Salsa20 specification

Daniel J. Bernstein *

Department of Mathematics, Statistics, and Computer Science (M/C 249)
The University of Illinois at Chicago
Chicago, IL 60607–7045
snuffle@box.cr.yp.to

Abstract. This document defines the Salsa20 hash function, the Salsa20 expansion function, and the Salsa20 encryption function.

1 Introduction

The core of Salsa20 is a hash function with 64-byte input and 64-byte output. The hash function is used in counter mode as a stream cipher: Salsa20 encrypts a 64-byte block of plaintext by hashing the key, nonce, and block number and xor'ing the result with the plaintext.

This document defines Salsa20 from bottom up, starting from three simple operations on 4-byte words, continuing through the Salsa20 hash function, and finishing with the Salsa20 encryption function.

In this document, a **byte** is an element of $\{0, 1, \ldots, 255\}$. There are many common ways to represent a byte as a sequence of electrical signals; the details of this representation are of no relevance to the definition of Salsa20.

2 Words

A **word** is an element of $\{0, 1, \ldots, 2^{32} - 1\}$. Words in this document are often written in hexadecimal, indicated by the symbols `0x`: for example, `0xc0a8787e` $= 12 \cdot 2^{28} + 0 \cdot 2^{24} + 10 \cdot 2^{20} + 8 \cdot 2^{16} + 7 \cdot 2^{12} + 8 \cdot 2^8 + 7 \cdot 2^4 + 14 \cdot 2^0 = 3232266366$.

The **sum** of two words u, v is $u + v \bmod 2^{32}$. The sum is denoted $u + v$; there is no risk of confusion. For example, `0xc0a8787e` + `0x9fd1161d` = `0x60798e9b`.

The **exclusive-or** of two words u, v, denoted $u \oplus v$, is the sum of u and v with carries suppressed. In other words, if $u = \sum_i 2^i u_i$ and $v = \sum 2^i v_i$ then $u \oplus v = \sum_i 2^i (u_i + v_i - 2u_i v_i)$. For example, `0xc0a8787e` $\oplus$ `0x9fd1161d` = `0x5f796e63`.

For each $c \in \{0, 1, 2, 3, \ldots\}$, the **$c$-bit left rotation** of a word u, denoted $u \lll c$, is the unique nonzero word congruent to $2^c u$ modulo $2^{32} - 1$, except that $0 \lll c = 0$. In other words, if $u = \sum_i 2^i u_i$ then $u \lll c = \sum_i 2^{i+c \bmod 32} u_i$. For example, `0xc0a8787e` $\lll 5$ = `0x150f0fd8`.

* The author was supported by the National Science Foundation under grant CCR–9983950, and by the Alfred P. Sloan Foundation. Date of this document: 2005.04.27.

3 The quarterround function

Inputs and outputs

If y is a 4-word sequence then quarterround(y) is a 4-word sequence.

Definition

If $y = (y_0, y_1, y_2, y_3)$ then quarterround(y) $= (z_0, z_1, z_2, z_3)$ where

$$\begin{aligned} z_1 &= y_1 \oplus ((y_0 + y_3) \lll 7), \\ z_2 &= y_2 \oplus ((z_1 + y_0) \lll 9), \\ z_3 &= y_3 \oplus ((z_2 + z_1) \lll 13), \\ z_0 &= y_0 \oplus ((z_3 + z_2) \lll 18). \end{aligned}$$

