ChaCha20

簡介 -

1. 是一種 stream cipher，Salsa20 的改良版，更能抵抗密碼分析攻擊。
2. 加密過程：把 plaintext 和 keystream 做 xor 運算，解密過程：把 ciphertext 和 keystream 做 xor 運算，所以安全性取決於 PRNG 強度。
3. 有被運用在 FreeBSD, OpenBSD 和 NetBSD 等作業系統中的 arc4random 亂數生成器，取代已經脆弱的 RC4。
4. 不能保證 authenticity，攻擊者可能會操縱傳輸中的數據。
5. 承 4.為了防止這一點，ChaCha20 必須和 Message Authentication Code 一起使用，來達成網路安全通訊，例如 ChaCha20-Poly1305。
6. 在沒有對 AES 指令做加速的 CPU 上 ChaCha20 的 performance 通常比 AES 好

運算過程 -

1. 初始矩陣：

key (256 bit)、nonce (96 bit)、counter (32 bit)、constant(128 bit)，填充在 32-bit 數組中，排列如下：

constant[0] 0x61707865	constant[1] 0x3320646e	constant[2] 0x79622d32	constant[3] 0x6b206574
key[0]	key[1]	key[2]	key[3]
key[4]	key[5]	key[6]	key[7]
counter	nonce[0]	nonce[1]	nonce[2]

2. 置換：

ChaCha20 有 20 個 round，在每一輪執行 QUARTERROUND 之前必須先做置換
奇數 round - 對矩陣做行置換; 偶數 round - 對矩陣做列置換

行置換 - $M_{next}[i][j] = M_{current}[j][i]$

ex.

X[0]	X[1]	X[2]	X[3]
X[4]	X[5]	X[6]	X[7]
X[8]	X[9]	X[A]	X[B]
X[C]	X[D]	X[E]	X[F]

↓

X[0]	X[4]	X[8]	X[C]
X[1]	X[5]	X[9]	X[D]
X[2]	X[6]	X[A]	X[E]
X[3]	X[7]	X[B]	X[F]

i 為列數，從上至下編號為 0-3
j 為行數，從左至右編號為 0-3

列置換 - $M_{next}[i][j] = M_{current}[(i+j)\%4][j]$

ex.

X[0]	X[4]	X[8]	X[C]
X[1]	X[5]	X[9]	X[D]
X[2]	X[6]	X[A]	X[E]
X[3]	X[7]	X[B]	X[F]

↓

X[0]	X[5]	X[A]	X[F]
X[1]	X[6]	X[B]	X[C]
X[2]	X[7]	X[8]	X[D]
X[3]	X[4]	X[9]	X[E]

3. QUARTERROUND：

一次對一列(4 個)數組做運算，因為有四列，一次只做 1/4，所以是 quarter。

ex. 做完第一次行置換的矩陣如下 -

X[0]	X[4]	X[8]	X[C]
X[1]	X[5]	X[9]	X[D]
X[2]	X[6]	X[A]	X[E]
X[3]	X[7]	X[B]	X[F]

依序執行 QUARTERROUND -

QUARTERROUND（X[0]，X[4]，X[8]，X[C]）；

QUARTERROUND（X[1]，X[5]，X[9]，X[D]）；

QUARTERROUND（X[2]，X[6]，X[A]，X[E]）；

QUARTERROUND（X[3]，X[7]，X[B]，X[F]）；

QUARTERROUND 做的事情如下(令輸入為 a, b, c, d)

```
a ⊞= b; d ⊕= a; d <<<= 16;
c ⊞= d; b ⊕= c; b <<<= 12;
a ⊞= b; d ⊕= a; d <<<= 8;
c ⊞= d; b ⊕= c; b <<<= 7;
```

⊞是 bitwise add,

⊕是 bitwise xor,

<<<是 constant-distance rotation operation

例如 a=0x11111111，b=0x01020304，c=0x77777777，d=0x01234567

```
c ⊞= d; b ⊕= c; b <<<= 7
```

c ⊞= d = 0x77777777 ⊞ 0x01234567 = 0x789abcde

b ⊕= d = 0x01020304 ⊕ 0x789abcde = 0x7998bfda

b <<< 7 = 0x7998bfda <<< 7

= 01111001100110001011111111011010 <<< 7

= 11001100010111111110110100111100 = 0xcc5fed3c

4. 生成 keystream：

初始矩陣經過 20 round 之後，會得到一個新的 4*4 矩陣，將新矩陣與初始矩陣做向量相加，得到 4*4*64=512bits(=64bytes)的 keystream。

5. 加密：

將明文和 keystream 按位做 xor，得到密文，若明文長度>512bits，則將 counter+1，再按照上述步驟執行一遍生成新的 keystream。

因為 counter 有 32 bits，理論上可以生成 2^32*64 bytes (=256GiB)的 keystream。

6. 解密：

用事先協商好的初始 key、nonce、constant、counter 依照 ChaCha20 產生與加密時相同的 keystream，和密文做 xor 運算，即可得到明文。

實驗方法和結果分析 -

[Security]

因為 ChaCha20 是 stream cipher，安全性取決於 PRNG 的強度，故可以透過 NIST SP 800-22 及 AIS-31 兩種 test-suite 來作為衡量其安全性的標準之一。

[Efficiency]

透過 pyRAPL 來測量加解密過程中的 energy consumption，作為衡量效率的標準之一。

1. NIST SP 800-22：

(1) 使用 randomgen 裡的 ChaCha，可以改變 round 數(必須>=2, 且為偶數)

