Improved Distinguishing Attacks on HC-256*

Gautham Sekar[1,2,**] and Bart Preneel[1,2]

[1] Department of Electrical Engineering ESAT/SCD-COSIC,
Katholieke Universiteit Leuven, Kasteelpark Arenberg 10,
B-3001 Leuven-Heverlee, Belgium
[2] Interdisciplinary 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 [15,16]. HC-256 was published in 2004. The ciphers were also submitted to the ECRYPT eSTREAM competition [5] 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

* This work was supported in part by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), and in part by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II.

** This author is supported by an FWO project.

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 [2,7].

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 [15,16]. 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 [4], 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 [17]. This attack requires 6148 precise cache-timing measurements, 2^{16} known plaintext bits, 3 MBytes of memory and a computational effort equivalent to testing about 2^{55} keys. However, the attack uses very strong assumptions - under these assumptions any unprotected implementation of a cipher based on lookup tables such as AES or RC4 could be broken easily. Recently, Maitra *et al.* presented some observations on HC-128 in [8]. There they exploit the results of [14] (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 [16] for the least significant bit can be extended for the other bits. Their paper also studies the aforementioned observation due to Dunkelman [4]. 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 [15]. 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 [15], but a more careful analysis shows that the bias probability was underestimated and thus the requirement of the keystream bits was overestimated in [15]. 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 attack are presented in Sect. 4 and Sect. 5, respectively. Our second observation is presented in Sect. 6. Finally, Sect. 7 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 $\|$.
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 $y \in \{0, \ldots, 31\}$, $x \in \{0, \ldots, 2^{32} - 1\}$.
$x \lll y$: $((x \ll y) \oplus (x \gg (32 - y))$, where $y \in \{0, \ldots, 31\}$, $x \in \{0, \ldots, 2^{32} - 1\}$.
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. Let $K = K[0] \| \ldots \| K[7]$ and $IV = IV[0] \| \ldots \| IV[7]$, where each $K[i]$ and $IV[i]$ ($i = 0, \ldots, 7$) is 32 bits in length. 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 specifications.

$$
\begin{aligned}
f_1(x) &= (x \ggg 7) \oplus (x \ggg 18) \oplus (x \ggg 3), \\
f_2(x) &= (x \ggg 17) \oplus (x \ggg 19) \oplus (x \ggg 10), \\
g_1(x, y) &= ((x \ggg 10) \oplus (y \ggg 23)) + Q[(x \oplus y) \bmod 1024], \\
g_2(x, y) &= ((x \ggg 10) \oplus (y \ggg 23)) + P[(x \oplus y) \bmod 1024], \\
h_1(x) &= Q[x^{(0)}] + Q[256 + x^{(1)}] + Q[512 + x^{(2)}] + Q[768 + x^{(3)}], \\
h_2(x) &= P[x^{(0)}] + P[256 + x^{(1)}] + P[512 + x^{(2)}] + P[768 + x^{(3)}].
\end{aligned}
$$

3.1 *K/IV* Setup Algorithm

1. The K and the IV are expanded into an array $W[0, \ldots, 2559]$ as follows.

$$
W[i] = \begin{cases} K[i] & 0 \le i \le 7; \\ IV[i - 8] & 8 \le i \le 15; \\ f_2(W[i-2]) + W[i-7] + f_1(W[i-15]) & \\ +W[i-16] + i & 16 \le i \le 2559. \end{cases}
$$

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

$$P[i] = W[i + 512], \text{for } 0 \leq i \leq 1023,$$
$$Q[i] = W[i + 1536], \text{for } 0 \leq i \leq 1023.$$

3. Run the cipher (i.e., the keystream generation algorithm provided in Sect. 3.2) 4096 steps without generating output.

3.2 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.
{
  $k = i \bmod 1024$;
  if ($i \bmod 2048$) $< 1024$
  {
    $P[k] = P[k] + P[k \boxminus 10] + g_1(P[k \boxminus 3], P[k \boxminus 1023])$;
    $s_i = h_1(P[k \boxminus 12]) \oplus P[k]$;
  }
  else
  {
    $Q[k] = Q[k] + Q[k \boxminus 10] + g_2(Q[k \boxminus 3], Q[k \boxminus 1023])$;
    $s_i = h_2(Q[k \boxminus 12]) \oplus Q[k]$;
  }
  end-if
  $i = i + 1$;
}
end-repeat
```

4 Motivational Observation

First, we recall the analysis provided by the designer in [15]. The analysis exploits weaknesses in the PRBG and is based on the assumption of a flawless K/IV setup. At the i-th step, if $(i \bmod 2048) < 1024$, the S-box P is updated as

$$P[i \bmod 1024] \leftarrow P[i \bmod 1024] + P[i \boxminus 10] + g_1(P[i \boxminus 3], P[i \boxminus 1023]).$$

Also, $s_i = h_1(P[i \boxminus 12]) \oplus P[i \bmod 1024]$. For $10 \leq (i \bmod 2048) < 1023$, this can also be written as

$$\begin{aligned} s_i \oplus h_1(z_i) = \; & (s_{i-2048} \oplus h'_1(z_{i-2048})) + (s_{i-10} \oplus h_1(z_{i-10})) + \\ & g_1(s_{i-3} \oplus h_1(z_{i-3}), s_{i-2047} \oplus h'_1(z_{i-2047})), \end{aligned} \tag{1}$$

where $h_1(x)$ and $h'_1(x)$ are different functions since they are related to different S-boxes (see Sect. 3.2) and z_i denotes the array element $P[i \boxminus 12]$ at the i-th step.

