# SALSA20

implementação

Yah! (USP)
P. Matias

# Criptologia

Estudo da codificação ou decodificação de mensagens e sinais.

# Criptografia por chave simétrica

Algoritmos que devem utilizar a mesma chave na codificação e decodificação de mensagens.

# Salsa20

Algorítmo proposto pelo matemático *S. Bernstein* em 2005, como alternativa ao algortimo AES. Funciona por meio de rotações binárias, muito rápidas de serem executadas pelo computador, executadas em 20 etapas (rounds).

# Codificação e Decodificação no Salsa-20

Para **CODIFICAR**:

**Mensagem Cifrada** = **Mensagem** **XOR** ***Z***

Para **DECODIFICAR**:

**Mensagem** = **Mensagem Cifrada** **XOR** ***Z***

# Codificação e Decodificação no Salsa-20

Para **CODIFICAR**:

**Mensagem Cifrada[i]** = **Mensagem[i]** **XOR** ***Z[i]***

Para **DECODIFICAR**:

**Mensagem[i]** = **Mensagem Cifrada[i]** **XOR** ***Z[i]***

**Operações com *i-ésimo* elemento!**

# Tabela Verdade XOR

| A | B | XOR AB |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

# Cálculo de Z

$$Z = X + DR(X)$$

# Cálculo de Z

***Z = X + DR(X)***

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix}$$

# Cálculo de Z

***Z = X + DR(X)***

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix}$$

***Double-Round***

# Double Round

***Execução de dois Quarter-Rounds (linhas e colunas) !***

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix}$$

$$\begin{cases} QR(x_0, x_4, x_8, x_{12}) \\ QR(x_5, x_9, x_{13}, x_1) \\ QR(x_{10}, x_{14}, x_2, x_6) \\ QR(x_{15}, x_3, x_7, x_{11}) \end{cases} ; \begin{cases} QR(x_0, x_1, x_2, x_3) \\ QR(x_5, x_6, x_7, x_4) \\ QR(x_{10}, x_{11}, x_8, x_9) \\ QR(x_{15}, x_{12}, x_{13}, x_{14}) \end{cases}$$

# Double Round

***Quarter-round executado nas colunas!***

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix}$$

$$\begin{cases} QR(x_0, x_4, x_8, x_{12}) \\ QR(x_5, x_9, x_{13}, x_1) \\ QR(x_{10}, x_{14}, x_2, x_6) \\ QR(x_{15}, x_3, x_7, x_{11}) \end{cases} ; \begin{cases} QR(x_0, x_1, x_2, x_3) \\ QR(x_5, x_6, x_7, x_4) \\ QR(x_{10}, x_{11}, x_8, x_9) \\ QR(x_{15}, x_{12}, x_{13}, x_{14}) \end{cases}$$

# Double Round

***Quarter-round executado nas colunas!***

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix}$$

$$\begin{cases} QR(x_0, x_4, x_8, x_{12}) \\ QR(x_5, x_9, x_{13}, x_1) \\ QR(x_{10}, x_{14}, x_2, x_6) \\ QR(x_{15}, x_3, x_7, x_{11}) \end{cases} ; \begin{cases} QR(x_0, x_1, x_2, x_3) \\ QR(x_5, x_6, x_7, x_4) \\ QR(x_{10}, x_{11}, x_8, x_9) \\ QR(x_{15}, x_{12}, x_{13}, x_{14}) \end{cases}$$

# Double Round

***Quarter-round executado nas colunas!***

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix}$$

$$\begin{cases} QR(x_0, x_4, x_8, x_{12}) \\ QR(x_5, x_9, x_{13}, x_1) \\ QR(x_{10}, x_{14}, x_2, x_6) \\ QR(x_{15}, x_3, x_7, x_{11}) \end{cases} ; \begin{cases} QR(x_0, x_1, x_2, x_3) \\ QR(x_5, x_6, x_7, x_4) \\ QR(x_{10}, x_{11}, x_8, x_9) \\ QR(x_{15}, x_{12}, x_{13}, x_{14}) \end{cases}$$

# Double Round

***Quarter-round executado nas colunas!***

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix}$$

$$\begin{cases} QR(x_0, x_4, x_8, x_{12}) \\ QR(x_5, x_9, x_{13}, x_1) \\ QR(x_{10}, x_{14}, x_2, x_6) \\ QR(x_{15}, x_3, x_7, x_{11}) \end{cases} ; \begin{cases} QR(x_0, x_1, x_2, x_3) \\ QR(x_5, x_6, x_7, x_4) \\ QR(x_{10}, x_{11}, x_8, x_9) \\ QR(x_{15}, x_{12}, x_{13}, x_{14}) \end{cases}$$

