

Improved Distinguishing Attacks on HC-256*

Gautham Sekar^{1,2,†}, and Bart Preneel^{1,2}

¹Department of Electrical Engineering ESAT/SCD-COSIC,
Katholieke Universiteit Leuven,
Kasteelpark Arenberg 10, B-3001 Heverlee, Belgium.

²Interdisciplinary Institute for BroadBand Technology (IBBT), Belgium.
{Gautham.Sekar, Bart.Preneel}@esat.kuleuven.be

Abstract

The software-efficient stream cipher HC-256 was proposed by Wu at FSE 2004. Due to its impressive performance, the cipher was also a well-received entrant to the ECRYPT eSTREAM competition. The closely related stream cipher HC-128, also designed by Wu, went on to find a place in the final portfolio of the eSTREAM contest. The cipher HC-256 is word-oriented, with 32 bits in each word, and uses a 256-bit key and a 256-bit *IV*. Since HC-256 was published in 2004, barring a cache-timing analysis of unprotected implementations, there has not been any attack on the cipher. This paper makes two contributions. First, we build a class of distinguishers on HC-256, each of which requires testing the validity of about $2^{276.8}$ linear equations involving binary keystream variables. Thereby, our attacks improve the data complexity of the hitherto best-known distinguisher (presented by the designer along with the specifications of the cipher) by a factor of about 12. We also present another observation that, we believe, can be further exploited to build more efficient distinguishing attacks on the cipher. It is hoped that the results of this paper would also find use in future security evaluations of the closely-related ciphers HC-128 and HC-256[†].

1 Introduction

HC-128 and HC-256 are software-oriented synchronous stream ciphers designed by Wu [12, 13]. HC-256 was published in 2004. The ciphers were also submitted to the ECRYPT eSTREAM competition [3] in 2005. On the Pentium M processor, the speed of HC-128 reaches 3.05 cycles/byte, while HC-256 requires about 4.15 cycles/byte on the Pentium 4. Due to these impressive performance figures, the ciphers were seen as forerunners in the stream cipher contest. In the absence of attacks, both HC-256 and HC-128 were advanced to Phase III of the competition as ‘focus’ ciphers. Since the main focus of eSTREAM was 128-bit security, HC-128 was recently selected for the final eSTREAM portfolio under Profile 1 (software-based stream ciphers). The ciphers belong to the family of array-based stream ciphers that include, among others, the RC4, ISAAC and Py [1, 4].

Barring a few interesting observations, HC-128 and HC-256 have not yet witnessed any serious attacks. The designer himself has presented distinguishers along with the specifications in [12, 13]. In the case of HC-256, each distinguisher requires testing the validity of 2^{280} equations (where each equation involves 10 keystream output bits). Another observation, made by Dunkelman in [2], shows that the keystream words of HC-128 leak information on the internal states. However, this observation has not yet been exploited to construct distinguishers or to recover the key. Zenner has presented cache-timing attacks on unprotected implementations of HC-256 that allow reconstruction of the inner state and also the key [14]. This attack requires 6148 precise cache timing measurements, 216 known plaintext bits, 3 MBytes of memory and a computational effort equivalent to testing about 255 keys. However, the attack uses very strong assumptions - under these assumptions any unprotected implementation of a

*The full version of this paper has been accepted to IWSEC 2009.

†This author is supported by an FWO project.

cipher based on lookup tables such as AES or RC4 could be broken easily. Recently, Maitra *et al.* have presented some observations on HC-128 in [5]. They have exploited the results of [11] (on linear approximation of modular addition of three integers) to show that the output generation of HC-128 can be well-approximated by linear functions. Using this they show that for HC-128, the distinguisher presented in [13] for the least significant bit can be extended for the other bits. Their paper also studies the aforementioned observation due to Dunkelman [2]. Yet, their paper does not show any improvement over the existing attacks (i.e., those presented by the designer along with the specifications of the cipher).

1.1 Contribution of This Paper

The main idea behind our distinguishers is to note that the keystream output word generation of HC-256 involves two elements of the state array directly which are 10 places apart. We exploit this to improve the distinguisher presented in [12]. Our attacks do not work immediately for HC-128 as in the keystream output generation no two elements of the state array are involved directly, but they are used with some rotation.

