Nicholas Whiteman
Nathan Metens

Salsa20 Notes

- Designed for both software and hardware implementations.
- Stream Cipher

$Salsa20(k, r) := \ H(k,\ (r, 0))\ ||\ H(k,\ (r,\ 1)\ ||\ ...$

$\{0,\ 1\}^{128\ or\ 256} *\ \{0, 1\}^{64} \ -> \ \{0, 1\}^{n}$
seed (key)   nonce   ciphertext

$Salsa20(k,\ r) := \ F(k,\ (r,\ 0))\ ||\ F(k,\ (r,\ 1))\ ||\ ...$
This goes on for as long as the plaintext is

- Each F() is a 64 byte block, so F(k, (r, 0)) represents bytes 0-63, F(k, (r, 1)) represents bytes 64-127.
- Can go on until maximum of $i\ =\ 2^{64}\ -\ 1$ but it is recommended not to do this and to restrict to about 2^{34} blocks (See ChaCha20).
- Randomizing blocks to xor the plain text is a bad idea as this breaks security (The attacker already knows your encryption, you are giving them more information than needed)

Here is what happens in each block:
1. The 64 bytes block is created
2. We begin filling each block
    - 4 byte const t
    - 16 byte key k
    - 4 byte t
    - 8 byte nonce r
    - 8 byte index i
    - 4 byte t
    - 16 byte k
    - 4 byte t
    - This is our initial state that every block in Salsa20 starts at
3. We then put it in our function h' to update the block
    - h': Our one-to-one function also known as "doubleround"
    - For this function we create a matrix of our 64 byte block, where each cell is 32 bits.
    - We start with the **columns**, we call this columnround.
        * Since we have 4 columns and 4 cells per each column (4x4 grid 32 bits), let's call our example cells: a, b, c, d.

* We then use this formula for each of our columns:

$b \oplus ROTL(a + d, 7)$

$c \oplus ROTL(b + a, 9)$

$d \oplus ROTL(c + b, 13)$

$a \oplus ROTL(d + c, 18)$

* Once that is done we do it again but instead of columns we do **rows** instead (rowround). This will give us the formula:

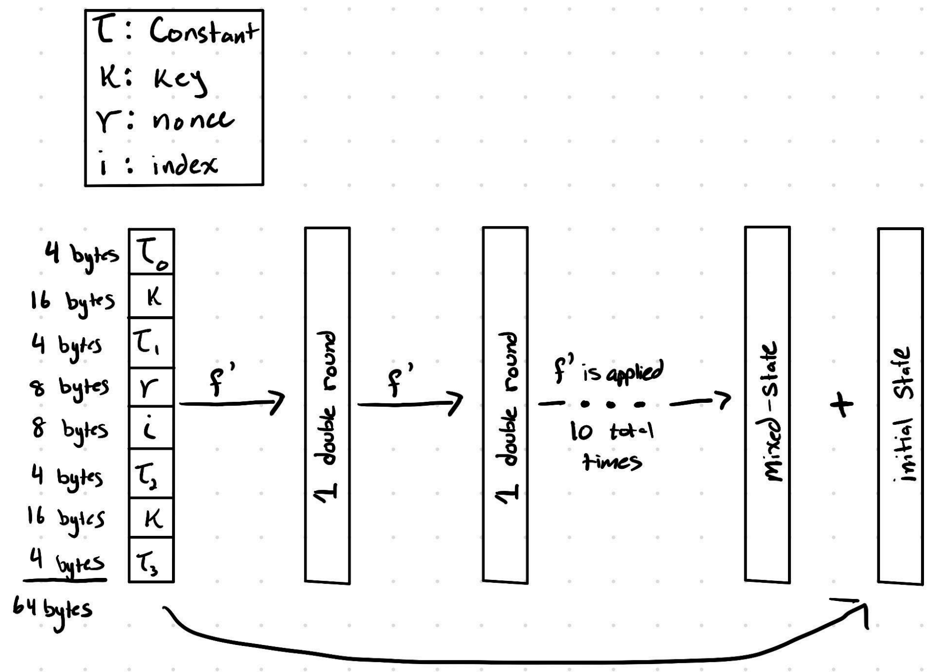
$f'(state) = rowround(columnround(state))$

also known as **doubleround**

* **NOTE:** we always rotate left by those values
* We then do this function 10 times, meaning we do 20 rounds total

4. Once we have done our 20 rounds, we then ADD the 20 round state with the initial state mod 2^32
   - $finalstate[i] = (mixedstate[i] + originalstate[i])\ mod\ 2^{32}$
   - This equation is pretty much just adding cell by cell respectively with each state.
   - For example, the first cell of the mixed state + the first cell of the original state mod 2^32 (for 32 bit math)
5. The final state will be a singular block in Salsa20.

- Since Salsa20 is a stream cipher you xor each corresponding block to the plain text to get your cipher text.



- Some important information:
    - The only difference between Salsa20/128 and Salsa20/256 is that 128 reuses the key twice in that initial state, while 256 splits the key in half and uses each part only once.
    - The cipher is only reversible if the adversary somehow knows the initial state.
    - Up until the addition of the mixed state and the initial state, the cipher is entirely reversible as xor, left rotate, and addition (mod 2^32) are all reversible

Sources:
Daniel J. Bernstein: The Salsa20 family of stream ciphers
Videos Coursera: 2 4 Real world stream ciphers 20 min)
Wikipedia: Salsa20
John L. Whiteman: Salsa20 (GitHub)
GPT 5.1: www.chatgpt.com