Differential Fault Analysis of HC-128

Aleksandar Kircanski and Amr M. Youssef

Concordia Institute for Information Systems Engineering
Concordia University, Montreal, Quebec, CANADA.

Abstract. HC-128 is a high speed stream cipher with a 128-bit secret key and a 128-bit initialization vector. It has passed all the three stages of the ECRYPT stream cipher project and is a member of the eSTREAM software portfolio. In this paper, we present a differential fault analysis attack on HC-128. The fault model in which we analyze the cipher is the one in which the attacker is able to fault a random word of the inner state of the cipher but cannot control its exact location nor its new faulted value. To perform the attack, we exploit the fact that some of the inner state words in HC-128 may be utilized several times without being updated. Our attack requires about 7968 faults and recovers the complete internal state of HC-128 by solving a set of 32 systems of linear equations over Z_2 in 1024 variables.

1 Introduction

HC-128 [9] is a high speed stream cipher that has passed all the three phases of the ECRYPT eSTREAM competition and is currently a member of the eSTREAM software portfolio. The cipher design is suitable for modern super-scalar processors. It uses a 128-bit secret key and 128-bit initialization vector. At each step, it produces a 32-bit keystream output word. The inner state of the cipher is relatively large and amounts to 32768 bits, consisting of two arrays, P and Q, of 512 32-bit words, each. HC-256 [10] is another cipher similar in structure to HC-128 but uses a 256-bit key and 256-bit IV. Only HC-128 participated in the eSTREAM competition. Along with the HC-128 proposal [9], an initial security analysis pointed out to a small bias in the least significant bit of the output words which allows a distinguisher based on 2^{151} outputs. Contrary to the claims of the cipher designer [9], in [6] it was shown that the distinguisher can be extended to other bits as well, due to the bias occurring in the operation of addition of three n-bit integers, which is utilized in HC-128. However, the initial security claim [9] that there exists no distinguisher for HC-128 that uses less than 2^{64} bits [9] has not been even nearly contradicted. In [11], Zenner presented a cache timing analysis of HC-256 but this attack is not directly applicable to HC-128.

In this paper, we present a differential fault analysis (DFA) attack on HC-128. Our attack requires around half the number of fault injections when compared to the attack [4] on RC4 in the equivalent fault model. In general, fault analysis attacks [2] fall under the category of implementation dependent attacks, which include side channel attacks such as timing analysis and power analysis. In fault analysis attacks, some kind of physical influence such as ionizing radiation is applied to the cryptographic device, resulting in a corruption of the internal memory or the computation process. The examination of the results under such faults often reveals some information about the cipher key or the secret inner state. The first fault analysis attack targeted the RSA cryptosystem in 1996 [2] and subsequently, fault analysis attacks were expanded to block ciphers (e.g., [1], [3]) and stream ciphers (e.g., [4], [5]). The threat of fault analysis attacks became more realistic after cheap and low-tech methods were found to induce faults.

Throughout our attack, we introduce a new technique which exploits what can be called the *reuse* of inner state words in different iterations of the cipher. Unlike in the fault analysis model which assumes that every fault inverts exactly one bit of the inner state, the fault model assumed in this paper allows only the assumption that the fault will be localized in one of the 32-bit inner state words and no assumption on the distribution of the newly induced value, which impedes the differential analysis. The *reuse* of inner state words allows us to overcome this difficulty as follows. After faulting the inner state value, at one of the next iterations of the cipher, say at iteration r, the faulty value participates in the output. Based on the output at iteration r, some information about the difference between the original and the faulty value can be learned. If the same inner state value is *reused* at iteration $r+t$ without being updated in the meantime, the faulty value enters the output transformation with a partially known difference and the differential analysis can be applied to deduce information about the other values that participated in the output. From the work provided in this paper, it follows that the *reuse* of inner state values without updating it may facilitate differential fault analysis in weaker fault analysis models.

The rest of the paper is organized as follows. In section 2, we introduce our notation and briefly review the relevant HC-128 specifications. An overview of the proposed attack is provided in section 3 and a preliminary analysis of the attack is given in section 4. The details of the attack are provided in section 5 and its complexity is analyzed in section 6.

2 HC-128 specifications and definitions

The following notation is used throughout the paper:

$+$ and $\boxminus$: addition mod 2^{32} and subtraction mod 512.
$\oplus$: bit-wise XOR.
$\ll$, $\gg$: left and right shift, respectively, defined on 32 bit values.
$\lll$, $\ggg$: left and right rotation, respectively, defined on 32 bit values.
x^b: The b^{th} bit of a word x.
$x^{c..b}$, where $c > b$: The word $x^c|x^{c-1}|..|x^b$.
$s_i'\langle P[f]\rangle, s_i'\langle Q[f]\rangle$: The faulty keystream, where the fault is inserted while the cipher is in state $i = 268$ and occurs at $P[f]$, $Q[f]$, respectively.

The secret inner state of HC-128 consists of the tables P and Q, each containing 512 32-bit words. The execution of the cipher is governed by two public

The HC-128 Keystream Generation Algorithm

```
1: $i = 0$
2: repeat until enough keystream bits are generated
3:      $j = i \bmod 512$
4:      if ($i \bmod 1024$) $< 512$
5:          $P[j] = P[j] + g_1(P[j \boxminus 3], P[j \boxminus 10], P[j \boxminus 511])$
6:          $s_i = h_1(P[j \boxminus 12]) \oplus P[j]$
7:      else
8:          $Q[j] = Q[j] + g_2(Q[j \boxminus 3], Q[j \boxminus 10], Q[j \boxminus 511])$
9:          $s_i = h_2(Q[j \boxminus 12]) \oplus Q[j]$
10:     $i = i + 1$
```

Fig. 1. The HC-128 Keystream Generation Algorithm

counters i and j. The functions g_1, g_2, h_1 and h_2 in Fig. 1, are defined as follows:

$$\begin{aligned}
&g_1(x, y, z) = ((x \ggg 10) \oplus (z \ggg 23)) + (y \ggg 8),\\
&g_2(x, y, z) = ((x \lll 10) \oplus (z \lll 23)) + (y \lll 8),\\
&h_1(x) = Q[x^{7..0}] + Q[256 + x^{23..16}], \;\; h_2(x) = P[x^{7..0}] + P[256 + x^{23..16}].
\end{aligned}$$

The key and IV initialization procedures are omitted since they are not relevant to our attack. We say that HC-128 is *in state i*, if i steps have been executed, counting from the initial inner state. We will denote the iteration in which the cipher goes from state i to $i + 1$ by *step i*.

Definition 1 *Let $P_s[j]$ denote the $P[j]$ value after it has been updated for s times by the HC-128 KGA. Similarly, let $Q_s[j]$ denote the $Q[j]$ value after it has been updated for s times, $j = 0, \dots 511$.*

Definition 1 allows representing P and Q values at different cipher states as follows. If $s \in \{1, 2, \dots\}$, $j \in \{0, \dots 511\}$ and HC-128 is in state i, then

$$P[j] = \begin{cases} P_0[j], & i \in \{0, \dots j\} \\ P_s[j], & i \in \{1024 \times (s-1) + j + 1, \dots 1024 \times s + j\} \end{cases}$$

$$Q[j] = \begin{cases} Q_0[j], & i \in \{0, \dots 512 + j\} \\ Q_s[j], & i \in \{1024 \times (s-1) + 512 + j + 1, \dots \\ & \qquad 1024 \times s + 512 + j\} \end{cases}$$

To simplify the notation, regardless of whether h_1 or h_2 was called, the input value will be called the *h input value*. Both functions take a 32-bit word on the input. However, only the least significant byte and third least significant byte of the input value are used. Let x denote the input to the corresponding h function called in step i. Define $A_i = x^{7..0}$ and $B_i = 256 + x^{23..16}$.