# Double Round

***Quarter-round executado nas linhas!***

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix}$$

$$\begin{cases} QR(x_0, x_4, x_8, x_{12}) \\ QR(x_5, x_9, x_{13}, x_1) \\ QR(x_{10}, x_{14}, x_2, x_6) \\ QR(x_{15}, x_3, x_7, x_{11}) \end{cases} ; \begin{cases} QR(x_0, x_1, x_2, x_3) \\ QR(x_5, x_6, x_7, x_4) \\ QR(x_{10}, x_{11}, x_8, x_9) \\ QR(x_{15}, x_{12}, x_{13}, x_{14}) \end{cases}$$

# Double Round

***Quarter-round executado nas linhas!***

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix}$$

$$\begin{cases} QR(x_0, x_4, x_8, x_{12}) \\ QR(x_5, x_9, x_{13}, x_1) \\ QR(x_{10}, x_{14}, x_2, x_6) \\ QR(x_{15}, x_3, x_7, x_{11}) \end{cases} ; \begin{cases} QR(x_0, x_1, x_2, x_3) \\ QR(x_5, x_6, x_7, x_4) \\ QR(x_{10}, x_{11}, x_8, x_9) \\ QR(x_{15}, x_{12}, x_{13}, x_{14}) \end{cases}$$

# Double Round

***Quarter-round executado nas linhas!***

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix}$$

$$\begin{cases} QR(x_0, x_4, x_8, x_{12}) \\ QR(x_5, x_9, x_{13}, x_1) \\ QR(x_{10}, x_{14}, x_2, x_6) \\ QR(x_{15}, x_3, x_7, x_{11}) \end{cases} ; \begin{cases} QR(x_0, x_1, x_2, x_3) \\ QR(x_5, x_6, x_7, x_4) \\ QR(x_{10}, x_{11}, x_8, x_9) \\ QR(x_{15}, x_{12}, x_{13}, x_{14}) \end{cases}$$

# Double Round

***Quarter-round executado nas linhas!***

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix}$$

$$\begin{cases} QR(x_0, x_4, x_8, x_{12}) \\ QR(x_5, x_9, x_{13}, x_1) \\ QR(x_{10}, x_{14}, x_2, x_6) \\ QR(x_{15}, x_3, x_7, x_{11}) \end{cases} ; \begin{cases} QR(x_0, x_1, x_2, x_3) \\ QR(x_5, x_6, x_7, x_4) \\ QR(x_{10}, x_{11}, x_8, x_9) \\ QR(x_{15}, x_{12}, x_{13}, x_{14}) \end{cases}$$

# Quarter Round

***QR(a, b, c, d)***

$$
\begin{aligned}
b &= b \oplus [(a+d) <<< 7], \\
c &= c \oplus [(b+a) <<< 9], \\
d &= d \oplus [(c+b) <<< 13], \\
a &= a \oplus [(d+c) <<< 18]
\end{aligned}
$$

# Quarter Round

***QR(a, b, c, d)***

$$
\begin{aligned}
b &= b \oplus [(a+d) <<< 7], \\
c &= c \oplus [(b+a) <<< 9], \\
d &= d \oplus [(c+b) <<< 13], \\
a &= a \oplus [(d+c) <<< 18]
\end{aligned}
$$

***Rotações!***

# Bloco de Codificação X

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} \phi_0 & k_0 & k_1 & k_2 \\ k_3 & \phi_1 & n_0 & n_1 \\ t_0 & t_1 & \phi_2 & k_4 \\ k_5 & k_6 & k_7 & \phi_3 \end{bmatrix}$$

# Cada elemento tem 32 bits

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} \phi_0 & k_0 & k_1 & k_2 \\ k_3 & \phi_1 & n_0 & n_1 \\ t_0 & t_1 & \phi_2 & k_4 \\ k_5 & k_6 & k_7 & \phi_3 \end{bmatrix}$$

# Cada elemento tem 32 bits

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} \phi_0 & k_0 & k_1 & k_2 \\ k_3 & \phi_1 & n_0 & n_1 \\ t_0 & t_1 & \phi_2 & k_4 \\ k_5 & k_6 & k_7 & \phi_3 \end{bmatrix}$$

x[0] = 0x00000000

# Cada elemento tem 32 bits

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} \phi_0 & k_0 & k_1 & k_2 \\ k_3 & \phi_1 & n_0 & n_1 \\ t_0 & t_1 & \phi_2 & k_4 \\ k_5 & k_6 & k_7 & \phi_3 \end{bmatrix}$$

x[0] = 0x000000**00**

Primeiro Byte!
(menos significativo)

# Cada elemento tem 32 bits

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} \phi_0 & k_0 & k_1 & k_2 \\ k_3 & \phi_1 & n_0 & n_1 \\ t_0 & t_1 & \phi_2 & k_4 \\ k_5 & k_6 & k_7 & \phi_3 \end{bmatrix}$$

x[0] = 0x0000**00**00

Segundo Byte!

