The Stream Cipher HC-128

Hongjun Wu

Katholieke Universiteit Leuven, ESAT/SCD-COSIC
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
wu.hongjun@esat.kuleuven.be

Statement 1. HC-128 supports 128-bit key and 128-bit initialization vector.
Statement 2. 2^{64} keystream bits can be generated from each key/IV pair.
Statement 3. There is no hidden flaw in HC-128.
Statement 4. The smallest period is expected to be much larger than 2^{128}.
Statement 5. Recovering the secret key is as difficult as exhaustive key search.
Statement 6. Distinguishing attack requires more than 2^{64} keystream bits.
Statement 7. There is no weak key in HC-128.
Statement 8. Encryption speed is 3.05 cycles/byte on Pentium M processor.
Statement 9. The key and IV setup takes about 27,300 clock cycles
Statement 10. HC-128 is not covered by any patent and it is freely available.

Remarks. When more than 2^{64} keystream bits are generated from each key/IV pair, the effect on the security of the message/key is negligible. Thus there is no need to implement any mechanism to restrict the keystream length in practice.

1 Introduction

Stream cipher HC-128 is the simplified version of HC-256 [15] for 128-bit security. HC-128 is a simple, secure, software-efficient cipher and it is freely-available.

HC-128 consists of two secret tables, each one with 512 32-bit elements. At each step we update one element of a table with non-linear feedback function. All the elements of the two tables get updated every 1024 steps. At each step, one 32-bit output is generated from the non-linear output filtering function.

HC-128 is suitable for the modern (and future) superscalar microprocessors. The dependency between operations in HC-128 is very small: three consecutive steps can be computed in parallel; at each step, the feedback and output functions can be computed in parallel. The high degree of parallelism allows HC-128 to run efficiently on the modern processor. We implemented HC-128 in C, and the encryption speed of HC-128 reaches 3.05 cycles/byte on the Intel Pentium M processor.

HC-128 is very secure. Our analysis shows that recovering the key of HC-128 is as difficult as exhaustive key search. To distinguish the keystream from random, we expect that more than 2^{64} keystream bits are required (our current analysis shows that about 2^{151} outputs are needed in the distinguishing attack).

This report is organized as follows. We introduce HC-128 in Section 2. The security analysis of HC-128 is given in Section 3 and Section 4. Section 5 discusses the implementation and performance of HC-128. Section 6 concludes this report.

2 Cipher Specifications

In this section, we describe the stream cipher HC-128. From a 128-bit key and a 128-bit initialization vector, it generates keystream with length up to 2^{64} bits.

2.1 Operations, variables and functions

The following operations are used in HC-128:

- $+$: $x + y$ means $x + y \bmod 2^{32}$, where $0 \le x < 2^{32}$ and $0 \le y < 2^{32}$
- $\boxminus$: $x \boxminus y$ means $x - y \bmod 512$
- $\oplus$: bit-wise exclusive OR
- $\|$: concatenation
- $\gg$: right shift operator. $x \gg n$ means x being right shifted n bits.
- $\ll$: left shift operator. $x \ll n$ means x being left shifted n bits.
- $\ggg$: right rotation operator. $x \ggg n$ means $((x \gg n) \oplus (x \ll (32-n))$ where $0 \le n < 32$, $0 \le x < 2^{32}$.
- $\lll$: left rotation operator. $x \lll n$ means $((x \ll n) \oplus (x \gg (32-n))$ where $0 \le n < 32$, $0 \le x < 2^{32}$.

Two tables P and Q are used in HC-128. The key and the initialization vector of HC-128 are denoted as K and IV. We denote the keystream being generated as s.

- P : a table with 512 32-bit elements. Each element is denoted as $P[i]$ with $0 \le i \le 511$.
- Q : a table with 512 32-bit elements. Each element is denoted as $Q[i]$ with $0 \le i \le 511$.
- K : the 128-bit key of HC-128.
- IV : the 128-bit initialization vector of HC-128.
- s : the keystream being generated from HC-128. The 32-bit output of the ith step is denoted as s_i. Then $s = s_0 \| s_1 \| s_2 \| \cdots$