Examples

$$\begin{aligned} &\text{quarterround}(\texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000000}) \\ &\qquad = (\texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000000}). \\ &\text{quarterround}(\texttt{0x00000001}, \texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000000}) \\ &\qquad = (\texttt{0x08008145}, \texttt{0x00000080}, \texttt{0x00010200}, \texttt{0x20500000}). \\ &\text{quarterround}(\texttt{0x00000000}, \texttt{0x00000001}, \texttt{0x00000000}, \texttt{0x00000000}) \\ &\qquad = (\texttt{0x88000100}, \texttt{0x00000001}, \texttt{0x00000200}, \texttt{0x00402000}). \\ &\text{quarterround}(\texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000001}, \texttt{0x00000000}) \\ &\qquad = (\texttt{0x80040000}, \texttt{0x00000000}, \texttt{0x00000001}, \texttt{0x00002000}). \\ &\text{quarterround}(\texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000001}) \\ &\qquad = (\texttt{0x00048044}, \texttt{0x00000080}, \texttt{0x00010000}, \texttt{0x20100001}). \\ &\text{quarterround}(\texttt{0xe7e8c006}, \texttt{0xc4f9417d}, \texttt{0x6479b4b2}, \texttt{0x68c67137}) \\ &\qquad = (\texttt{0xe876d72b}, \texttt{0x9361dfd5}, \texttt{0xf1460244}, \texttt{0x948541a3}). \\ &\text{quarterround}(\texttt{0xd3917c5b}, \texttt{0x55f1c407}, \texttt{0x52a58a7a}, \texttt{0x8f887a3b}) \\ &\qquad = (\texttt{0x3e2f308c}, \texttt{0xd90a8f36}, \texttt{0x6ab2a923}, \texttt{0x2883524c}). \end{aligned}$$

Comments

One can visualize the quarterround function as modifying y in place: first y_1 changes to z_1, then y_2 changes to z_2, then y_3 changes to z_3, then y_0 changes to z_0. Each modification is invertible, so the entire function is invertible.

4 The rowround function

Inputs and outputs

If y is a 16-word sequence then rowround(y) is a 16-word sequence.

Definition

If $y = (y_0, y_1, y_2, y_3, \ldots, y_{15})$ then $\text{rowround}(y) = (z_0, z_1, z_2, z_3, \ldots, z_{15})$ where

$$\begin{aligned}(z_0, z_1, z_2, z_3) &= \text{quarterround}(y_0, y_1, y_2, y_3),\\ (z_5, z_6, z_7, z_4) &= \text{quarterround}(y_5, y_6, y_7, y_4),\\ (z_{10}, z_{11}, z_8, z_9) &= \text{quarterround}(y_{10}, y_{11}, y_8, y_9),\\ (z_{15}, z_{12}, z_{13}, z_{14}) &= \text{quarterround}(y_{15}, y_{12}, y_{13}, y_{14}).\end{aligned}$$

Examples

$$\begin{aligned}\text{rowround}(&\texttt{0x00000001}, \texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000000},\\ &\texttt{0x00000001}, \texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000000},\\ &\texttt{0x00000001}, \texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000000},\\ &\texttt{0x00000001}, \texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000000})\\ =\ (&\texttt{0x08008145}, \texttt{0x00000080}, \texttt{0x00010200}, \texttt{0x20500000},\\ &\texttt{0x20100001}, \texttt{0x00048044}, \texttt{0x00000080}, \texttt{0x00010000},\\ &\texttt{0x00000001}, \texttt{0x00002000}, \texttt{0x80040000}, \texttt{0x00000000},\\ &\texttt{0x00000001}, \texttt{0x00000200}, \texttt{0x00402000}, \texttt{0x88000100}).\\ \text{rowround}(&\texttt{0x08521bd6}, \texttt{0x1fe88837}, \texttt{0xbb2aa576}, \texttt{0x3aa26365},\\ &\texttt{0xc54c6a5b}, \texttt{0x2fc74c2f}, \texttt{0x6dd39cc3}, \texttt{0xda0a64f6},\\ &\texttt{0x90a2f23d}, \texttt{0x067f95a6}, \texttt{0x06b35f61}, \texttt{0x41e4732e},\\ &\texttt{0xe859c100}, \texttt{0xea4d84b7}, \texttt{0x0f619bff}, \texttt{0xbc6e965a})\\ =\ (&\texttt{0xa890d39d}, \texttt{0x65d71596}, \texttt{0xe9487daa}, \texttt{0xc8ca6a86},\\ &\texttt{0x949d2192}, \texttt{0x764b7754}, \texttt{0xe408d9b9}, \texttt{0x7a41b4d1},\\ &\texttt{0x3402e183}, \texttt{0x3c3af432}, \texttt{0x50669f96}, \texttt{0xd89ef0a8},\\ &\texttt{0x0040ede5}, \texttt{0xb545fbce}, \texttt{0xd257ed4f}, \texttt{0x1818882d}).\end{aligned}$$

