A Combinatorial Analysis of HC-128

Goutam Paul[1], Subhamoy Maitra[2], Shashwat Raizada[2]

[1] Department of Computer Science and Engineering,
Jadavpur University, Kolkata 700 032, India.
goutam.paul@ieee.org
[2] Applied Statistics Unit,
Indian Statistical Institute, Kolkata 700 108, India.
subho@isical.ac.in, shashwat.raizada@gmail.com

Abstract. We show that the knowledge of any one of the two internal state arrays of HC-128 along with the knowledge of 2048 keystream words is sufficient to construct the other state array completely in 2^{42} time complexity. Though our analysis does not lead to any attack on HC-128, it reveals a structural insight into the cipher. In the process, we theoretically establish certain combinatorial properties of HC-128 keystream generation algorithm. We also suggest a modification to HC-128 that takes care of the recently known cryptanalytic results with little reduction in speed.

Keywords: Cryptography, eSTREAM, HC-128, Keystream, State Recovery, Stream Cipher.

1 Introduction

The stream cipher HC-128 [13] is part of the eSTREAM [4] Portfolio (revision 1, September 2008) in the Software category. The designer, Wu, performed some security analysis of the cipher [13, Sections 3,4]. Another observation by Dunkelman [3] in the eSTREAM discussion forum shows that the keystream words of HC-128 leak information regarding secret states. A generalization of these results has been studied in [9]. In [8], it has been shown that the key and IV setup algorithm can be reversed if both the P, Q arrays are available after the key scheduling. Recently, a fault analysis on HC-128 has been presented in [7]. None of the existing results on HC-128 disproves the security conjectures of the designer and frequent use of this cipher in commercial domain is expected.

There are two internal state arrays of HC-128, P and Q, each containing 512 many 32-bit words. The keystream is generated in blocks of 512 words. Within a block, one of these arrays gets updated and the keystream word is produced by XOR-ing the updated entry with the sum of two words from the other array. The role of the two arrays is reversed after every block of 512 keystream words generation. In this paper, we show that the knowledge of one internal state array of HC-128 reveals the other. This analysis can serve as a general model to study stream ciphers that have a similar dual state internal structure.

In Section 3, we show that if one knows one of P and Q completely, then one can reconstruct the complete other array efficiently. Without loss of generality, we consider four consecutive blocks B_1, B_2, B_3 and B_4 of keystream generation such that Q is updated in blocks B_1 and B_3 and P is updated in blocks B_2 and B_4. Suppose the keystream words corresponding to all of these four blocks are known. Henceforth, by the symbols P and Q, we will denote the arrays after the completion of block B_1 and before the start of block B_2. After the completion of block B_2, Q remains unchanged and P is updated to, say, P_N. After the completion of block B_3, Q would again be updated to, say, Q_N.

Block B_1:	**Block B_2:**	**Block B_3:**
P unchanged, Q updated. (Q denotes the updated array)	P updated to P_N, Q unchanged.	P_N unchanged, Q updated to Q_N.

Block B_4, that is not shown in the diagram, would only be used for verifying if our reconstruction is correct or not.

In Section 3.2, we present Algorithm 1 (called *ReconstructState*), that takes as inputs the 512 words of the array P and (assuming that the 2048 keystream words corresponding to the four blocks B_1, B_2, B_3 and B_4 are known) produces as output 512 words of the array Q_N. Since the update of an array depends only on itself, it turns out that from block B_3 onwards the complete state becomes known. The proof of correctness of the algorithm is established through Lemma 2 and Theorems 1, 2 and 3 and the data and time complexity requirements are analyzed in Theorem 4.

In Section 4, we propose little modification to the existing HC-128 that escapes the currently known cryptanalytic results over HC-128. We also argue the motivation of such design and present performance comparisons with the existing design of HC-128.

2 Description of HC-128

This is adapted from [13, Section 2]. 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, each with 512 many 32-bit elements are used as internal states of HC-128. A 128-bit key array $K[0, \ldots, 3]$ and a 128-bit initialization vector $IV[0,\ldots,3]$ are used, where each entry of the array is a 32-bit element. Let s_t denote the keystream word generated at the t-th step, $t = 0, 1, 2, \ldots$.

The following six functions are used in HC-128:

$$\boxed{\begin{array}{l}
f_1(x) = (x \ggg 7) \oplus (x \ggg 18) \oplus (x \gg 3),\\
f_2(x) = (x \ggg 17) \oplus (x \ggg 19) \oplus (x \gg 10),\\
g_1(x, y, z) = ((x \ggg 10) \oplus (z \ggg 23)) + (y \ggg 8),\\
g_2(x, y, z) = ((x \lll 10) \oplus (z \lll 23)) + (y \lll 8),\\
h_1(x) = Q[x^{(0)}] + Q[256 + x^{(2)}],\\
h_2(x) = P[x^{(0)}] + P[256 + x^{(2)}].
\end{array}}$$

Here $x = x^{(3)} \| x^{(2)} \| x^{(1)} \| x^{(0)}$, x is a 32-bit word and $x^{(0)}$ (least significant byte), $x^{(1)}$, $x^{(2)}$ and $x^{(3)}$ (most significant byte) are four bytes.