For the least significant bit, our analysis is similar to that in [12], but a more careful analysis shows that the bias probability was underestimated and thus the requirement of the keystream bits was overestimated in [12]. Our analysis improves the probability and thus our distinguishers require fewer keystream words. Each of our distinguishers requires examining about $2^{276.8}$ equations where each equation involves 8 keystream output bits.

This paper is organised as follows. Section 2 lists the notations used in the paper. Section 3 details the specifications of HC-256. Our main observation and the resulting distinguishing attacks are presented in Sect. 4. Our second observation is presented in Sect. 5. Section 6 concludes the paper and presents a few interesting open problems.

2 Notation and Convention

We use the following notations and conventions.

The set of natural numbers is denoted by \mathbb{N} .

The $+$ operator denotes addition modulo 2^{32} .

The $-$ operator denotes subtraction modulo 2^{32} .

The symbol \boxminus denotes subtraction modulo 1024.

The symbol \oplus denotes bitwise exclusive-OR.

Concatenation is denoted by \parallel .

The complement of event E is denoted by E^c .

$x \gg y$: x is shifted to the right by y bit-positions.

$x \ll y$: x is shifted to the left by y bit-positions.

$x \ggg y$: $((x \gg y) \oplus (x \ll (32 - y)))$, where $y \in \{0, \dots, 31\}$, $x \in \{0, \dots, 2^{32} - 1\}$.

$x \lll y$: $((x \ll y) \oplus (x \gg (32 - y)))$, where $y \in \{0, \dots, 31\}$, $x \in \{0, \dots, 2^{32} - 1\}$.

PRBG denotes the pseudorandom bit generation algorithm of the cipher.

The keystream word generated at round i (i.e., the $(i + 1)$ -th iteration of the PRBG) is denoted by s_i .

The terms $s_{i(j)}$, $(h_1(x))_j$, $(h_2(x))_j$, $r_{i(j)}$ and $(Q[r])_j$ denote the j -th bits ($j = 0$ for the least significant bit) of s_i , $h_1(x)$, $h_2(x)$, r_i and $Q[r]$, respectively.

The term *word* denotes a 32-bit integer.

If x is a word, then $x^{(i)}$ denotes the i -th byte of x , where $x^{(0)}$ is the least significant byte and $x^{(3)}$ is the most significant byte.

3 Specifications of HC-256

The cipher uses a 256-bit key K and a 256-bit IV . Like the designer's analysis in [12], our analysis is also based on the assumption of a perfect K/IV setup and focusses only on the PRBG of HC-256. Therefore, we omit the

description of the K/IV setup algorithm. The internal state of HC-256 consists of two tables P and Q , each with 1024 32-bit elements. The following functions are used in the PRBG.

$$\begin{aligned} g_1(x, y) &= ((x \ggg 10) \oplus (y \ggg 23)) + Q[(x \oplus y) \bmod 1024], \\ g_2(x, y) &= ((x \ggg 10) \oplus (y \ggg 23)) + P[(x \oplus y) \bmod 1024], \\ h_1(x) &= Q[x^{(0)}] + Q[256 + x^{(1)}] + Q[512 + x^{(2)}] + Q[768 + x^{(3)}], \\ h_2(x) &= P[x^{(0)}] + P[256 + x^{(1)}] + P[512 + x^{(2)}] + P[768 + x^{(3)}]. \end{aligned}$$

3.1 The PRBG

The PRBG of HC-256 updates only one of the two tables P and Q in each round and outputs one word.

```

i = 0;
repeat until enough keystream bits are generated.
{
  k = i mod 1024;
  if (i mod 2048) < 1024
  {
    P[k] = P[k] + P[k ⊕ 10] + g1(P[k ⊕ 3], P[k ⊕ 1023]);
    si = h1(P[k ⊕ 12]) ⊕ P[k];
  }
  else
  {
    Q[k] = Q[k] + Q[k ⊕ 10] + g2(Q[k ⊕ 3], Q[k ⊕ 1023]);
    si = h2(Q[k ⊕ 12]) ⊕ Q[k];
  }
  end-if
  i = i + 1;
}
end-repeat