Comments

One can visualize the input $(y_0, y_1, \ldots, y_{15})$ as a square matrix:

$$\begin{pmatrix} y_0 & y_1 & y_2 & y_3\\ y_4 & y_5 & y_6 & y_7\\ y_8 & y_9 & y_{10} & y_{11}\\ y_{12} & y_{13} & y_{14} & y_{15}\end{pmatrix}$$

The rowround function modifies the rows of the matrix in parallel by feeding a permutation of each row through the quarterround function. In the first row, the rowround function modifies y_1, then y_2, then y_3, then y_0; in the second row, the rowround function modifies y_6, then y_7, then y_4, then y_5; in the third row, the rowround function modifies y_{11}, then y_8, then y_9, then y_{10}; in the fourth row, the rowround function modifies y_{12}, then y_{13}, then y_{14}, then y_{15}.

5 The columnround function

Inputs and outputs

If x is a 16-word sequence then columnround(x) is a 16-word sequence.

Definition

If $x = (x_0, x_1, x_2, x_3, \ldots, x_{15})$ then columnround(x) $= (y_0, y_1, y_2, y_3, \ldots, y_{15})$ where

$$(y_0, y_4, y_8, y_{12}) = \text{quarterround}(x_0, x_4, x_8, x_{12}),$$
$$(y_5, y_9, y_{13}, y_1) = \text{quarterround}(x_5, x_9, x_{13}, x_1),$$
$$(y_{10}, y_{14}, y_2, y_6) = \text{quarterround}(x_{10}, x_{14}, x_2, x_6),$$
$$(y_{15}, y_3, y_7, y_{11}) = \text{quarterround}(x_{15}, x_3, x_7, x_{11}).$$

Equivalent formula: $(y_0, y_4, y_8, y_{12}, y_1, y_5, y_9, y_{13}, y_2, y_6, y_{10}, y_{14}, y_3, y_7, y_{11}, y_{15}) =$ rowround$(x_0, x_4, x_8, x_{12}, x_1, x_5, x_9, x_{13}, x_2, x_6, x_{10}, x_{14}, x_3, x_7, x_{11}, x_{15})$.

Examples

columnround(0x00000001, 0x00000000, 0x00000000, 0x00000000,
0x00000001, 0x00000000, 0x00000000, 0x00000000,
0x00000001, 0x00000000, 0x00000000, 0x00000000,
0x00000001, 0x00000000, 0x00000000, 0x00000000)
= (0x10090288, 0x00000000, 0x00000000, 0x00000000,
0x00000101, 0x00000000, 0x00000000, 0x00000000,
0x00020401, 0x00000000, 0x00000000, 0x00000000,
0x40a04001, 0x00000000, 0x00000000, 0x00000000).

columnround(0x08521bd6, 0x1fe88837, 0xbb2aa576, 0x3aa26365,
0xc54c6a5b, 0x2fc74c2f, 0x6dd39cc3, 0xda0a64f6,
0x90a2f23d, 0x067f95a6, 0x06b35f61, 0x41e4732e,
0xe859c100, 0xea4d84b7, 0x0f619bff, 0xbc6e965a)
= (0x8c9d190a, 0xce8e4c90, 0x1ef8e9d3, 0x1326a71a,
0x90a20123, 0xead3c4f3, 0x63a091a0, 0xf0708d69,
0x789b010c, 0xd195a681, 0xeb7d5504, 0xa774135c,
0x481c2027, 0x53a8e4b5, 0x4c1f89c5, 0x3f78c9c8).