The key scheduling of HC-128 is as follows.

```
Key and IV Setup:
1. Let $K[0,\ldots,3]$ be the secret key and $IV[0,\ldots,3]$ be the initialization vector.
Let $K[i+4] = K[i]$ and $IV[i+4] = IV[i]$ for $0 \le i \le 3$.
2. The key and IV are expanded into an array $W[0,\ldots,1279]$ as follows.
      $W[i] = K[i]$,                                  for $0 \le i \le 7$;
            $= IV[i-8]$,                              for $8 \le i \le 15$;
            $= f_2(W[i-2]) + W[i-7]$
              $+ f_1(W[i-15]) + W[i-16] + i$,         for $16 \le i \le 1279$.
3. Update the tables $P$ and $Q$ with the array $W$ as follows.
      $P[i] = W[i+256]$, for $0 \le i \le 511$
      $Q[i] = W[i+768]$, for $0 \le i \le 511$
4. Run the cipher 1024 steps and use the outputs
to replace the table elements as follows.
      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 keystream is generated using the following steps.

The Keystream Generation Algorithm:

```
$i = 0$;
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 Reconstruction of One Internal Array from Another

We introduce a few notations for the ease of analysis. We describe them first and then move on to the actual strategy.

3.1 Our Notations and Problem Formulation

As discussed in Section 1, we consider four consecutive blocks B_1, B_2, B_3 and B_4. In B_1 and B_3, Q is updated. Let Q denote the updated array after the completion of block B_1 and let Q_N be the new array after Q is updated in block B_3. In B_1, P remains unchanged and in B_2, it is updated to P_N. Let $s_{b,i}$ denote the i-th keystream word produced in block B_b, $1 \leq b \leq 4$, $0 \leq i \leq 511$.

The update of P (or Q) depends only on itself, i.e.,

$$P_N[i] = \begin{cases} P[i] + g_1(P[509+i], P[502+i], P[i+1]), & \text{for } 0 \leq i \leq 2; \\ P[i] + g_1(P_N[i-3], P[502+i], P[i+1]), & \text{for } 3 \leq i \leq 9; \\ P[i] + g_1(P_N[i-3], P_N[i-10], P[i+1]), & \text{for } 10 \leq i \leq 510; \\ P[i] + g_1(P_N[i-3], P_N[i-10], P_N[i-511]), & \text{for } i = 511. \end{cases} \quad (1)$$

Thus, if one knows the 512 words of P (or Q) corresponding to any one block, then one can easily derive the complete P (or Q) array corresponding to any subsequent block.

Consider that the keystream words $s_{b,i}$, $1 \leq b \leq 4$, $0 \leq i \leq 511$, are observable. We formulate a special state reconstruction problem as follows.

$$\boxed{\begin{array}{l}\text{Given the partial state information } P[0\ldots511], \\ \text{reconstruct the complete state } (P_N[0\ldots511], Q_N[0\ldots511]).\end{array}}$$

Since the update of each of P and Q depends only on P and Q respectively, once we determine P_N and Q_N, we essentially recover the complete state information for all subsequent steps.

3.2 State Reconstruction Strategy

Our state reconstruction proceeds in five phases. The **First Phase** would be to determine P_N from P using (1).

The keystream generation of block B_2 follows the equation

$$s_{2,i} = \begin{cases} h_1(P[500+i]) \oplus P_N[i], & \text{for } 0 \le i \le 11; \\ h_1(P_N[i-12]) \oplus P_N[i], & \text{for } 12 \le i \le 511. \end{cases} \tag{2}$$

Since $h_1(x) = Q[x^{(0)}] + Q[256 + x^{(2)}]$, we can rewrite (2) as

$$Q[l_i] + Q[u_i] = s_{2,i} \oplus P_N[i] \tag{3}$$

$$\left.\begin{array}{l} \text{where for } 0 \le i \le 11,\ \ l_i = (P[500+i])^{(0)} \text{ and } u_i = 256 + (P[500+i])^{(2)} \\ \text{and for } 12 \le i \le 511,\ \ l_i = (P_N[i-12])^{(0)} \text{ and } u_i = 256 + (P_N[i-12])^{(2)}. \end{array}\right\} \tag{4}$$

Here l_i, u_i and the right hand side $s_{2,i} \oplus P_N[i]$ of system (3) of equations are known for all $i = 0, 1, \ldots, 511$. Thus, there are 512 equations in ≤ 512 unknowns. Simply applying Gauss elimination would require a complexity of $512^3 = 2^{27}$. However, according to Lemma 1, a unique solution does not exist for any such system and hence we have to take a different approach to solve the system. Though the proof of Lemma 1 is simple, we include here for easy reference.