There are six functions being used in HC-256. $f_1(x)$ and $f_2(x)$ are the same as the $\sigma_0^{\{256\}}(x)$ and $\sigma_1^{\{256\}}(x)$ being used in the message schedule of SHA-256 [14]. For $h_1(x)$, the table Q is used as S-box. For $h_2(x)$, the table P is used as S-box.

$$
\begin{aligned}
f_1(x) &= (x \ggg 7) \oplus (x \ggg 18) \oplus (x \gg 3) \\
f_2(x) &= (x \ggg 17) \oplus (x \ggg 19) \oplus (x \gg 10) \\
g_1(x,y,z) &= ((x \ggg 10) \oplus (z \ggg 23)) + (y \ggg 8) \\
g_2(x,y,z) &= ((x \lll 10) \oplus (z \lll 23)) + (y \lll 8) \\
h_1(x) &= Q[x_0] + Q[256 + x_2] \\
h_2(x) &= P[x_0] + P[256 + x_2]
\end{aligned}
$$

where $x = x_3 \| x_2 \| x_1 \| x_0$, x is a 32-bit word, x_0, x_1, x_2 and x_3 are four bytes. x_3 and x_0 denote the most significant byte and the least significant byte of x, respectively.

2.2 Initialization process (key and IV setup)

The initialization process of HC-128 consists of expanding the key and initialization vector into P and Q (similar to the message setup in SHA-256) and running the cipher 1024 steps (with the outputs being used to update P and Q).

1. Let $K = K_0 \parallel K_1 \parallel K_2 \parallel K_3$ and $IV = IV_0 \parallel IV_1 \parallel IV_2 \parallel IV_3$, where each K_i and IV_i denotes a 32-bit number. Let $K_{i+4} = K_i$, and $IV_{i+4} = IV_i$ for $0 \le i < 4$. The key and IV are expanded into an array W_i $(0 \le i \le 1279)$ as:

$$W_i = \begin{cases} K_i & 0 \le i \le 7 \\ IV_{i-8} & 8 \le i \le 15 \\ f_2(W_{i-2}) + W_{i-7} + f_1(W_{i-15}) + W_{i-16} + i & 16 \le i \le 1279 \end{cases}$$

2. Update the tables P and Q with the array W.

$$\begin{aligned} P[i] &= W_{i+256} \qquad \text{for } 0 \le i \le 511 \\ Q[i] &= W_{i+768} \qquad \text{for } 0 \le i \le 511 \end{aligned}$$

3. Run the cipher 1024 steps and use the outputs to replace the table elements as follows ("$\boxminus$" denotes "$-$" modulo 512).

for $i = 0$ to 511, do
$P[i] = (P[i] + g_1(P[i \boxminus 3], P[i \boxminus 10], P[i \boxminus 511])) \oplus h_1(\, P[i \boxminus 12]\,)$;

for $i = 0$ to 511, do
$Q[i] = (Q[i] + g_2(Q[i \boxminus 3], Q[i \boxminus 10], Q[i \boxminus 511])) \oplus h_2(\, Q[i \boxminus 12]\,)$;

The initialization process completes and the cipher is ready to generate keystream.

2.3 The keystream generation algorithm

At each step, one element of a table is updated and one 32-bit output is generated. Each S-box is used to generate only 512 outputs, then it is updated in the next 512 steps. The keystream generation algorithm of HC-128 is given below ("$\boxminus$" denotes "$-$" modulo 512, s_i denotes the output of the i-th step).

$i = 0$;
repeat until enough keystream bits are generated.
{
  $j = i \bmod 512$;
  if ($i \bmod 1024) < 512$
  {
    $P[j] = P[j] + g_1(P[j \boxminus 3], P[j \boxminus 10], P[j \boxminus 511])$;
    $s_i = h_1(\, P[j \boxminus 12]\,) \oplus P[j]$;
  }
  else
  {
    $Q[j] = Q[j] + g_2(Q[j \boxminus 3], Q[j \boxminus 10], Q[j \boxminus 511])$;
    $s_i = h_2(\, Q[j \boxminus 12]\,) \oplus Q[j]$;
  }
  end-if
  $i = i + 1$;
}
end-repeat

3 Security Analysis of HC-128

The security analysis of HC-128 is similar to that of HC-256. The output and feedback functions of HC-128 are non-linear, so it is impossible to apply the fast correlation attacks [12, 9, 13, 5, 11] and algebraic attacks [1, 6–8] to recover the secret key of HC-128. The large secret S-box of HC-128 is updated during the keystream generation process, so it is very difficult to develop linear relations linking the input and output bits of the S-box.