Since addition and exclusive-OR are the same at the least significant bit-position[1], from (1) we get:

$$\begin{aligned}&s_{i(0)} \oplus s_{i-2048(0)} \oplus s_{i-10(0)} \oplus s_{i-3(10)} \oplus s_{i-2047(23)}\\ &= (h_1(z_i))_0 \oplus (h'_1(z_{i-2048}))_0 \oplus (h_1(z_{i-10}))_0\\ &\oplus (h_1(z_{i-3}))_{10} \oplus (h'_1(z_{i-2047}))_{23} \oplus (Q[r_i])_0,\end{aligned} \tag{2}$$

where $10 \leq (i \bmod 2048) < 1023$, $r_i = (s_{i-3} \oplus h_1(z_{i-3}) \oplus s_{i-2047} \oplus h'_1(z_{i-2047}))$ mod 1024. Similarly, when $2048 \cdot \alpha + 10 \leq i, j < 2048 \cdot \alpha + 1023$,[2] and $i \neq j$,

$$\begin{aligned}&s_{j(0)} \oplus s_{j-2048(0)} \oplus s_{j-10(0)} \oplus s_{j-3(10)} \oplus s_{j-2047(23)}\\ &= (h_1(z_j))_0 \oplus (h'_1(z_{j-2048}))_0 \oplus (h_1(z_{j-10}))_0\\ &\oplus (h_1(z_{j-3}))_{10} \oplus (h'_1(z_{j-2047}))_{23} \oplus (Q[r_j])_0.\end{aligned} \tag{3}$$

For the LHS of (2) and (3) to be equal, i.e., for

$$\begin{aligned}&s_{i(0)} \oplus s_{i-2048(0)} \oplus s_{i-10(0)} \oplus s_{i-3(10)} \oplus s_{i-2047(23)} =\\ &s_{j(0)} \oplus s_{j-2048(0)} \oplus s_{j-10(0)} \oplus s_{j-3(10)} \oplus s_{j-2047(23)}\end{aligned} \tag{4}$$

to hold for $2048 \cdot \alpha + 10 \leq i, j < 2048 \cdot \alpha + 1023$ $(i \neq j)$, we require that

$$\begin{aligned}&(h_1(z_i))_0 \oplus (h'_1(z_{i-2048}))_0 \oplus (h_1(z_{i-10}))_0\\ &\oplus (h_1(z_{i-3}))_{10} \oplus (h'_1(z_{i-2047}))_{23} \oplus (Q[r_i])_0 =\\ &(h_1(z_j))_0 \oplus (h'_1(z_{j-2048}))_0 \oplus (h_1(z_{j-10}))_0\\ &\oplus (h_1(z_{j-3}))_{10} \oplus (h'_1(z_{j-2047}))_{23} \oplus (Q[r_j])_0.\end{aligned} \tag{5}$$

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 (5) as

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

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 $x_2 = z_{j-3} \| z_{j-10} \| z_{j-2047} \| z_{j-2048} \| r_j$.

We now restate Theorem 1 and its proof from [15].

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. Given $x_1 = x_2$, $H(x_1) = H(x_2)$. If $x_1 \neq x_2$, then $H(x_1) = H(x_2)$ with probability 2^{-n}. Since the probability that $x_1 = x_2$ is 2^{-m}, then $x_1 \neq x_2$ with probability $1 - 2^{-m}$. The probability that $H(x_1) = H(x_2)$ is, therefore, $2^{-m} + 2^{-n} - 2^{-m-n}$. □

[1] For more significant bits, addition may be approximated by exclusive-OR with some biased probability.

[2] 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).

From Theorem 1, (6) and hence (4) holds with probability $1/2 + 2^{-139}$ given $2048 \cdot \alpha + 10 \leq i, j < 2048 \cdot \alpha + 1023$ and $i \neq j$. In Sect. 4.1, we show that (4) holds with a marginally higher probability when $i = j + 10$.