Lemma 1. *Suppose $r + s$ linear equations are formed using variables from the set $\{x_1, x_2, \ldots, x_r, y_1, y_2, \ldots, y_s\}$. If each equation is of the form $x_i + y_j = b_{ij}$ for some i in $[1, r]$ and some j in $[1, s]$, where b_{ij}'s are all known, then such a system does not have a unique solution.*

Proof. Consider the $(r + s) \times (r + s)$ coefficient matrix A of the system with the columns denoted by $C_1, \ldots, C_{r+s}$, such that the first r columns $C_1, \ldots, C_r$ correspond to the variables $x_1, \ldots, x_r$ and the last s columns $C_{r+1}, \ldots, C_{r+s}$ correspond to the variables $y_1, \ldots, y_s$. Every row of A has the entry 1 in exactly two places and the entry 0 elsewhere. The first 1 in each row appears in one of the columns $C_1, \ldots, C_r$ and the second 1 in one of the columns $C_{r+1}, \ldots, C_{r+s}$. After the elementary column transformations $C_1 \leftarrow C_1 + \ldots + C_r$ and $C_{r+1} \leftarrow C_{r+1} + \ldots + C_{r+s}$, the two columns C_1 and C_{r+1} has 1's in all the rows and hence become identical. This implies that the matrix is not of full rank and hence unique solution does not exist for the system. $\square$

The left hand side of every equation in system (3) is of the form $Q[l_i] + Q[u_i]$, where $0 \le l_i \le 255$ and $256 \le u_i \le 511$. Taking $r = s = 256$, $x_i = Q[i-1]$, $1 \le i \le 256$ and $y_j = Q[255 + j]$, $1 \le j \le 256$, we see that Lemma 1 directly applies to this system, establishing the non-existence of a unique solution. At this stage, one could remove the redundant rows to find a linear space which contains the solution. However, it is not clear how many variables need to be guessed to arrive at the final solution. Below we formulate a graph theoretic approach to derive the entries of the array Q efficiently, by guessing the value of only a single variable.

Definition 1. *System (3) of 512 equations can be represented in the form of a bipartite graph $G = (V_1, V_2, E)$, where $V_1 = \{0, \ldots, 255\}$, $V_2 = \{256, \ldots, 511\}$ and for each term $Q[l_i] + Q[u_i]$ of (3), there is an edge $\{l_i, u_i\} \in E$, $l_i \in V_1$ and $u_i \in V_2$. Thus, $|E| = 512$ (counting repeated edges, if any). We call such a graph G with the vertices as the indices of one internal array of HC-128 the* index graph *of the state of HC-128.*

Lemma 2. *Let M be the size of the largest connected component of the index graph G corresponding to block B_2. Then M out of 512 words of the array Q can be derived in 2^{32} search complexity.*

Proof. Consider any one of the 512 equations of System (3). Since the sum $Q[l_i] + Q[u_i]$ is known, knowledge of one of $Q[l_i]$, $Q[u_i]$ reveals the other. Thus, if we know one word of Q at any index of a connected component, we can immediately derive the words of Q at all the indices of the same component. Since this holds for each connected component, we can guess any one 32-bit word in the largest connected component correctly in 2^{32} attempts and thereby the result follows. $\square$

Since the arrays P, Q and the keystream of HC-128 are assumed to be random, our *index graph G* can be considered to be a random bipartite graph. Theoretical analysis of the size distribution of the connected components of random finite graphs is a vast area of research in applied probability and there have been several works [6, 12, 10, 5, 2] in this direction under different graph models. In [12], the model considered is a bipartite graph $G(n_1, n_2, T)$ with n_1 vertices in the first part, n_2 vertices in the second one and the graph is constructed by T independent trials, each of them consists of drawing an edge which joins two vertices chosen independently of each other from distinct parts. This coincides with our index graph model of Definition 1 with $n_1 = |V_1|$, $n_2 = |V_2|$ and $T = |E|$.

In general, let $n_1 \geq n_2$, $\alpha = \frac{n_2}{n_1}$, $\beta = (1 - \alpha) \ln n_1$, $n = n_1 + n_2$. Let $\xi_{n_1,n_2,T}$ and $\chi_{n_1,n_2,T}$ respectively denote the number of isolated vertices and the number of connected components in $G(n_1, n_2, T)$. We have the following result from [12].

