A sobre M.
  - Cuando se envían certificados se puede enviar solo el propio o ademas adjuntar toda la cadena de certificación.
- **Validar un certificado:**
  - Primero se debe tener una clave publica de la Autoridad de Certificado, para poder empezar a confiar en una. El problema es como la validamos, hay que dar un primer paso, aunque no tengamos confianza inicialmente.
    - La Autoridad de Certificado raíz se autoafirma, es la única que lo hace.
  - Verificamos la firma emitida por AC que aparece en el certificado.
    - Para ello usamos la clave publica de AC.
  - Verificamos la **fecha de uso** del certificado y si **no ha sido revocado**.
- Se basa en un modelo jerárquico, en el que la autoridad de certificado superior certifica a las inferiores y la que esta en la cima es la AC raiz.
- Se debe validar tambien toda su cadena de certificación hasta llegar a un certificado raíz.