CRIPTOGRAFÍA

MAT - FIB

2022-23 Q1

1. Chacha20

Para expandir la clave de 256 bits que utiliza Chacha20 se inicializan unos estados donde la única diferencia es el valor del contador. Teniendo en cuenta que en muchas situaciones se puede saber cuáles son los primeros bytes en claro de un archivo cifrado, y por tanto cuáles son los primeros bits expandidos de la clave, es importante que a partir de ellos no se puedan inferir los siguientes bits expandidos de la clave, por lo que no hay una correlación sencilla entre los bits expandidos a partir de diferentes contadores.

`Se puede reutilizar cualquier implementación de Chacha20.`

1.1. Propagación de pequeños cambios

Con una clave K de 256 bits cualquiera haced una estadística de los bits que cambian en la salida para diferentes valores del `Counter=2,3,...,4096` comparándolos con `Counter=1`:

1. Para cada valor de `Counter=2,3,...,4096` contad el número de bits que han variado respecto a la salida con `Counter=1`. Dibujad una gráfica con las frecuencias del número total de bits que cambian.[1]
2. Para cada valor de `Counter=2,3,...,4096` anotad las posiciones de los bits que han cambiado respecto a la salida con `Counter=1`. Dibujad una gráfica con las frecuencias de las posiciones que cambian.[2]

Comentad las gráficas obtenidas y el porqué de los resultados. ¿Qué ocurriría si la primero o la segunda gráfica fueran diferentes?

1.2. Efectos de las funciones elementales

Repetid el apartado anterior con las modificaciones siguientes en cada caso:

1. Eliminad los `QUARTERROUND` de los *Column rounds*:

```
QUARTERROUND(0, 4, 8, 12)
QUARTERROUND(1, 5, 9, 13)
QUARTERROUND(2, 6, 10, 14)
QUARTERROUND(3, 7, 11, 15)
```

[1] En el eje horizontal el número r de bits que han cambiado y en el eje vertical el número de veces que r bits han cambiado.

[2] En el eje horizontal la posición i-ésima del bit que ha cambiado y en el eje vertical el número de veces que el bit i-ésimo ha cambiado.

2. Eliminad los `QUARTERROUND` de los *Diagonal rounds*:

```
QUARTERROUND(0, 5, 10, 15)
QUARTERROUND(1, 6, 11, 12)
QUARTERROUND(2, 7, 8, 13)
QUARTERROUND(3, 4, 9, 14)
```

3. Eliminad los `QUARTERROUND`:

```
QUARTERROUND(0, 4, 8, 12)
QUARTERROUND(1, 6, 11, 12)
```

Comentad las gráficas obtenidas y el porqué de los resultados.

2. El cuerpo finito $GF(2^8)$

Los elementos de este cuerpo se pueden representar por **bytes**. Los expresaremos en forma binaria, hexadecimal o polinómica, según convenga.

El byte $\mathtt{b}_7\mathtt{b}_6\mathtt{b}_5\mathtt{b}_4\mathtt{b}_3\mathtt{b}_2\mathtt{b}_1\mathtt{b}_0$ será el polinomio $b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$.

Por ejemplo, `01010111`=`0x57` será $x^6 + x^4 + x^2 + x + 1$.

Suma

La suma de dos elementos del cuerpo es la suma de polinomios binarios (coeficientes en $\mathbb{Z}_2$).

Por ejemplo, `01010111`+`10000011` será

$$(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2 = \mathtt{11010100}$$

Se corresponde con la operación XOR, que se denotará $\oplus$. El elemento neutro de la suma es `00000000=0x00` y el opuesto de cada elemento es el mismo.

Multiplicación

El producto de dos elementos del cuerpo se corresponde con el producto de polinomios binarios seguido de una reducción módulo un polinomio irreducible[3] $\mathbf{m}$ de grado 8.

Por ejemplo, si $\mathbf{m} = \mathbf{x^8} + \mathbf{x^4} + \mathbf{x^3} + \mathbf{x^2} + \mathbf{1}$[4]

$$\begin{aligned}(x^6 + x^4 + x^2 + x + 1)(x^7 + x + 1) &= x^{13} + x^{11} + x^9 + x^8 + x^7 + \\ &\quad x^7 + x^5 + x^3 + x^2 + x + \\ &\quad x^6 + x^4 + x^2 + x + 1 \\ &= x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1\end{aligned}$$

[3] Un polinomio $\mathbf{m}$ es irreducible si sus únicos divisores son 1 y $\mathbf{m}$.

[4] El polinomio que se usa en el AES es $\mathbf{x^8} + \mathbf{x^4} + \mathbf{x^3} + \mathbf{x} + \mathbf{1}$.

$$x^{13}+x^{11}+x^9+x^8+x^6+x^5+x^4+x^3+1 \pmod{x^8+x^4+x^3+x^2+1} = x^5+x^4+1.$$

El elemento neutro de la multiplicación es `00000001 = 0x01 = 1`.

En $GF(2^8)$ todo elemento diferente de `0x00` tiene inverso multiplicativo. El inverso del polinomio **a** es el único polinomio **b** tal que

$$\mathbf{a}\,\mathbf{b} \equiv 1 \mod \mathbf{m}.$$

Se puede calcular usando el algoritmo extendido de Euclides.

