El formalismo matemático de cifrados simétricos y asimétricos: 0

Cifrado simétrico:

Alice y Bob comparten una clave secreta k
Se llama un cifrado simétrico porque Alice y Bob pueden hacer todo y tienen (simétrico) conocimiento de la clave.

$$K = \{\text{claves}\}$$

$$M = \{\text{mensajes}\}$$
$$C = \{\text{texto cifrados}\}.$$

El proceso de cifrar se puede pensar como una función

$$e : K \times M \to C$$

y el proceso de descifrar como una función d

$$d : K \times C \to M$$

Y queremos que ~~sean inversas~~ cumplan:

$$d(k, e(k,m)) = m. \quad \forall k \in K \text{ y } m \in M$$

También se escribe: fijada K y se $e_k : M \to C$ y $d_k : C \to M$ t.q

$$d_k(e_k(m)) = m \quad \forall m \in M$$

En particular, esto implica que la función e_k es inyectiva:

Si $e_k(m) = e_k(m') \Rightarrow m = d_k(e_k(m)) = d_k(e_k(m')) = m'$.

El principio de Kerckhoff. Uno se debería pensar que cualquier adversario Eve sabe las funciones e y d. O sea, la seguridad del sistema criptográfico es determinada por la seguridad de la clave.

Si (K, M, C, e, d) es un cifrado razonable debería cumplir

1) $\forall k \in K$ y $\forall m \in M$ debería ser fácil calcular $e_k(m)$

2) $\forall k \in K$ y $\forall c \in C$ debería ser fácil calcular $d_k(m)$

3) dado $c_1, c_2, \ldots, c_n \in C$ cifrados con una clave $k \in K$, debería ser muy difícil calcular los textos planos correspondientes $d_k(c_1), \ldots, d_k(c_n)$ sin saber la clave k.

4) **difícil de acontecer** dados pares (m_1, c_1), donde (m_2, c_2)... (m_n, c_n)

m_i es el texto plano que si cifre a c_i debería ser

difícil descifrar cualquier $c \in C$ que no esté en la lista de pares

(chosen plaintext attack).

Ej: 1) Un cifrado de César: con un solo ~~texto~~ par se puede deducir la clave.

2) Un cifrado de sustitución: con un solo par se puede deducir bastante de la tabla de cifrar. (no se sabe para las letras que no ~~se~~ están en el texto plano).

3) Sea $k = (k_1, k_2)$, $\text{mcd}(k_1, p) = 1$ y

$$e_k(m) = k_1 m + k_2 \pmod p$$

¿Cuál sería $d_k(c)$?

$$d_k(c) = k_1^{-1} (c - k_2) \pmod p.$$

Supongamos que sabemos p y que (m_1, c_1) y (m_2, c_2). Tenemos que hallar $k = (k_1, k_2)$:

$c_1 = k_1 m_1 + k_2 \pmod p$ Si m_1 es primo con p, se puede resolver $k_1 = (c_1 - k_2) m_1^{-1}$

y sustituir en $k_1 m_2 + k_2 = c_2$ y encontrar k.

¿Y si no se sabe el p?

$k_1 m_1 + k_2 \equiv c_1 \pmod p$

$k_1 m_2 + k_2 \equiv c_2 \pmod p$

Se puede atacar si (4) se sabe tres pares: (m_1, c_2), (m_2, c_2) y (m_3, c_3).

Ej.

$k_1 m_1 + k_2 \equiv k_1 m_2 + k_2 \pmod p$

$\Rightarrow m_1 - m_2 \equiv 0 \pmod p$.

$\Rightarrow p$ es un divisor de $m_1 - m_2$.

Iteramos por los primos que dividen a $m_1 - m_2$.

¡Ahora procedemos como antes usando los pares (m_2, c_2) y (m_3, c_3).

¡!

~~No hemos~~ ¿Qué nos falta?

No hemos hablado de lo que quiere decir fácil ni difícil. P.ej. ~~factorizar~~ encontrar los factores primos de un número grande es difícil en general!

Por ahora: fácil: < 1 segundo en una computadora común

difícil: > año usando todas las computadoras del mundo

Sistemas de codificación:

Def: Un sistema de codificación es un método de convertir una clase de datos a otra. P. ej, letras a números.

P. ej ASCII

␣	32	00100000
A	65	01000001
a	97	01100001

etc

Usando un sistema de codificación es conveniente pensar los elementos de M de longitud fija. O sea, cada $m \in M$ es una cadena de exactamente B_m ceros y unos.

$$e_k \quad \{0,1\}^{B_m} \longrightarrow \{0,1\}^{B_c}$$

Para hacer las cosa más concreta, como $\#\{0,1\}^B = 2^B$ identificamos M con $\{0,1,\dots,2^{B_m}-1\}$. O sea

$$e_k : \{0,1,\dots,2^{B_m}-1\} \longrightarrow \{0,1,\dots,2^{B_c}-1\}$$
$$\| \qquad\qquad\qquad\qquad \|$$
$$M \qquad\qquad\qquad\qquad C$$

$$K = \{0,1,\dots,2^{B_k}-1\}$$

¿Qué tamaño debería ser B_k? (6)

- Si k es demasiado chico se puede hacer un ataque de fuerza bruta: o sea, iterar por todo $k \in K$ hasta que, usando d_k, se descifra a algo razonable.