Comments

One can visualize the input $(x_0, x_1, \ldots, x_{15})$ as a square matrix, as in Section 4:

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

The columnround function is, from this perspective, simply the transpose of the rowround function: it modifies the columns of the matrix in parallel by feeding a permutation of each column through the quarterround function. In the first column, the columnround function modifies y_4, then y_8, then y_{12}, then y_0; in the second column, the columnround function modifies y_9, then y_{13}, then y_1, then y_5; in the third column, the columnround function modifies y_{14}, then y_2, then y_6, then y_{10}; in the fourth column, the columnround function modifies y_3, then y_7, then y_{11}, then y_{15}.

6 The doubleround function

Inputs and outputs

If x is a 16-word sequence then doubleround(x) is a 16-word sequence.

Definition

A double round is a column round followed by a row round: doubleround(x) = rowround(columnround(x)).

Examples

doubleround(0x00000001, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000)
= (0x8186a22d, 0x0040a284, 0x82479210, 0x06929051,
0x08000090, 0x02402200, 0x00004000, 0x00800000,
0x00010200, 0x20400000, 0x08008104, 0x00000000,
0x20500000, 0xa0000040, 0x0008180a, 0x612a8020).

doubleround(0xde501066, 0x6f9eb8f7, 0xe4fbbd9b, 0x454e3f57,
0xb75540d3, 0x43e93a4c, 0x3a6f2aa0, 0x726d6b36,
0x9243f484, 0x9145d1e8, 0x4fa9d247, 0xdc8dee11,
0x054bf545, 0x254dd653, 0xd9421b6d, 0x67b276c1)
= (0xccaaf672, 0x23d960f7, 0x9153e63a, 0xcd9a60d0,
0x50440492, 0xf07cad19, 0xae344aa0, 0xdf4cfdfc,
0xca531c29, 0x8e7943db, 0xac1680cd, 0xd503ca00,
0xa74b2ad6, 0xbc331c5c, 0x1dda24c7, 0xee928277).

Comments

One can visualize a double round as modifying the columns of the input in parallel, and then modifying the rows in parallel. Each word is modified twice.

7 The littleendian function

Inputs and outputs

If b is a 4-byte sequence then littleendian(b) is a word.

Definition

If $b = (b_0, b_1, b_2, b_3)$ then littleendian$(b) = b_0 + 2^8 b_1 + 2^{16} b_2 + 2^{24} b_3$.

Examples

$$\text{littleendian}(0, 0, 0, 0) = \texttt{0x00000000}.$$
$$\text{littleendian}(86, 75, 30, 9) = \texttt{0x091e4b56}.$$
$$\text{littleendian}(255, 255, 255, 250) = \texttt{0xfaffffff}.$$

Comments

Note that littleendian is invertible.

8 The Salsa20 hash function

Inputs and outputs

If x is a 64-byte sequence then Salsa20(x) is a 64-byte sequence.

Definition

In short: Salsa20$(x) = x +$ doubleround$^{10}(x)$, where each 4-byte sequence is viewed as a word in little-endian form.

In detail: Starting from $x = (x[0], x[1], \ldots, x[63])$, define

$$\begin{aligned} x_0 &= \text{littleendian}(x[0], x[1], x[2], x[3]), \\ x_1 &= \text{littleendian}(x[4], x[5], x[6], x[7]), \\ x_2 &= \text{littleendian}(x[8], x[9], x[10], x[11]), \\ &\vdots \\ x_{15} &= \text{littleendian}(x[60], x[61], x[62], x[63]). \end{aligned}$$

Define $(z_0, z_1, \ldots, z_{15}) = \text{doubleround}^{10}(x_0, x_1, \ldots, x_{15})$. Then Salsa20($x$) is the concatenation of

$$\begin{aligned} &\text{littleendian}^{-1}(z_0 + x_0), \\ &\text{littleendian}^{-1}(z_1 + x_1), \\ &\text{littleendian}^{-1}(z_2 + x_2), \\ &\vdots \\ &\text{littleendian}^{-1}(z_{15} + x_{15}). \end{aligned}$$