3 The attack overview

The fault model in which we analyze the cipher is the one in which the attacker is able to fault a random word of the inner state tables P and Q but cannot control its exact location nor its new faulted value. We also assume that the attacker is able to reset the cipher arbitrary number of times. To perform the attack, the faults are induced while the cipher is in state 268 instead of state 0. Such a choice reduces the number of required faults to perform the attack. Throughout the rest of the paper, whenever it is referred to a fault occurrence, it is assumed that the fault occurs when the cipher is in step $i = 268$. The aim of the attack is to recover the tables P_1 and Q_1, i.e. P and Q tables of the cipher in step $i = 1024$. Since the iteration function of HC-128 is $1-1$, the inner state can then be rewind to the initial state $i = 0$. The attack can now be summarized as follows. First, the faults are induced and the corresponding output is collected as follows:

- Repeat the following steps until all of the P, Q words have been faulted at least once
  - Reset the cipher, iterate it for 268 steps and then induce the fault
  - Store the resulting faulty keystream words s'_i, $i = 268, \dots 1535$

Then, the h input values, as defined in the previous section, are recovered for certain steps as follows:

- Recover the h input values in steps $512, \dots 1023$ (details are provided in section 5.1)
- Recover a subset of the h input values in steps $1024, \dots 1535$ (the size of the recovered subset is quantified in section 5.2)

The inner state is recovered, bit by bit, in 32 phases. In phase $b = 0$, the bits $P_1^0[i], Q_1^0[i]$, $i = 0, \dots 512$ are recovered. Then, in phases $b = 1, \dots 30$, assuming the knowledge of $P_1^{b-1..0}[i], Q_1^{b-1..0}[i]$, $i = 0, \dots 512$, the bits $P_1^b[i], Q_1^b[i]$ are recovered. In each phase, a system of linear equations over Z_2 in $P_1^b[i], Q_1^b[i]$ is generated as follows:

- Generate 512 equations of the form $(P_1^b[A_i] + P_1^b[B_i]) \oplus Q_1^b[i] = s_i^b$, $i = 512, \dots 1023$ (section 5.3)
- Recover a subset of the $P_1^b[0], \dots P_1^b[255]$ and a subset of $Q_1^b[0], \dots Q_1^b[255]$ values and add the recovered information to the system (section 5.4)
- Generate more equations in $P_1^b[i], Q_1^b[i]$ values by considering the relations between faulty and non-faulty keystreams (section 5.5)
- Solve the obtained system of linear equations

Finally, the most significant bits of all the P and Q words are recovered by phase $b = 31$.

4 The faulty value position and difference

In this section, two algorithms are provided. The first one is used to recover the XOR difference between certain faulty and non-faulty inner state values after the fault has been induced and the cipher is iterated for certain number of steps. The algorithm is useful since the XOR differences between the non-faulty and the faulty inner state values is used to perform differential cryptanlaysis when the corresponding inner state values are *reused* in future cipher iterations. The second algorithm is used to recover the position of the induced fault. Before describing these two algorithms, an analysis of how the fault propagates as the cipher iterates is provided. Namely, we show that the position of the fault in the P or the Q tables uniquely determines the way by which the difference propagates through the corresponding table. This is due to the fact that, in HC-128, the update steps 5 and 8 in Fig. 1 use indices which are independent of the current state. Furthermore, although the indices used in the keystream output generation steps 6 and 9 depend on the inner state information, this does not impede the recovery of initial fault position, as will be shown below. To illustrate the above argument, assume that the fault occurred at $Q[f]$ while the cipher is in state $i = 268$. Since, according to line 5 of Fig. 1, the faulty value $Q'[f]$ is surely not referenced is during steps $i = 0, \dots 511$, it follows that $P'_1[l] = P_1[l]$, $l = 0, \dots 511$. Also, according to the update line 8 of Fig. 1, by which values $Q[j]$, $Q[j \boxminus 3]$, $Q[j \boxminus 10]$ and $Q[j \boxminus 511]$ are referenced, the first time in which $Q'[f]$ will be referenced is during the state in which $Q[f-1]$ is updated, i.e., in step $i = 512 + f - 1$. Thus, $Q_1[f-1] \neq Q'_1[f-1]$. More generally, define

$$\Delta Q_1[j] = \begin{cases} 0, & \text{if } \; Q_1[j] = Q'_1[j] \\ 1, & \text{if } \; Q_1[j] \neq Q'_1[j]. \end{cases} \tag{1}$$

Applying the same logic to follow the propagation until state 1024, for $1 \leq f \leq 501$, it is straightforward to check that

$$(\Delta Q_1[j])_{j=0}^{512} = \underbrace{00\dots0}_{j=0,\dots f-2} \; 110110110 \; \underbrace{111\dots11}_{j=f+8,\dots 511}$$

The difference propagation in the inner state is also partially projected to the keystream. For instance, if the fault occurs at $Q[f]$, then $s_j = s'_j$ holds for $512 \leq j < 512 + f - 1$. The first difference occurs at $i = 512 + j$, $j = f - 1$, after the value $Q[f-1]$ is affected and then referenced for the output in the same step. We define

$$\Delta s_i = \begin{cases} 0 & \text{if } \; s_i = s'_i \\ 1 & \text{if } \; s_i \neq s'_i \end{cases} \tag{2}$$

to track the difference propagation in the keystream output. In the presented reasoning, we implicitly assume that any difference in the right-hand side values of lines 5,6,8 or 9 of Fig. 1 always causes a difference in the corresponding left-hand sides. For $100,000$ times, the inner state of HC-128 has been randomly initialized, iterated for 268 times and then faulted at random word. In all the

100, 000 experiments, the correctness of our assumption was verified. The following Lemmas provide the complete difference propagation patterns for both the inner state and the keystream. The proofs are omitted since they are straightforward.

Lemma 1 *If the fault occurred in the P table, its position f uniquely determines the sequence $(\Delta P_1[j])_{j=0}^{512}$. Similarly, if the fault occurred in the Q table, its position f uniquely determines the sequence $(\Delta Q_1[j])_{j=0}^{512}$. The corresponding sequences, depending on the fault positions, are given in Table 1.*

Lemma 2 *If the fault occurred in the P table, the fault position uniquely determines sequence $(\Delta s_i)_{i=256}^{511}|(\Delta s_i)_{i=1024}^{1279}$. Similarly, if the fault occurred in the Q table, the fault position uniquely determines sequence $(\Delta s_i)_{i=512}^{1023}$. The corresponding sequences, depending on the fault position, are provided in Table 2.*

Fault at $P[f]$	$(\Delta P_1[j])_{j=0}^{512}$
$f = 0$	$1\ \underbrace{0\ldots0}_{j=1\ldots510}\ 1$
$f \in \{1, \ldots 257\}$	$\underbrace{0\ldots0}_{j=0\ldots f-1}\ 1\ \underbrace{0\ldots0}_{j=f+1\ldots511}$
$f \in \{258, \ldots 264\}$	$\underbrace{0\ldots0}_{j=0\ldots f-1}\ 1000000000100100100110110110\ \underbrace{1\ldots1}_{j=f+28\ldots511}$
$f \in \{265, 266, 267, 268\}$	$\underbrace{0\ldots0}_{j=0\ldots f-1}\ 100100100110110110\ \underbrace{1\ldots1}_{j=f+18\ldots511}$
$f \in \{269, \ldots 511\}$	$\underbrace{0\ldots0}_{j=0\ldots f-2}\ 110110110\ \underbrace{1\ldots1}_{j=f+8\ldots511}$
Fault at $Q[f]$	$(\Delta Q_1[j])_{j=0}^{512}$
$f = 0$	$100100100110110110\ \underbrace{1\ldots1}_{j=18\ldots511}$
$f \in \{1, \ldots 501\}$	$\underbrace{0\ldots0}_{j=0\ldots f-2}\ 110110110\ \underbrace{1\ldots1}_{j=f+8\ldots511}$
$f \in \{502, \ldots 508\}$	$\underbrace{0\ldots0}_{j=0\ldots f-503}\ 100100100110110110\ \underbrace{1\ldots1}_{j=f-484\ldots511}$
$f \in \{509, 510, 511\}$	$\underbrace{0\ldots0}_{j=0\ldots f-510}\ 100100110110110\ \underbrace{1\ldots1}_{j=f-493\ldots511}$