(2) bit stream length = 512000, number of bit streams = 264

(3) 因為據說 ChaCha20, ChaCha12, ChaCha8 是比較安全的，所以先分別測試 round = 20/12/8 的結果，發現都可以通過全部測試。

(4) 之後再遞減 round 數做測試，發現 ChaCha6 和 ChaCha4 基本上也都可以通過全部測試，多測幾次偶爾會發生有 1, 2 項測試沒過的情形；ChaCha2 則是完全無法通過全部測試。

2. AIS-31：

(1) 使用 randomgen 裡的 ChaCha，round = 20。

(2) 執行截圖如下 -

(3) ChaCha20 所產生的 random bits, 通過了所有測試(P1: T0, P1:T1-T5, P2)

拿 P2 的結果翻譯成英文看一下：

[18:44:07]

TEST STARTED.

[18:44:07] TEST SUITE: P2 (specific tests)

[18:44:07] Filename: chacha20.bin

[18:44:07] ISSUE DETAILS: Enabled.

[18:44:07] DATA FORMAT: 1 file byte contains 8 random bits.

[18:44:07] TEST START: Normal test.

[18:44:07] RND BIT WIDTH: 256 bits.

[18:44:07] The file is being read.

[18:44:07] Copy BitStream file to RAM ...

[18:44:08] Convert file data to ByteStream ...

[18:44:09] Write remaining file: chacha20.bin_rest

[18:53:13] 7200000 elements copied to RAM.

[18:53:13] The file was read.

[18:53:13] Test procedure T6a for verification of P2.i) (vii.a) started.

[18:53:13] | P (1) - 0.5 | = 2.2999999999995246E-4

[18:53:13] Last element: 100000

[18:53:13] Test procedure T6a passed.

[18:53:13] Test procedure T6b for verification of P2.i) (vii.b) started.

[18:53:13] p (01) = 0.49882

[18:53:13] p (11) = 0.50245

[18:53:13] | p_ (01) - p_ (11) | = 0.0036299999999999666

[18:53:13] Last element: 500396

[18:53:13] Test procedure T6b passed.

[18:53:13] Test procedure T7a for verification of P2.i) (vii.c) started.

[18:53:13] Test size [0] = 0.6055202799925774

[18:53:13] Test size [1] = 3.362037956063709

[18:53:13] Last element: 1706360

[18:53:13] Test procedure T7a passed.

[18:53:13] Test procedure T7b for verification of P2.i) (vii.d) started.

[18:53:13] Test size [0] = 0.0028800007787522105

[18:53:13] Test size [1] = 0.8323235320481406

[18:53:13] Test size [2] = 0.05832000149299203

[18:53:13] Test size [3] = 1.1045001512060706

[18:53:13] Last element: 4926360

[18:53:13] Test procedure T7b passed.

[18:53:13] Test T8 to verify P2.i) (vii.e) started.

[18:53:14] Test size = 7.999502481982367

[18:53:14] Last element: 6994840

[18:53:14] Test T8 passed.

[18:53:14] Run finished successfully.

3. Energy Consumption

(1) 使用 pyRAPL 裡的 measureit，可以指定某段程式碼要執行、測量幾次。

(2) 測量時，把所有其他程式關掉，以免影響結果。

(3) 一次執行包含加密和解密，各執行、測量 100 次。

(4) 部分結果截圖：

	A	B	C	D	E	F
1	label	timestamp	duration	pkg	dram	socket
2	chacha_once	1622986486	37.31137	623.16	36.01	0
3	chacha_once	1622986486	38.15675	558.47	29.29	0
4	chacha_once	1622986486	33.37269	378.42	17.7	0
5	chacha_once	1622986486	43.62725	484.62	20.75	0
6	chacha_once	1622986486	39.40691	532.23	33.57	0
7	chacha_once	1622986486	33.43253	494.99	24.41	0
8	chacha_once	1622986486	39.27992	548.71	28.08	0
9	chacha_once	1622986486	38.93365	519.4	31.74	0
10	chacha_once	1622986486	44.98399	617.68	32.35	0

Duration：執行一次經過的時間 單位：μs

PKG : CPU energy consumption 單位：μJ

DRAM : RAM energy consumption 單位：μJ

(5) 程式碼截圖：

```
import os
from Crypto.Cipher import ChaCha20
import pyRAPL

pyRAPL.setup()

csv_output = pyRAPL.outputs.CSVOutput('chacha20_result.csv')

@pyRAPL.measureit(number=100, output=csv_output)
def chacha_once():
  plaintext = b'test'
  key = os.urandom(32)
  nonce = os.urandom(12)

  encoder = ChaCha20.new(key=key, nonce=nonce)
  ciphertext = encoder.encrypt(plaintext)

  decoder = ChaCha20.new(key=key, nonce=nonce)
  plaintext = decoder.decrypt(ciphertext)

for _ in range(100):
  chacha_once()

csv_output.save()
```


參考資料 -

1. https://zh.wikipedia.org/wiki/Salsa20
2. https://baike.baidu.com/item/chacha20-poly1305
3. https://bashtage.github.io/randomgen/devel/bit_generators/generated/randomgen.chacha.ChaCha.random_raw.html#randomgen.chacha.ChaCha.random_raw
4. https://pycryptodome.readthedocs.io/en/latest/src/cipher/chacha20.html
5. https://pypi.org/project/pyRAPL/