Examples

$$\begin{aligned}
\text{Salsa20}(&\ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\\
&\ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\\
&\ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\\
&\ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)\\
= (&\ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\\
&\ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\\
&\ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\\
&\ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0).
\end{aligned}$$

$$\begin{aligned}
\text{Salsa20}(&211,159, 13,115, 76, 55, 82,183, 3,117,222, 37,191,187,234,136,\\
&49,237,179, 48, 1,106,178,219,175,199,166, 48, 86, 16,179,207,\\
&31,240, 32, 63, 15, 83, 93,161,116,147, 48,113,238, 55,204, 36,\\
&79,201,235, 79, 3, 81,156, 47,203, 26,244,243, 88,118,104, 54)\\
= (&109, 42,178,168,156,240,248,238,168,196,190,203, 26,110,170,154,\\
&29, 29,150, 26,150, 30,235,249,190,163,251, 48, 69,144, 51, 57,\\
&118, 40,152,157,180, 57, 27, 94,107, 42,236, 35, 27,111,114,114,\\
&219,236,232,135,111,155,110, 18, 24,232, 95,158,179, 19, 48,202).
\end{aligned}$$

$$\begin{aligned}
\text{Salsa20}(&88,118,104, 54, 79,201,235, 79, 3, 81,156, 47,203, 26,244,243,\\
&191,187,234,136,211,159, 13,115, 76, 55, 82,183, 3,117,222, 37,\\
&86, 16,179,207, 49,237,179, 48, 1,106,178,219,175,199,166, 48,\\
&238, 55,204, 36, 31,240, 32, 63, 15, 83, 93,161,116,147, 48,113)\\
= (&179, 19, 48,202,219,236,232,135,111,155,110, 18, 24,232, 95,158,\\
&26,110,170,154,109, 42,178,168,156,240,248,238,168,196,190,203,\\
&69,144, 51, 57, 29, 29,150, 26,150, 30,235,249,190,163,251, 48,\\
&27,111,114,114,118, 40,152,157,180, 57, 27, 94,107, 42,236, 35).
\end{aligned}$$

$$\begin{aligned}
\text{Salsa20}^{1000000}(&\ 6,124, 83,146, 38,191, 9, 50, 4,161, 47,222,122,182,223,185,\\
&\ 75, 27, 0,216, 16,122, 7, 89,162,104,101,147,213, 21, 54, 95,\\
&225,253,139,176,105,132, 23,116, 76, 41,176,207,221, 34,157,108,\\
&\ 94, 94, 99, 52, 90,117, 91,220,146,190,239,143,196,176,130,186)\\
= (&\ 8, 18, 38,199,119, 76,215, 67,173,127,144,162,103,212,176,217,\\
&192, 19,233, 33,159,197,154,160,128,243,219, 65,171,136,135,225,\\
&123, 11, 68, 86,237, 82, 20,155,133,189, 9, 83,167,116,194, 78,\\
&122,127,195,185,185,204,188, 90,245, 9,183,248,226, 85,245,104).
\end{aligned}$$

9 The Salsa20 expansion function

Inputs and outputs

If k is a 32-byte *or* 16-byte sequence and n is a 16-byte sequence then $\text{Salsa20}_k(n)$ is a 64-byte sequence.

Definition

Define $\sigma_0 = (101, 120, 112, 97)$, $\sigma_1 = (110, 100, 32, 51)$, $\sigma_2 = (50, 45, 98, 121)$, and $\sigma_3 = (116, 101, 32, 107)$. If k_0, k_1, n are 16-byte sequences then $\text{Salsa20}_{k_0,k_1}(n) = \text{Salsa20}(\sigma_0, k_0, \sigma_1, n, \sigma_2, k_1, \sigma_3)$.

Define $\tau_0 = (101, 120, 112, 97)$, $\tau_1 = (110, 100, 32, 49)$, $\tau_2 = (54, 45, 98, 121)$, and $\tau_3 = (116, 101, 32, 107)$. If k, n are 16-byte sequences then $\text{Salsa20}_k(n) = \text{Salsa20}(\tau_0, k, \tau_1, n, \tau_2, k, \tau_3)$.

