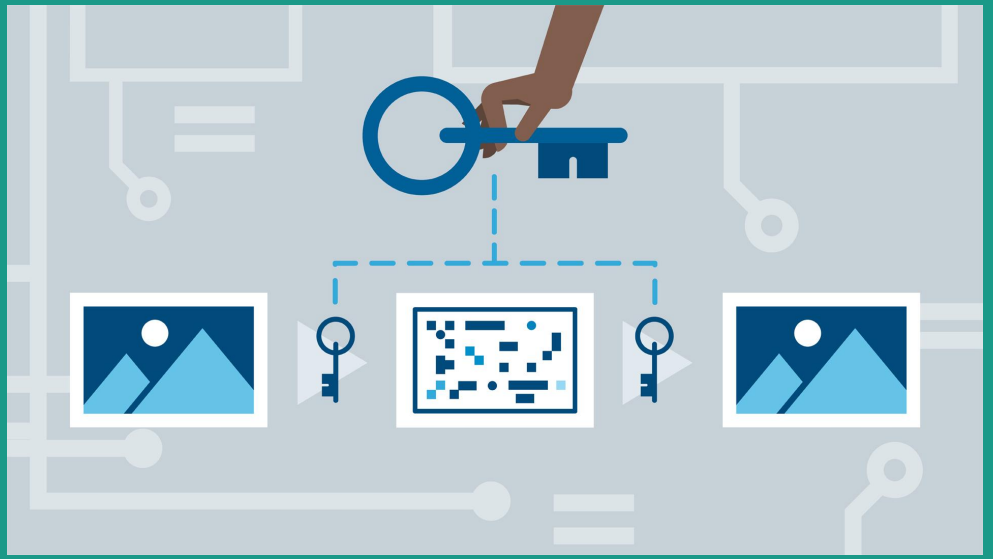
Propagating cipher block chaining (PCBC)

PARTE 1

¿Qué es?

Describa brevemente la forma en que funciona
2.
¿Es seguro usarlo? ¿Por qué?
3.
¿Se ha implementado/usado? ¿Dónde?
4.
¿Cuáles son las principales ventajas/desventajas?
5.
Agregue un diagrama del funcionamiento (recuerde citar la fuente)



¿Qué es?

Any attack has not been reported against these ciphers till date.

CIPHER	SPEED
Py & TPy	2.80
Py6 & TPy6	2.80
Pypy & TPypy	4.58
RCR-32	4.45
RCR-64	2.70
RC4	7.30

Es uno de los candidatos eSTREAM más rápidos con alrededor de 2.8 ciclos por byte en algunas plataformas. Tiene una estructura similar a RC4, pero agrega una matriz de 260 palabras de 32 bits que se indexan utilizando una permutación de bytes, y produce 64 bits en cada ronda.

¿Cómo funciona?

Py funciona en tres fases: un algoritmo de configuración clave, un algoritmo de configuración IV y una función redonda que genera dos palabras de salida cada palabra de salida tiene 4 bytes de longitud. El estado interno de Py contiene dos cajas S y, P y una variable s.

¿Cómo funciona?

Además, contiene 260 elementos, cada uno de los cuales tiene 32 bits de longitud. los los elementos de Y están indexados por [-3, -2, ..., 256]. P es una permutación de los elementos. de {0, ..., 255}. La característica principal del cifrado de flujo Py es que los S-boxes son actualizado como "arrays de arrays" todos los elementos se rotan cíclicamente en una posición.

Usos de Cipher

Basada en la investigación, se encontró que el Stream Cipher fue muy utilizado en los años 2005 hasta 2008 ya que la versión más corta llamada PY6 sustituyó a PY a pasar de tener más inseguridades (sin ser comprobado oficialmente). Además, sus uso específicos se basaban en donde se necesitara rapidez.

Algoritmo (extra)

```
//EJEMPLO DE ALGORITMO PARA CLASE DE CIFRADO DE LA INFORMACIÓN
//GRUPO 6 TEMA 3
//ALGORITMO PY DE 256

#include <stdint.h>
#define ROTL32(x, s) ((x)<<(s) | (x)>>(32-(s)))
uint8_t  *P;    // P[0] through P[255] are active
uint32_t *Y;    // Y[-3] through Y[256] are active
uint32_t s;
uint32_t *output;

while (output_words--) {
        int i = Y[185] % 256;
        P[256] = P[i];     // This effectively swaps P[0] and P[i]
        P[i] = P[0];       // Then copies P[0] to P[256]
        P++;  // Prior P[1] is new P[0], just-written P[256] is new P[255]

        s += Y[P[72]] - Y[P[239]];
        s = ROTL32(s, (P[116] + 18) % 32);

        *output++ = (ROTL32(s, 25) ^ Y[256]) + Y[P[26]];  // This line omitted from Pypy & TPypy
        *output++ = (       s      ^ Y[-1] ) + Y[P[208]];
        Y[257]    = (ROTL32(s, 14) ^ Y[-3] ) + Y[P[153]];
        Y++;  // Prior P[-2] is new P[-3], just-written P[257] is new P[256]
}
```

Referencias

T. Baign`eres, P. Junod and S. Vaudenay, “How Far Can We Go Beyond Linear Cryptanalysis?,”Asiacrypt 2004 (P. Lee, ed.), vol. 3329 of LNCS, pp. 432–450, Springer-Verlag, 2004

E. Biham, J. Seberry, “Py (Roo): A Fast and Secure Stream Cipher using Rolling, Arrays,” eSTREAM, ECRYPT Stream Cipher Project, Report 2005/023, 2005.

Daniel. J. Bernstein, “Comparison of 256-bit stream ciphers at the beginning of

2006,” Workshop Record of SASC 2006 – Stream Ciphers Revisited, ECRYPT Network of Excellence in Cryptology, pp. 70–83

Gracias.