In this section, we will analyze the period of HC-128, the security of the secret key and the security of the initialization process. The randomness of the keystream will be analyzed separately in Section 4.

3.1 Period

The 32778-bit state of HC-128 ensures that the period of the keystream is extremely large. But the exact period of HC-128 is difficult to predict. The average period of the keystream is estimated to be much more than 2^{256}. The large number of states also eliminates the threat of the time-memory-data tradeoff attack on stream ciphers [4] (also [2, 10]).

3.2 Security of the secret key

We note that the output function and the feedback function of HC-128 are non-linear. The non-linear output function leaks small amount of partial information at each step. The non-linear feedback function ensures that the secret key can not be recovered from those leaked partial information.

3.3 Security of the initialization process (key/IV setup)

The initialization process of the HC-128 consists of two stages, as given in Subsection 2.2. We expand the key and IV into P and Q. At this stage, every bit of the key/IV affects all the bits of the two tables and any difference in the related keys/IVs results in uncontrollable differences in P and Q. Note that the constants in the expansion function at this stage play significant role in reducing the effect of related keys/IVs. After the expansion, we run the cipher 1024 steps and using the outputs to update the P and Q. After the initialization process, we expect that any difference in the keys/IVs would not result in biased keystream.

4 Randomness of the keystream

Our initial analysis shows that the distinguishing attack on HC-128 requires more than 2^{128} outputs. The analysis is given below.

We recall that at the ith step, if $(i \bmod 1024) < 512$, the table P is updated as

$$P[i \bmod 512] = P[i \bmod 512] + g_1(\, P[i \boxminus 3], P[i \boxminus 10], P[i \boxminus 511]\,)$$

We know that $s_i = h_1(P[i \boxminus 12]) \oplus P[i \bmod 512]$. For $10 \leq (i \bmod 1024) < 511$, this feedback function can be written alternatively as

$$\begin{aligned} s_i \oplus h_1(z_i) = (s_{i-1024} \oplus h_1'(z_{i-1024})) + \\ g_1(s_{i-3} \oplus h_1(z_{i-3}), s_{i-10} \oplus h_1(z_{i-10}), s_{i-1023} \oplus h_1'(z_{i-1023})) \end{aligned} \quad (1)$$

where $h_1(x)$ and $h_1'(x)$ indicate two different functions since they are related to different S-boxes; z_j denotes the $P[j \boxminus 12]$ at the j-th step.

We note that there are two '+' operations in the feedback function. We will first investigate the least significant bits in the feedback function since they are not affected by the '+' operations. Denote the i-th least significant bit of a as a^i. From (1), we obtain that for $10 \leq (i \bmod 1024) < 511$,

$$\begin{aligned} & s_i^0 \oplus s_{i-1024}^0 \oplus s_{i-3}^{10} \oplus s_{i-10}^8 \oplus s_{i-1023}^{23} \\ = & (h_1(z_i))^0 \oplus (h_1'(z_{i-1024}))^0 \oplus (h_1(z_{i-3}))^{10} \oplus \\ & \oplus (h_1(z_{i-10}))^8 \oplus (h_1'(z_{i-1023}))^{23} \end{aligned} \quad (2)$$

Similarly, for $1024 \times \alpha + 10 \leq i, j < 1024 \times \alpha + 511$ and $j \neq i$, we obtain

$$\begin{aligned} & s_j^0 \oplus s_{j-1024}^0 \oplus s_{j-3}^{10} \oplus s_{j-10}^8 \oplus s_{j-1023}^{23} \\ = & (h_1(z_j))^0 \oplus (h_1'(z_{j-1024}))^0 \oplus (h_1(z_{j-3}))^{10} \oplus \\ & \oplus (h_1(z_{j-10}))^8 \oplus (h_1'(z_{j-1023}))^{23} \end{aligned} \quad (3)$$

