Salsa and ChaCha

Syntax and Correctness: PRG

Salsa and ChaCha are families of stream ciphers built on a PRG with the following syntax:

Let $\mathcal{S} = \{0,1\}^{256}$ be our seed space, $\mathcal{N} = \{0,1\}^{64}$ be our nonce space, and

$\mathcal{L} = \{0,1\}^{512 \times 2^{64}}$ be our output space. Then our PRG is defined by

$$G : \mathcal{S} \times \mathcal{N} \to \mathcal{L}$$

The PRG is correct if $G(s_1, n_1) = G(s_2, n_2) \iff s_1 = s_2 \text{ and } n_1 = n_2$

For now, we will focus on cases where $n = 0^{64}$

Syntax and Correctness: Stream Cipher

The stream cipher built off of the PRG we just defined has the following syntax:

$$E : \{0,1\}^{256} \times \{0,1\}^L \rightarrow \{0,1\}^L$$
$$D : \{0,1\}^{256} \times \{0,1\}^L \rightarrow \{0,1\}^L$$

for all $L \leq 512 \times 2^{64}$

We define $E_s(m) = G(s)[0..|m|-1] \oplus m$

$$D_s(c) = G(s)[0..|c|-1] \oplus c$$

The cipher is correct if it satisfies $\forall m \in \{0,1\}^L, D_s(E_s(m)) = m$

Construction: ChaCha20, pad(s, j, n)

The padding function: format the input into a 4 x 4 matrix of 32-bit words

$$\begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{pmatrix} \longleftarrow \begin{pmatrix} c_0 & c_1 & c_2 & c_3 \\ s_0 & s_1 & s_2 & s_3 \\ s_4 & s_5 & s_6 & s_7 \\ j_0 & j_1 & n_0 & n_1 \end{pmatrix}$$

c_i terms represent 32-bit constants, “expa”, “nd 3”, “2-by”, and “te k”;
s_i terms represent the seed broken up into 32-bit words;
j_i terms represent a 64-bit counter that increments with every block of the output;
n_i terms represent the 64-bit nonce (which we assume is 0 for now).

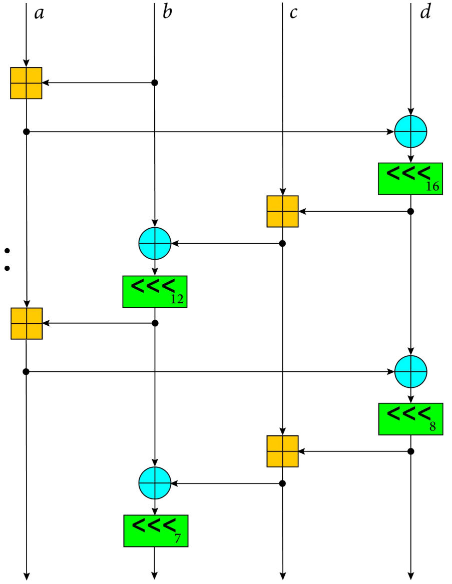
Construction: ChaCha20, π

The permutation $\pi : \{0,1\}^{512} \rightarrow \{0,1\}^{512}$ is defined by repeating the following steps 10 times

$$\begin{array}{ll}
(1)\ \text{QuarterRound}(x_0, x_4, x_8, x_{12}), & (2)\ \text{QuarterRound}(x_1, x_5, x_9, x_{13}), \\
(3)\ \text{QuarterRound}(x_2, x_6, x_{10}, x_{14}), & (4)\ \text{QuarterRound}(x_3, x_7, x_{11}, x_{15}), \\
(5)\ \text{QuarterRound}(x_0, x_5, x_{10}, x_{15}), & (6)\ \text{QuarterRound}(x_1, x_6, x_{11}, x_{12}), \\
(7)\ \text{QuarterRound}(x_2, x_7, x_8, x_{13}), & (8)\ \text{QuarterRound}(x_3, x_4, x_9, x_{14}).
\end{array}$$

