Stream Ciphers

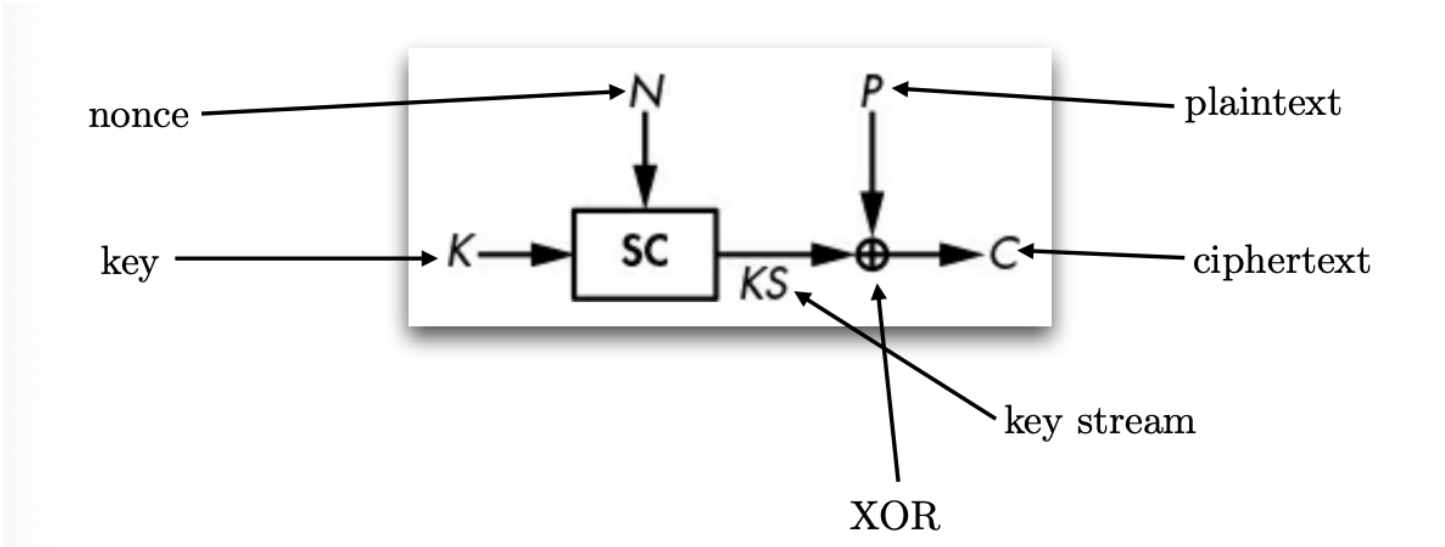
Teórica #4 de Criptografia Aplicada

Como funcionam Stream Ciphers?

Stream Ciphers usam **Geradores de Bits Aleatórios Determinístcos (DRBG)** em vez de **Geradores de Pseudo Bits Aleatórios (PRBG)**, dado que é a propriedade **deterministica** do RBG (Random Bit Generator) que permite a correta decriptação dos dados.

PRBG => Permite encriptar mas não decriptar

DRBG => Permite encriptar e decriptar



A figura em cima representa o funcionamento geral de uma stream cipher. Tipicamente esta recebe uma chave (K) de **128** ou **256** bits e um nonce (N) também com comprimento entre **64** e **128** bits.

Ao contrario da chave, o nonce não tem que ser secreto mas deve ser único para cada chave. Encriptando duas mensagens com a mesma chave e nonce gera problemas pois, mesmo as mensagens sendo diferentes, a keystream produzida é igual.

```
KS1 = SC(K1, N1) (keystream)
C1 = KS1 XOR P1 (encriptar)
P1 = KS1 XOR P1 (decriptar)
```

Se encriptarmos duas mensagens com a mesma keystream (mesma chave com a mesmo nonce), o seguinte acontece:

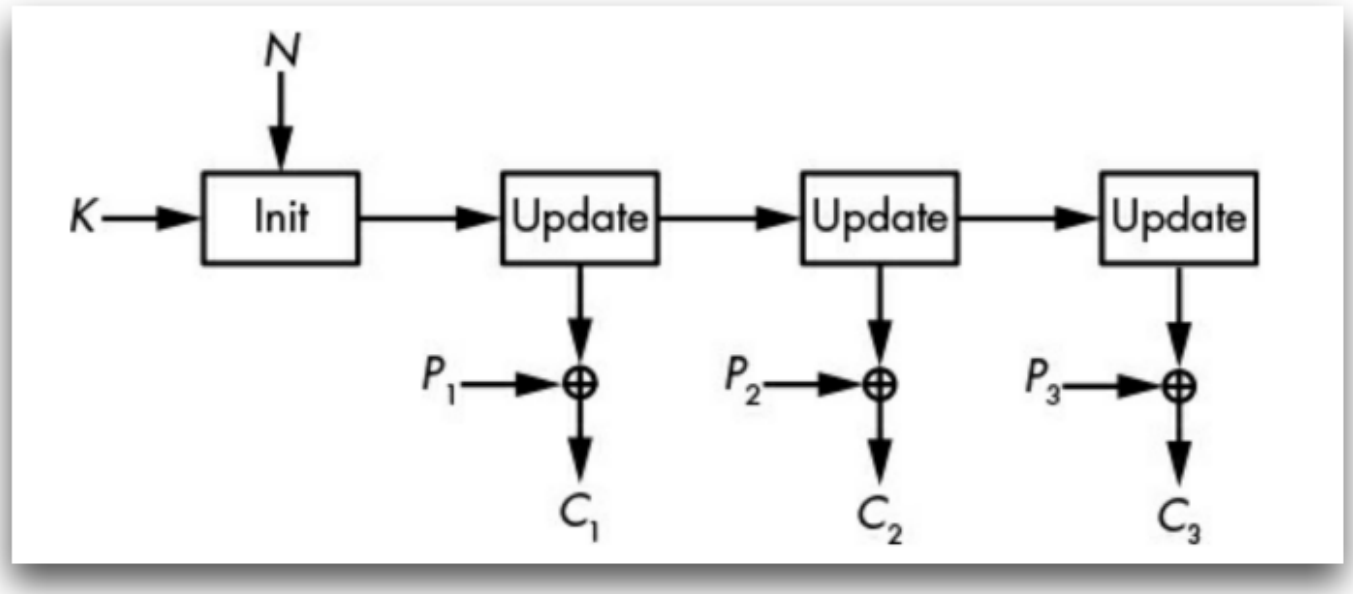
```
C1 = KS XOR P1
C2 = KS XOR P2

Obtendo o plain text P1, é possivel obter a keystream usada
```

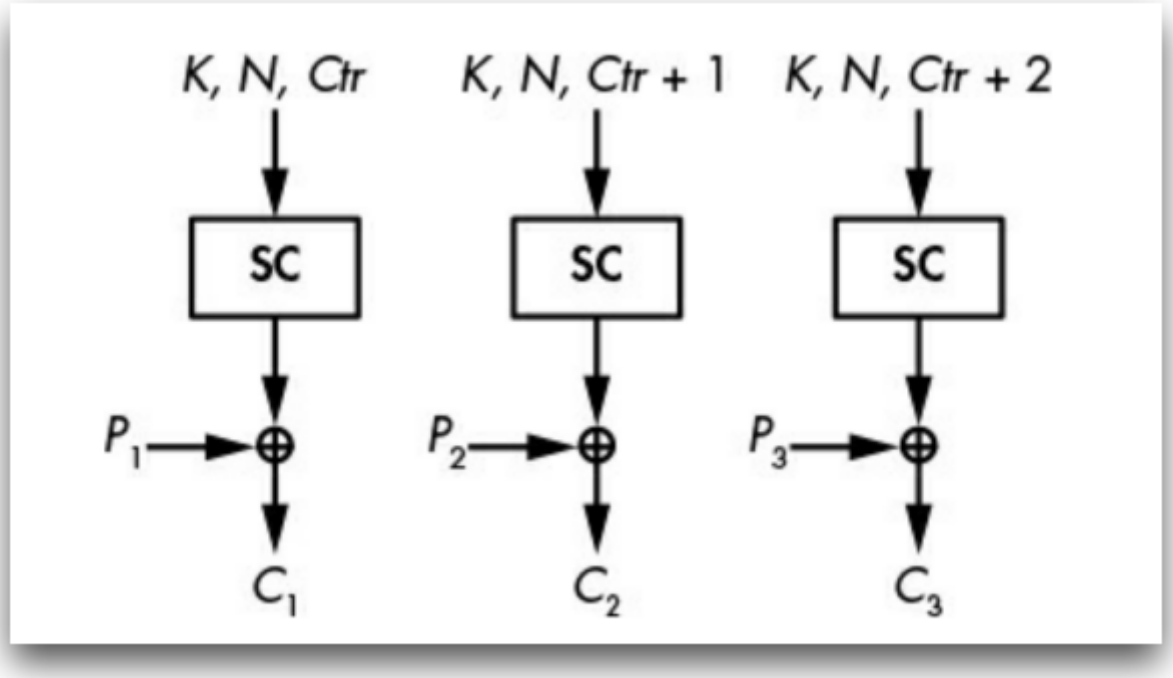
```
KS = C1 XOR P1

E tendo a keystream…

C1 XOR C2 XOR P1 = KS XOR P1 XOR KS XOR C2 XOR P1
                 = (KS XOR KS) XOR (P1 XOR P1) XOR P2
                 = P2
```