Proposition 1. *If $n \to \infty$ and $(1+\alpha)T = n \ln n + Xn + o(n)$, where X is a fixed number, then $Prob\,(\chi_{n_1,n_2,T} = \xi_{n_1,n_2,T} + 1) \to 1$ and for any $k = 0, 1, 2, \ldots$, $Prob\,(\xi_{n_1,n_2,T} = k) - \frac{\lambda^k e^{-\lambda}}{k!} \to 0$, where $\lambda = \frac{e^{-X}(1+e^{-\beta})}{1+\alpha}$.*

In other words, if n is sufficiently large and n_1, n_2, T are related by $(1 + \alpha)T = n \ln n + Xn + o(n)$, then the graph contains one giant connected component and isolated vertices whose number follows a Poisson distribution with parameter λ given above.

Corollary 1. *If M is the size of the largest component of the index graph G, then the mean and standard deviation of M is respectively given by $E(M) \approx 442.59$ and $sd(M) \approx 8.33$.*

Proof. For our index graph, $n_1 = n_2 = 256$, $n = n_1 + n_2 = 512$, $T = 512$, $\alpha = \frac{n_2}{n_1} = 1$, $\beta = (1 - \alpha)\ln n_1 = 0$. The relation $(1+\alpha)T = n \ln n + Xn + o(n)$ is equivalent to $\frac{(1+\alpha)}{n}T = \ln n + X + \frac{o(n)}{n}$. As $n \to \infty$, the ratio $\frac{o(n)}{n} \to 0$ and hence $X \to \frac{(1+\alpha)}{n}T - \ln n$. Substituting $\alpha = 1$, $T = 512$ and $n = 512$, we get $X \approx -4.24$. By Proposition 1, the limiting distribution of the random variable $\xi_{n_1,n_2,T}$ is Poisson with mean (as well as variance) $\lambda = \frac{e^{-X}(1+e^{-\beta})}{1+\alpha} \approx e^{4.24} \approx 69.41$. Moreover, in the limit, $\chi_{n_1,n_2,T} = \xi_{n_1,n_2,T} + 1$ and this implies that all the vertices except the isolated ones would be in a single giant component. So $M = n - \xi_{n_1,n_2,T}$ and the expectation $E(M) = n - E(\xi_{n_1,n_2,T}) = n - \lambda \approx 512 - 69.41 = 442.59$. Again, the variance $Var(M) = Var(n - \xi_{n_1,n_2,T}) = Var(\xi_{n_1,n_2,T}) = \lambda$, giving $sd(M) = sd(\xi_{n_1,n_2,T}) = \sqrt{\lambda} \approx 8.33$. $\square$

Simulations with 10 million trials, each time with 1024 consecutive words of keystream generation for the complete arrays P and Q, gives the average of the number $\xi_{n_1,n_2,T}$ of isolated vertices of the index graph of the state of HC-128 as 69.02 with a standard deviation of 6.41. These values closely match with the theoretical estimates of the mean $\lambda \approx 69.41$ and standard deviation $\sqrt{\lambda} \approx 8.33$ of $\xi_{n_1,n_2,T}$ derived in Corollary 1.

Again from Corollary 1, theoretical estimates of the mean and standard deviation of the size M of the largest component is 442.59 and 8.33 respectively. From the same simulation described above, the empirical average and standard deviation of M are found to be $407.91 \approx 408$ and 9.17 respectively.

In the limit when $n \to \infty$, each vertex is either an isolated one or part of the single giant component. In practice, on the other hand, except the isolated vertices (≈ 69 in number) and the vertices of the giant component (≈ 408 in number), the remaining few ($\approx 512 - 69 - 408 = 35$ in number) vertices form some small components. However, the low (9.17) empirical standard deviation of M implies that the empirical estimate 408 of $E(M)$ is robust. We would see later that as a consequence of Theorem 2, any $M > 200$ is sufficient for our purpose.

If $C = \{y_1, y_2, \ldots, y_M\}$ be the largest component of G, then we can guess the word corresponding to any fixed index, say y_1. As explained in the proof of Lemma 2, each guess of $Q[y_1]$ uniquely determines the values of $Q[y_2], \ldots, Q[y_M]$. According to Corollary 1 and the discussion following it, we can guess around 408 words of Q in this method. This is the **Second Phase** of our solution.

We use the following result, which we call *Propagation Theorem*, to determine the remaining unknown words.

Theorem 1 (Propagation Theorem). *If $Q[y]$ is known for some y in $[0, 499]$, then $m = \lfloor \frac{511-y}{12} \rfloor$ more words of Q, namely, $Q[y+12], Q[y+24], \ldots, Q[y+12m]$, can all be determined from $Q[y]$ in a time complexity that is linear in the size of Q.*

Proof. Consider block B_1. Following our notation in Section 3.1, the equation for keystream generation is

$s_{1,i} = h_2(Q[i-12]) \oplus Q[i]$, for $12 \le i \le 511$.

Written in another way, it becomes

$Q[i] = s_{1,i} \oplus \left(P\left[(Q[i-12])^{(0)}\right] + P\left[256 + (Q[i-12])^{(2)}\right]\right)$.

