Crypto Project

Wednesday, November 12, 2025 12:55 PM

Salsa20 operates on 32-bit words.

- additions are done modulo 2^{32}.

Example

1 bit: either 1 or 0 $\longrightarrow$ $2^1 = 2$ options

2 bits: {00, 01, 10, 11} $\longrightarrow$ $2^2 = 4$ options

4 bits: {0000, 0001, ..., 1110, 1111} $\longrightarrow$ $2^4 = 16$ options

⋮

32 bits: $2^{32} = 4,294,967,296$ options.

- Each character in the alphabet can be represented w/ 8 bits:

$$\underbrace{A}_{\text{letter}} = \underbrace{65}_{\text{dec}} = 01000001$$

$$Y = 89 = 01011001$$

$$f = 102 = 01100110$$

So, the function "_u32" in the Salsa20.py turns an integer into a 32 bit value.

Example

$$0x\underbrace{ffffffff}_{\text{8 f's}} = \underbrace{1111111\cdots111}_{\text{32 ones}}$$

- The function takes a value, no matter the size, and turns it into a 32 bit value by performing the "&" operation on it.

0xf = 1111 , if the value x = 3.

then

$$\begin{array}{r} 1111 \\ \& \; 0011 \\ \hline 0011 \end{array} = 3$$

x was able to fit inside of the 4 bit limitation of <u>0xf</u>.

If x = 28 , then

$$\left.\begin{array}{l} \text{oxf} = 1111 \\ 28 = 11100 \end{array}\right\} \Rightarrow \begin{array}{r} 1111 \\ \& \, 11100 \\ \hline \cancel{1}\underline{1100} \end{array}$$

since x here is 5 bits long instead of 4, then the 5th bit is truncated. So the result of this 4 bit operation is 1100 = 12.

The "_u32" method works the same way to truncate values w/in 32 bits.