Table 1. The effect of faults induced during state 268 on the P and Q tables

4.1 Recovering the differences between faulty and non-faulty words

After a fault is introduced, other P and Q values are affected as the cipher iterates. In this section, we show how to derive the difference between these affected faulty values and their original counterparts. For illustration, assume

Fault at $Q[f]$	$(\varDelta s_i)_{i=512}^{1023}$
$f = 0$	$100100100110110110\ \underbrace{1\ldots1}_{i=18\ldots511}$
$f \in \{1,\ldots499\}$	$\underbrace{0\ldots0}_{i=0\ldots f-2}\ 110110110\ \underbrace{1\ldots1}_{i=f+8\ldots511}$
$f = \{500, 501\}$	$\underbrace{0\ldots0}_{i=0..f-501}\ 1\ \underbrace{0\ldots0}_{i=f-499\ldots498}\ 1101101101111$
$f \in \{502,\ldots508\}$	$\underbrace{0\ldots0}_{i=0\ldots f-503}\ 101100100110110110\ \underbrace{1\ldots1}_{i=f-484\ldots511}$
$f = \{509, 510, 511\}$	$\underbrace{0\ldots0}_{0..f-510}\ 100100110110110\ \underbrace{1\ldots1}_{i=f-494\ldots511}$
Fault at $P[f]$	$(\varDelta s_i)_{i=256}^{511} \vert (\varDelta s_i)_{i=1024}^{1279}$
$f \in \{0,\ldots247\}$	$\underbrace{0\ldots0}_{i=0\ldots254+f}\ 110110110\ \underbrace{1\ldots1}_{i=263+f\ldots511}$
$f \in \{248,\ldots255\}$	$\underbrace{0\ldots0}_{i=0\ldots254+f}\ \underbrace{110110110}_{i=255+f\ldots511}$
$f = 256$	$\underbrace{0\ldots0}_{i=0\ldots11}\ 1\ \underbrace{0\ldots0}_{i=13\ldots510}\ 1$
$f = 257$	$\underbrace{0\ldots0}_{i=0\ldots12}\ 1\ \underbrace{0\ldots0}_{i=14\ldots511}$
$f \in \{258,\ldots264\}$	$\underbrace{0\ldots0}_{i=0\ldots f-247}\ 101100100110110110\ \underbrace{1\ldots1}_{i=f-228\ldots511}$
$f \in \{265,\ldots267\}$	$\underbrace{0\ldots0}_{i=0\ldots f-254}\ 100100110110110\ \underbrace{1\ldots1}_{i=f-238\ldots511}$
$f = 268$	$\underbrace{0\ldots0}_{i=0\ldots11}\ 100100100110110110\ \underbrace{1\ldots1}_{i=30\ldots511}$
$f \in \{269,\ldots511\}$	$\underbrace{0\ldots0}_{i=0\ldots f-258}\ 110110110\ \underbrace{1\ldots1}_{i=f-248\ldots511}$

Table 2. The effect of faults induced during state 268 on the keystream

that the fault occurred at $Q[f]$. In step $i = 512+f-1$, the faulty and non-faulty keystream words will be produced by

$$s_{512+f-1} = h_2(Q[f-13]) \oplus Q[f-1], \quad s'_{512+f-1} = h_2(Q'[f-13]) \oplus Q'[f-1].$$

However, since $Q'[f-13] = Q[f-13]$, it follows that $s_{512+f-1} \oplus s'_{512+f-1} = Q[f-1] \oplus Q'[f-1]$, which allows the recovery of $Q_1[f-1] \oplus Q'_1[f-1]$. For a fault position f, define the set $S(f)$ as follows:

$$l \in S(f) \Leftrightarrow 0 \leq l \leq 511 \text{ and } l \in \{f-1, f, f+2, f+3, f+5, f+6, \\ f+8, f+9, f+10, f+13, f+16, f+19\} \tag{3}$$

where "+" and "-" denote addition and subtraction in the set of integers Z. In other words, given a fault at position f in the P or Q tables, the set $S(f)$ defines the set of positions for which the difference from the original counterpart words can be recovered as given by the following two Lemmas.

Lemma 3 *Let HC-128 be in step* 268 *when a fault occurs in* $P[f]$, $269 \leq f \leq 511$. *Then, for* $l \in S(f)$, *we have*

$$P_1[l] \oplus P_1'[l] = s_l \oplus s_l' \tag{4}$$

Proof. The distribution of corrupted values in P_1 when $f \geq 269$ is provided in Table 1. If $l = f - 1$, then

$$s_{f-1} = h_1(P_1[f-13]) \oplus P_1[f-1], \ \ s_{f-1}' = h_1(P_1'[f-13]) \oplus P_1'[f-1]$$

According to Table 1, $P_1[f-13] = P_1'[f-13]$ and since there is no corrupted values in the Q table, (4) follows. Similar proof follows for the other $l \in S(f)$ values, $269 \leq f \leq 511$.

Lemma 4 *Let HC-128 be in state* 268, *when a fault occurs in word* $Q[f]$, $0 \leq f \leq 501$. *Then, for* $l \in S(f)$, *we have*

$$Q_1[l] \oplus Q_1'[l] = s_{512+l} \oplus s_{512+l}' \tag{5}$$

The proof of Lemma 4 is analogous to the proof of Lemma 3. Note that the upper bound on f in Lemma 4 allows a simplified treatment of recoverable differences. Namely, if the fault is on $Q[f]$ for $f > 501$, the propagation starts as early as in step $i = 512$ and the set of recoverable differences differs from $S(f)$.

Given the fault position $P[f]$ or $Q[f]$, the above two Lemmas establish that for $l \in S(f)$, $P[l] \oplus P'[l]$ or $Q[l] \oplus Q'[l]$ can be recovered. A converse question can also be posed: Given a position, say $Q[l]$, which fault positions in the Q table will allow the recovery of $Q_1[l] \oplus Q_1'[l]$? For that purpose, it is convenient to define the set $S_Q^{-1}(l)$ for $0 \leq l \leq 511$ as follows

$$\begin{aligned} f \in S_Q^{-1}(l) \Leftrightarrow 0 \leq f \leq 501 \text{ and } f \in \{&l+1, l, l-2, l-3, l-5, l-6, \\ &l-8, l-9, l-10, l-13, l-16, l-19\} \end{aligned} \tag{6}$$

Now, given a position $Q[l]$, the set $S_Q^{-1}(l)$ provides all fault positions such that $Q_1[l] \oplus Q_1'[l] = s_{512+l} \oplus s_{512+l}'$ according to Lemma 4. Similarly, given a position $268 \leq l \leq 511$ in the P table, the set $S_P^{-1}(l)$ defined by

$$\begin{aligned} f \in S_P^{-1}(l) \Leftrightarrow 269 \leq f \leq 511 \text{ and } f \in \{&l+1, l, l-2, l-3, l-5, l-6, \\ &l-8, l-9, l-10, l-13, l-16, l-19\} \end{aligned} \tag{7}$$

provides the fault positions f such that $P_1[l] \oplus P_1'[l] = s_l \oplus s_l'$ can be recovered according to Lemma 3.

4.2 Recovering the position of the fault

In this section, we provide an algorithm to deduce the position where the fault occurred. Since, according to Lemmas 1 and 2, the fault position uniquely determines the corresponding sequences, the following functions can be defined:

$$\phi^P : f \mapsto (\Delta s_i)_{i=256}^{511} | (\Delta s_i)_{i=1024}^{1279}, \quad \phi^Q : f \mapsto (\Delta s_i)_{i=512}^{1023}$$

The functions are explicitly given in Table 2. By checking that no two right-hand side sequences in both parts of the Table 2 are equal, it follows that

Lemma 5 *The functions ϕ^P and ϕ^Q are 1-1.*

Let $\Delta^P = \phi^P(\{0,\dots 511\})$ and $\Delta^Q = \phi^Q(\{0,\dots 511\})$. If the fault does not cause $(\Delta s_i)_{512}^{1023} \in \Delta^P$ and $(\Delta s_i)_{i=256}^{511}|(\Delta s_i)_{i=1024}^{1279} \in \Delta^Q$ at the same time, which, as will be shown, happens with negligible probability, then Algorithm 1 returns the fault position.