4.1 Our Improvement

Similar to the analysis above, our analysis is also based on the assumption of a perfect K/IV setup. When $2048 \cdot \alpha + 10 \leq i, j < 2048 \cdot \alpha + 1023$ and $i = j + 10$, (4) and (5) respectively become:

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

$$\begin{aligned} &(h_1(z_{j+10}))_0 \oplus (h'_1(z_{j-2038}))_0 \oplus (h_1(z_{j+7}))_{10} \oplus (h'_1(z_{j-2037}))_{23} \oplus (Q[r_{j+10}])_0 = \\ &(h_1(z_{j-10}))_0 \oplus (h'_1(z_{j-2048}))_0 \oplus (h_1(z_{j-3}))_{10} \oplus (h'_1(z_{j-2047}))_{23} \oplus (Q[r_j])_0. \end{aligned} \tag{8}$$

Let L denote the event that (8) is satisfied. We now examine the following cases under the assumption of a perfect K/IV setup.

Case 1:

Let E denote the event $z_{j-2038} \| z_{j+7} \| z_{j-2037} = z_{j-2048} \| z_{j-3} \| z_{j-2047}$. Since each z-term is a 32-bit variable distributed uniformly at random, the probability that E occurs $Pr[E] = 2^{-96}$. When E occurs, (8) reduces to:

$$(h_1(z_{j+10}))_0 \oplus (Q[r_{j+10}])_0 = (h_1(z_{j-10}))_0 \oplus (Q[r_j])_0. \tag{9}$$

We know that,

$$\begin{aligned} (h_1(z_{j+10}))_0 = \; &(Q[z^{(0)}_{j+10}])_0 \oplus (Q[256 + z^{(1)}_{j+10}])_0 \\ &\oplus (Q[512 + z^{(2)}_{j+10}])_0 \oplus (Q[768 + z^{(3)}_{j+10}])_0. \end{aligned} \tag{10}$$

Similarly,

$$\begin{aligned} (h_1(z_{j-10}))_0 = \; &(Q[z^{(0)}_{j-10}])_0 \oplus (Q[256 + z^{(1)}_{j-10}])_0 \\ &\oplus (Q[512 + z^{(2)}_{j-10}])_0 \oplus (Q[768 + z^{(3)}_{j-10}])_0. \end{aligned} \tag{11}$$

Let $z = z^{(3)} \| z^{(2)} \| z^{(1)} \| z^{(0)}$, where z is a 32-bit integer, $z^{(0)}$ is the least significant byte of z, and $z^{(3)}$ is the most significant byte of z. Let F denote the event $z^{(2)}_{j+10} \| z^{(1)}_{j+10} \| z^{(0)}_{j+10} = z^{(2)}_{j-10} \| z^{(1)}_{j-10} \| z^{(0)}_{j-10}$. Now, recall that

$$z_j = z_{j-2048} + z_{j-10} + g_1(z_{j-3}, z_{j-2047}). \tag{12}$$

Therefore,

$$z_{j+10} = z_{j-2038} + z_j + g_1(z_{j+7}, z_{j-2037}). \tag{13}$$

Observation 1: When event E occurs, it follows from (12) and (13) 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, $Pr[z_{j+10} = z_{j-10}] = 2^{-31} = Pr[z_{j+10(31)} = z_{j-10(31)}]$,[3] 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.

When F occurs, (9) reduces to

$$(Q[768 + z^{(3)}_{j+10}])_0 \oplus (Q[r_{j+10}])_0 = (Q[768 + z^{(3)}_{j-10}])_0 \oplus (Q[r_j])_0. \qquad (14)$$

Now, if $z^{(3)}_{j+10(7)} \neq z^{(3)}_{j-10(7)}$, that is, $z_{j+10(31)} \neq z_{j+10(31)}$ (this is because $z^{(3)}_{j+10(7)}$ is the most significant bit of z_{j+10}, i.e., $z_{j+10(31)}$), then $768+z^{(3)}_{j+10} \neq 768+z^{(3)}_{j-10}$. Given this, if $r_{j+10}\|r_j = 768 + z^{(3)}_{j+10}\|768 + z^{(3)}_{j-10}$ (probability is 2^{-20} since r_j is a 10-bit variable) or $r_{j+10}\|r_j = 768 + z^{(3)}_{j-10}\|768 + z^{(3)}_{j+10}$, then (14) holds. Note that we cannot have both the relations $r_{j+10}\|r_j = 768 + z^{(3)}_{j+10}\|768 + z^{(3)}_{j-10}$ and $r_{j+10}\|r_j = 768 + z^{(3)}_{j-10}\|768 + z^{(3)}_{j+10}$ to be satisfied; otherwise, $z^{(3)}_{j+10(7)} \neq z^{(3)}_{j-10(7)}$ is violated.