For the left sides of (2) and (3) to be equal, i.e., for the following equation

$$\begin{aligned} &s_i^0 \oplus s_{i-1024}^0 \oplus s_{i-3}^{10} \oplus s_{i-10}^8 \oplus s_{i-1023}^{23} = \\ &s_j^0 \oplus s_{j-1024}^0 \oplus s_{j-3}^{10} \oplus s_{j-10}^8 \oplus s_{j-1023}^{23} \end{aligned} \tag{4}$$

to hold, we require that

$$\begin{aligned} &(h_1(z_i))^0 \oplus (h_1'(z_{i-1024}))^0 \oplus (h_1(z_{i-3}))^{10} \\ &\oplus (h_1(z_{i-10}))^8 \oplus (h_1'(z_{i-1023}))^{23} \\ =\ &(h_1(z_j))^0 \oplus (h_1'(z_{j-1024}))^0 \oplus (h_1(z_{j-3}))^{10} \\ &\oplus (h_1(z_{j-10}))^8 \oplus (h_1'(z_{j-1023}))^{23} \end{aligned} \tag{5}$$

Approximate (5) as

$$H(x_1) = H(x_2) \tag{6}$$

where H denotes a random secret 80-bit-to-1-bit S-box, x_1 and x_2 are two 80-bit random inputs, $x_1 = \overline{z}_i \parallel \overline{z}_{i-3} \parallel \overline{z}_{i-10} \parallel \overline{z}_{i-1023} \parallel \overline{z}_{i-1024}$ and $x_2 = \overline{z}_j \parallel \overline{z}_{j-3} \parallel \overline{z}_{j-10} \parallel \overline{z}_{j-1023} \parallel \overline{z}_{j-1024}$, where $\overline{z}$ indicates the concatenation of the least significant byte and the second most significant byte of z. The following theorem gives the collision rate of the outputs of $H(x)$.

Theorem 1. *Let H be an m-bit-to-n-bit S-box and all those n-bit elements are randomly generated, where $m \geq n$. Let x_1 and x_2 be two m-bit random inputs to H. Then $H(x_1) = H(x_2)$ with probability $2^{-m} + 2^{-n} - 2^{-m-n}$.*

Proof. If $x_1 = x_2$, then $H(x_1) = H(x_2)$. If $x_1 \neq x_2$, then $H(x_1) = H(x_2)$ with probability 2^{-n}. $x_1 = x_2$ with probability 2^{-m} and $x_1 \neq x_2$ with probability $1 - 2^{-m}$. The probability that $H(x_1) = H(x_2)$ is $2^{-m} + (1 - 2^{-m}) \times 2^{-n}$.

According to Theorem 1, (6) holds with probability $\frac{1}{2} + 2^{-81}$. So (4) holds with probability $\frac{1}{2} + 2^{-81}$. After testing the validity of 2^{164} equations (4), the output of the cipher can be distinguished from random signal with success rate 0.9772 (with false negative rate and false positive rate as 0.0228). Note that only about 2^{17} equations (4) can be obtained from every 512 outputs, this distinguishing attack requires about 2^{156} outputs.

We note that the attack above only deals with the least significant bit in (1). It may be possible to consider the rest of the 31 bits bit-by-bit. But due to the effect of the two '+' operations in the feedback function, the attack exploiting those 31 bits is not as effective as that exploiting the least significant bit. Thus more than 2^{151} outputs are needed in this distinguishing attack.

It may be possible that the distinguishing attack against HC-128 can be improved in the future. However, it is very unlikely that our security goal can be breached since the security margin is extremely large. We thus conjecture that it is computationally impossible to distinguish 2^{64} bits keystream of HC-128 from random.

5 Implementation and Performance of HC-128

The optimized implementation of HC-128 is similar to that of HC-256. On the Pentium M processor, the speed of HC-128 reaches 3.05 cycles/bye, while the speed of HC-256 is about 4.4 cycles/byte.