# Cada elemento tem 32 bits

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} \phi_0 & k_0 & k_1 & k_2 \\ k_3 & \phi_1 & n_0 & n_1 \\ t_0 & t_1 & \phi_2 & k_4 \\ k_5 & k_6 & k_7 & \phi_3 \end{bmatrix}$$

x[0] = 0x00**00**0000

Terceiro Byte!

# Cada elemento tem 32 bits

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} \phi_0 & k_0 & k_1 & k_2 \\ k_3 & \phi_1 & n_0 & n_1 \\ t_0 & t_1 & \phi_2 & k_4 \\ k_5 & k_6 & k_7 & \phi_3 \end{bmatrix}$$

x[0] = 0x**00**000000

Quarto Byte!
(mais significativo)

# Bloco de Codificação X com 512 bits (16 x 32 bits)

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} \phi_0 & k_0 & k_1 & k_2 \\ k_3 & \phi_1 & n_0 & n_1 \\ t_0 & t_1 & \phi_2 & k_4 \\ k_5 & k_6 & k_7 & \phi_3 \end{bmatrix}$$

# Constantes

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} \phi_0 & k_0 & k_1 & k_2 \\ k_3 & \phi_1 & n_0 & n_1 \\ t_0 & t_1 & \phi_2 & k_4 \\ k_5 & k_6 & k_7 & \phi_3 \end{bmatrix}$$

# Nounce (número aleatório)

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} \phi_0 & k_0 & k_1 & k_2 \\ k_3 & \phi_1 & n_0 & n_1 \\ t_0 & t_1 & \phi_2 & k_4 \\ k_5 & k_6 & k_7 & \phi_3 \end{bmatrix}$$

# Contador sequêncial de bloco

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} \phi_0 & k_0 & k_1 & k_2 \\ k_3 & \phi_1 & n_0 & n_1 \\ t_0 & t_1 & \phi_2 & k_4 \\ k_5 & k_6 & k_7 & \phi_3 \end{bmatrix}$$

# Chave secreta

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} \phi_0 & k_0 & k_1 & k_2 \\ k_3 & \phi_1 & n_0 & n_1 \\ t_0 & t_1 & \phi_2 & k_4 \\ k_5 & k_6 & k_7 & \phi_3 \end{bmatrix}$$

# Implementação

# Implementação

| A | B | XOR AB |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

```
X = A ^ B;
```

```
X = X ^ Y    ->    X ^= Y
```

# Implementação

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} \phi_0 & k_0 & k_1 & k_2 \\ k_3 & \phi_1 & n_0 & n_1 \\ t_0 & t_1 & \phi_2 & k_4 \\ k_5 & k_6 & k_7 & \phi_3 \end{bmatrix}$$

# Implementação

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} \phi_0 & k_0 & k_1 & k_2 \\ k_3 & \phi_1 & n_0 & n_1 \\ t_0 & t_1 & \phi_2 & k_4 \\ k_5 & k_6 & k_7 & \phi_3 \end{bmatrix}$$

X[0] = Phi[0];
X[5] = Phi[1];
X[10]= Phi[2];
X[15]= Phi[3];

# Implementação

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} \phi_0 & k_0 & k_1 & k_2 \\ k_3 & \phi_1 & n_0 & n_1 \\ t_0 & t_1 & \phi_2 & k_4 \\ k_5 & k_6 & k_7 & \phi_3 \end{bmatrix}$$

X[0] = **0x61707865***;
X[5] = Phi[1];
X[10]= Phi[2];
X[15]= Phi[3];

*especificação do algoritmo!

# Implementação

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} \phi_0 & k_0 & k_1 & k_2 \\ k_3 & \phi_1 & n_0 & n_1 \\ t_0 & t_1 & \phi_2 & k_4 \\ k_5 & k_6 & k_7 & \phi_3 \end{bmatrix}$$

X[0] = 0x61707865;
X[5] = **0x3320646e**;
X[10]= Phi[2];
X[15]= Phi[3];

# Implementação

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} \phi_0 & k_0 & k_1 & k_2 \\ k_3 & \phi_1 & n_0 & n_1 \\ t_0 & t_1 & \phi_2 & k_4 \\ k_5 & k_6 & k_7 & \phi_3 \end{bmatrix}$$

X[0] = 0x61707865;
X[5] = 0x3320646e;
X[10]= **0x79622d32**;
X[15]= Phi[3];

# Implementação

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} \phi_0 & k_0 & k_1 & k_2 \\ k_3 & \phi_1 & n_0 & n_1 \\ t_0 & t_1 & \phi_2 & k_4 \\ k_5 & k_6 & k_7 & \phi_3 \end{bmatrix}$$

X[0] = 0x61707865;
X[5] = 0x3320646e;
X[10]= 0x79622d32;
X[15]= **0x6b206574**;

# Implementação

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} \phi_0 & k_0 & k_1 & k_2 \\ k_3 & \phi_1 & n_0 & n_1 \\ t_0 & t_1 & \phi_2 & k_4 \\ k_5 & k_6 & k_7 & \phi_3 \end{bmatrix}$$