Algorithm 1: Fault Position Recovery

```
INPUT: $(\Delta s_i)_{i=256}^{1279} = (s_i \oplus s'_i)_{i=256}^{1279}$
OUTPUT: The position where the fault occurred
  1: If both $(\Delta s_i)_{512}^{1023} \in \Delta^P$ and $(\Delta s_i)_{i=256}^{511}|(\Delta s_i)_{i=1024}^{1279} \in \Delta^Q$, return undefined
  2: If $(\Delta s_i)_{i=512}^{1023} \in \Delta^P$, return $\phi_P^{-1}((\Delta s_i)_{i=512}^{1023})$
  3: If $(\Delta s_i)_{i=256}^{511}|(\Delta s_i)_{i=1024}^{1279} \in \Delta^Q$, return $\phi_Q^{-1}((\Delta s_i)_{i=256}^{511}|(\Delta s_i)_{i=1024}^{1279})$
```

From line 1 of Algorithm 1, if there is conflicting information on whether the fault occurred in the P or the Q table, the algorithm returns *undefined*. To estimate the probability of this unwanted event, let $F_{P[f]}$ and $F_{Q[f]}$ denote the event that the fault occurs at position $P[f]$ and $Q[f]$, respectively. Let U denote the event that Algorithm 1 returns $undefined$. Then we have

$$\mathrm{Prob}[U] = \sum_{f=0}^{511} \mathrm{Prob}[U \cap F_{P[f]}] + \sum_{f=0}^{511} \mathrm{Prob}[U \cap F_{Q[f]}] =$$
$$\frac{1}{1024}\Big(\sum_{f=0}^{511} \mathrm{Prob}[U|F_{P[f]}] + \sum_{f=0}^{511} \mathrm{Prob}[U|F_{Q[f]}]\Big) \tag{8}$$

where $\mathrm{Prob}[U \cap F_{P[f]}] = \mathrm{Prob}[F_{P[f]}]\mathrm{Prob}[U|F_{P[f]}]$ and also $\mathrm{Prob}[U \cap F_{Q[f]}] = \mathrm{Prob}[F_{Q[f]}]\mathrm{Prob}[U|F_{Q[f]}]$. To expand the probability $\mathrm{Prob}[U|F_{P[f]}]$, let n_0 and n_1 denote the number of faulty values among the values $P[0],\dots P[255]$ and $P[256]\dots P[511]$, respectively, at state 512, given that the fault occurred at $P[f]$. Also, let $p = \frac{n_0+n_1}{256} - \frac{n_0 n_1}{256^2}$. If $n(\delta'_i)$ is the number of 1 values in a 512-element sequence $\delta'_i \in \Delta_Q$, then

$$\mathrm{Prob}[U|F_{P[f]}] = \sum_{\delta'_i \in \Delta^Q} \mathrm{Prob}[(\Delta s_i)_{i=512}^{1023} = \delta'_i|F_{P[f]}] = \sum_{\delta'_i \in \Delta^Q} p^{n(\delta'_i)}(1-p)^{512-n(\delta'_i)}$$

As for the probability $\mathrm{Prob}[U|F_{Q[f]}]$, let n_0 and n_1 denote the number of faulty words among $Q[0],\dots Q[255]$ and $Q[256],\dots Q[511]$, respectively, at state 268, given that the fault occurred at $Q[f]$. Let n_2 and n_3 denote the number of faulty words among $Q[0],\dots Q[255]$ and among $Q[256],\dots Q[511]$, respectively, at state 1024, given that the fault occurred at $Q[f]$. Let $p_0 = \frac{n_0+n_1}{256} - \frac{n_0 n_1}{256^2}$ and $p_1 = \frac{n_2+n_3}{256} - \frac{n_2 n_3}{256^2}$. If $m_0(\delta'_i)$ and $m_1(\delta'_i)$, denote the number of 1 values among

$\delta'_{12}, \ldots \delta'_{255}$ and $\delta'_{256}, \ldots \delta'_{511}$, respectively, where $\delta'_i \in \Delta_P$, then

$$\mathrm{Prob}[U|F_{Q[f]}] = \sum_{\delta'_i \in \Delta^P} \mathrm{Prob}[(\Delta s_i)_{i=256}^{511} | (\Delta s_i)_{i=1024}^{1279} = \delta'_i | F_{Q[f]}] =$$

$$= \sum_{\delta'_i \in \Delta^P} p_0^{m_0(\delta'_i)} (1-p_0)^{244-m_0(\delta'_i)} p_1^{m_1(\delta'_i)} (1-p_1)^{256-m_1(\delta'_i)}$$

Calculating the sets Δ_P and Δ_Q and substituting the corresponding values using Table 2 allows the computation of the sums in Eq. (8) as $\frac{1}{1024}\sum_{f=0}^{511} \mathrm{Prob}[U|F_{P[f]}] = 2^{-66.293}$ and $\frac{1}{1024}\sum_{f=0}^{511} \mathrm{Prob}[U|F_{Q[f]}] = 2^{-30.406}$. Thus, the probability that Algorithm 1 returns *undefined* as fault position is is $\mathrm{Prob}[U] = 2^{-30.406}$.

5 Using DFA to generate equations

As described in section 3, the attack is performed by introducing faults until every P and Q word is faulted. Let T be the number of fault injections required to fault each of the 1024 words in the P and Q tables at least once. The expected number of required faults, $E(T)$, is given by $E(T) = 7698.4$ (see the coupons' collector problem in [7].) After inducing that number of faults, the average number of faults at a particular word $P[i]$ or $Q[i]$ will be $7698.4/1024 \approx 7.52$. As stated in section 3, the attack proceeds in 32 phases. Each phase b relies on the knowledge of $P_1^{b-1..0}[i]$, $Q_1^{b-1..0}[i]$, $i = 0, \ldots 511$, recovered in previous phases. Only the first phase, $b = 0$, does not require any previous bit knowledge. In each phase, a linear system of equations over Z_2 in $P_1^b[i]$, $Q_1^b[i]$, $i = 0, \ldots 511$ is generated and solved. Phase $b = 31$ proceeds with minor modifications compared to phases $0 \le b \le 30$, as explained below.

5.1 The recovery of h input values for steps 512, ... 1023

In every HC-128 step, one of the two h functions is called, i.e., either h_1 or h_2. The input for the h functions is a 32-bit value, out of which only 16 bits, A_i, B_i, play a role in the computation. In this section, we describe a method to recover all of the A_i, B_i values, for $i = 512, \ldots 1023$.

To recover A_i, assume that the fault occurred at $P[f]$ while the cipher was in state 268. As can be seen from Table 1, if $1 \le f \le 255$, then as the cipher iterates through steps $i = 512, \ldots 1023$, no other P values gets corrupted. Also, the Q table does not get corrupted. Thus, in case $1 \le f \le 255$, the non-faulty and the faulty keystream words in step $512 \le i \le 1023$ are

$$s_i = (P_1[A_i] + P_1[B_i]) \oplus Q_1[j],\ s'_i\langle P[f]\rangle = (P'_1[A_i] + P_1[B_i]) \oplus Q_1[j]$$

Since $P'_1[A_i] \ne P_1[A_i]$ implies that $A_i = f$, then we have

$$s_i \ne s'_i\langle P_1[f]\rangle \Rightarrow A_i = f \tag{9}$$

In case $f = 0$, the fault does propagate to $P[511]$ and if $s_i \neq s'_i\langle P[0]\rangle$, then it is unclear whether $A_i = 0$ or $B_i = 511$, or both equalities hold. However, if there exists no faulty keystream for $1 \leq f \leq 255$ such that (9) is true, then $A_i = 0$. As for B_i, assume that a fault is inserted at word $P[f]$, $256 \leq f \leq 268$, while the cipher is in state 268. From Table 1, it is clear that at state 1024, none of the $P[0], \ldots P[f-1]$ values will be corrupted and the value $P[f]$ will necessarily be corrupted. Similarly, if the fault is inserted at $P[f]$ where $269 \leq f \leq 511$, none of the values $P[0], \ldots P[f-2]$ get corrupted and the value $P[f-1]$ will necessarily be corrupted. Thus, if f_{max} denotes the maximal f such that $s_i \neq s_i\langle P[f]\rangle$, then

$$B_i = \begin{cases} f_{max} & \text{if } f_{max} \in \{256, \ldots 268\} \\ f_{max} - 1 & \text{if } f_{max} \in \{269, \ldots 510\} \end{cases}$$

Finally, if $f_{max} = 511$, it is not clear whether $B_i = 510$ or $B_i = 511$. To differentiate between these two cases, it should be verified whether $s_i \neq s'_i$ also holds for any f which does not corrupt $P[511]$, for instance for $f = 507$. The recovery of A_i, B_i, for all $512 \leq i \leq 1023$ is given by Algorithm 2.