Setting $y = i - 12$, we have, for $0 \le y \le 499$,

$$Q[y+12] = s_{1,y+12} \oplus \left(P\left([Q[y])^{(0)}\right] + P\left[256 + (Q[y])^{(2)}\right]\right) \tag{5}$$

This is a recursive equation, in which all s_1 values and the array P are completely known. Clearly, if we know one $Q[y]$, we know all subsequent $Q[y+12k]$, for $k = 1, 2, \ldots$, as long as $y + 12k \le 511$. This means $k \le \frac{511-y}{12}$. The number m of words of Q that can be determined is then the maximum allowable value of k, i.e., $m = \lfloor \frac{511-y}{12} \rfloor$. $\square$

By recursively applying (5) to the words of the Q array determined from the maximum size connected component of the index graph, we derive many of around $104 (= 512 - 408)$ unknown words in the array. This is the **Third Phase** of our solution. If we imagine the words initially labeled as 'known' or 'unknown', then this step can be visualized as propagation of the 'known' labels in the forward direction. Even after this step, some words remain unknown. However, as Theorem 2 implies, we observe that through this propagation, all the words $Q[500], Q[501], \ldots, Q[511]$ can be 'known' with probability almost 1.

Theorem 2. *After the Third Phase, the expected number of unknown words amongst* $Q[500]$, $Q[501]$, $\ldots$, $Q[511]$ *is approximately* $8 \cdot (1 - \frac{43}{512})^M + 4 \cdot (1 - \frac{42}{512})^M$, *where M is the size of the largest component of the index graph G.*

Proof. After the Second Phase, exactly M words $Q[y_1]$, $Q[y_2]$, $\ldots$, $Q[y_M]$ are known corresponding to the distinct indices y_1, y_2, $\ldots$, y_M in the largest component C of size M in G. Since G is a random bipartite graph, each of indices $y_1, y_2, \ldots y_M$ can be considered to be drawn from the set $\{0, 1, \ldots, 511\}$ uniformly at random (without replacement). We partition this sample space into 12 disjoint residue classes modulo 12, denoted by, $[0], [1], \ldots, [11]$. Then, each of the indices $y_1, y_2, \ldots, y_M$ can be considered to be drawn from the set $\{[0], [1], \ldots, [11]\}$ (this time with replacement; this is a reasonable approximation because $M \gg 12$) with probabilities proportional to the sizes of the residue classes. Thus, for $1 \le j \le M$, $Prob(y_j \in [r]) = \frac{43}{512}$ if $0 \le r \le 7$ and $\frac{42}{512}$ if $8 \le r \le 11$.

Let $m_r = 1$, if none of $y_1, y_2, \ldots, y_M$ are from $[r]$; otherwise, let $m_r = 0$. Hence, the total number of residue classes from which no index is selected is $Y = \sum_{r=0}^{11} m_r$. Now, in the Third Phase, we propagate the known labels in the forward direction using (5) (see Theorem 1, the Propagation Theorem). The indices $\{500, 501, \ldots, 511\}$ are to the extreme right end of the array Q and hence they also form the set of "last" indices where the propagation eventually stops. Further, each index in the set $\{500, 501, \ldots, 511\}$ belongs to exactly one of the sets $[r]$. Hence, the number of unknown words amongst $Q[500]$, $Q[501]$, $\ldots$, $Q[511]$ is also given by Y.

We have,

$$E(m_r) = Prob(m_r = 1) = \begin{cases} (1 - \frac{43}{512})^M & \text{for } 0 \leq r \leq 7; \\ (1 - \frac{42}{512})^M & \text{for } 8 \leq r \leq 11. \end{cases}$$

Thus, $E(Y) = \sum_{r=0}^{11} E(m_r) = 8 \cdot (1 - \frac{43}{512})^M + 4 \cdot (1 - \frac{42}{512})^M$. □

Substituting M by its theoretical mean estimate 443 as well as by its empirical mean estimate 408 yields $E(Y) \approx 0$. In fact, for any $M > 200$, the expression $(1 - \frac{43}{512})^M + 4 \cdot (1 - \frac{42}{512})^M$ for $E(Y)$ becomes vanishingly small. Our experimental data also supports that in every instance, none of the words $Q[500], Q[501], \ldots, Q[511]$ remains unknown.

Remarks:

1. The probability that one particular 8-bit pattern is missing from 512 many randomly selected 8-bit segments of the P array, that are used to form the indices u_i's in (3), is $(1-2^{-8})^{512} \approx 0.13$. Assuming that the missing unknown is equally likely to be one of $\{Q[256], \ldots, Q[511]\}$, the probability that there is one missing unknown and it is in $\{Q[500], \ldots, Q[511]\}$ is $\approx 0.13 \cdot \frac{12}{256} \approx 0.0061$. Thus, one may be tempted to conclude that the probability that at least one of $\{Q[500], \ldots, Q[511]\}$ remains unknown is non-negligible.
   However, we like to point out that the analysis in the above paragraph corresponds to the unknown values in (3), i.e., after the First Phase of the solution. On the other hand, the analysis in Theorem 2 corresponds to the unknown values after we have propagated the known indices using the Propagation Theorem, i.e., after the Third Phase of the solution.
2. Changing bytes 1 or 3 of $Q[y]$ yields no change in equation (5). Combining this with the Second Phase, we could form a new set of equations and attempt to solve them. However, as Theorem 2 establishes, this is not required; propagation of known $Q[y]$ values in steps of 12 covers all the unknowns.