5.1 The optimized implementation of HC-128

In the optimized code, loop unrolling is used and only one branch decision is made for every 16 steps. The details of the implementation are given below. The feedback function of P is given as

$$P[i \bmod 512] = P[i \bmod 512] + P[i \boxminus 10] + g_1(\, P[i \boxminus 3], P[i \boxminus 511]\,)$$

A register X containing 16 elements is introduced for P. If $(i \bmod 1024) < 512$ and $i \bmod 16 = 0$, then at the begining of the ith step, $X[j] = P[(i - 16 + j) \bmod 512]$ for $j = 0, 1, \cdots 15$, i.e. the X contains the values of $P[i \boxminus 16], P[i \boxminus 15], \cdots, P[i \boxminus 1]$. In the 16 steps starting from the ith step, the P and X are updated as

$$\begin{aligned}
P[i] &= P[i] + g_1(\, X[13], X[6], P[i+1]\,);\\
X[0] &= P[i];\\
P[i+1] &= P[i+1] + g_1(\, X[14], X[7], P[i+2]\,);\\
X[1] &= P[i+1];\\
P[i+2] &= P[i+2] + g_1(\, X[15], X[8], P[i+3]\,);\\
X[2] &= P[i+2];\\
P[i+3] &= P[i+3] + g_1(\, X[0], X[9], P[i+4]\,);\\
X[3] &= P[i+3];\\
&\cdots\\
P[i+14] &= P[i+14] + g_1(\, X[11], X[4], P[i+15]\,);\\
X[14] &= P[i+14];\\
P[i+15] &= P[i+15] + g_1(\, X[12], X[5], P[(i+1) \bmod 512]\,);\\
X[15] &= P[i+15];
\end{aligned}$$

Note that at the ith step, two elements of $P[i \boxminus 10]$ and $P[i \boxminus 3]$ can be obtained directly from X. Also for the output function $s_i = h_1(P[i \boxminus 12]) \oplus P[i \bmod 1024]$, the $P[i \boxminus 12]$ can be obtained from X. In this implementation, there is no need to compute $i \boxminus 3$, $i \boxminus 10$ and $i \boxminus 12$.

A register Y with 16 elements is used in the implementation of the feedback function of Q in the same way as that given above.

5.2 The performance of HC-128

Encryption Speed. We use the C codes submitted to the eStream to measure the encryption speed. The processor used in the measurement is the Intel Pentium M (1.6 GHz, 32 KB Level 1 cache, 2 MB Level 2 cache).

Using the eStream performance testing framework, the highest encryption speed of HC-128 is 3.05 cycles/byte with the compiler gcc (there are three optimization options leading to this encryption speed: k8 O3-ual-ofp, prescott O2-ofp and athon O3-ofp). Using the Intel C++ Compiler 9.1 in Windows XP (SP2), the speed is 3.3 cycles/byte. Using the Microsoft Visual C++ 6.0 in Windows XP (SP2), the speed is 3.6 cycles/byte.

Initialization Process. The key setup of HC-128 requires about 27,300 clock cycles. There are two large S-boxes in HC-128. In order to eliminate the threat of related key/IV attack, the tables should be updated with the key and IV thoroughly and this process requires a lot of computations. It is thus undesirable to use HC-128 in the applications where key (or IV) is updated very frequently.

6 Conclusion

In this report, a software-efficient stream cipher HC-128 is illustrated. Our analysis shows that HC-128 is very secure. However, the extensive security analysis of any new cipher requires a lot of efforts from many researchers. We encourage the readers to analyze the security of HC-128.

References

1. F. Armknecht, M. Krause, "Algebraic Attacks on Combiners with Memory", in *Advances in Cryptology – Crypto 2003*, LNCS 2729, pp. 162-75, Springer-Verlag, 2003.
2. S. Babbage, "A Space/Time Tradeoff in Exhaustive Search Attacks on Stream Ciphers", European Convention on Security and Detection, IEE Conference publication, No. 408, May 1995.
3. E. Biham, A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems", in *Advances in Cryptology – Crypto'90*, LNCS 537, pp. 2-21, Springer-Verlag, 1991.
4. A.Biryukov, A.Shamir, "Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers", in *Advances in Cryptography – ASIACRYPT'2000*, LNCS 1976, pp.1–13, Springer-Verlag, 2000.
5. V.V. Chepyzhov, T. Johansson, and B. Smeets. "A Simple Algorithm for Fast Correlation Attacks on Stream Ciphers", in *Fast Software Encryption (FSE'00)*, LNCS 1978, pp. 181-195, Springer-Verlag, 2000.
6. N. Courtois, "Higher Order Correlation Attacks, XL algorithm and Cryptanalysis of Toyocrypt", in *(ICISC 2002)*, LNCS 2587, pp. 182-199, Springer-Verlag, 2002.
7. N. Courtois, and W. Meier, "Algebraic Attacks on Stream Ciphers with Linear Feedback", in *Advances in Cryptology – Eurocrypt 2003*, LNCS 2656, pp. 345?59, Springer-Verlag, 2003.
8. N. Courtois, "Fast Algebraic Attacks on Stream Ciphers with Linear Feedback", in *Advances in Cryptology – Crypto 2003*, LNCS 2729, pp. 176?94, Springer-Verlag, 2003.
9. J. D. Golić, "Towards Fast Correlation Attacks on Irregularly Clocked Shift Registers", in *Advances in Cryptography – Eurocrypt'95*, pages 248-262, Springer-Verlag, 1995.