Stream Ciphers Stateful e Counter-Based

Numa perspetiva ampla, podemos categorizar as stream ciphers em dois modos: stateful e counter based:

- Stateful stream ciphers partilham um estado interno que vai sendo atualizado ao longo da geração da keystream (existe apenas uma stream cipher para todo o processo de cifragem, podendo-se se assemelhar o estado a uma keystream) (figura 2);
- Counter-Based stream ciphers produzem *chunks* da keystream a partir de uma chave, nonce e valor de contador, não partilhando nenhum estado durante a geração da keystream (figura 3).





Dependendo da plataforma alvo onde a cifra se decorre, existem dois paradigmas de stream ciphers: **Hardware-Oriented** e **Software-Oriented**.

O que são stream ciphers orientadas a hardware?

Implementações orientadas a hardware são representadas por circuitos eletrónicos que implementam o algoritmo da stream cipher ao bit level, não sendo possível usar o mesmo circuito para mais nada (dedicated hardware).

O que são Feedback Shift Registers (FSR)?

FSR é um array de bits que tem associado uma função de *update feedback*. O estado do FSR é armazenado no array ou num registo e é atualizado com base na função de feedback, produzindo um bit. O calculo do estado funciona com shifts à esquerda e OR lógicos:

```
Sendo Rt o estado do FSR a um determinado tempo *t* e f(Rt) a função de feedback:

Rt + 1 = (Rt << 1) | f(Rt)
```

O **periodo** do FSR é dado pelo numero de valores únicos capazes de serem gerados até que o valor do estado seja igual ao estado inicial. Quanto **maior** o período, mais seguro é o FSR.

O que são Linear Feedback Shift Registers (LFSRs)?

LFSRs são FSRs que utilizam um função de feeback linear. A escolha dos bits para o LFSR é crucial para o seu periodo e por consequência a sua segurança.

Considerando *n* o número de bits e o polinómio

`1 + x + x^2 + … + x^n`

então, o periodo é máximo se o polinómio for primitivo.

LFSRs são inseguros porque estes são lineares. Se o atacante conhece o tamanho *n* de bits, então consegue recuperar o estado inicial.

O que são filtered LFSRs

LFSRs, só que antes de produzir os bits de output, aplica-se uma função não linear `g`.

O que são Nonlinear FSRs?

LFSRs não lineares, isto é, tem como função e feedback uma função não linear.

NAO ENTENDI O QUE É O GRAIN-128

NAO ENTENDI O QUE É O A5/1

O que são stream ciphers orientadas ao Software?

Stream ciphers orientadas ao Software tiram proveito da capacidade do hardware embebido e da abstração entre as camadas software-hardware. Não estão restritos ao bit level, podendo especificar chaves e nonces em bytes.

RC4

RC4 é a stream cipher com mais anos de uso ao nível do software, sendo usado no WEP (Wireless Equivalent Privacy) e no protocolo TLS para estabelecer conexões HTTPS.

Como funciona o RC4?

(0). O estado interno do RC4 é representado por um array de 256 posições, contendo um byte incrementado de 0 a 255. ([0] = 0, [1] = 1, … [255] = 255).

(1). Uma vez estabelecido o array, este é inicializado através de um *KSA* (Key Scheduling Algorithm), que altera a posição de cada um dos elementos:

```
j=0
S = range(256)
for i in range(256):
    j = (j + S[i] + K[i % n]) % 256
    S[i], S[j] = S[j], S[i]
```

(2). De seguida, usando o estado inicial `S` previamente inicializado, é gerado uma keystream KS, **com o mesmo tamanho do plaintext** P, e calcula a cipher text C através de uma operação XOR.