Algorithm 2: Recovery of A_i and B_i, for some $i = 512, \ldots 1023$

```
INPUT: Step $i \in \{512, \ldots 1023\}$
OUTPUT: $A_i$, $B_i$
  1: If exists $1 \leq f \leq 255$ such that $s_i \neq s'_i\langle P[f]\rangle$: $A_i = f$
  2: else $A_i = 0$
  3: Find $f_{max}$, the maximum $f$ such that $s_i \neq s_i\langle P[f]\rangle$
  4: If $256 \leq f_{max} \leq 268$: $B_i = f_{max}$
  5: else if $269 \leq f_{max} \leq 510$: $B'_i = f_{max} - 1$
  6: else if $s_i = s_i\langle P[507]\rangle$: $B_i = 510$
  7: else $B_i = 511$
  9: Return $A_i$, $B_i$
```

Given the definition of h_2 and by noting that $j = i \bmod 512$, the recovered A_i and B_i values are in fact

$$A_i = \begin{cases} Q_0^{7..0}[j \boxminus 12] & \text{if } i \in \{512..523\} \\ Q_1^{7..0}[j \boxminus 12] & \text{if } i \in \{524..1023\} \end{cases} \tag{10}$$

$$B_i = \begin{cases} Q_0^{23..16}[j \boxminus 12] + 256 & \text{if } i \in \{512..523\} \\ Q_1^{23..16}[j \boxminus 12] + 256 & \text{if } i \in \{524..1023\} \end{cases} \tag{11}$$

5.2 The recovery of the h input values for steps 1024, ... 1535

In this subsection, A_i, B_i values for a subset of $i = 1024, \ldots 1535$ are recovered. While B_i values will be recovered by a method similar to the one from the previous subsection, the same method is not applicable for A_i recovery and we will utilize the *reuse* of inner state words to recover the A_i values.

As for the recovery of B_i for $i = 1024, \ldots 1535$, from Table 1 it can be observed that if for some $1 \leq f \leq 501$, $Q[f]$ is faulted at step 268, the value $Q[f+7]$ will remain unchanged and the values $Q[f+8], \ldots Q[511]$ will surely be corrupted. Thus, if f_{min} denotes the minimal $249 \leq f \leq 501$ such that $s_i = s'_i\langle Q[f]\rangle$, then $B_i = f_{min} + 7$. Also, since $Q[509]$, $Q[510]$ and $Q[511]$ will get corrupted regardless of the fault position in Q, it is not possible to distinguish which of values 509, 510 or 511 B_i was equal to.

Thus, if for given step i, $B_i < 509$ holds, B_i will be recovered. Moreover, if $B_i < 500$, $Q_1^{7..0}[B_i]$ will be recovered by (11). Assuming that $B_i < 500$ for step i, then A_i can be recovered as follows. Consider the faulty keystream $s'_i\langle Q[f]\rangle$ where $f \in S_Q^{-1}(B_i)$. According to Lemma 4 and (6)

$$Q_1[B_i] \oplus Q'_1[B_i] = s_{512+B_i} \oplus s'_{512+B_i}$$

Thus, $Q_1'^{7..0}[B_i]$ can be recovered by $Q_1'^{7..0}[B_i] = Q_1^{7..0}[B_i] \oplus s^{7..0}_{512+B_i} \oplus s'^{7..0}_{512+B_i}$. After being used in step $512 + B_i$, the value $Q[B_i]$ is *reused* in step i as follows

$$s_i = (Q_1[A_i] + Q_1[B_i]) \oplus P_2[j],\ s'_i\langle Q[f]\rangle = (Q'_1[A_i] + Q'_1[B_i]) \oplus P'_2[j]$$

If $257 \leq f \leq 501$, $Q_1[A_i] = Q'_1[A_i]$ holds according to Table 1. Also, the P table remains uncorrupted and thus $P_2[j] = P'_2[j]$. Thus, focusing on the least significant byte and XORing the previous two values yields

$$s_i^{7..0} \oplus s_i'^{7..0}\langle Q[f]\rangle = (Q_1^{7..0}[A_i] + Q_1^{7..0}[B_i]) \oplus (Q_1^{7..0}[A_i] + Q_1'^{7..0}[B_i]) \qquad (12)$$

Since $s_i^{7..0} \oplus s_i'^{7..0}$, $Q_1^{7..0}[B_i]$ and $Q_1'^{7..0}[B_i]$ are known, (12) represents a test that allows eliminating some wrong candidates for $Q_1^{7..0}[A_i]$ value. One test of the form (12) will be generated for each faulty instance for which the fault position is $Q[f]$, where $f \in S_Q^{-1}(B_i)$. Consequently, an $0 \leq A_i \leq 255$ can be discarded if the corresponding $Q_1^{7..0}[A_i]$ recovered by (11) does not satisfy (12).

The test (12) can be reformulated so that the third least significant byte is used as follows

$$\begin{gathered} s_i^{23..16} \oplus s_i'^{23..16} = \\ (Q_1^{23..16}[A_i] + Q_1^{23..16}[B_i] + \sigma_i) \oplus (Q_1^{23..16}[A_i] + Q_1'^{23..16}[B_i] + \sigma'_i) \qquad (13) \end{gathered}$$

where σ_i is a carry corrector defined to be 1 if $Q_1^{15..0}[A_i] + Q_1^{15..0}[B_i] \geq 2^{16}$ and 0 otherwise. Another carry corrector, σ'_i, is defined analogously. The value $Q_1'^{23..16}[B_i]$ is obtained in the same way as the value $Q_1'^{7..0}[B_i]$ above. If $0 \leq A_i \leq 255$ and the corresponding $Q_1^{23..16}[A_i]$ are substituted in (13) and none of $\sigma_i, \sigma'_i \in \{0, 1\}$ satisfy the test, then A_i is discarded.

In what follows, we estimate the expected number of steps for which both the A_i and B_i values are recovered by the presented method. Let $1024 \leq i \leq 1535$ be a step of HC-128. If, for example, for step i, $257 \leq B_i \leq 492$, then the B_i value will surely be recovered as provided by the above method. Furthermore, for such a particular value B_i, $|S_Q^{-1}(B_i)| = 12$ will hold. Since for each $f \in S_Q^{-1}(B_i)$ around 7.52 faults occur at $Q[f]$, as shown at the beginning of section 5, around $7.52 \times 12 = 90.24$ tests given by Eq. (12) and the same number of

Algorithm 3: Recovery of A_i and B_i, for some $i = 1024, \dots 1535$

INPUT: Step $i \in \{1024, \dots 1535\}$
OUTPUT: A_i or $undef$, B_i or $undef$

```
1: Calculate $F = \{257 \leq f \leq 501 | s_i = s'_i\langle Q[f]\rangle\}$
2: If $|F| = 0$: $B_i = undef$
3: Else $B_i = min(F) + 7$
4: If $B_i > 500$ $A_i = undef$; Return $A_i$, $B_i$
5: Else: let $Cand(A_i) = \{0, 1, ..255\}$
6: For each $f \in S_Q^{-1}(B_i)$
7:     Deduce $Q_1'^{7..0}[B_i] = Q_1^{7..0}[B_i] \oplus s_{512+B_i}^{7..0} \oplus s_{512+B_i}'^{7..0}$
8:     For $A_i = 0, \dots 255$
9:        If $s_i^{7..0} \oplus s_i'^{7..0} \neq (Q_1^{7..0}[A_i] + Q_1^{7..0}[B_i]) \oplus (Q_1^{7..0}[A_i] + Q_1'^{7..0}[B_i])$
10:            Eliminate $A_i$ from $Cand(A_i)$
11:       If for every $\sigma_i$,$\sigma'_i \in \{0, 1\}$, $s_i^{23..16} \oplus s_i'^{23..16} \neq d$, where
12:       $d = (Q_1^{23..16}[A_i] + Q_1^{23..16}[B_i] + \sigma_i) \oplus (Q_1^{23..16}[A_i] + Q_1'^{23..16}[B_i] + \sigma'_i)$
13:            Eliminate $A_i$ from $Cand(A_i)$
14: If $|cand(A_i)| = 1$, let $A_i$ be the unique $cand(A_i)$ member
15: Else: $A_i = undef$
16: Return $A_i$, $B_i$
```

tests given by Eq. (13) will be applied to the set of candidates for A_i. According to our experimental results, such a number of tests is sufficient to discard all the false candidates for A_i. In particular, an experiment in which Algorithm 3 was executed for all 512 steps $i \in \{1024, \dots 1535\}$ for $10,000$ times, with random HC-128 instantiations, was conducted. On average, in 472.7 out of the 512 steps, both A_i and B_i values were recovered.