  ↑ Kerckhoff

  Con tecnología actual si $B_k \geq 80$ el ataque es no viable.

- En la práctica para PKC hay refinamientos que nos dejan cambiar el tamaño de B_k con la raíz cuadrada de B_k. Se llaman collision attacks o meet-in-the-middle attacks y si existen para el cifrado que Alice y Bob están usando, $B_k \geq 160$.

Ejemplos de cifrados simétricos:

Ej: $K = M = C = \{1, 2, \dots, p-1\}$

$k \in K$

$e_k(m) \equiv k \cdot m \pmod p \Rightarrow d_k(c) \equiv k^{-1} c \pmod p$

Prop 2) ¿es fácil calcular k^{-1}? Sí, el algoritmo de Euclides nos dice que se calcula k^{-1} en $2 \log_2 p + 2$ pasos

Prop 3) Dada c es difícil encontrar k? Sí: la función $e_k : M \to C$ es sobreyectiva

Si p es grande $2^{159} < p < 2^{160}$ no se puede hacer fuerza bruta. Entonces dado C para calcular m sin saber k sería difícil por el c y el m determinan la clave $k \equiv m^{-1} c \pmod{p}$

Prop 4) La última equación dice que este cifrado no cumple Prop. 4.

Ej: $e_k(m) = km$ $\quad k \in \mathbb{Z}$

Prop 1 & 2) ✓

Prop 3) $c = km$ hay que factorizar (no tan difícil)

Si sabe que $c_1, c_2, \dots, c_n$,

entonces $\gcd(c_1, \dots, c_n) = \gcd(km_1, km_2, \dots, km_n)$

$= k \gcd(m_1, \dots, m_n)$

sería un múltiplo pequeño de k.

⇒ Ganamos algo importante Reduciendo módulo p

Ej: $e_k(m) \equiv m + k \pmod{p}$ César

$e_k(m) \equiv k_1 m + k_2 \pmod{p}$ afín

$e_k\begin{pmatrix} m_1 \\ m_2 \\ \vdots \\ m_n \end{pmatrix} = K\begin{pmatrix} m_1 \\ \vdots \\ m_n \end{pmatrix} + \begin{pmatrix} k_1 \\ \vdots \\ k_n \end{pmatrix}$ cifrado de Hill

No cumplen Prop 4 como ya hemos visto

Ej: Sumar más rápido que multiplicar. Pero XOR es más rápido que sumar

$\beta \in \{0,1\}$ bit $\beta' \in \{0,1\}$ otro bit $\beta \oplus \beta' = \begin{cases} 0 & \text{si } \beta = \beta' \\ 1 & \text{si } \beta \neq \beta' \end{cases}$

(sumar módulo 2)

$K = M = C = \{0,1\}^B$

$e_k(m) = k \oplus m$ y $d_k(c) = k \oplus c$

$e_k = d_k$ como funciones.

Sucesiones aleatorias de bits y cifrados simétricos

La pregunta fundamental de como crear cifrados simétricos y eficientes.

Es posible usar una clave k corta (≈ ~~128 bits~~ 100 bits) para poder mandar seguramente y eficientemente mensajes de largo arbitrario?

Una posible construcción. Construimos una función

$R: K \times \mathbb{Z} \to \{0,1\}$

que cumple

1) fácil calcular $R(k,j)$.
2) dada una sucesión de largo arbitrario y dados todos los valores $R(k,j_1), R(k,j_2), \ldots R(k,j_m)$ es difícil adivinar $R(k,j)$ para j no en la lista (con mejor de 50% chance)

Si existe tal función R, se podría usar con un one time pad

La sucesión $R(k,1), R(k,2), \ldots$, es una sucesión pseudo aleatoria porque depende de la función R.

Existen tales funciones? Después de 25 años nadie ha demostrado la existencia de tal función.

Existen varios candidatos para tal función y que esos cifrados no hayan sido rotos indica que tal vez son sucesiones pseudoaleatorias

Dos maneras de construir tal función R:

- aplicar una combinación ad hoc de funciones que mezclan el input
- construir una función R que tiene como base un problema matemático

② Cifrados asimétricos

¿Es posible intercambiar claves ~~sin~~ no personalmente?

Se usa un cifrado asimétrico (con clave pública)

K, M, C como siempre. Pero ahora

$$K = (K_{priv}, K_{pub})$$

$e_{K_{pub}}: M \to C$ ← cualquier persona puede cifrar

$d_{K_{priv}}: M C \to M$ ← solo Alice puede descifrar

Verifican la propiedad:

$$d_{k_{priv}}\left(e_{k_{pub}}(m)\right) = m \quad \forall m \in M.$$

Diffie y Hellman inventaron este esquema sin definir las funciones de cifrar y descifrar. De manera que hablamos de DH en la práctica en varias otras.