Examples

Define $k_0 = (1, 2, 3, 4, 5, \ldots, 16)$, $k_1 = (201, 202, 203, 204, 205, \ldots, 216)$, and $n = (101, 102, 103, 104, 105, \ldots, 116)$. Then

$\text{Salsa20}_{k_0,k_1}(n) =$
$\text{Salsa20}(101,120,112, 97, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,$
$13, 14, 15, 16,110,100, 32, 51,101,102,103,104,105,106,107,108,$
$109,110,111,112,113,114,115,116, 50, 45, 98,121,201,202,203,204,$
$205,206,207,208,209,210,211,212,213,214,215,216,116,101, 32,107)$
$= (69, 37, 68, 39, 41, 15,107,193,255,139,122, 6,170,233,217, 98,$
$89,144,182,106, 21, 51,200, 65,239, 49,222, 34,215,114, 40,126,$
$104,197, 7,225,197,153, 31, 2,102, 78, 76,176, 84,245,246,184,$
$177,160,133,130, 6, 72,149,119,192,195,132,236,234,103,246, 74)$

and

$\text{Salsa20}_{k_0}(n) =$
$\text{Salsa20}(101,120,112, 97, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,$
$13, 14, 15, 16,110,100, 32, 49,101,102,103,104,105,106,107,108,$
$109,110,111,112,113,114,115,116, 54, 45, 98,121, 1, 2, 3, 4,$
$5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,116,101, 32,107)$
$= (39,173, 46,248, 30,200, 82, 17, 48, 67,254,239, 37, 18, 13,247,$
$241,200, 61,144, 10, 55, 50,185, 6, 47,246,253,143, 86,187,225,$
$134, 85,110,246,161,163, 43,235,231, 94,171, 51,145,214,112, 29,$
$14,232, 5, 16,151,140,183,141,171, 9,122,181,104,182,177,193).$

Comments

"Expansion" refers to the expansion of (k, n) into $\text{Salsa20}_k(n)$. It also refers to the expansion of k into a long stream of Salsa20_k outputs for various n's; see Section 10.

The constants $\sigma_0\sigma_1\sigma_2\sigma_3$ and $\tau_0\tau_1\tau_2\tau_3$ are "`expand 32-byte k`" and "`expand 16-byte k`" in ASCII.

10 The Salsa20 encryption function

Inputs and outputs

Let k be a 32-byte *or* 16-byte sequence. Let v be an 8-byte sequence. Let m be an ℓ-byte sequence for some $\ell \in \{0, 1, \ldots, 2^{70}\}$. The **Salsa20 encryption of** m **with nonce** v **under key** k, denoted $\text{Salsa20}_k(v) \oplus m$, is an ℓ-byte sequence.

Normally k is a secret key (preferably 32 bytes); v is a nonce, i.e., a unique message number; m is a plaintext message; and $\text{Salsa20}_k(v) \oplus m$ is a ciphertext message. Or m can be a ciphertext message, in which case $\text{Salsa20}_k(v) \oplus m$ is the original plaintext message.

Definition

$\text{Salsa20}_k(v)$ is the 2^{70}-byte sequence

$$\text{Salsa20}_k(v, \underline{0}), \text{Salsa20}_k(v, \underline{1}), \text{Salsa20}_k(v, \underline{2}), \ldots, \text{Salsa20}_k(v, \underline{2^{64} - 1}).$$

Here $\underline{i}$ is the unique 8-byte sequence $(i_0, i_1, \ldots, i_7)$ such that $i = i_0 + 2^8 i_1 + 2^{16} i_2 + \cdots + 2^{56} i_7$.

The formula $\text{Salsa20}_k(v) \oplus m$ implicitly truncates $\text{Salsa20}_k(v)$ to the same length as m. In other words,

$$\text{Salsa20}_k(v) \oplus (m[0], m[1], \ldots, m[\ell - 1]) = (c[0], c[1], \ldots, c[\ell - 1])$$

where $c[i] = m[i] \oplus \text{Salsa20}_k(v, \underline{\lfloor i/64 \rfloor})[i \bmod 64]$.

Comments

The definition of Salsa20 could easily be generalized from byte sequences to bit sequences, given an encoding of bytes as sequences of bits. However, there is no apparent application of this generalization.