Criptografía

Criptografía simétrica moderna
Campos finitos
Criptografía asimétrica

Criptografía simétrica moderna

Criptografía simétrica moderna

- Utiliza las mismas ideas que la criptografía clásica: trasposiciones y sustituciones
- Se busca que los algoritmos sean tan complejos que un criptoanalista necesite para romperlo un poder computacional demasiado grande
- Se busca combatir a un intruso o atacante hipotético

Criptografía simétrica moderna

- Modelo de encriptación:

- El intruso puede escuchar en el canal. A veces también puede enviar mensajes o alterar los que están en tránsito.

DES

- Desarrollado por IBM en 1976
- Estándar de la NSA desde 1977
- Implementable en software y en hardware
- Claves de 64 bits. El octavo bit de cada byte es de paridad, por lo que la clave efectiva es de 56 bits.

DES

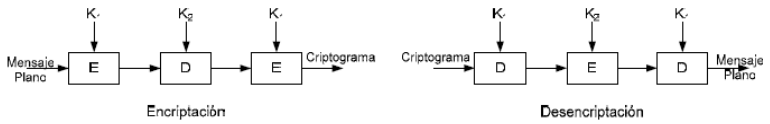
- No se ha publicado a la fecha ningún criptoanálisis que permita leer tráfico encriptado con DES a un costo menor que fuerza bruta
- El ataque por fuerza bruta hoy es factible, una máquina dedicada de USD 10.000 de costo puede barrer el espacio de claves en 2 días. Por este motivo se buscó estandarizar una nueva función.

Triple DES (TDES, 3DES)

- Se busca mejorar la fuerza de encriptación
- El algoritmo DES no tiene estructura de grupo, entonces $E_{k1}(E_{k2}(m))$ tiene el doble de fuerza que E_{k3} para k1, k2, k3 de 56 bits. Pero esto resulta incompatible con sistemas legacy que utilicen el DES simple, y además es susceptible a ataques meet-in-the-middle.

Triple DES

- Por esos dos motivos, se utiliza el siguiente sistema:



- Este sistema permite tener compatibilidad con sistemas con DES simple al hacer k1=k2, ya que la transformación 3DES en esas condiciones equivale a DES
- La fuerza de encriptación es idealmente de 112 bits, pero la fuerza efectiva es de 80 bits debido a algunos ataques publicados para este algoritmo

Ataque meet-in-the-middle

- Se desea atacar un sistema: $E_{k1}(E_{k2}(m))=c$ conociendo un mensaje m y su criptograma c
- Se computa $E_k(m)$ para todo k y se guarda en una tabla
- Se computa $D_k(c)$ para cada k y se busca el resultado en la tabla. Si se encuentra, es muy probable haber hallado k1 y k2
- Con 2^N memoria y 2^{N+1} operaciones se evitaron hacer 2^{2N} operaciones

AES

- En 1997 se inició un concurso para reemplazar a DES como estándar
- Los 5 algoritmos que llegaron a la final de 1999 fueron MARS, RC6, Rijndael, Serpent y Twofish
- En 2000 se estandarizó Rijndael como AES
- El algoritmo puede encriptar bloques de 128, 192 o 256 utilizando claves de esos tamaños
- Actualmente no tiene ataques conocidos que reduzcan su seguridad de forma significativa

Criptografía asimétrica

Criptografía asimétrica

- Cuando se busca encriptar para confidencialidad, el modelo de encriptación es:

- Debe ser imposible (o computacionalmente inviable) calcular K_{pr} a partir de K_{Pu}

Criptografía asimétrica

- Se resuelve la distribución de las claves, que era un problema en criptografía simétrica
- Los algoritmos son computacionalmente mucho más costosos
- Los cifradores asimétricos están basados en algún problema matemático computacionalmente complejo de resolver

RSA

- Publicado en 1978, se utiliza para proveer tanto confidencialidad como firma digital (autenticación)
- Está basado en el problema de la factorización de enteros largos
- La forma más eficiente de criptoanalizarlo es intentando factorizar el entero largo asociado

RSA

- Generación de las claves:
  - Generar dos números largos aleatorios primos y distintos, p y q. Ambos deben poseer la misma cantidad de dígitos (típicamente 1024 bits).
  - Obtener n = p x q y φ(n) = (p-1) x (q-1).
  - Seleccionar un número relativamente primo de φ(n) y denominarlo d. Se debe cumplir: gcd (d,φ(n)) = 1. “gcd” significa “greatest common divisor”. Entonces lo que se pretende es que d y φ(n) no tengan ningún divisor común.
  - Finalmente se calcula el entero e, con respecto a p,q y d que será el inverso multiplicativo de d, modulo φ(n). Se debe cumplir: e x d = 1 (mod φ(n)).
  - La clave pública consiste en el par (e,n) y la privada en el par (d,n).

RSA

- Encriptación y desencriptación:
  - Representar el mensaje como enteros de 0 a n-1. Se debe partir el mensaje en una serie de bloques y representar cada bloque como un entero. Se puede utilizar cualquier representación estándar. El propósito de este paso no es encriptar el mensaje, sino obtener una representación numérica requerida para la encriptación.
  - Se calcula $c = m^e \bmod n$. Entonces el criptograma es el resto de dividir m^e por n.
  - Para desencriptar se utiliza la clave privada (d,n) y se calcula $c^d \bmod n$. Posteriormente se deberá hacer la presentación numérica inversa utilizada antes de la encriptación.