Summarising the above results, we have (8) to be satisfied when the following set of conditions (say S_1) simultaneously occur:

1. $z_{j-2048}\|z_{j+7}\|z_{j-2037} = z_{j-2038}\|z_{j-3}\|z_{j-2047}$ (probability 2^{-96}),
2. $z^{(2)}_{j+10}\|z^{(1)}_{j+10}\|z^{(0)}_{j+10} = z^{(2)}_{j-10}\|z^{(1)}_{j-10}\|z^{(0)}_{j-10}$ (from Observation 1, this probability is 2^{-23} given condition 1),
3. $z^{(3)}_{j+10(7)} \neq z^{(3)}_{j-10(7)}$, 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. $r_{j+10}\|r_j = 768 + z^{(3)}_{j+10}\|768 + z^{(3)}_{j-10}$ (probability 2^{-20}) or $r_{j+10}\|r_j = 768 + z^{(3)}_{j-10}\|768 + z^{(3)}_{j+10}$ (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, $Pr[S_1] = 2^{-96} \cdot 2^{-23} \cdot (1 - 2^{-8}) \cdot 2^{-19} \approx 2^{-138}$.

[3] This 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 $Pr[X = Y] = Pr[X_{(15)} = Y_{(15)}] = 2^{-15}$. We performed several similar experiments and the results are tabulated in Appendix A.

Case 2:

Proceeding along the lines of the above arguments, we define S_2 as follows:

1. $z_{j-2038} \| z_{j+7} \| z_{j-2037} = z_{j-2048} \| z_{j-3} \| z_{j-2047}$ (probability 2^{-96}),
2. $z^{(3)}_{j+10} \| z^{(2)}_{j+10} \| z^{(1)}_{j+10} \| z^{(0)}_{j+10} = z^{(3)}_{j-10} \| z^{(2)}_{j-10} \| z^{(1)}_{j-10} \| z^{(0)}_{j-10}$, 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}).

From (9), (10) and (11), it is easy to see that the event L occurs when S_2 occurs. 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 $2^{-136.35}$; therefore, the gain over $Pr[S_1 \cup S_2]$ is negligible. When none of these events occur, it follows that we will have at least two terms of one of the following forms in (8):

(a) $(Q[X])_m$, $(Q[Y])_m$ (where $X \neq Y$).
(b) $(Q[X])_m$, $(Q[Y])_n$ (where $m \neq n$).

In each case, it is easy to see that the two terms do not cancel out with biased probability. Besides, at least one of the two terms does not cancel out with any other term in (8) with biased probability. In other words, when Q is a random S-box, if $X = Y$ with probability $p \neq 0$, then $(Q[X])_m = (Q[Y])_m$ holds with probability $1/2 + p/2$ by Theorem 1. When none of the S-like events occurs, we find that Theorem 1 may, even in the best case, be applied in the same way to all pairs of terms in (8) except one. We also illustrate it with an example in Appendix B.

Therefore, when $(S_1 \cup S_2)^c$ occurs, (8) and hence (7) holds with uniform probability 1/2 under the assumption of a perfect K/IV setup (also confirmed by a large number of experiments). Applying Bayes' rule, we obtain:

$$\begin{aligned} Pr[L] &= Pr[L|(S_1 \cup S_2)] \cdot Pr[S_1 \cup S_2] + Pr[L|(S_1 \cup S_2)^c] \cdot Pr[(S_1 \cup S_2)^c] \\ &= 1 \cdot 2^{-136.4} + 0.5 \cdot (1 - 2^{-136.4}) = 1/2 + 2^{-137.4}. \end{aligned} \tag{15}$$

Note that:
(i) had HC-256 been an ideal cipher, this probability would have been 1/2,
(ii) in [15], this bias was $1/2 + 2^{-139}$.

5 The Distinguisher

A distinguisher is an algorithm that distinguishes one probability distribution from another. In cryptography, it is an algorithm that distinguishes a stream of bits from a stream of bits uniformly distributed at random (i.e., bitstream generated by an ideal stream cipher). In this section we build a distinguishing attack on HC-256 using the results of Sect. 4. Let N denote the total number of equations (7). Let p and p' respectively denote the probability that (7) holds given the outputs are collected from HC-256 and the probability that (7) holds given the outputs are generated by an ideal cipher. That is, $p = 0.5 + 2^{-137.4}$ (from (15)) and $p' = 0.5$. Let D and D' denote the distributions of the XOR-sum of the 8 output bits in (7) from HC-256 and an ideal cipher, respectively. Then, $\mu = Np$ and $\mu' = Np'$ are the respective means of D and D'. Similarly, $\sigma = \sqrt{Np(1-p)}$ and $\sigma' = \sqrt{Np'(1-p')}$ denote the respective standard deviations of D and D'. When N is large, both these binomial distributions can be approximated with the normal distribution. Now, if $|\mu - \mu'| > 2(\sigma + \sigma')$, i.e., $N > 2^{276.8}$, the cipher can be distinguished from random signal with success rate 0.9772 (since the cumulative distribution function gives the value 0.9772 at $\mu + 2\sigma$). In [15], $N > 2^{280}$ for the same success rate. In [15], there was one advantage though. Every 1024 consecutive output words, there are many more equations (4) when compared to equations (7) and therefore more number of equations (4) per (K, IV) pair.