5.3 Equations of the form $P_1^b[A_i] \oplus P_1^b[B_i] \oplus Q_1^b[j] = s_i^b \oplus c_{i,b}$

After the steps given by subsections 5.1 and 5.2 have been executed, the attack proceeds in 32 phases, each consisting of 3 parts, as presented by the attack overview in section 3. In this subsection, the first part of b-th attack phase is presented.

The first part of b-th phase, in which starting 512 equations are generated, proceeds as follows. In steps $i \in \{512, \dots 1023\}$, the keystream output word is generated as $(P_1[A_i] + P_1[B_i]) \oplus Q_1[j] = s_i$ where $j = i$ mod 512. Since A_i and B_i for $i \in \{512, \dots 1023\}$ have been recovered in subsection (5.1), focusing on the b-th bit yields 512 bits equations of the form

$$P_1^b[A_i] \oplus P_1^b[B_i] \oplus Q_1^b[j] = s_i^b \oplus c_{i,b}, i = 512, \dots 1023 \tag{14}$$

where $c_{i,b}$ is a known carry corrector which is equal to 1 if there is carry in $(P_1^{b-1..0}[A_i] + P_1^{b-1..0}[B_i])$ and 0 otherwise. In case $b \in \{0, \dots 7\}$ or $b \in \{16, \dots 23\}$, relying on the knowledge obtained by (10) and (11), the system can be extended by adding information $Q_1^b[w] = a_w, w = 0, \dots 499$, regarded as equations. However, for $b \notin \{0, \dots 7, 16, \dots 23\}$ such equations are unavailable.

Hence, a method to systematically add more equations to the system (14) that works for all $b = 0, \dots 31$, i.e., that makes the corresponding system of rank 1024, is necessary. In order to provide a generic treatment for all b values, in what follows, equations derived from information given by (10) and (11) will not be utilized.

5.4 Recovering bits $P_1^b[0], \dots P_1^b[255]$ and $Q_1^b[0], \dots Q_1^b[255]$

In the second part of the b-th phase of the attack, the system of equations given by (14) is expanded. Note that in steps $512 \leq i \leq 1023$, the output is generated by $s_i = (P_1[A_i] + P_1[B_i]) \oplus Q_1[j]$, whereas in steps $1024 \leq i \leq 1535$, the output is generated by $s_i = (Q_1[A_i] + Q_1[B_i]) \oplus P_2[j]$. The idea is to corrupt $P_1[B_i]$ and $Q_1[B_i]$ in the previous two relations and recover $P_1[A_i]$ and $Q_1[A_i]$ by observing how these values react to addition of different values. The difference of the corrupted values is controlled by utilizing the *reuse* of $P_1[B_i]$ and $Q_1[B_i]$ over different states of the cipher. The analysis results in the recovery of a subset of the $P_1^b[0], \dots P_1^b[255]$ and also a subset of the $Q_1^b[0], \dots Q_1^b[255]$ values.

As for recovering $P_1^b[0], \dots P_1^b[255]$, let $512 \leq i \leq 1023$ and $268 \leq B_i \leq 511$. Consider a fault at position $P[f]$, so that $f \in S_P^{-1}(B_i)$. Using Lemma 3 and (7), define $\delta = s_{B_i} \oplus s'_{B_i} = P_1[B_i] \oplus P'_1[B_i]$. Assume that for the faulty cipher instance in question, $\delta^b = 1$. Consider the difference

$$\Delta = s_i \oplus s'_i = (P_1[A_i] + P_1[B_i]) \oplus (P_1[A_i] + P'_1[B_i]),$$

and denote by c_b and c'_b the carry from the $b-1$ to b-th bit in the sums $P_1[A_i] + P_1[B_i]$ and $P_1[A_i] + P'_1[B_i]$, respectively. If $c_b = c'_b$, then the bit $P^b[A_i]$ is recovered as follows

$$P_1^b[A_i] = \begin{cases} \delta^{b+1} \oplus \Delta^{b+1}, & \text{if } c_b = c'_b = 0 \\ \delta^{b+1} \oplus \Delta^{b+1} \oplus 1, & \text{if } c_b = c'_b = 1 \end{cases} \tag{15}$$

If $c_b \neq c'_b$, the bit $P_1^b[B_i]$ is not uniquely determined and will not be recovered.

Algorithm 4: Recovery of $P_1^b[A_i]$, for some $i = 512, \dots 1023$

INPUT: Step $i \in \{512, \dots 1023\}$
OUTPUT: Bit $P_1^b[A_i]$, or *undef*

```
1: For every faulty keystream, where the fault occurred at $P[f]$, $f \in S_P^{-1}(B_i)$
2:     Calculate $\delta = s_{B_i} \oplus s'_{B_i}$ and $\Delta = s_i \oplus s'_i$
3:     If $P_1^{b-1..0}[x] + P_1^{b-1..0}[B_i] < 2^b$, set $c_b = 0$, else set $c_b = 1$
4:     If $P_1^{b-1..0}[x] + P_1'^{b-1..0}[B_i] < 2^b$, set $c'_b = 0$, else set $c'_b = 1$
5:     If $c_b = c'_b$:
6:         Return $P_1^b[A_i]$ calculated according to (15)
7: Return undef
```

To recover $P_1^b[A_i]$, the explained procedure is repeatedly applied using each fault occurring at $P[f]$, $f \in S_P^{-1}(B_i)$. Let $p_1 = Prob[\delta^b = 1]$ and $p_2 = Prob[c_b =$

$c'_b]$, then the probability of success can be lower bounded as follows:

$$Prob[P_1^b[A_i] \text{ recovery succeeds}] \geq \sum_{B_i=256}^{511} \frac{1}{256}\left(1-(1-p_1p_2)^{|S_P^{-1}(B_i)|}\right) \quad (16)$$

The values $|S_P^{-1}(B_i)|$ are given by Table 4 and $Prob[\delta^b = 1] = \frac{1}{2}$. As for $Prob[c_b = c'_b]$, it can be modelled as the probability that there exists a carry at bit b in two random sums [8]. It achieves a minimum for $b = 31$ and thus the lower bound for the success probability over possible bit positions is given by $Prob[\text{Recovery of } P_1^b[A_i] \text{ succeeds}] \geq 0.908$.

Now, the probability that for some particular $k \in \{0, \dots 255\}$, the value $P_1^b[k]$ will not be recovered in some particular step i is then less than $1 - \frac{1}{256} \times 0.908$. Let $Z_k = 1$ if $P_1^b[k]$ has not been recovered after applying the algorithm for all steps $i = 512, \dots 1023$. Otherwise, let $Z_k = 0$. The number of $P_1^b[k]$, $0 \leq k \leq 255$ values not recovered can then be estimated as

$$E(\sum_{k=0}^{255} Z_k) = \sum_{k=0}^{255} E(Z_k) \leq 256 \times (1 - \frac{1}{256} \times 0.908)^{512} = 41.511 \quad (17)$$

Thus, the method presented in this section when applied on steps $i = 512, \dots 1023$, is expected to recover more than $256 - 41.511 = 214.49$ of the $P_1^b[0], \dots P_1^b[255]$ values. The exact procedure is presented by Algorithm 4.