```

4 Motivational Observation

First, we recall the analysis in [12] in which it is derived that for

$$\begin{aligned} s_{i(0)} \oplus s_{i-2048(0)} \oplus s_{i-10(0)} \oplus s_{i-3(10)} \oplus s_{i-2047(23)} = \\ s_{j(0)} \oplus s_{j-2048(0)} \oplus s_{j-10(0)} \oplus s_{j-3(10)} \oplus s_{j-2047(23)} \end{aligned} \quad (1)$$

to hold for $2048 \cdot \alpha + 10 \leq i, j < 2048 \cdot \alpha + 1023$ ($i \neq j$),¹ it is required that

$$\begin{aligned} (h_1(z_i))_0 \oplus (h'_1(z_{i-2048}))_0 \oplus (h_1(z_{i-10}))_0 \\ \oplus (h_1(z_{i-3}))_{10} \oplus (h'_1(z_{i-2047}))_{23} \oplus (Q[r_i])_0 = \\ (h_1(z_j))_0 \oplus (h'_1(z_{j-2048}))_0 \oplus (h_1(z_{j-10}))_0 \\ \oplus (h_1(z_{j-3}))_{10} \oplus (h'_1(z_{j-2047}))_{23} \oplus (Q[r_j])_0, \end{aligned} \quad (2)$$

where $h_1(x)$ and $h'_1(x)$ are different functions since they are related to different S-boxes (see Sect. 3.1); z_j denotes the array element $P[j \boxplus 12]$ at the j -th step and $r_i = (s_{i-3} \oplus h_1(z_{i-3}) \oplus s_{i-2047} \oplus h'_1(z_{i-2047})) \bmod 1024$. Now, using the fact that $z_i = z_{i-2048} + z_{i-10} + g_1(z_{i-3}, z_{i-2047})$ and $z_j = z_{j-2048} + z_{j-10} + g_1(z_{j-3}, z_{j-2047})$, we approximate (2) as

$$H(x_1) = H(x_2), \quad (3)$$

¹ α is an element in \mathbb{N} such that $2048 \cdot \alpha + 1023 < 2^{123}$ (since HC-256 generates a maximum of 2^{123} outputs or 2^{128} output bits from a single (K, IV) pair).

where H denotes a random secret 138-bit-to-1-bit S-box, x_1 and x_2 are two 138-bit random inputs, $x_1 = z_{i-3} \| z_{i-10} \| z_{i-2047} \| z_{i-2048} \| r_i$ and $x_2 = z_{j-3} \| z_{j-10} \| z_{j-2047} \| z_{j-2048} \| r_j$. If $x_1 = x_2$, $H(x_1) = H(x_2)$. If $x_1 \neq x_2$, then $H(x_1) = H(x_2)$ with probability 2^{-1} . Since the probability that $x_1 = x_2$ is 2^{-138} , the probability that $H(x_1) = H(x_2)$ is, therefore, $2^{-1} + 2^{-138} - 2^{-139} = 1/2 + 2^{-139}$. Hence, this is the probability that (1) holds.

4.1 Our Improvement

When $2048 \cdot \alpha + 10 \leq i, j < 2048 \cdot \alpha + 1023$ and $i = j + 10$, (1) and (2) respectively become:

$$\begin{aligned} s_{j-2038(0)} \oplus s_{j+10(0)} \oplus s_{j+7(10)} \oplus s_{j-2037(23)} = \\ s_{j-2048(0)} \oplus s_{j-10(0)} \oplus s_{j-3(10)} \oplus s_{j-2047(23)} \end{aligned} \quad (4)$$

$$\begin{aligned} (h_1(z_{j+10}))_0 \oplus (h'_1(z_{j-2038}))_0 \oplus (h_1(z_{j+7}))_{10} \oplus (h'_1(z_{j-2037}))_{23} \oplus (Q[r_{j+10}])_0 = \\ (h_1(z_{j-10}))_0 \oplus (h'_1(z_{j-2048}))_0 \oplus (h_1(z_{j-3}))_{10} \oplus (h'_1(z_{j-2047}))_{23} \oplus (Q[r_j])_0. \end{aligned} \quad (5)$$

We recall that

$$z_j = z_{j-2048} + z_{j-10} + g_1(z_{j-3}, z_{j-2047}). \quad (6)$$

Therefore,

$$z_{j+10} = z_{j-2038} + z_j + g_1(z_{j+7}, z_{j-2037}). \quad (7)$$

Now, we have the following observation.