Now, each equation (4) has 10 keystream bits, whereas each equation (7) has only 8 output bits. Therefore, for our distinguisher, $8 \cdot 2^{276.8} = 2^{279.8}$ keystream bits are required. Whereas, in [15], $10 \cdot 2^{280} = 2^{283.3}$ keystream bits are needed to build the distinguisher. Thus our attacks require about 12 times fewer keystream bits. We like to point out one issue here. It is actually possible to mount the distinguishing attack with fewer keystream bits. For example, if the adversary has 2^{106} sets of keystream bits $(s_{j-2038(0)}, s_{j+10(0)}, s_{j+7(10)}, s_{j-2037(23)}, s_{j-2048(0)}, s_{j-10(0)}, s_{j-3(10)}, s_{j-2047(23)})$ from $2^{170.8}$ random (K, IV) pairs, then a total $2^{279.8}$ output bits are available and the distinguishing attack can be mounted.

Thus the conjecture [15] that HC-256 will require more than 2^{174} keystream output words (or, equivalently 2^{179} output bits) for distinguishing attack should be restated.

6 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.

$$r_j = (s_{j-3} \oplus h_1(z_{j-3}) \oplus s_{j-2047} \oplus h_1'(z_{j-2047})) \bmod 1024. \tag{16}$$

Hence, for the following equation to hold:

$$\begin{aligned} &s_{j-2038(0)} \oplus s_{j+10(0)} \oplus s_{j+7(10)} \oplus s_{j-2037(23)} \oplus s_{j+7(7)} \oplus s_{j-2037(7)} = \\ &s_{j-2048(0)} \oplus s_{j-10(0)} \oplus s_{j-3(10)} \oplus s_{j-2047(23)} \oplus s_{j-3(7)} \oplus s_{j-2047(7)}, \end{aligned} \tag{17}$$

we require that,

$$\begin{aligned}&(h_1(z_{j+10}))_0 \oplus (h_1'(z_{j-2038}))_0 \oplus (h_1(z_{j+7}))_{10} \oplus (h_1'(z_{j-2037}))_{23}\\ &\oplus (Q[r_{j+10}])_0 \oplus (h_1(z_{j+7}))_7 \oplus (h_1'(z_{j-2037}))_7 \oplus r_{j+10(7)}\\ &= (h_1(z_{j-10}))_0 \oplus (h_1'(z_{j-2048}))_0 \oplus (h_1(z_{j-3}))_{10} \oplus (h_1'(z_{j-2047}))_{23}\\ &\oplus (Q[r_j])_0 \oplus (h_1(z_{j-3}))_7 \oplus (h_1'(z_{j-2047}))_7 \oplus r_{j(7)},\end{aligned} \tag{18}$$

is satisfied. When event E occurs, (18) reduces to:

$$(h_1(z_{j+10}))_0 \oplus (Q[r_{j+10}])_0 \oplus r_{j+10(7)} = (h_1(z_{j-10}))_0 \oplus (Q[r_j])_0 \oplus r_{j(7)}. \tag{19}$$

Now, we have the following four possibilities.

1. $r_{j+10(7)} = z^{(3)}_{j+10(7)}$ and $r_{j(7)} = z^{(3)}_{j-10(7)}$.
2. $r_{j+10(7)} \neq z^{(3)}_{j+10(7)}$ and $r_{j(7)} = z^{(3)}_{j-10(7)}$.
3. $r_{j+10(7)} = z^{(3)}_{j+10(7)}$ and $r_{j(7)} \neq z^{(3)}_{j-10(7)}$.
4. $r_{j+10(7)} \neq z^{(3)}_{j+10(7)}$ and $r_{j(7)} \neq z^{(3)}_{j-10(7)}$.

Let G denote the event $z^{(3)}_{j+10(7)} = z^{(3)}_{j-10(7)}$ (note that $z^{(3)}_{j+10(7)}$ is the most significant bit of z_{j+10}, i.e., $z_{j+10(31)}$). When E occurs, from Observation 1 in Sect. 4.1, we get $Pr[G] = 2^{-31}$ and when G occurs we have $z_{j+10} = z_{j-10}$. Given E occurs, we examine the above cases one by one.

Case 1: $r_{j+10(7)} = z^{(3)}_{j+10(7)}$ and $r_{j(7)} = z^{(3)}_{j-10(7)}$.