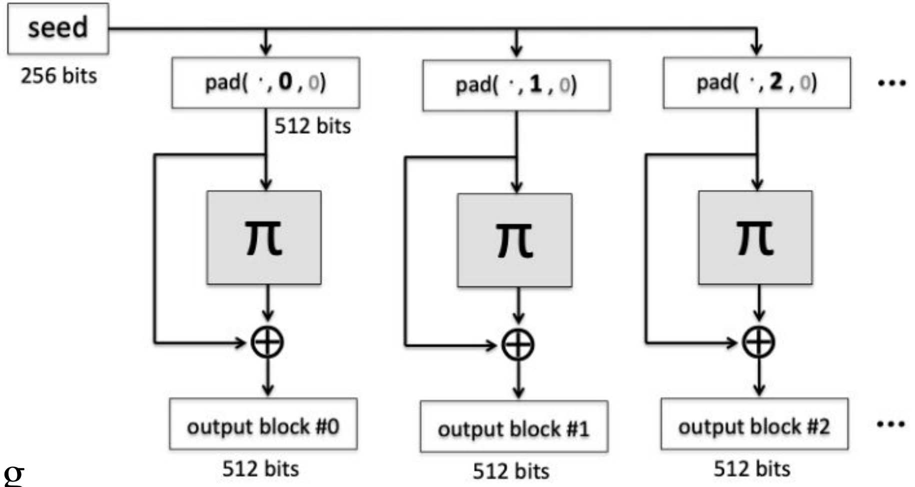
QuarterRound(a,b,c,d) is a process defined by the following C code that uses a macro ROTL(a,b) that rotates left a 32-bit word a by b bits:

```
#define ROTL(a,b) (((a) << (b)) | ((a) >> (32 - (b))))
a += b;  d ^= a;  ROTL(d, 16);
c += d;  b ^= c;  ROTL(b, 12);
a += b;  d ^= a;  ROTL(d, 8);
c += d;  b ^= c;  ROTL(b, 7);
```



Construction: ChaCha20 PRG

```
input: seed $s \in \{0,1\}^{256}$
1.    for $j \leftarrow 0$ to $L-1$
2.        $h_j \leftarrow \text{pad}(s, j, 0) \in \{0,1\}^{512}$
3.        $r_j \leftarrow \pi(h_j) \oplus h_j$
4.    output $(r_0, \ldots, r_{L-1})$.
```



Note: Rather than XOR, $\oplus$ represents breaking up the operands into 32-bit words and adding them word-wise mod 2^{32}

Construction: Nonces

A major flaw of PRGs is that their outputs can only be used once. Nonces are the solution to this problem.

Instead of sharing a new seed for every new PRG output, we can change the nonce. Sending the nonce along with the message does not compromise the seed.

This turns the PRG that ChaCha20 relies on into a PRF, with the seed as the secret key and the nonce as the message; then, ChaCha20 acts like CTR mode.

Security: PRF* Definition

We do not have a good pre-established security definition for ChaCha20's underlying PRG, so we posit the following game (inspired by PRF security for block ciphers):

```
Subroutine Initialize()
    $b \overset{\$}{\leftarrow} \{0,1\}$
    $s \overset{\$}{\leftarrow} \{0,1\}^{256}$
    $N \leftarrow \emptyset$

Subroutine Finalize($d$)
    return $d = b$
```