As for recovering $Q_1^b[0], \dots Q_1^b[255]$, an analogous technique, applied on steps $i = 1024, \dots 1535$, is used. The exact procedure is presented by Algorithm 5. The expected number of recovered values is calculated analogously to (16), whereas it needs to be taken into account that A_i and B_i, $i \in \{1024, \dots 1535\}$, need to be successfully recovered by subsection 5.2. The $|S_Q^{-1}(B_i)|$ values, given at Table 3, are more favorable than the corresponding $|S_P^{-1}(B_i)|$ values in (16). The expected number of $Q_1^b[0], \dots Q_1^b[255]$ values to be recovered is 218.01.

5.5 Utilizing equations in faulty bits

In this subsection, the system constructed in subsections 5.3 and 5.4 is expanded further for the purpose of attaining the full rank of the system. Consider the faulty output word in steps $512, \dots 1023$, $s'_i = h_2(Q'_1[j \boxminus 12]) \oplus Q'_1[j]$. Evidently,

l	0	1	2	3	4	5	6	7	8	9	
$\|S_Q^{-1}(l)\|$	2	2	3	4	4	5	6	6	7	8	
l	10	11	12	13	14	15	16	17	18	19, ... 500	
$\|S_Q^{-1}(l)\|$	9	9	9	10	10	10	11	11	11	12	
l	501	502	503	504	505	506	507	508	509	510	511
$\|S_Q^{-1}(l)\|$	11	10	10	9	8	8	7	6	6	5	4

Table 3. The number of fault positions which allow the recovery of $Q_1[l] \oplus Q'_1[l]$

l	268	269	270	271	272	273	274	275	276	277	278
$\lvert S_P^{-1}(l)\rvert$	1	2	2	3	4	4	5	6	6	7	8
l	279	280	281	282	283	284	285	286	287	$288,\ldots 510$	511
$\lvert S_P^{-1}(l)\rvert$	9	9	9	10	10	10	11	11	11	12	11

Table 4. The number of fault positions which allow the recovery of $P_1[l] \oplus P'_1[l]$

Algorithm 5: Recovery of $Q_1^b[A_i]$, for some $i = 1024, \ldots 1535$

INPUT: Step $i \in \{1024, \ldots 1535\}$
OUTPUT: Bit $Q_1^b[A_i]$, or *undef*

```
1: If $A_i$ or $B_i$ are unknown, return undef
2: For every faulty keystream, where the fault occurred at $Q[f]$, $Q \in S_Q^{-1}(B_i)$
3:     Calculate $\delta = s_{512+B_i} \oplus s'_{512+B_i}$ and $\varDelta = s_i \oplus s'_i$
4:     If $Q_1^{b-1..0}[x] + Q_1^{b-1..0}[B_i] < 2^b$, set $c_b = 0$, else set $c_b = 1$
5:     If $Q_1^{b-1..0}[x] + Q_1'^{b-1..0}[B_i] < 2^b$, set $c'_b = 0$, else set $c'_b = 1$
6:     If $c_b = c'_b = 0$: Return: $\delta^{b+1} \oplus \varDelta^{b+1}$
6:     If $c_b = c'_b = 1$: Return $\delta^{b+1} \oplus \varDelta^{b+1} \oplus 1$
8: Return undef
```

regarding the previous relation as an equation is useless since it includes faulty inner state bits. Below, a method to transform the faulty inner state bits participating in the equation to original inner state bits is provided. Again, the *reuse* of inner state words is utilized.

Let the fault position be $Q[f]$, where $f \in S_Q^{-1}(l)$ and $244 \leq l \leq 499$. The non-faulty and the faulty instances of the cipher in step $i_0 = 512 + l + 12$ are

$$s_{i_0} = h_2(Q_1[l]) \oplus Q_1[l+12],\ s'_{i_0} = h_2(Q'_1[l]) \oplus Q'_1[l+12] \tag{18}$$

Note that $Q_1^{7..0}[l] = A_{i_0}$ and $Q_1^{23..16}[l] = B_{i_0} - 256$ are known according to subsection 5.1 and that the difference $Q'_1[l] \oplus Q_1[l]$ can be calculated as $\delta = Q'_1[l] \oplus Q_1[l] = s_{512+l} \oplus s'_{512+l}$, according to Lemma 4. Thus, A'_{i_0} and B'_{i_0} can be recovered as

$$A'_{i_0} = Q_1^{7..0}[l] \oplus \delta^{7..0},\ B'_{i_0} = Q^{23..16}[l] \oplus \delta^{23..16} + 256 \tag{19}$$

So, the second equation in line (18), considering only bit b, can be rewritten as

$$s_{i_0}^{'b} \oplus c_{i_0,b} = P_1^b[A'_{i_0}] \oplus P_1^b[B'_{i_0}] \oplus Q_1^{'b}[l+12] \tag{20}$$

where A'_{i_0} and B'_{i_0} are known and $c_{i_0,b}$ is an indicator of the carry in $P_1^{b-1..0}[A'_{i_0}] + P_1^{b-1..0}[B'_{i_0}]$ which is also known due to the assumption that bits $b-1, \ldots 0$ of all the P and Q words are known. Finally, to add equation (20) to the system constructed in the previous sections, the variable $Q_1^{'b}[l+12]$ needs to be eliminated. Once again, to reexpress $Q_1^{'b}[l+12]$, the idea is to wait for this value to be *reused* once more during steps $1024, \ldots 1535$.

Due to the assumed lower bound $244 \leq l$, it follows that $l + 12 \geq 256$. Hence, it is possible for the B_i index in some step $1024 \leq i \leq 1535$ to take the value $l+12$ which was used in step i_0. If such a step exists, denote it by i_1. Also, assume that $A_{i_1} < f-1$, so that $Q_1[A_{i_1}] = Q'_1[A_{i_1}]$. Finally, assume that both A_{i_1} and B_{i_1} have been successfully recovered by the procedure given in subsection 5.2. Then, if $j_1 = i_1 \bmod 512$, the non-faulty and faulty keystream words are $s_{i_1} = (Q_1[A_{i_1}] + Q_1[B_{i_1}]) \oplus P_2[j_1]$ and $s'_{i_1} = (Q_1[A_{i_1}] + Q'_1[B_{i_1}]) \oplus P_2[j_1]$ and the difference can be computed as

$$s_{i_1} \oplus s'_{i_1} = (Q_1[A_{i_1}] + Q_1[B_{i_1}]) \oplus (Q_1[A_{i_1}] + Q'_1[B_{i_1}]) \tag{21}$$

Extracting bit b from (21) and cancelling out $Q_1[A_{i_1}]$ yields

$$s^b_{i_1} \oplus s'^b_{i_1} = Q^b_1[B_{i_1}] \oplus c_{i_1,b} \oplus Q'^b_1[B_{i_1}] \oplus c'_{i_1,b} \tag{22}$$

where $c_{i_1,b}$ and $c'_{i_1,b}$ are carry indicators for $Q^{b-1..0}_1[A_{i_1}] + Q^{b-1..0}_1[B_{i_1}]$ and $Q^{b-1..0}_1[A_{i_1}] + Q'^{b-1..0}_1[B_{i_1}]$, respectively. The carry indicator $c_{i_1,b}$ is calculated trivially and as for $c'_{i_1,b}$, it is necessary to find $Q'^{b-1..0}_1[B_{i_1}]$. For that, it suffices to focus on the bits $b-1, \ldots 0$ in equation (21), since all values except $Q'^{b-1..0}_1[B_{i_1}]$ are known and the required value can be calculated as

$$Q'^{b-1..0}_1[B_{i_1}] = ((s^{b-1..0}_{i_1} \oplus s'^{b-1..0}_{i_1}) \oplus (Q^{b-1..0}_1[A_{i_1}] + Q^{b-1..0}_1[B_{i_1}])) - Q^{b-1..0}_1[A_{i_1}] \tag{23}$$

After finding $c_{i_1,b}$ and $c'_{i_1,b}$, from (22) and since $B_{i_1} = l+12$, $Q'_1[l+12]$ can be expressed in terms of Q_1 bits as

$$Q'^b_1[l+12] = s^b_{i_1} \oplus s'^b_{i_1} \oplus Q^b_1[B_{i_1}] \oplus c_{i_1,b} \oplus c'_{i_1,b} \tag{24}$$

Substituting (24) in (20) yields

$$s'^b_{i_0} \oplus c_{i_0,b} = P^b_1[A'_{i_0}] \oplus P^b_1[B'_{i_0}] \oplus s^b_{i_1} \oplus s'^b_{i_1} \oplus Q^b_1[B_{i_1}] \oplus c_{i_1,b} \oplus c'_{i_1,b} \tag{25}$$

which is added to the system of equations without introducing any new variables. The described procedure is summarized by Algorithm 6.