X[0] = 0x61707865; t[0] = 0;
X[5] = 0x3320646e;
X[10]= 0x79622d32; t[1] = 0;
X[15]= 0x6b206574;

# Implementação

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} \phi_0 & k_0 & k_1 & k_2 \\ k_3 & \phi_1 & n_0 & n_1 \\ t_0 & t_1 & \phi_2 & k_4 \\ k_5 & k_6 & k_7 & \phi_3 \end{bmatrix}$$

X[0] = 0x61707865; t[0] = 0; k[0] = 0x04030201; k[4] =
X[5] = 0x3320646e; k[1] = k[5] =
X[10]= 0x79622d32; t[1] = 0; k[2] = k[6] =
X[15]= 0x6b206574; k[3] = k[7] =

# Implementação

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} \phi_0 & k_0 & k_1 & k_2 \\ k_3 & \phi_1 & n_0 & n_1 \\ t_0 & t_1 & \phi_2 & k_4 \\ k_5 & k_6 & k_7 & \phi_3 \end{bmatrix}$$

X[0] = 0x61707865; t[0] = 0; k[0] = 0x0403020**01**; k[4] =
X[5] = 0x3320646e; k[1] = k[5] =
X[10]= 0x79622d32; t[1] = 0; k[2] = k[6] =
X[15]= 0x6b206574; k[3] = k[7] =

# Implementação

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} \phi_0 & k_0 & k_1 & k_2 \\ k_3 & \phi_1 & n_0 & n_1 \\ t_0 & t_1 & \phi_2 & k_4 \\ k_5 & k_6 & k_7 & \phi_3 \end{bmatrix}$$

X[0] = 0x61707865; t[0] = 0; k[0] = 0x04030201; k[4] =
X[5] = 0x3320646e; k[1] = k[5] =
X[10]= 0x79622d32; t[1] = 0; k[2] = k[6] =
X[15]= 0x6b206574; k[3] = k[7] =

# Implementação

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} \phi_0 & k_0 & k_1 & k_2 \\ k_3 & \phi_1 & n_0 & n_1 \\ t_0 & t_1 & \phi_2 & k_4 \\ k_5 & k_6 & k_7 & \phi_3 \end{bmatrix}$$

X[0] = 0x61707865; t[0] = 0; k[0] = 0x04**03**0201; k[4] =
X[5] = 0x3320646e; k[1] = k[5] =
X[10]= 0x79622d32; t[1] = 0; k[2] = k[6] =
X[15]= 0x6b206574; k[3] = k[7] =

# Implementação

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} \phi_0 & k_0 & k_1 & k_2 \\ k_3 & \phi_1 & n_0 & n_1 \\ t_0 & t_1 & \phi_2 & k_4 \\ k_5 & k_6 & k_7 & \phi_3 \end{bmatrix}$$

X[0] = 0x61707865; t[0] = 0; k[0] = 0x**04**030201; k[4] =
X[5] = 0x3320646e; k[1] = k[5] =
X[10]= 0x79622d32; t[1] = 0; k[2] = k[6] =
X[15]= 0x6b206574; k[3] = k[7] =

# Implementação

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} \phi_0 & k_0 & k_1 & k_2 \\ k_3 & \phi_1 & n_0 & n_1 \\ t_0 & t_1 & \phi_2 & k_4 \\ k_5 & k_6 & k_7 & \phi_3 \end{bmatrix}$$

X[0] = 0x61707865; t[0] = 0; k[0] = 0x04030201; k[4] =
X[5] = 0x3320646e; k[1] = 0x08070605; k[5] =
X[10]= 0x79622d32; t[1] = 0; k[2] = k[6] =
X[15]= 0x6b206574; k[3] = k[7] =

# Implementação

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} \phi_0 & k_0 & k_1 & k_2 \\ k_3 & \phi_1 & n_0 & n_1 \\ t_0 & t_1 & \phi_2 & k_4 \\ k_5 & k_6 & k_7 & \phi_3 \end{bmatrix}$$

X[0] = 0x61707865;
X[5] = 0x3320646e;
X[10]= 0x79622d32;
X[15]= 0x6b206574;

t[0] = 0;
t[1] = 0;

k[0] = 0x04030201;
k[1] = 0x08070605;
k[2] = 0x0c0b0a09;
k[3] =

k[4] =
k[5] =
k[6] =
k[7] =

# Implementação

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} \phi_0 & k_0 & k_1 & k_2 \\ k_3 & \phi_1 & n_0 & n_1 \\ t_0 & t_1 & \phi_2 & k_4 \\ k_5 & k_6 & k_7 & \phi_3 \end{bmatrix}$$