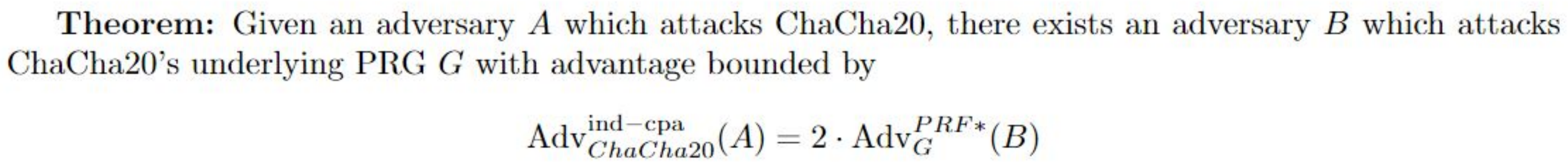
```
Subroutine ENC($n$)
    if $|n| \neq 64$ then
        return $\perp$
    end if
    if $n \in N$ then
        return $\perp$
    end if
    $N = N \cup \{n\}$
    if $b = 0$ then
        $c \overset{\$}{\leftarrow} \{0,1\}^{512 \times 2^{64}}$
        return $c$
    else
        $c \leftarrow G_s(n)$
        return $c$
    end if
```

```
$\mathrm{Exp}_G^{PRF*}(A)$
    Initialize()
    $d \leftarrow A^{ENC}$
    return Finalize($d$)
```

$$\boxed{\mathrm{Adv}_G^{PRF*}(A) = 2 \cdot \Pr[\mathrm{Exp}_G^{PRF*}(A) \to \mathrm{true}] - 1}$$

Security: IND-CPA Reduction

A simple reduction can be performed to bound the advantage of an attacker of ChaCha20 to a factor of the advantage of an attacker of G:



Theorem: Given an adversary A which attacks ChaCha20, there exists an adversary B which attacks ChaCha20's underlying PRG G with advantage bounded by

$$\mathrm{Adv}^{\mathrm{ind-cpa}}_{ChaCha20}(A) = 2 \cdot \mathrm{Adv}^{PRF*}_{G}(B)$$

Security: IND-CPA Reduction

Proof. Suppose we have an adversary A which attacks ChaCha20. Then let us construct Subroutine SimEnc(m_0, m_1):

```
static $n \leftarrow 0^{64}$
**if** $n \geq 2^{64}$ **then**
    **return** $\perp$
**end if**
**if** $|m_0| \neq |m_1|$ **then**
    **return** $\perp$
**end if**
$m \leftarrow m_b$
**if** $|m| = 0$ or $|m| > 512 \times 2^{64}$ **then**
    **return** $\perp$
**end if**
$p \leftarrow G(n)$
**return** $n \,\|\, (p \oplus m)$
```

Security: IND-CPA Reduction

and adversary B^G:

```
$b \xleftarrow{\$} \{0,1\}$
Run $A^{\text{SimEnc}}$ until it returns $d$
**return** d=b
```

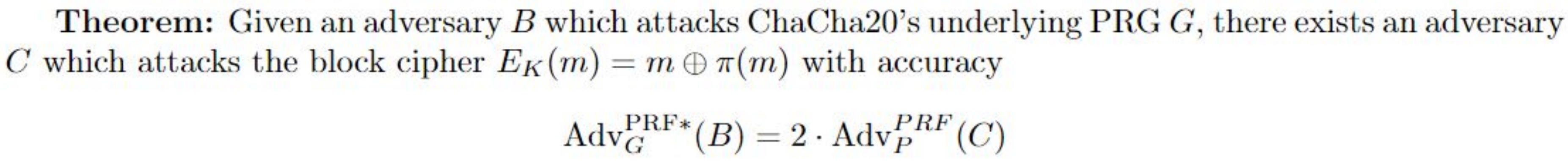
Then notice that when G is the real PRG, the SimEnc(m_0, m_1) subroutine returns exactly the output of ChaCha20 when asked to encrypt m_b, and when G outputs a random bitstring, A has a probability $\frac{1}{2}$ to guess b correctly, so B's advantage is

$$\begin{aligned}
\text{Adv}_G^{PRF*}(B) &= 2 \cdot \Pr[\text{Exp}_G^{PRF*}(B) \to \text{true}] - 1 \\
&= 2 \cdot (\frac{1}{2}\Pr[\text{Exp}_{ChaCha20}^{\text{ind-cpa}}(A) \to \text{true}] + \frac{1}{4}) - 1 \\
&= \Pr[\text{Exp}_{ChaCha20}^{\text{ind-cpa}}(A) \to \text{true}] - \frac{1}{2} \\
&= \frac{1}{2}(2 \cdot \Pr[\text{Exp}_{ChaCha20}^{\text{ind-cpa}}(A) \to \text{true}] - 1) \\
&= \frac{1}{2}\text{Adv}_{ChaCha20}^{\text{ind-cpa}}(A)
\end{aligned}$$