(a) When G occurs: we get $r_{j+10(7)} = r_{j(7)} \Rightarrow s_{j+7(7)} \oplus s_{j-2037(7)} \oplus s_{j-3(7)} \oplus s_{j-2047(7)} = 0$. Besides, (19) reduces to:

$$(Q[r_{j+10}])_0 \oplus r_{j+10(7)} = (Q[r_j])_0 \oplus r_{j(7)}. \tag{20}$$

Given this, if $r_{j+10} = r_j$ (probability 2^{-9}), then (20) and hence (17) holds with probability 1. Else, in (20), we will have two terms $(Q[r_{j+10}])_0$ and $(Q[r_j])_0$ from two different positions in the Q array. Under the assumption that the elements of Q are uniformly distributed at random, equation (20), therefore, holds with probability 1/2. This implies that (17) holds with probability 1/2.

(b) When G^c occurs: we get $r_{j+10(7)} \neq r_{j(7)} \Rightarrow s_{j+7(7)} \oplus s_{j-2037(7)} \oplus s_{j-3(7)} \oplus s_{j-2047(7)} = 1$.

Case 2: $r_{j+10(7)} \neq z^{(3)}_{j+10(7)}$ and $r_{j(7)} = z^{(3)}_{j-10(7)}$.

(a) When G occurs: we get $r_{j+10(7)} \neq r_{j(7)}$, that is, $s_{j+7(7)} \oplus s_{j-2037(7)} \oplus s_{j-3(7)} \oplus s_{j-2047(7)} = 1$.

(b) When G^c occurs: we get $r_{j+10(7)} = r_{j(7)} \Rightarrow s_{j+7(7)} \oplus s_{j-2037(7)} \oplus s_{j-3(7)} \oplus s_{j-2047(7)} = 0$.

Case 3: $r_{j+10(7)} = z^{(3)}_{j+10(7)}$ and $r_{j(7)} \neq z^{(3)}_{j-10(7)}$.

(a) When G occurs: we get $r_{j+10(7)} \neq r_{j(7)} \Rightarrow s_{j+7(7)} \oplus s_{j-2037(7)} \oplus s_{j-3(7)} \oplus s_{j-2047(7)} = 1$.

(b) When G^c occurs: we get $r_{j+10(7)} = r_{j(7)}$, i.e., $s_{j+7(7)} \oplus s_{j-2037(7)} \oplus s_{j-3(7)} \oplus s_{j-2047(7)} = 0$.

Case 4: $r_{j+10(7)} \neq z^{(3)}_{j+10(7)}$ and $r_{j(7)} \neq z^{(3)}_{j-10(7)}$.

(a) When G occurs: we get $r_{j+10(7)} = r_{j(7)} \Rightarrow s_{j+7(7)} \oplus s_{j-2037(7)} \oplus s_{j-3(7)} \oplus s_{j-2047(7)} = 0$. Given this, if $r_{j+10} = r_j$ (probability 2^{-9}), then (20) and hence (17) holds with probability 1. Else, using similar arguments as in Case 1(a), it follows that (17) holds with probability 1/2.

(b) When G^c occurs: we get $r_{j+10(7)} \neq r_{j(7)}$, i.e., $s_{j+7(7)} \oplus s_{j-2037(7)} \oplus s_{j-3(7)} \oplus s_{j-2047(7)} = 1$.

Suppose $r_{j+10(b)} = r_{j(b)}$ for $b \in \{0, \ldots, 9\}$, $b \neq 7$ and (17) does not hold. Then, given E occurs, $s_{j+7(7)} \oplus s_{j-2037(7)} \oplus s_{j-3(7)} \oplus s_{j-2047(7)} = 0$ is satisfied only in Case 2(b) and Case 3(b). In Case 2(b), $r_{j+10(7)} \neq z^{(3)}_{j+10(7)}$ (probability 0.5), $r_{j(7)} = z^{(3)}_{j-10(7)}$ (probability 0.5) and G^c occurs (probability is $1 - 2^{-31}$ given E occurs). Therefore, given the occurrence of event E, Case 2(b) happens with probability $0.5 \cdot 0.5 \cdot (1 - 2^{-31}) = 2^{-2} \cdot (1 - 2^{-31})$ and Case 3(b) also happens with the same probability. Thereby, we have the following observation.

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

$s_{j+7(b)} \oplus s_{j-2037(b)} = s_{j-3(b)} \oplus s_{j-2047(b)}$, for all $b \in \{0, \ldots, 9\}$, $b \neq 7$, and

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

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:

$$s_{j+7(b)} \oplus s_{j-2037(b)} = s_{j-3(b)} \oplus s_{j-2047(b)}, \tag{21}$$

for all $b \in \{0, \ldots, 9\}$. Then, from (16), 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 (15). The relation (21) was also exploited in [15], but resulting in a comparatively smaller bias of 2^{-129} and hence a distinguisher requiring about 2^{261} output words (for 0.9772 success rate).

7 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 [15] and requires 2^{280} equations (each involving 10 keystream output bits) to be tested for a success rate 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 [15] by a factor of about 12. We have also provided leads for further cryptanalysis of the cipher.