Observation 1: When the event $z_{j-2038} \| z_{j+7} \| z_{j-2037} = z_{j-2048} \| z_{j-3} \| z_{j-2047}$ (call it event E) occurs, it follows from (6) and (7) that z_{j+10} and z_{j-10} take the forms $z_{j+10} = A + B + C \bmod 2^{32}$ and $z_{j-10} = -A + B - C \bmod 2^{32}$, respectively. Therefore, the least significant bits of z_{j+10} and z_{j-10} are identical and hence $Pr[F] = 2^{-23}$. Besides, the most significant bits of z_{j+10} and z_{j-10} are equal if and only if $z_{j+10} = z_{j-10}$ (which, in turn, happens with probability 2^{-31} since their least significant bits are identical). In other words, $Pr[z_{j+10} = z_{j-10}] = 2^{-31} = Pr[z_{j+10(31)} = z_{j-10(31)}]$,² where $z_{j(k)}$ denotes the k -th significant bit of z_j ($k = 0$ denotes the least significant bit). We use this observation throughout the paper.

Let L denote the event that (5) is satisfied. We observe that L occurs when one of the following sets of conditions occurs. We denote the following set of conditions as S_1 .

1. $z_{j-2048} \| z_{j+7} \| z_{j-2037} = z_{j-2038} \| z_{j-3} \| z_{j-2047}$ (probability 2^{-96}),
2. $z_{j+10}^{(2)} \| z_{j+10}^{(1)} \| z_{j+10}^{(0)} = z_{j-10}^{(2)} \| z_{j-10}^{(1)} \| z_{j-10}^{(0)}$ (from Observation 1, this probability is 2^{-23} given condition 1),
3. $z_{j+10(7)}^{(3)} \neq z_{j-10(7)}^{(3)}$, i.e., $z_{j+10(31)} \neq z_{j-10(31)}$ (from Observation 1, this probability is $1 - 2^{-8}$ given condition 1 and condition 2),
4. $r_{j+10} \| r_j = 768 + z_{j+10}^{(3)} \| 768 + z_{j-10}^{(3)}$ (probability 2^{-20}) or $r_{j+10} \| r_j = 768 + z_{j-10}^{(3)} \| 768 + z_{j+10}^{(3)}$ (we have just observed that the two events are mutually exclusive given condition 3; their combined probability is therefore $2^{-20} + 2^{-20} = 2^{-19}$).

Therefore, $Pr[S_1] = 2^{-96} \cdot 2^{-23} \cdot (1 - 2^{-8}) \cdot 2^{-19} \approx 2^{-138}$. Similarly, we denote by S_2 the following set of conditions.

1. $z_{j-2038} \| z_{j+7} \| z_{j-2037} = z_{j-2048} \| z_{j-3} \| z_{j-2047}$ (probability 2^{-96}),

²This is confirmed by our simple experiments with 8-bit and 16-bit integers. We first considered the equations $X = A + B + C \bmod 256$, $Y = -A + B - C \bmod 256$, and evaluated $Pr[X = Y]$, $Pr[X_{(7)} = Y_{(7)}]$ varying A, B, C over all possible 8-bit values. We obtained $Pr[X = Y] = Pr[X_{(7)} = Y_{(7)}] = 2^{-7}$. With 16-bit values, when $X = A + B + C \bmod 2^{16}$ and $Y = -A + B - C \bmod 2^{16}$, we obtained $Pr[X = Y] = Pr[X_{(15)} = Y_{(15)}] = 2^{-15}$.

2. $z_{j+10}^{(3)} \| z_{j+10}^{(2)} \| z_{j+10}^{(1)} \| z_{j+10}^{(0)} = z_{j-10}^{(3)} \| z_{j-10}^{(2)} \| z_{j-10}^{(1)} \| z_{j-10}^{(0)}$, i.e., $z_{j+10} = z_{j-10}$ (from Observation 1, this probability is 2^{-31} given condition 1),
3. $r_{j+10} = r_j$ (probability 2^{-10}).