RSA

- Ejemplo:
  - Supongamos que tenemos el mensaje plano *criptografia* y elegimos p=47, q=59, n=pq = 47 x 59 = 2773 y d = 157. Entonces φ(2773) = 46 x 58 = 2668. Según el ejemplo obtuvimos que e = 17.
  - Representación numérica a utilizar:

A	B	C	D	E	F	G	H	I	J	K	L	M
00	01	02	03	04	05	06	07	08	09	10	11	12

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

RSA

Mensaje Plano: criptografia → 02170815191406170005080 0

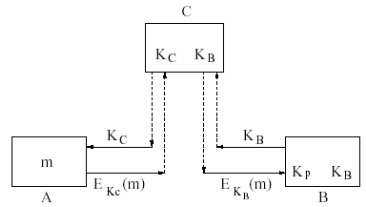
Paso	Mensaje	Cálculo	Cifrado
1	0217	0217^{17} mod 2773	1219
2	0815	0815^{17} mod 2773	0635
3	1914	1914^{17} mod 2773	2575
4	0617	0617^{17} mod 2773	2375
5	0005	0005^{17} mod 2773	0508
6	0800	0800^{17} mod 2773	1505

Desencriptación:

Paso	Cifrado	Cálculo	Mensaje
1	1219	1219^{157} mod 2773	0217
2	0635	0635^{157} mod 2773	0815
3	2575	2575^{157} mod 2773	1914
4	2375	2375^{157} mod 2773	0617
5	0508	0508^{157} mod 2773	0005
6	1505	1505^{157} mod 2773	0800

RSA

- Criptoanálisis:
  - Para descartar la posibilidad de ataques de fuerza bruta se recomienda usar claves de al menos 1024 bits.
  - Como todos los algoritmos asimétricos, es vulnerable a ataques de intermediario:



  - La única forma de evitar este tipo de ataques es que cada parte esté segura de que la clave pública de su contraparte es verdadera.

Merkle-Hellman (Knapsack)

- Publicado en 1978, fue roto en 1982 por Adi Shamir (inventor de RSA)
- Operaciones más simples que en RSA
- Sólo puede cifrar en un sentido: la clave pública sólo sirve para cifrar, la privada sólo para descifrar. Por lo tanto no sirve para firma digital.
- Se basa en el problema matemático de la suma de subconjuntos

Merkle-Hellman (Knapsack)

- Problema de la suma de subconjuntos: dado un conjunto de enteros, ¿existe algún subconjunto cuya suma sea cero?
- Si la secuencia es superincremental $s_{n+1} > \sum_{j=1}^{n} s_j$ entonces el problema es trivial
- Si no lo es, el problema es NP-completo
- La clave pública será un conjunto no superincremental, mientras que la privada será un conjunto superincremental

Merkle-Hellman (Knapsack)

- Generación de claves:
  - Se elige una secuencia superincremental:
    - w = (w_1, w_2, …, w_n)
  - Un entero aleatorio *q* que cumpla $q > \sum_{i=1}^{n} w_i$
  - Un entero aleatorio r coprimo con q (esto se pide para que exista r^{-1} mod q)
  - Y se calcula β = (β_1, β_2, …, β_n) con $\beta_i = rw_i \bmod q$
  - La clave pública es β y la privada es (r, w, q)
  - β es una secuencia no superincremental, y w es una secuencia superincremental

Merkle-Hellman (Knapsack)

- Encriptación del mensaje $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$:
  - $\alpha_1 \ldots \alpha_N$ son los N bits del mensaje
  - $c = \sum_{i=1}^{n} \alpha_i \beta_i.$
- Desencriptación del criptograma c:
  - Sea $s=r^{-1}$ y sea $c'=cs \pmod q$
  - Luego $c' \equiv cs \equiv \sum_{i=1}^{n} \alpha_i \beta_i s \pmod q.$
  - Pero con $\beta_i s \equiv w_i r s \equiv w_i \pmod q.$
  - Resulta $c' \equiv \sum_{i=1}^{n} \alpha_i w_i \pmod q.$ y se debe resolver el problema para la secuencia superincremental

Campos finitos

Operaciones matemáticas en Z_n

Operaciones en campos finitos

- Computar el GCD: algoritmo de Euclides
  - GCD(a, b) = GCD(b, a mod b)
- Ejemplo:
  - gcd(1970,1066)
  - gcd(1066, 904)
  - gcd(904, 162)
  - gcd(162, 94)
  - gcd(94, 68)
  - gcd(68, 26)
  - gcd(26, 16)
  - gcd(16, 10)
  - gcd(10, 6)
  - gcd(6, 4)
  - gcd(4, 2)
  - gcd(**2**, 0)

Operaciones en campos finitos

- Computar a^{-1} mod b: algoritmo extendido de Euclides:
- Sea A1=1, A2=0, A3=b, B1=0, B2=1, B3=a
- Iterar hasta que B3 sea 1 o 0:
  - Q_i= floor($A3_{i-1}/B3_{i-1}$)
  - $Aj_i=Bj_{i-1}$
  - $Bj_{i-1}=Aj_{i-1}-Q_i.Bj_{i-1}$
- Si B3 es 1, la inversa es B2; si B3 es 0 la inversa no existe

Ejemplo

Q	A1	A2	A3	B1	B2	B3
	1	0	1759	0	1	550
3	0	1	550	1	-3	109
5	1	-3	109	-5	16	5
21	-5	16	5	106	-339	4
1	106	-339	4	-111	**355**	1

Operaciones en campos finitos

- Computar a^b mod q: exponenciación por cuadrados
  - Cualquier numero tiene una expresión: $\sum_i \alpha_i 2^i$
  - Hago $a_i = a_{i-1}^2$ mod q tantas veces como bits tenga b
  - El resultado es $\prod_i bit_i(b) a_i$ mod q