X[0] = 0x61707865; t[0] = 0; k[0] = 0x04030201; k[4] =
X[5] = 0x3320646e; k[1] = 0x08070605; k[5] =
X[10]= 0x79622d32; t[1] = 0; k[2] = 0x0c0b0a09; k[6] =
X[15]= 0x6b206574; k[3] = 0x100f0e0d; k[7] =

# Implementação

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} \phi_0 & k_0 & k_1 & k_2 \\ k_3 & \phi_1 & n_0 & n_1 \\ t_0 & t_1 & \phi_2 & k_4 \\ k_5 & k_6 & k_7 & \phi_3 \end{bmatrix}$$

X[0] = 0x61707865;
X[5] = 0x3320646e;
X[10]= 0x79622d32;
X[15]= 0x6b206574;

t[0] = 0;
t[1] = 0;

k[0] = 0x04030201;
k[1] = 0x08070605;
k[2] = 0x0c0b0a09;
k[3] = 0x100f0e0d;

k[4] = 0x14131211;
k[5] =
k[6] =
k[7] =

# Implementação

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} \phi_0 & k_0 & k_1 & k_2 \\ k_3 & \phi_1 & n_0 & n_1 \\ t_0 & t_1 & \phi_2 & k_4 \\ k_5 & k_6 & k_7 & \phi_3 \end{bmatrix}$$

X[0] = 0x61707865;
X[5] = 0x3320646e;
X[10]= 0x79622d32;
X[15]= 0x6b206574;

t[0] = 0;
t[1] = 0;

k[0] = 0x04030201;
k[1] = 0x08070605;
k[2] = 0x0c0b0a09;
k[3] = 0x100f0e0d;

k[4] = 0x14131211;
k[5] = 0x18171615;
k[6] =
k[7] =

# Implementação

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} \phi_0 & k_0 & k_1 & k_2 \\ k_3 & \phi_1 & n_0 & n_1 \\ t_0 & t_1 & \phi_2 & k_4 \\ k_5 & k_6 & k_7 & \phi_3 \end{bmatrix}$$

X[0] = 0x61707865;
X[5] = 0x3320646e;
X[10]= 0x79622d32;
X[15]= 0x6b206574;

t[0] = 0;
t[1] = 0;

k[0] = 0x04030201;
k[1] = 0x08070605;
k[2] = 0x0c0b0a09;
k[3] = 0x100f0e0d;

k[4] = 0x14131211;
k[5] = 0x18171615;
k[6] = 0x1c1b1a19;
k[7] =

# Implementação

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} \phi_0 & k_0 & k_1 & k_2 \\ k_3 & \phi_1 & n_0 & n_1 \\ t_0 & t_1 & \phi_2 & k_4 \\ k_5 & k_6 & k_7 & \phi_3 \end{bmatrix}$$

X[0] = 0x61707865;
X[5] = 0x3320646e;
X[10]= 0x79622d32;
X[15]= 0x6b206574;

t[0] = 0;
t[1] = 0;

k[0] = 0x04030201;
k[1] = 0x08070605;
k[2] = 0x0c0b0a09;
k[3] = 0x100f0e0d;
k[4] = 0x14131211;
k[5] = 0x18171615;
k[6] = 0x1c1b1a19;
k[7] = 0x201f1e1d;

# Implementação

***rotate()***

$$a = a \oplus [(d + c) <<< 18]$$

```
unsigned long rotate(unsigned long x, int n) {
    return (x << n);
}
```

# Implementação

***rotate()***

$$a = a \oplus [(d + c) <<< 18]$$

```
unsigned long rotate(unsigned long x, int n) {
    return (x << n);
}
```

# Implementação

***rotate()***

$$a = a \oplus [(d + c) <<< 18]$$

```
unsigned long rotate(unsigned long x, int n) {
    return (x << n);
}
```

# Implementação

***quarterround()***

$$a = a \oplus [(d + c) <<< 18]$$

```
void step(uint32_t *s, int i, int j, int k, int r) {
   s[i] ^= rotate(s[j] + s[k], r);
}

void quarterround(uint32_t *s, int i0, int i1, int i2, int i3) {
  step(s, i1, i0, i3, 7);
  step(s, i2, i1, i0, 9);
  step(s, i3, i2, i1, 13);
  step(s, i0, i3, i2, 18);
}
```

# Implementação

***quarterround()***

$$a = a \oplus [(d + c) <<< 18]$$

```
void step(uint32_t *s, int i, int j, int k, int r) {
   s[i] ^= rotate(s[j] + s[k], r);
}

void quarterround(uint32_t *s, int i0, int i1, int i2, int i3) {
  step(s, i1, i0, i3, 7);
  step(s, i2, i1, i0, 9);
  step(s, i3, i2, i1, 13);
  step(s, i0, i3, i2, 18);
}
```