In [3], Crowley employs a Hidden Markov Model to combine several biases in the keystream of the cipher Py and improves the attacks described in [11]. Given the structural similarities between Py and HC-256, it may be possible to apply similar techniques here to construct a more efficient distinguisher. We leave it as an open problem.

A variant of HC-256, named HC-256', was also proposed by Wu in [15, Section 6] but without any accompanying cryptanalysis. Investigating whether our attacks could also applied to HC-256' is another interesting open problem.

Acknowledgments

It is a great pleasure to thank Hongjun Wu for valuable discussions. We also thank the anonymous reviewers of IWSEC2009 for their constructive comments on our work.

References

1. Baignères, T., Junod, P., Vaudenay, S.: How far can we go beyond linear cryptanalysis? In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 432–450. Springer, Heidelberg (2004)
2. Biham, E., Seberry, J.: Py (Roo): A Fast and Secure Stream Cipher using Rolling Arrays. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/023 (2005)
3. Crowley, P.: Improved Cryptanalysis of Py. In: Workshop Record of SASC 2006 – Stream Ciphers Revisited, ECRYPT Network of Excellence in Cryptology, Leuven, Belgium, February 2006, pp. 52–60 (2006)
4. Dunkelman, O.: A Small Observation on HC-128. November 14 (2007), `http://www.ecrypt.eu.org/stream/phorum/read.php?1,1143`
5. The eSTREAM Project, `http://www.ecrypt.eu.org/stream/`
6. Goldreich, O. (ed.): Lecture Notes on Pseudorandomness–Part-I. Department of Computer Science. Weizmann Institute of Science, Rehovot, Israel (January 2001)
7. Jenkins Jr., R.J.: ISAAC. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 41–49. Springer, Heidelberg (1996)

8. Maitra, S., Paul, G., Raizada, S.: Some Observations on HC-128. In: Workshop on Coding Theory and Cryptography, (to appear, 2009), http://eprint.iacr.org/2008/499.pdf
9. Mantin, I., Shamir, A.: A practical attack on broadcast RC4. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 152–164. Springer, Heidelberg (2002)
10. Paul, S., Preneel, B.: On the (In)security of Stream Ciphers Based on Arrays and Modular Addition. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 69–83. Springer, Heidelberg (2006)
11. Paul, S., Preneel, B., Sekar, G.: Distinguishing Attacks on the Stream Cipher Py. In: Robshaw, M.J.B. (ed.) FSE 2006. LNCS, vol. 4047, pp. 405–421. Springer, Heidelberg (2006)
12. Sarkar, P.: On Approximating Addition by Exclusive OR, http://eprint.iacr.org/2009/047.pdf
13. Sekar, G., Paul, S., Preneel, B.: New Weaknesses in the Keystream Generation Algorithms of the Stream Ciphers TPy and Py. In: Garay, J.A., Lenstra, A.K., Mambo, M., Peralta, R. (eds.) ISC 2007. LNCS, vol. 4779, pp. 249–262. Springer, Heidelberg (2007)
14. Staffelbach, O., Meier, W.: Cryptographic Significance of the Carry for Ciphers Based on Integer Addition. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 601–613. Springer, Heidelberg (1991)
15. Wu, H.: A New Stream Cipher HC-256. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 226–244. Springer, Heidelberg (2004), http://eprint.iacr.org/2004/092.pdf
16. Wu, H.: The Stream Cipher HC-128. In: Robshaw, M.J.B., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 39–47. Springer, Heidelberg (2008)
17. Zenner, E.: A Cache Timing Analysis of HC-256. Selected Areas in Cryptography (2008)(to appear)

A Experimental Results

Here, we elaborate on footnote 4 in Sect. 4.1. Consider the equations:

$$X = A + B + C \bmod 2^k, \tag{22}$$
$$Y = -A + B - C \bmod 2^k, \tag{23}$$

where X, Y, A, B, C are k-bit random variables. Let $X_{(b)}$ denote the b-th bit of X ($b = 0$ denotes the least significant bit and $b = k-1$ denotes the most significant bit). Since the '+' and '-' operators are the same as the exclusive-OR for the least significant bit position, we have from (22) and (23):

$$X_{(0)} = A_{(0)} \oplus B_{(0)} \oplus C_{(0)},$$
$$Y_{(0)} = A_{(0)} \oplus B_{(0)} \oplus C_{(0)}.$$

Therefore, $X_{(0)} = Y_{(0)} \Rightarrow Pr[X = Y] = 2^{-(k-1)}$. Now, we ran simulations to evaluate $Pr[X_{(k-1)} = Y_{(k-1)}]$ and $Pr[X \neq Y | X_{(k-1)} = Y_{(k-1)}]$ for different values of k. As there was no need to vary B, we fixed it to zero and varied A and C over all possible k-bit values. The results are provided in Table 1. Following this trend, we obtain that for $k = 32$, $Pr[X_{(31)} = Y_{(31)}] = 2^{-31}$ and $Pr[X \neq Y | X_{(31)} = Y_{(31)}] = 0$.