Next, we use the following result to determine the entire Q_N array.

Theorem 3. *Suppose the complete array P_N and the 12 words $Q[500]$, $Q[501]$, $\ldots$, $Q[511]$ from the array Q are known. Then the entire Q_N array can be reconstructed in a time complexity linear in the size of Q.*

Proof. Following our notation in Section 3.1, the equation for the keystream generation of the first 12 steps of block B_3 is $s_{3,i} = h_2(Q[500 + i]) \oplus Q_N[i]$, $0 \leq i \leq 11$. Expanding $h_2(.)$, we get, for $0 \leq i \leq 11$,

$$Q_N[i] = s_{3,i} \oplus \left(P_N\left[(Q[500 + i])^{(0)}\right] + P_N\left[256 + (Q[500 + i])^{(2)}\right]\right).$$

Thus, we can determine $Q_N[0], Q_N[1], \ldots Q_N[11]$ from $Q[500], Q[501], \ldots Q[511]$. Now, applying Theorem 1 on these first 12 words of Q_N, we can determine all the words of Q_N in linear time (in size of Q). □

Applying Theorem 3 constitute the **Fourth Phase** of our solution.

After Q_N is derived, we need to verify its correctness. For this, we update P_N as it would be updated in block B_4 and generate 512 keystream words with this P_N and the derived Q_N. If the generated keystream words entirely match with the observed keystream words $\{s_{4,0}, s_{4,1}, \ldots, s_{4,511}\}$ of block B_4, then our guess is correct. This verification is the **Fifth** (and final) **Phase** of the algorithm. If we find a mismatch, then we repeat the procedure with the next guess, i.e., with another possible value in $[0, 2^{32} - 1]$ of the word $Q[y_1]$.

Once Q_N is correctly determined, the words of the Q array for all the succeeding blocks can be deterministically computed from the update rule for Q.

The above discussion is formalized in Algorithm 1, called *ReconstructState* (see the last page).

Theorem 4. *The data complexity of Algorithm 1 is* 2^{16} *and its time complexity is* 2^{42}*.*

Proof. For the First Phase, we do not need any keystream word. For each of the Second, Third, Fourth and Fifth Phases, we need a separate block of 512 keystream words. Thus, the required amount of data is $4 \cdot 512 = 2^{11}$ no. of 32 $(= 2^5)$-bit keystream words.

From Step 1 in the First Phase up to Step 7 of the Second Phase, the total time required is linear in the size of P (or Q), i.e., of complexity 2^9. Step 8 in the Second Phase of Algorithm 1 can be performed through depth-first search which requires $O(|V_1| + |V_2| + |E|)$ time complexity. For $|V_1| = 256$, $|V_2| = 256$ and $|E| = 512$, the value turns out to be 2^{10}. After this, the guess in Step 10 of Algorithm 1 consumes 2^{32} time and for each such guess, the complete Phases 3, 4 and 5 together take time that is linear in the size of the array Q, i.e., of complexity 2^9. Thus, the total time required is $2^9 + 2^{10} + 2^{32} \cdot 2^9 < 2^{42}$. □

Note that for system (3) of equations, one must verify the solution by first generating some keystream words and then matching them with the observed keystream, as is done in the Fifth Phase of Algorithm 1. During Step 10 in the Second Phase, one may exploit the cycles of the largest component to verify correctness of the guess. If the guessed value of a variable in a cycle does not match with the value of the variable derived when the cycle is closed, we can discard that guess. However, in the worst case, all the 2^{32} guesses have to be tried and if there is no conflict in a cycle, the guess has to be verified by keystream matching. Thus, it is not clear if there is any significant advantage by detecting and exploiting the cycles and so we have not considered this in the description of the algorithm.

4 Design Modification with respect to Known Observations

We have two design goals:

– to guard against the available analysis in literature and

– not to sacrifice the speed in the process.

Thus, we attempt to keep the same structure as the original HC-128 with minimal changes.

Apart from the present paper, we are aware of three other works on the analysis of keystream generation algorithm of HC-128, one by the designer Wu [13], the next as in [9] and the most recent one from [7].