10. J. D. Golić, "Cryptanalysis of Alleged A5 Stream Cipher", in *Advances in Cryptology – Eurocrypt'97*, LNCS 1233, pp. 239 - 255, Springer-Verlag, 1997.
11. T. Johansson and F. Jönsson. "Fast Correlation Attacks through Reconstruction of Linear Polynomials", in *Advances in Cryptology – CRYPTO 2000*, LNCS 1880, pp. 300-315, Springer-Verlag, 2000.
12. W. Meier and O. Staffelbach, "Fast Correlation Attacks on Certain Stream Ciphers". *Journal of Cryptography*, 1(3):159-176, 1989.
13. M. Mihaljević, M.P.C. Fossorier, and H. Imai, "A Low-Complexity and High-Performance Algorithm for Fast Correlation Attack", in *Fast Software Encryption (FSE'00)*, pp. 196-212, Springer-Verlag, 2000.
14. National Institute of Standards and Technology, "Secure Hash Standard (SHS)", Federal Information Processing Standards Publication (FIPS) 180-2. Available at http://csrc.nist.gov/publications/fips/
15. H. Wu, "A New Stream Cipher HC-256", in *Fast Software Encryption (FSE'04)*, LNCS 3017, pp. 226-244. The full version is available at http://eprint.iacr.org/2004/092.pdf

A Test Vectors of HC-128

Let $K = K_0||K_1||\cdots||K_7$ and $IV = IV_0||IV_1||\cdots||IV_7$. The first 512 bits of keystream are given for different values of key and IV. Note that for each 32-bit output given below, the least significant byte leads the most significant byte in the keystream. For example, if S and T are 32-bit words, and $S = s_3||s_2||s_1||s_0$, $T = t_3||t_2||t_1||t_0$, where each s_i and t_i is one byte, and s_0 and t_0 denote the least significant bytes, then the keystream S, T is related to the keystream $s_0, s_1, s_2, s_3, t_0, t_1, t_2, t_3$.

1. The key and IV are set as 0.

```
73150082  3bfd03a0  fb2fd77f  aa63af0e
de122fc6  a7dc29b6  62a68527  8b75ec68
9036db1e  81896005  00ade078  491fbf9a
1cdc3013  6c3d6e24  90f664b2  9cd57102
```

2. The key is set as 0, the IV is set as 0 except that $IV_0 = 1$.

```
c01893d5  b7dbe958  8f65ec98  64176604
36fc6724  c82c6eec  1b1c38a7  c9b42a95
323ef123  0a6a908b  ce757b68  9f14f7bb
e4cde011  aeb5173f  89608c94  b5cf46ca
```

3. The IV is set as 0, the key is set as 0 except that $K_0 = 0x55$.

```
518251a4  04b4930a  b02af931  0639f032
bcb4a47a  5722480b  2bf99f72  cdc0e566
310f0c56  d3cc83e8  663db8ef  62dfe07f
593e1790  c5ceaa9c  ab03806f  c9a6e5a0
```

Let $A_i = \bigoplus_{j=0}^{0x\text{fffff}} s_{16j+i}$ for $i = 0, 1, \cdots, 15$, i.e. set a 512-bit buffer as 0 and encrypt it repeatedly for 2^{20} times. Set the key and IV as 0, the value of $A_0||A_1||\cdots||A_{15}$ is given below:

```
a4eac026  7e491126  6a2a384f  5c4e1329
da407fa1  55e6b1ae  05c6fdf3  bbdc8a86
7a699aa0  1a4dc117  63658ccc  d3e62474
9cf8236f  0131be21  c3a51de9  d12290de
```