# Implementação

## *quarterround()*

$$a = a \oplus [(d + c) <<< 18]$$

```
void step(uint32_t *s, int i, int j, int k, int r) {
    s[i] ^= rotate(s[j] + s[k], r);
}

void quarterround(uint32_t *s, int i0, int i1, int i2, int i3) {
  step(s, i1, i0, i3, 7);
  step(s, i2, i1, i0, 9);
  step(s, i3, i2, i1, 13);
  step(s, i0, i3, i2, 18);
}
```

# Implementação

***quarterround()***

$$a = a \oplus [(d + c) <<< 18]$$

```
void step(uint32_t *s, int i, int j, int k, int r) {
    s[i] ^= rotate(s[j] + s[k], r);
}

void quarterround(uint32_t *s, int i0, int i1, int i2, int i3) {
  step(s, i1, i0, i3, 7);
  step(s, i2, i1, i0, 9);
  step(s, i3, i2, i1, 13);
  step(s, i0, i3, i2, 18);
}
```

# Implementação

***quarterround()***

$$a = a \oplus [(d + c) <<< 18]$$

```
void step(uint32_t *s, int i, int j, int k, int r) {
    s[i] ^= rotate(s[j] + s[k], r);
}

void quarterround(uint32_t *s, int i0, int i1, int i2, int i3) {
  step(s, i1, i0, i3, 7);
  step(s, i2, i1, i0, 9);
  step(s, i3, i2, i1, 13);
  step(s, i0, i3, i2, 18);
}
```

# Implementação

## *quarterround()*

$$a = a \oplus [(d + c) <<< 18]$$

```
void step(uint32_t *s, int i, int j, int k, int r) {
    s[i] ^= rotate(s[j] + s[k], r);
}

void quarterround(uint32_t *s, int i0, int i1, int i2, int i3) {
  step(s, i1, i0, i3, 7);
  step(s, i2, i1, i0, 9);
  step(s, i3, i2, i1, 13);
  step(s, i0, i3, i2, 18);
}
```

# Implementação

***quarterround()***

$$a = a \oplus [(d + c) <<< 18]$$

```
void step(uint32_t *s, int i, int j, int k, int r) {
    s[i] ^= rotate(s[j] + s[k], r);
}

void quarterround(uint32_t *s, int i0, int i1, int i2, int i3) {
  step(s, i1, i0, i3, 7);
  step(s, i2, i1, i0, 9);
  step(s, i3, i2, i1, 13);
  step(s, i0, i3, i2, 18);
}
```

# Implementação

***quarterround()***

$$a = a \oplus [(d + c) <<< 18]$$

```
void step(uint32_t *s, int i, int j, int k, int r) {
    s[i] ^= rotate(s[j] + s[k], r);
}

void quarterround(uint32_t *s, int i0, int i1, int i2, int i3) {
  step(s, i1, i0, i3, 7);
  step(s, i2, i1, i0, 9);
  step(s, i3, i2, i1, 13);
  step(s, i0, i3, i2, 18);
}
```

# Implementação

***quarterround()***

$$a = a \oplus [(d + c) <<< 18]$$

```
void step(uint32_t *s, int i, int j, int k, int r) {
    s[i] ^= rotate(s[j] + s[k], r);
}

void quarterround(uint32_t *s, int i0, int i1, int i2, int i3) {
  step(s, i1, i0, i3, 7);
  step(s, i2, i1, i0, 9);
  step(s, i3, i2, i1, 13);
  step(s, i0, i3, i2, 18);
}
```

# Implementação

## *quarterround()*

$$a = a \oplus [(d + c) <<< 18]$$

```
void step(uint32_t *s, int i, int j, int k, int r) {
    s[i] ^= rotate(s[j] + s[k], r);
}

void quarterround(uint32_t *s, int i0, int i1, int i2, int i3) {
  step(s, i1, i0, i3, 7);
  step(s, i2, i1, i0, 9);
  step(s, i3, i2, i1, 13);
  step(s, i0, i3, i2, 18);
}
```

# Implementação

## *doubleround()*

$$\begin{cases} QR(x_0, x_4, x_8, x_{12}) \\ QR(x_5, x_9, x_{13}, x_1) \\ QR(x_{10}, x_{14}, x_2, x_6) \\ QR(x_{15}, x_3, x_7, x_{11}) \end{cases} ; \begin{cases} QR(x_0, x_1, x_2, x_3) \\ QR(x_5, x_6, x_7, x_4) \\ QR(x_{10}, x_{11}, x_8, x_9) \\ QR(x_{15}, x_{12}, x_{13}, x_{14}) \end{cases}$$

```
void columnround(uint32_t *s) {
   quarterround(s, 0, 4, 8, 12);
   quarterround(s, 5, 9, 13, 1);
   quarterround(s, 10, 14, 2, 6);
   quarterround(s, 15, 3, 7, 11);
}
```

```
void rowround(uint32_t *s) {
   quarterround(s, 0, 1, 2, 3);
   quarterround(s, 5, 6, 7, 4);
   quarterround(s, 10, 11, 8, 9);
   quarterround(s, 15, 12, 13, 14);
}
```

```
void doubleround(uint32_t *s) {
   columnround(s);
   rowround(s);
}
```