The probability that S_2 occurs $Pr[S_2] = 2^{-96} \cdot 2^{-31} \cdot 2^{-10} = 2^{-137}$. From condition 3 of S_1 and condition 2 of S_2 , we have S_1 and S_2 to be mutually exclusive. Therefore, $Pr[S_1 \cup S_2] = 2^{-138} + 2^{-137} = 2^{-136.4}$.

Actually, there are a few other such favourable events which result in the occurrence of L . However, from a large number of experiments we found that each of them occurs with much lesser probability when compared to $Pr[S_1]$ or $Pr[S_2]$. The combined probability of these mutually exclusive events was found to be approximately $2^{-136.35}$; hence, the gain over $Pr[S_1 \cup S_2]$ is negligible. We therefore approximate the union of these events by $(S_1 \cup S_2)$. When $(S_1 \cup S_2)^c$ occurs, (5) and hence (4) holds with uniform probability $1/2$ under the assumption of a perfect K/IV setup (due to space constraints, we could not provide the supporting arguments here and so we refer the interested reader to [10]). Applying Bayes' rule, we obtain:

$$\begin{aligned} Pr[L] &= Pr[L|(S_1 \cup S_2)] \cdot Pr[S_1 \cup S_2] + Pr[L|(S_1 \cup S_2)^c] \cdot Pr[(S_1 \cup S_2)^c] \\ &= 1 \cdot 2^{-136.4} + 0.5 \cdot (1 - 2^{-136.4}) = 1/2 + 2^{-137.4}. \end{aligned} \quad (8)$$

Thereby, we see that we obtain a marginal improvement (of $2^{1.6}$) in the bias probability. Besides, each equation (1) has 10 keystream bits, whereas each equation (4) has only 8 output bits. This gives an additional improvement. Combining these two, we obtain that for our distinguisher to yield a success probability of 0.9772, about $2^{279.8}$ keystream bits are required (the detailed calculations can be obtained from [10]). Whereas, in [12], $2^{283.3}$ keystream bits are needed to build the distinguisher for the same success probability. Thus our attacks require about 12 times fewer keystream bits.

5 Another Observation

In this section, we present another observation on the cipher that stems from Observation 1 (see Sect. 4.1) and the following relation.

$$r_j = (s_{j-3} \oplus h_1(z_{j-3}) \oplus s_{j-2047} \oplus h'_1(z_{j-2047})) \bmod 1024. \quad (9)$$

Due to space constraints, we could not furnish the proof here, but it may be obtained from [10].

Observation 2: When the following relations exist among keystream bits:

$$s_{j+7(b)} \oplus s_{j-2037(b)} = s_{j-3(b)} \oplus s_{j-2047(b)}, \quad (10)$$

for all $b \in \{0, \dots, 9\}$, $b \neq 7$, and

$$\begin{aligned} s_{j-2038(0)} \oplus s_{j+10(0)} \oplus s_{j+7(10)} \oplus s_{j-2037(23)} \oplus s_{j+7(7)} \oplus s_{j-2037(7)} \neq \\ s_{j-2048(0)} \oplus s_{j-10(0)} \oplus s_{j-3(10)} \oplus s_{j-2047(23)} \oplus s_{j-3(7)} \oplus s_{j-2047(7)}, \end{aligned} \quad (11)$$

and event E occurs, then $s_{j+7(7)} \oplus s_{j-2037(7)} \oplus s_{j-3(7)} \oplus s_{j-2047(7)} = 0$ holds with probability $1/2 \cdot (1 - 2^{-31})$. We believe this observation can be further exploited to construct more efficient distinguishers on the HC-256. Before we conclude, we make one final remark.

Remark: Suppose the following relations exist among keystream bits:

$$s_{j+7(b)} \oplus s_{j-2037(b)} = s_{j-3(b)} \oplus s_{j-2047(b)}, \quad (12)$$

for all $b \in \{0, \dots, 9\}$. Then, from (9), we observe that when the conditions 1 and 2 of S_2 (see Sect. 4.1) are satisfied, condition 4 is also satisfied. Therefore, in this case, $Pr[S_2] = 2^{-96} \cdot 2^{-31} = 2^{-127}$ and $Pr[S_1 \cup S_2] = 2^{-138} + 2^{-127} \approx 2^{-127}$. Therefore, $Pr[L] = 1/2 + 2^{-128}$. This is a notable improvement over the probability obtained in (8). The relation (12) was also exploited in [12], but resulting in a comparatively smaller bias of 2^{-129} and hence a distinguisher requiring about 2^{261} output words (for 0.9772 success probability).