□

Security: PRF* Reduction

Even more rudimentary than the PRG is the underlying operation $\pi(h_j) \oplus h_j$. Assuming π is an ideal permutation (both the challenger and adversary have access to it), we can analyze this operation under PRF.



Theorem: Given an adversary B which attacks ChaCha20's underlying PRG G, there exists an adversary C which attacks the block cipher $E_K(m) = m \oplus \pi(m)$ with accuracy

$$\mathrm{Adv}_G^{\mathrm{PRF}*}(B) = 2 \cdot \mathrm{Adv}_P^{PRF}(C)$$

Security: PRF* Reduction

Proof. Suppose we have an adversary B which attacks G. Then let us construct Subroutine SimEnc(n):

```
static $N = \emptyset$
**if** $|n| \neq 64$ **then**
    **return** $\perp$
**end if**
**if** $n \in N$ **then**
    **return** $\perp$
**end if**
$N = N \cup \{n\}$
**if** $b = 0$ **then**
    $c \xleftarrow{\$} \{0,1\}^{512 \times 2^{64}}$
    **return** $c$
**else**
    $c \leftarrow 0^{512 \times 2^{64}}$
    Parse $c$ as 512-bit blocks $c[0]c[1] \ldots c[2^{64}-2]c[2^{64}-1]$
    **for** $i = 0; i < 2^{64}; i++$ **do**
        $y \leftarrow \text{pad}(m, i, n)$
        $c[i] = P(y)$
    **end for**
    **return** $c$
**end if**
```

Security: PRF* Reduction

and adversary C^P:

```
$b \xleftarrow{\$} \{0,1\}$
$m \xleftarrow{\$} \{0,1\}^{256}$
Run $B^{\mathrm{SimEnc}}$ until it returns $d$
**return** d=b
```

Then notice that when P is really $m \oplus \pi(m)$, the $\mathrm{SimEnc}(n)$ subroutine returns exactly the output of G when asked to encipher n, and when P outputs a random bitstring, B has a probability $\frac{1}{2}$ to guess b correctly, so C's advantage is

$$\begin{aligned}
\mathrm{Adv}_G^{PRF}(C) &= 2 \cdot \Pr[\mathrm{Exp}_G^{PRF}(C) \to \mathrm{true}] - 1 \\
&= 2 \cdot (\frac{1}{2} \Pr[\mathrm{Exp}_G^{\mathrm{PRF}*}(B) \to \mathrm{true}] + \frac{1}{4}) - 1 \\
&= \Pr[\mathrm{Exp}_G^{\mathrm{PRF}*}(B) \to \mathrm{true}] - \frac{1}{2} \\
&= \frac{1}{2}(2 \cdot \Pr[\mathrm{Exp}_G^{\mathrm{PRF}*}(A) \to \mathrm{true}] - 1) \\
&= \frac{1}{2}\mathrm{Adv}_G^{\mathrm{PRF}*}(B)
\end{aligned}$$

$\square$

Security: Final Thoughts

We now know that given an adversary that can perform an IND-CPA attack on ChaCha20 with advantage A, we can construct an adversary that can perform a PRF attack on P with advantage ¼A. This seems like a weak security definition, since P lacks security; however, the next step would be to bound the advantage of an attacker on P by the number of queries to P and π it makes (See Exercise 4.23 in Boneh and Shoup). Then we can show that an attacker against ChaCha20 with advantage A would require at least a certain number of queries to P or π.

References

1. Boneh, Dan and Shoup, Victor. *A Graduate Course in Applied Cryptography*. Version 0.6, 2023.