# Implementação

## *doubleround()*

$$\begin{cases} QR(x_0, x_4, x_8, x_{12}) \\ QR(x_5, x_9, x_{13}, x_1) \\ QR(x_{10}, x_{14}, x_2, x_6) \\ QR(x_{15}, x_3, x_7, x_{11}) \end{cases} ; \begin{cases} QR(x_0, x_1, x_2, x_3) \\ QR(x_5, x_6, x_7, x_4) \\ QR(x_{10}, x_{11}, x_8, x_9) \\ QR(x_{15}, x_{12}, x_{13}, x_{14}) \end{cases}$$

```
void columnround(uint32_t *s) {
   quarterround(s, 0, 4, 8, 12);
   quarterround(s, 5, 9, 13, 1);
   quarterround(s, 10, 14, 2, 6);
   quarterround(s, 15, 3, 7, 11);
}
```

```
void rowround(uint32_t *s) {
   quarterround(s, 0, 1, 2, 3);
   quarterround(s, 5, 6, 7, 4);
   quarterround(s, 10, 11, 8, 9);
   quarterround(s, 15, 12, 13, 14);
}
```

```
void doubleround(uint32_t *s) {
   columnround(s);
   rowround(s);
}
```

# Implementação

## *doubleround()*

$$\begin{cases} QR(x_0, x_4, x_8, x_{12}) \\ QR(x_5, x_9, x_{13}, x_1) \\ QR(x_{10}, x_{14}, x_2, x_6) \\ QR(x_{15}, x_3, x_7, x_{11}) \end{cases} ; \begin{cases} QR(x_0, x_1, x_2, x_3) \\ QR(x_5, x_6, x_7, x_4) \\ QR(x_{10}, x_{11}, x_8, x_9) \\ QR(x_{15}, x_{12}, x_{13}, x_{14}) \end{cases}$$

```
void columnround(uint32_t *s) {
   quarterround(s, 0, 4, 8, 12);
   quarterround(s, 5, 9, 13, 1);
   quarterround(s, 10, 14, 2, 6);
   quarterround(s, 15, 3, 7, 11);
}
```

```
void rowround(uint32_t *s) {
   quarterround(s, 0, 1, 2, 3);
   quarterround(s, 5, 6, 7, 4);
   quarterround(s, 10, 11, 8, 9);
   quarterround(s, 15, 12, 13, 14);
}
```

```
void doubleround(uint32_t *s) {
   columnround(s);
   rowround(s);
}
```

# Implementação

## *doubleround()*

$$\begin{cases} QR(x_0, x_4, x_8, x_{12}) \\ QR(x_5, x_9, x_{13}, x_1) \\ QR(x_{10}, x_{14}, x_2, x_6) \\ QR(x_{15}, x_3, x_7, x_{11}) \end{cases} ; \begin{cases} QR(x_0, x_1, x_2, x_3) \\ QR(x_5, x_6, x_7, x_4) \\ QR(x_{10}, x_{11}, x_8, x_9) \\ QR(x_{15}, x_{12}, x_{13}, x_{14}) \end{cases}$$

```
void columnround(uint32_t *s) {
   quarterround(s, 0, 4, 8, 12);
   quarterround(s, 5, 9, 13, 1);
   quarterround(s, 10, 14, 2, 6);
   quarterround(s, 15, 3, 7, 11);
}
```

```
void rowround(uint32_t *s) {
   quarterround(s, 0, 1, 2, 3);
   quarterround(s, 5, 6, 7, 4);
   quarterround(s, 10, 11, 8, 9);
   quarterround(s, 15, 12, 13, 14);
}
```

```
void doubleround(uint32_t *s) {
   columnround(s);
   rowround(s);
}
```

# Implementação

## *doubleround()*

$$\begin{cases} QR(x_0, x_4, x_8, x_{12}) \\ QR(x_5, x_9, x_{13}, x_1) \\ QR(x_{10}, x_{14}, x_2, x_6) \\ QR(x_{15}, x_3, x_7, x_{11}) \end{cases} ; \begin{cases} QR(x_0, x_1, x_2, x_3) \\ QR(x_5, x_6, x_7, x_4) \\ QR(x_{10}, x_{11}, x_8, x_9) \\ QR(x_{15}, x_{12}, x_{13}, x_{14}) \end{cases}$$

```
void columnround(uint32_t *s) {
   quarterround(s, 0, 4, 8, 12);
   quarterround(s, 5, 9, 13, 1);
   quarterround(s, 10, 14, 2, 6);
   quarterround(s, 15, 3, 7, 11);
}
```

```
void rowround(uint32_t *s) {
   quarterround(s, 0, 1, 2, 3);
   quarterround(s, 5, 6, 7, 4);
   quarterround(s, 10, 11, 8, 9);
   quarterround(s, 15, 12, 13, 14);
}
```

```
void doubleround(uint32_t *s) {
   columnround(s);
   rowround(s);
}
```