También podemos escribir los elementos diferentes del `0x00` como potencia de un generador. Por ejemplo, si $g = x =$ `00000010` $=$ `0x02`, entonces

$$GF(2^8) = \{g, g^2, \ldots, g^{254}, g^{255}(= g^0 = 1)\} \cup \{0\}$$

El producto de dos elementos $a = g^i$ y $b = g^j$, diferentes de `0x00`, es $ab = g^i g^j = g^{i+j}$, y el inverso de a es $a^{-1} = (g^i)^{-1} = g^{-i} = g^{255-i}$. En este caso, la multiplicación y el cálculo de inversos se reducen a la búsqueda en una tabla de 255 elementos.

Definid en **Python 3** las funciones (*El polinomio que a usar para definir las operaciones en el cuerpo es* $\mathbf{m} = \mathbf{x^8+x^7+x^3+x+1}$):

I) `irreducible_polynomials()`

entrada:
salida: lista de polinomios binarios irreducibles de grado `8` representados como enteros entre 256 ($= 2^8$) y 511 ($= 2^{8+1}-1$).

II) `GF_product_p(a, b)`

entrada: `a` y `b` elementos del cuerpo representados por enteros entre 0 y 255;
salida: un elemento del cuerpo representado por un entero entre 0 y 255 que es el producto en el cuerpo de `a` y `b` calculado usando la definición en términos de polinomios.

III) `GF_es_generador(a)`

entrada: `a` elemento del cuerpo representado por un entero entre 0 y 255;
salida: `True` si `a` es generador del cuerpo, `False` si no lo es.

IV) `GF_tables()`

entrada:
salida: dos tablas (*exponencial* y *logaritmo*), la primera tal que en la posición i tenga $a = g^i$ y la segunda tal que en la posición a tenga i tal que $a = g^i$. (g generador del cuerpo finito representado por el menor entero entre 0 y 255.)

v) `GF_product_t(a, b)`

entrada:	`a` y `b` elementos del cuerpo representados por enteros entre 0 y 255;
salida:	un elemento del cuerpo representado por un entero entre 0 y 255 que es el producto en el cuerpo de `a` y `b` calculado usando las tablas *exponencial* y *logaritmo*.

vi) `GF_invers(a)`

entrada:	`a` elemento del cuerpo representado por un entero entre 0 y 255;
salida:	`0` si `a=0x00`, inverso de `a` en el cuerpo si `a`≠`0x00` representado por un entero entre 1 y 255.

Haced tablas comparativas de los tiempos de ejecución usando las diferentes funciones:

- `GF_product_p` *vs* `GF_product_t`,
- `GF_product_p(a,0x02)` *vs* `GF_product_t(a,0x02)`,
- `GF_product_p(a,0x03)` *vs* `GF_product_t(a,0x03)`,
- `GF_product_p(a,0x09)` *vs* `GF_product_t(a,0x09)`,
- `GF_product_p(a,0x0B)` *vs* `GF_product_t(a,0x0B)`,
- `GF_product_p(a,0x0D)` *vs* `GF_product_t(a,0x0D)`,
- `GF_product_p(a,0x0E)` *vs* `GF_product_t(a,0x0E)`,

Apuntad una explicación que justifique que en algunos casos las diferencias relativas son notable y en otros no.

¡Atención! Se considerará un error grave si:

- `GF_product_p(a, b)`≠`GF_product_t(a, b)` para algún par (`a`, `b`),
- `GF_product_p(a, b)`≠`GF_product_p(b, a)` para algún par (`a`, `b`),
- `GF_product_p(a, GF_invers(a))`≠1 para `a`≠0.

3. Criptografia de clave secreta

`Podéis usar cualquier implementación del AES.`

1. En *Atenea* encontraréis el directorio AES donde hay una serie de ficheros del tipo:

   - `AES_`*`nombre.apellido_fecha`*`.enc` que son el resultado de cifrar ciertos ficheros con el AES.

   Descifrad el fichero que lleva vuestro nombre. La información necesaria para descifrarlo la encontraréis en los ficheros con igual nombre pero distintas extensiones.

2. Descifrad el fichero `AES_`*`nombre.apellido_fecha`*`.puerta_trasera.enc` que ha sido cifrado usando AES-128 (clave 128 bits) con *padding* PKCS7 y modo de operación CBC.

   La clave secreta `K` y el vector inicial `IV` se ha obtenido a partir de la información aportada por 8 participantes de forma que fuera necesario el concurso de todos ellos para recuperar `K` e `IV`:

   a) Cada participante ha elegido 2 caracteres ASCII (8 bits) entre el conjunto:

   `abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789`

   por ejemplo `a` e `y`, y ha formado su clave $K_i =$ `aaaaaaaayyyyyyyy`

   b) Se ha calculado `preMasterKey`$=K_1 \oplus K_2 \oplus \cdots \oplus K_8$ i `H=sha256(preMasterKey)`.

   c) La clave secreta `K` está formada por los primeros 128 bits de `H` y el vector inicial `IV` por los últimos 128 bits de `H`.

Referencias

- RFC 8439: ChaCha20 and Poly1305 for IETF Protocols `https://tools.ietf.org/html/rfc8439`
- Federal Information Processing Standards Publication (FIPS) 197: Advanced Encryption Standard (AES) `http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf`
- NIST Special Publication 800-38A: Recommendation for Block Cipher Modes of Operation. `http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38A.pdf`
- Padding PKCS7: section 6.3 RFC 5652. `http://tools.ietf.org/html/rfc5652#section-6.3`

Para leer

- Bruce Schneier *NSA and Bush's Illegal Eavesdropping.* (December 20, 2005)
- Schmid, Gerhard (11 July 2001). *On the existence of a global system for the interception of private and commercial communications (ECHELON interception system), (2001/2098(INI))*. European Parliament: Temporary Committee on the ECHELON Interception System.