Algorithm 6: Add equations by expressing the faulty with non-faulty bits

INPUT: Faulty keystream for a fault occuring at $Q[f]$, $f \in S^{-1}_P(l)$, $244 \le l \le 499$
OUTPUT: An equation of form (25)

```
1: Let $\delta = s_{512+l} \oplus s'_{512+l}$ and $i_0 = 512 + l + 12$
2: Calculate $A'_{i_0}$ and $B'_{i_0}$ according to (19)
3: If $P^{b-1..0}_1[A'_{i_0}] + P^{b-1..0}_1[B'_{i_0}] < 2^b$ set $c_{i_0,b} = 0$, else $c_{i_0,b} = 1$
4: For $1024 \le i_1 \le 1535$ such that $A_{i_0}$ and $B_{i_0}$ are known
5:     If $B_{i_0} = l + 12$ and $A_{i_0} < f - 1$
6:         If $Q^{b-1..0}_1[A_{i_1}] + Q^{b-1..0}_1[B_{i_1}] < 2^b$, let $c_{i_1,b} = 0$, else $c_{i_1,b} = 1$
7:         Calculate $Q'^{b-1..0}_1[B_{i_1}]$ according to (23)
8:         If $Q^{b-1..0}_1[A_{i_1}] + Q'^{b-1..0}_1[B_{i_1}] < 2^b$, let $c'_{i_1,b} = 0$, else $c'_{i_1,b} = 1$
9:         Return equation (25)
```

Let N denote the number of equations generated by repeating the procedure above for all $f \in S^{-1}_Q(l)$ and $244 \le l \le 499$. To estimate $E(N)$, let $\rho(l+12)$ be the step number i, $1024 \leq i \leq 1535$, for which $B_i = l + 12$, if such a step exists. Also, let I denote the indicator function, returning 1 if the condition in question is true and returning 0 otherwise. Finally, let $FLT_Q[f]$ be the number of faults that occurr at position $Q[f]$. Then

$$N = \sum_{l=244}^{499} \sum_{f \in S_Q^{-1}(l)} FLT_Q[f] \times I[\rho(l+12) \text{ exists}] \times I[A_{\rho(l+12)} < f - 1] \\ \times I[A_{\rho(l+12)}, B_{\rho(l+12)} \text{ known}]$$

Recall that $E(FLT_Q[f]) = 7.52$. Also, $E(I[\rho(l+12) \text{ exists}]) \approx 1 - (\frac{255}{256})^{512}$. If $f > 257$, $E(I[A_{\rho(l+12)} < f - 1]) = 1$ and otherwise $\frac{f-2}{256}$. Finally, according to subsection 5.2, $E(I[A_{\rho(l+12)}, B_{\rho(l+12)} \text{ known}]) \geq \frac{472.7}{512}$. Substituting the values above and using additivity of $E(\cdot)$ yields that $E(N) \geq 18380.1$.

6 Attack complexity and experimental results

Adding the number of equations generated by the algorithms presented in subsections 5.3, 5.4 and 5.5 gives a lower bound of $512+214.49+218.01+18380.1 = 19324.6$ equations expected to be in the final system for bits $b \in \{0, \ldots 30\}$. The correctness of the system and the uniqueness of the solution have been verified experimentally as follows. For 100 times HC-128 was randomly initialized and the faults have been simulated as specified by the attack. The procedures specified by by subsections 5.3, 5.4 and 5.5 have been executed and the rank of the resulting system of equations for bits $b \in \{0, \ldots 30\}$ was verified to be 1024 in all the 100 times. As for bit $b = 31$, the procedures from subsection 5.4 are not applicable, leaving out the system to be generated only by subsections 5.3 and 5.5, yielding about $512 + 18380.1 = 18892.1$ equations. Again, throughout the 100 experiments, the rank of resulting system for bit $b = 31$ was 1022 each time. Thus, to yield a complete HC-128 inner state, the missing two bits need to be guessed. The correctness of the guessed bits is easily verified by running the cipher and comparing the resulting key stream with the observed one. As for the attack complexity, around 7698.4 faults at random inner state words are required, as given by the beginning of section 5. The most expensive computational factor in the attack is solving the linear system of equations in 1024 bit variables for 32 times.

7 Conclusion

In this paper, a DFA attack on HC-128 was presented. The adopted attack model assumes that the attacker is able to fault a random word of the inner state of the cipher but cannot control its exact location nor its new faulted value. The attack operates by constructing 32 systems of linear equations over Z_2, each of 1024 bit variables representing the inner state values. It also utilizes what we called the *reuse* of inner state words in different states of the cipher in order to facilitate the differential fault analysis.

References

1. Biham, E., Shamir, A.: Differential Fault Analysis of Secret Key Cryptosystems. In: Kaliski, B.S. (ed) Proceedings of CRYPTO 1997. LNCS, vol. 1294, pp. 513-525, Springer-Verlag (1997)
2. Boneh, D., Demillo, R.,A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults, In: Fumy, W. (ed) Proceedings of Eurocrypt 1997, LNCS, vol. 1233, pp. 37-51, Springer, Heidelberg (1997)
3. Dusart, P., Letourneux, G., Vivolo, O.: Differential fault analysis on AES, In: Zhou, J., Yung, M., Han, Y. (eds), Applied Cryptography and Network Security, ACNS 2003. LNCS, vol. 2846, pp. 293-306, Springer, Heidelberg (2003)
4. Hoch, J., Shamir, A.: Fault Analysis of Stream Ciphers. In: Joye, M., Quisqater, J. J. (eds) Cryptographic Hardware and Embedded Systems, CHES 2004, LNCS, vol. 3156, pp. 240-253, Springer, Heidelberg (2004)
5. Kircanski, A., Youssef, M., A.: Differential Fault Analysis of Rabbit, In: Jacobson, M., J. J., Rijmen, V., Safavi-Naini, R. (eds), Selected Areas in Cryptography, SAC 2009, LNCS-5867, pp. 197-214 (2009)
6. Maitra, S., Paul, G., Raizada, S.: Some observations on HC-128, In: Proceedings of the International Workshop on Coding and Cryptography, WCC, May 10-15, 2009, Ullensvang, Norway, pages 527-539
7. Mitzenmacher, M., Upfal, E.: Probability and Computing, Cambrige University Press, ISBN-10: 0521835402
8. Staffelbach, O., Meier, W.. In: Menezes, A., Vanstone, S., A.: Cryptographic Significance of the Carry for Ciphers Based on Integer Addition, Proceedings of CRYPTO '90, LNCS, Vol. 537, Springer (1990)
9. Wu, H.: The Stream Cipher HC-128. In: Robshaw, M., Billet, O. (eds) New Stream Cipher Designs, LNCS, vol. 4986, pp. 39-47, Springer Berlin Heidelberg (2008)
10. Wu, H.: A new stream cipher HC-256. In: Roy, K., B., Meier, W. (eds), FSE 2004, LNCS, vol. 3017, pp. 226-244, Springer Berlin-Heidelberg (2004)
11. Zenner, E.: A Cache Timing Analysis of HC-256. In: Avanzi, R., M., Keliher, L., Sica, F.: Selected Areas in Cryptograhy, SAC 2008, LNCS, vol. 5381, Springer (2008)