# Implementação

## *doubleround()*

$$\begin{cases} QR(x_0, x_4, x_8, x_{12}) \\ QR(x_5, x_9, x_{13}, x_1) \\ QR(x_{10}, x_{14}, x_2, x_6) \\ QR(x_{15}, x_3, x_7, x_{11}) \end{cases} ; \begin{cases} QR(x_0, x_1, x_2, x_3) \\ QR(x_5, x_6, x_7, x_4) \\ QR(x_{10}, x_{11}, x_8, x_9) \\ QR(x_{15}, x_{12}, x_{13}, x_{14}) \end{cases}$$

```
void columnround(uint32_t *s) {
   quarterround(s, 0, 4, 8, 12);
   quarterround(s, 5, 9, 13, 1);
   quarterround(s, 10, 14, 2, 6);
   quarterround(s, 15, 3, 7, 11);
}
```

```
void rowround(uint32_t *s) {
   quarterround(s, 0, 1, 2, 3);
   quarterround(s, 5, 6, 7, 4);
   quarterround(s, 10, 11, 8, 9);
   quarterround(s, 15, 12, 13, 14);
}
```

```
void doubleround(uint32_t *s) {
   columnround(s);
   rowround(s);
}
```

# Implementação

```
rounds(s, 20);

void rounds(uint32_t *s, int nrounds) {
    uint32_t s1[16];
    int i;

    /* copiar s para s1 */
    for(i = 0; i < 16; i++)
        s1[i] = s[i];

    while(nrounds >= 2) {
        doubleround(s1);
        nrounds -= 2;
    }

    for(i = 0; i < 16; i++)
        s[i] += s1[i];
}
```

# Implementação

```
rounds(s, 20);

void rounds(uint32_t *s, int nrounds) {
    uint32_t s1[16];
    int i;

    /* copiar s para s1 */
    for(i = 0; i < 16; i++)
        s1[i] = s[i];

    while(nrounds >= 2) {
        doubleround(s1);
        nrounds -= 2;
    }

    for(i = 0; i < 16; i++)
        s[i] += s1[i];
}
```

**Chamada do salsa mom 20 rounds!**

# Implementação

```
rounds(s, 20);

void rounds(uint32_t *s, int nrounds) {
    uint32_t s1[16];
    int i;

    /* copiar s para s1 */
    for(i = 0; i < 16; i++)
        s1[i] = s[i];

    while(nrounds >= 2) {
        doubleround(s1);
        nrounds -= 2;
    }

    for(i = 0; i < 16; i++)
        s[i] += s1[i];
}
```

***10 rounds duplos***
***(20 rounds total)***
***passo 2 decrescente***

# Implementação

```
rounds(s, 20);

void rounds(uint32_t *s, int nrounds) {
   uint32_t s1[16];
   int i;

   /* copiar s para s1 */
   for(i = 0; i < 16; i++)
      s1[i] = s[i];

   while(nrounds >= 2) {
      doubleround(s1);
      nrounds -= 2;
   }

   for(i = 0; i < 16; i++)
      s[i] += s1[i];
}
```

**Z = X + DR(X)**

# Implementação

```
void block(uint32_t *s, uint32_t *pos, uint32_t *nonce, uint32_t *key) {
   int i;
   /* s[::5] = o */
   s[ 0] = o[0];
   s[ 5] = o[1];
   s[10] = o[2];
   s[15] = o[3];
   /* s[1:5] = key[:4] */
   s[1] = key[0];
   s[2] = key[1];
   s[3] = key[2];
   s[4] = key[3];
   /* s[6:10] = nonce ++ pos  */
   s[6] = nonce[0];
   s[7] = nonce[1];
   s[8] = pos[0];
   s[9] = pos[1];
   /* s[11:15] = key[4:] */
   s[11] = key[4];
   s[12] = key[5];
   s[13] = key[6];
   s[14] = key[7];

   rounds(s, 20);  /* Salsa20/20 */
}
```

# Implementação

```
int main() {
   int i;
   uint32_t s[16];
   /* Exemplo de "The Salsa20 family of stream ciphers", Daniel J. Bernstein
    * http://cr.yp.to/snuffle/salsafamily-20071225.pdf */
   uint32_t key[8] = {
      0x04030201,
      0x08070605,
      0x0c0b0a09,
      0x100f0e0d,
      0x14131211,
      0x18171615,
      0x1c1b1a19,
      0x201f1e1d,
   };
   uint32_t nonce[] = {
      0x01040103,
      0x06020905,
   };
   uint32_t pos[] = {
      0x7,
      0x0,
   };
   block(s, pos, nonce, key);

   for(i = 0; i < 16; i++)
      printf("0x%08x\n", s[i]);
   return 0;
}
```

FIM