The works [13, 9] exploit the fact that $h_1(.)$ as well as $h_2(.)$ makes use of only 16 bits from the 32-bit input. Our current work also uses this fact to form equation (3), that eventually leads to reconstruction of the state. Thus, all of these results indicate that the form of $h_1(.), h_2(.)$ need to be modified so as to incorporate all the 32 bits of their inputs. In our new versions of these functions (equation (6)), we suggest XOR-ing the entire input with the existing output (sum of two array entries). However, certain precautions may need to be taken so that other security threats do not come into play.

We replace h_1 and h_2 as follows.

$$\left.\begin{aligned} h_{N1}(x) &= (Q[x^{(0)}] + Q[256 + x^{(2)}]) \oplus x, \\ h_{N2}(x) &= (P[x^{(0)}] + P[256 + x^{(2)}]) \oplus x. \end{aligned}\right\} \tag{6}$$

We need to modify the update functions g_1 and g_2 with the twin motivation of preserving the internal state as well as making sure that the randomness of the keystream is ensured. We propose the following:

$$\left.\begin{aligned} g_{N1}(x, y, z) &= \big((x \ggg 10) \oplus (z \ggg 23)\big) + Q[(y \gg 7) \wedge 1FF], \\ g_{N2}(x, y, z) &= \big((x \lll 10) \oplus (z \lll 23)\big) + P[(y \gg 7) \wedge 1FF]. \end{aligned}\right\} \tag{7}$$

We keep f_1 and f_2 the same as in original HC-128.

We include a randomly chosen word from the Q array in the update of P array elements and a randomly chosen word from the P array while updating the Q array elements. This would ensure that each new block of P (or Q) array is dependent on the previous block of Q(or P) array. Thus, our analysis of Section 3 would not apply and the internal state would be preserved even if half the internal state elements are known.

Likewise, in the equation of the distinguisher proposed by the designer [13, Section 4], the term $P[i \boxminus 10]$ will get replaced by some random term of Q array. With this replacement, it is not obvious how a similar distinguishing attack can be mounted. The similar situation will happen for the distinguishers proposed in [9].

Now let us concentrate on the fault attack presented in [7]. The fault analysis in [7] assumes that if a fault occurs at $Q[f]$ in the block in which P is updated, then $Q[f]$ is not referenced until step $f - 1$ of the next block (in which Q would be updated). This assumption does not hold for our design due to our nesting use of P and Q in the updates of one another (equation (7)). Thus, on our modified design, the fault position recovery algorithm given in [7, Section 4.2] would not work immediately. In particular, Lemma 1 and Lemma 2 of [7] would not hold on our modified cipher.

The security of any stream cipher is always a conjecture. We have tried to circumvent the known weaknesses of HC-128. The way we have modified the design, it appears that no new security holes are introduced. However, the new design is open to the community for further analysis.

4.1 Performance Evaluation

We evaluated the performance of our new design using the eSTREAM testing framework [1]. The C-implementation of the testing framework was installed in a machine with Intel(R) Pentium(R) D CPU, 2.8 GHz Processor Clock, 2048 KB Cache Size, 1 GB DDR RAM on Ubuntu 7.04 (Linux 2.6.20-17-generic) OS. A benchmark implementation of HC-128 and HC-256 [14] is available within the test suite. We implemented our modified version of HC-128, maintaining the API compliance of the suite. Test vectors were generated in the NESSIE [11] format. The results presented below correspond to tests with null IV using the gcc-3.4_prescott_O3-ofp compiler.

	HC-128	Our Proposal	HC-256
Stream Encryption (cycles/byte)	4.13	4.29	4.88

The encryption speed of our proposed design is of the same order as that of original HC-128. We also observe that the extra array element access in the new update rules (equation (7)) as compared to the original update rules does not affect the performance much. HC-128 was designed as a lightweight version of HC-256. The idea of cross-referencing each other in the update rules of P and Q has also been used in the design of HC-256 and that is why the half state exposure does not reveal the full state in case of HC-256. However, our modification to HC-128 removes the known weaknesses of HC-128 but keeps the speed much better than HC-256, with only little reduction in speed compared to HC-128.

5 Conclusion

The eSTREAM candidate HC-128 uses two internal arrays, each containing 512 many 32-bit words. In this paper, we show that one of the two arrays is redundant in the sense that if one knows only one array completely and has access to 2048 consecutive keystream words then the other array can be completely reconstructed in 2^{42} time complexity. This reveals that the security margin of HC-128 with two internal arrays is equivalent to that with one internal array, when the same structure and operations are considered. As a remedy, we propose a design modification of HC-128. We also evaluate the performance of our proposal in the eSTREAM testing framework and compare the speed with that of HC-128 and HC-256.