Table 1.

k	$Pr[X_{(k-1)} = Y_{(k-1)}]$	$Pr[X \neq Y \mid X_{(k-1)} = Y_{(k-1)}]$
4	2^{-3}	0
5	2^{-4}	0
6	2^{-5}	0
7	2^{-6}	0
8	2^{-7}	0
9	2^{-8}	0
10	2^{-9}	0
11	2^{-10}	0
12	2^{-11}	0
13	2^{-12}	0
14	2^{-13}	0
15	2^{-14}	0
16	2^{-15}	0
17	2^{-16}	0
18	2^{-17}	0
19	2^{-18}	0

B A Note on the Randomness of Keystream Bits When S_2 Does Not Occur

We restate (7) and (8) here:

$$s_{j-2038(0)} \oplus s_{j+10(0)} \oplus s_{j+7(10)} \oplus s_{j-2037(23)} = \\ s_{j-2048(0)} \oplus s_{j-10(0)} \oplus s_{j-3(10)} \oplus s_{j-2047(23)},$$

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

Suppose the 1024 elements of Q are uniformly distributed at random; same with the elements of Q'. We now examine the case when $z_{j-2037} \| z_{j+7} \| z_{j+10} = z_{j-2047} \| z_{j-3} \| z_{j-10}$, $r_{j+10} = r_j$, but $z_{j-2038} \neq z_{j-2048}$, i.e., one of the events comprising S_2^c. When $z_{j-2037} \| z_{j+7} \| z_{j+10} = z_{j-2047} \| z_{j-3} \| z_{j-10}$ and $r_{j+10} = r_j$, (8) reduces to:

$$(h'_1(z_{j-2038}))_0 = (h'_1(z_{j-2048}))_0. \tag{24}$$

Similar to (10) and (11), we have:

$$\begin{aligned} (h'_1(z_{j-2038}))_0 = \; & (Q'[z^{(0)}_{j-2038}])_0 \oplus (Q'[256 + z^{(1)}_{j-2038}])_0 \\ & \oplus (Q'[512 + z^{(2)}_{j-2038}])_0 \oplus (Q'[768 + z^{(3)}_{j-2038}])_0, \qquad (25) \\ (h'_1(z_{j-2048}))_0 = \; & (Q'[z^{(0)}_{j-2048}])_0 \oplus (Q'[256 + z^{(1)}_{j-2048}])_0 \\ & \oplus (Q'[512 + z^{(2)}_{j-2048}])_0 \oplus (Q'[768 + z^{(3)}_{j-2048}])_0. \qquad (26) \end{aligned}$$

Note that on the right-hand side of (25), we have four distinct array indices. That is, we access four elements of Q' from four different positions; similarly in (23). If $z_{j-2038} \neq z_{j-2048}$, then at least one of the following holds:

1. $z^{(0)}_{j-2038} \neq z^{(0)}_{j-2048}$,
2. $z^{(1)}_{j-2038} \neq z^{(1)}_{j-2048}$,
3. $z^{(2)}_{j-2038} \neq z^{(2)}_{j-2048}$,
4. $z^{(3)}_{j-2038} \neq z^{(3)}_{j-2048}$.

If the first case alone happens, from (25) and (26), we get:

$$(h'_1(z_{j-2038}))_0 \oplus (h'_1(z_{j-2038}))_0 = (Q'[z^{(0)}_{j-2038}])_0 \oplus (Q'[z^{(0)}_{j-2048}])_0. \tag{27}$$

Since $Q'[z^{(0)}_{j-2038}]$ and $Q'[z^{(0)}_{j-2048}]$ are two 32-bit elements from different positions in the same Q' array, they are equal with uniform probability. That is, $(Q'[z^{(0)}_{j-2038}])_0$ and $(Q'[z^{(0)}_{j-2048}])_0$ are equal with probability 1/2. This implies that (27) and hence (24) holds with probability 1/2. This, in turn, implies that (8) and hence (7) holds with uniform probability 1/2. Now, let us suppose only two of the above cases occurs; for example, cases 1 and 2. Then, we will have four terms - $(Q'[z^{(0)}_{j-2038}])_0$, $(Q'[z^{(0)}_{j-2048}])_0$, $(Q'[z^{(1)}_{j-2038}])_0$ and $(Q'[z^{(1)}_{j-2048}])_0$ - with four different array indices, and hence their XOR-sum is zero with uniform probability 1/2. Hence, it follows that (7) holds with probability 1/2.

Extending the above argument to other events that result in the outcome $z_{j-2038} \neq z_{j-2048}$ (for example, the occurrence of cases 1, 2 and 3 but not case 4), one can similarly verify that (7) holds with probability 1/2. For the other events comprising S^c_2, we arrive at the same result; however, a complete treatment is beyond the scope of this paper.