6 Conclusions and Future Work

In this paper, we have presented distinguishing attacks on the stream cipher HC-256. The hitherto best-known distinguisher on the cipher has been presented in [12] and requires 2^{280} equations (each involving 10 keystream output bits) to be examined for a success probability of 0.9772. Each of our distinguishers requires $2^{276.8}$ equations (with 8 keystream bits in every equation) to be examined for the same success probability. Thereby, we have improved the data requirement in [12] by a factor of about 12. We have also provided leads for further cryptanalysis of the cipher.

A variant of HC-256, named HC-256', was also proposed by Wu in [12, Section 6] but without any accompanying cryptanalysis. Investigating whether our attacks could also be applied to HC-256' is left as an open problem.

References

- [1] E. Biham, J. Seberry, "Py (Roo): A Fast and Secure Stream Cipher using Rolling Arrays," eSTREAM, ECRYPT Stream Cipher Project, Report 2005/023, 2005.
- [2] O. Dunkelman, "A Small Observation on HC-128," November 14, 2007, available at <http://www.ecrypt.eu.org/stream/phorum/read.php?1,1143>.
- [3] The eSTREAM Project, available at <http://www.ecrypt.eu.org/stream/>.
- [4] R.J. Jenkins Jr., "ISAAC," *Fast Software Encryption 1996* (D. Gollmann, ed.), vol. 1039 of *LNCS*, pp. 41-49, Springer-Verlag, 1996.
- [5] S. Maitra, G. Paul, S. Raizada, "Some Observations on HC-128," *Workshop on Coding Theory and Cryptography 2009* (to appear), also available at <http://eprint.iacr.org/2008/499.pdf>.
- [6] S. Paul, B. Preneel "On the (In)security of Stream Ciphers Based on Arrays and Modular Addition," *Asiacrypt 2006* (X. Lai and K. Chen, eds.), vol. 4284 of *LNCS*, pp. 69-83, Springer-Verlag, 2006.
- [7] S. Paul, B. Preneel, G. Sekar, "Distinguishing Attacks on the Stream Cipher Py," *Fast Software Encryption 2006* (M. Robshaw, ed.), vol. 4047 of *LNCS*, pp. 405-421, Springer-Verlag, 2006.
- [8] P. Sarkar, "On Approximating Addition by Exclusive OR," available at <http://eprint.iacr.org/2009/047.pdf>.
- [9] G. Sekar, S. Paul, B. Preneel, "New Weaknesses in the Keystream Generation Algorithms of the Stream Ciphers TPy and Py," *Information Security Conference 2007* (J. Garay, et al., eds.), vol. 4779 of *LNCS*, pp. 249-262, Springer-Verlag, 2007.
- [10] G. Sekar, B. Preneel, "Improved Distinguishing Attacks on HC-256," *International Workshop on Security 2009* (to appear), available at <http://www.cosic.esat.kuleuven.be/publications/article-1258.pdf>.
- [11] O. Staffelbach and W. Meier, "Cryptographic Significance of the Carry for Ciphers Based on Integer Addition," *CRYPTO 1990* (A. Menezes and S.A. Vanstone, eds.), vol. 537 of *LNCS*, pp. 601-614, Springer-Verlag, 1990.
- [12] H. Wu, "A New Stream Cipher HC-256," available at <http://eprint.iacr.org/2004/092.pdf>; extended version of the paper that appeared in *Fast Software Encryption 2004* (B.K. Roy and W. Meier, eds.), vol. 3017 of *LNCS*, pp. 226-244, Springer-Verlag, 2004.
- [13] H. Wu, "The Stream Cipher HC-128," *New Stream Cipher Designs* (M. Robshaw and O. Billet, eds.), vol. 4986 of *LNCS*, pp. 39-47, Springer-Verlag, 2008.
- [14] E. Zenner, "A Cache Timing Analysis of HC-256," *Selected Areas in Cryptography 2008* (to appear).