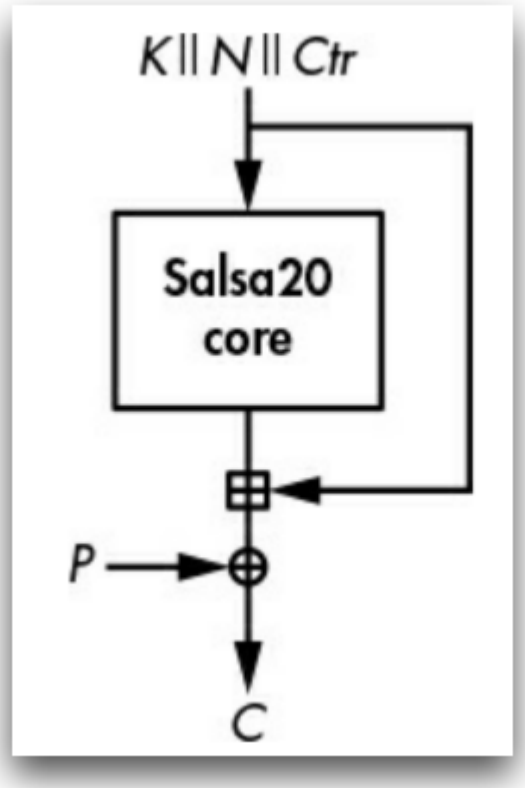
```
i=0
j=0
```

```
    for b in range(m):
        i = (i + 1) % 256
        j = (j + S[i]) % 256
        S[i], S[j] = S[j], S[i]
        KS[b] = S[(S[i] + S[j]) % 256]

    C = P XOR KS
```

O que é o Salsa20?

Stream cipher orientada a software, otimizada para CPUs modernos. Esta está implementada em diversos protocolos e bibliotecas.

É uma counter-based stream cipher que gera a sua keystream ao processar repetidamente um contador, que é incrementado para cada bloco a ser cifrado.



(1). Dado uma chave (K) (256-bits), nonce (N) e o valor do contador (Ctr), é criado um bloco de 512-bits. A função Salsa20 core transforma esse bloco em um bloco aleatório.

(2). De seguida, mistura-se o valor original do bloco ao bloco transformado para se obter a keystream do bloco.

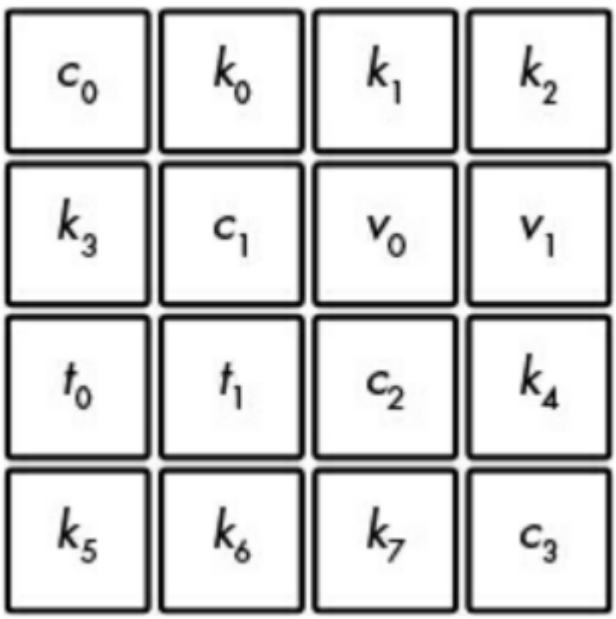
(3). Por final, aplica-se a keystream na operação XOR com o plain text e obtém-se o text encriptado.

C = P XOR KS

A função/algoritmo Salsa20 core utiliza uma função chamada *Quarter-Round (QR)+ para transformar quatro 32-bit words (4 bytes), a, b, c e d.

```
b = b XOR ((a + d) <<< 7)
c = c XOR ((b + a) <<< 9)
d = d XOR ((c + b) <<< 13)
a = a XOR ((d + c) <<< 18)
```

O estado inicial é representado então pelo bloco de 512-bits, dividido por 16 parcelas de 32 bits cada.



Usando a função QR, o Salsa20 core aplica a mesma primeiramente nas quatro colunas do estado e de seguida nas quatro linhas do estado, criando o processo de nome ***double-round***.

O Salsa20 aplica então o double-round 10 vezes, ao longo de 20 rondas (200 double-ronda) (daí o 20 em Salsa20).

Apenas após 4 rondas, uma alteração única propaga alteração em todos os 512 bits, dando-nos a propriedade criptográfica de *full difusion*.

O que pode correr mal em stream ciphers?

- Fragilidade dos algoritmos;
- Designs inseguros;
- Implementação incorretas;
- Reutilização do nonce (erro mais comum!).