References

1. C. D. Cannière. eSTREAM testing framework. Available at `http://www.ecrypt.eu.org/stream/perf` [last accessed on April 22, 2010].
2. C. Cooper and A. Frieze. The Size of the Largest Strongly Connected Component of a Random Digraph with a Given Degree Sequence. *Combinatorics, Probability and Computing*, vol. 13, no. 3, 2004, pages 319-337.
3. O. Dunkelman. A small observation on HC-128. `http://www.ecrypt.eu.org/stream/phorum/read.php?1,1143` Date: November 14, 2007 [last accessed on April 22, 2010].
4. `http://www.ecrypt.eu.org/stream/` [last accessed on April 22, 2010].
5. J. Hansen and J. Jaworski. Large components of bipartite random mappings. *Random Structures & Algorithms*, vol. 17, no. 3-4, October 2000, pages 317-342.
6. I. B. Kalugin. The number of components in a random bipartite graph. *Diskretnaya Matematika*, vol. 1, no. 3, 1989, pages 62-70.
7. A. Kircanski and A. M. Youssef. Differential Fault Analysis of HC-128. Africacrypt 2010, pages 360-377, vol. 6055, Lecture Notes in Computer Science, Springer.
8. Y. Liu and T. Qin. The key and IV setup of the stream ciphers HC-256 and HC-128. International Conference on Networks Security, Wireless Communications and Trusted Computing, pages 430-433, 2009.
9. S. Maitra, G. Paul and S. Raizada. Some Observations on HC-128. Pre-Proceedings of the International Workshop on Coding and Cryptography (WCC), May 10-15, 2009, Ullensvang, Norway, pages 527-539. An extended version of this paper is accepted in Designs, Codes & Cryptography.
10. M. Molloy and B. Reed. The Size of the Giant Component of a Random Graph with a Given Degree Sequence. *Combinatorics, Probability and Computing*, vol. 7, 1998, pages 295-305.
11. New European Schemes for Signatures, Integrity, and Encryption. Available at `https://www.cosic.esat.kuleuven.be/nessie` [last accessed on April 22, 2010].
12. A. I. Saltykov. The number of components in a random bipartite graph. *Diskretnaya Matematika*, vol. 7, no. 4, 1995, pages 86-94.
13. H. Wu. The Stream Cipher HC-128. `http://www.ecrypt.eu.org/stream/hcp3.html` [last accessed on April 22, 2010].
14. H. Wu. A New Stream Cipher HC-256. FSE 2004, pages 226-244, vol. 3017, Lecture Notes in Computer Science, Springer. The full version is available at `http://eprint.iacr.org/2004/092.pdf` [last accessed on April 22, 2010].

```
Input: $P[0 \dots 511]$.
Output: $P_N[0 \dots 511], Q_N[0 \dots 511]$.

First Phase:
1  for $i \leftarrow 0$ to $511$ do
2      Determine $P_N[i]$ using (1);
   end

Second Phase:
3  Form a bipartite graph $G = (V_1, V_2, E)$ as follows;
4  $V_1 \leftarrow \{0, \dots, 255\}$; $V_2 \leftarrow \{256, \dots, 511\}$; $E \leftarrow \emptyset$;
5  for $i \leftarrow 0$ to $511$ do
6      Determine $l_i$ and $u_i$ using (4);
7      $E \leftarrow E \cup \{l_i, u_i\}$;
   end
8  Find all connected components of $G$;
9  Let $C = \{y_1, y_2, \dots, y_M\}$ be the largest component with size $M$;
10 Guess $Q[y_1]$ and thereby determine $Q[y_2], \dots, Q[y_M]$ from (3); and
   for each such guess of $Q[y_1]$, repeat the Third, Fourth and Fifth
   Phases below;

Third Phase:
11 for $j \leftarrow 1$ to $M$ do
12     $y \leftarrow y_j$;
13     while $y \leq 499$ do
14         if $Q[y + 12]$ is still unknown then
15             $Q[y + 12] \leftarrow$
               $s_{1,y+12} \oplus \left(P\left[(Q[y])^{(0)}\right] + P\left[256 + (Q[y])^{(2)}\right]\right)$;
           end
16         $y \leftarrow y + 12$;
       end
   end

Fourth Phase:
17 for $i \leftarrow 0$ to $11$ do
18     $Q_N[i] \leftarrow$
       $s_{3,i} \oplus \left(P_N\left[(Q[500 + i])^{(0)}\right] + P_N\left[256 + (Q[500 + i])^{(2)}\right]\right)$;
20     $y \leftarrow i$;
21     while $y \leq 499$ do
22         $Q_N[y + 12] \leftarrow$
           $s_{3,y+12} \oplus \left(P_N\left[(Q_N[y])^{(0)}\right] + P_N\left[256 + (Q_N[y])^{(2)}\right]\right)$;
23         $y \leftarrow y + 12$;
       end
   end

Fifth Phase:
24 With the new $Q_N$, generate 512 keystream words by updating $P_N$;
25 Verify correctness of the guess in Step 10 by matching these
   keystream words with the observed keystream words of block $B_4$;
```

Algorithm 1: ReconstructState