A New Class of RC4 Colliding Key Pairs With Greater Hamming Distance

Jiageng Chen * and Atsuko Miyaji**

Japan Advanced Institute of Science and Technology, 1-1 Asahidai, Nomi, Ishikawa 923-1292, Japan
{jg-chen, miyaji}@jaist.ac.jp

Abstract. In this paper, we discovered a new class of colliding key pairs of RC4, namely, two different secret keys generate the same internal state after RC4's key scheduling algorithm. This is to our knowledge the first discovery of RC4 colliding keys with hamming distance greater than one, that is, the colliding key pairs we found can differ from each other at three different positions, and the value difference between these positions needs not be fixed. We analyzed the transition pattern and evaluated the probability of the existence of this new class of colliding key pairs. Our result shows that RC4 key collision could be achieved by two keys with greater hamming distance than the ones found in [1] and [2]. And this newly discovered class of colliding key pairs reveals the weakness that RC4's key scheduling algorithm could generate even more colliding keys. We also proposed an algorithm for searching colliding key pairs within this new class. Some concrete colliding key pairs are demonstrated in this paper, among which 55-byte colliding key pair is the shortest one we found by using our algorithm within one hour time.

1 Introduction

The RC4 stream cipher is one of the most famous ciphers widely used in real world applications such as Microsoft Office, Secure Socket Layer (SSL), Wired Equivalent Privacy (WEP), etc. Due to its popularity and simplicity, RC4 has become a hot cryptanalysis target since its specification was made public on the Internet in 1994 [3]. Various general weaknesses of RC4 have been discovered in some previous works including [4–6], etc. Another popular cryptanalysis area for RC4 is in the WEP environment. Such works include [7–10], etc.

Our paper focuses on RC4 key collisions. It describes the existence of secret key pairs that generate the same initial states after key scheduling algorithm. The study of "colliding keys" of RC4 can be dated back to 2000. It was pointed out by Grosul and Wallach [1] that RC4 has related-key pairs that generate hundreds of substantially similar output bytes when the key size is close to the full 256 bytes. In Matsui's paper [2], much stronger key collisions with shorter key

* This author is supported by the Graduate Research Program
** This work is supported by Grant-in-Aid for Scientific Research (B), 20300003.

size were explored. A state transition sequence of the key scheduling algorithm for a related key pair of an arbitrary fixed length which can lead to key collisions was presented.

In both [1] and [2], the colliding keys found only have hamming distance of 1. In this paper, we demonstrate another class of colliding key pairs. Differing from [1] and [2], we showed for the first time that there exist colliding key pairs whose hamming distance is 3, which is greater than 1. That is, two colliding keys we found could differ at three positions and the value differences at these three different positions do not need to be fixed. Our result shows that there exist many more colliding key pairs than previously expected. We estimated the number of colliding key pairs within this newly found class and also we proposed a search algorithm which can be used to easily find such colliding key pairs.

Structure of the paper. In Section 2, we briefly describe the RC4 algorithm followed by some previous works on RC4 key collision in Section 3. In Section 4, we explain our new transition pattern which will lead to the collision. We give probability evaluations in Section 5 and finally we describe an efficient colliding key pair search algorithm. Some techniques needed for the probability analysis, the total number of colliding key pairs we found and some concrete colliding key examples are summarized in the Appendix.

2 Preparation

2.1 Description of RC4

The internal state of RC4 consists of a permutation S of the numbers $0, ..., N-1$ and two indices $i, j \in \{0, ..., N-1\}$. The index i is determined and known to the public, while j and permutation S remain secret. RC4 consists of two algorithms: The Key Scheduling Algorithm (KSA) and the Pseudo Random Generator Algorithm (PRGA). The KSA generates an initial state from a random key K of k bytes as described in Algorithm 1. It starts with an array $\{0, 1, ..., N-1\}$ where $N = 256$ by default. At the end, we obtain the initial state S_{N-1}.
Once the initial state is created, it is used by PRGA to generate a keystream of bytes which will be XORed with plaintext to generate ciphertext. PRGA is described in Algorithm 2. In this paper, we focus only on KSA.

Algorithm 1. KSA

for $i = 0$ **to** $N-1$ **do**
    $S[i] \leftarrow i$
end for
$j \leftarrow 0$
for $i = 0$ **to** $N-1$ **do**
    $j \leftarrow j + S[i] + K[i \bmod l]$
    swap$(S[i], S[j])$
end for

Algorithm 2. PRGA

$i \leftarrow 0$
$j \leftarrow 0$
loop
    $i \leftarrow i + 1$
    $j \leftarrow j + S[i]$
    swap$(S[i], S[j])$
keystream byte $z_i = S[S[i] + S[j]]$
end loop

2.2 Previous Research on RC4 key collisions

Two important studies on RC4 key collisions are [1] and [2]. In [1], the authors pointed out that for secret keys with lengths close to 256 bytes, it's possible for two keys to generate similar internal states after KSA and thus may generate hundreds of similar bytes output during PRGA. Intuition in [1] comes from that for two keys K_1, K_2, assume $K_1[i] = K_2[i]$ except when $i = t$. When t is close to 255, internal state will be substantially similar. However, their discovery cannot be strictly called key collision and their result only works for key length close to 256.

In [2], RC4 key collision for shorter key length was first explained and a 24-byte key collision was found. The key pattern is almost same as in [1], namely, two keys differ at only one byte position ($K_1[i] = K_2[i]$ except when $i = t$) and value difference is $1(K_1[t] = K_2[t] - 1)$. Under the above condition, [2] is able to find a transition pattern that could lead to a total collision after KSA. The intuition behind the transition pattern is that from the first time i touches the different position t, the pattern ensures that there will always be only two differences in the internal state as the key scheduling process goes on. And the difference is absorbed when i touches t for the last time. Please refer to [2] for the detailed description.

Our result shows that RC4 key collision can also be achieved even if the hamming distance between two keys is greater than 1. Also, differing from [2], our transition pattern is a self-absorbing type, which means that once the internal states differences are generated due to the key differences, they will be absorbed very quickly and the internal states returns to the same till i touches the next key difference. We show this interesting pattern in the following section.

3 New transition pattern

3.1 Notations and intuitions

The following are the notations used in this paper.

K_1, K_2: Secret keys which satisfy $K_1[d] = K_2[d] - t$, $K_1[d+1] = K_2[d+1] + t$, $K_1[d+t+1] = K_1[d+t+1] - t$. $K_1[i] = K_2[i]$ for $i \neq d, d+1, d+t+1$.

$S_{1,i}, S_{2,i}$: S-boxes corresponding to $K_1(K_2)$ at time i. When $S_{1,i}[p] = S_{2,i}[p]$ for some p, we use $S_i[p]$ to denote.

$i, j_{1,i}, j_{2,i}$: Internal state. $j_{1,i}(j_{2,i})$ corresponds to $K_1(K_2)$ at time i. When $j_{1,i} = j_{2,i}$, we use j_i to denote.

d: Index of the first position where the values of the two secret keys differ from each other.

t: The value difference between two keys at index $d(t = K_1[d] - K_2[d])$.

k: The length(bytes) of the secret key.

n: The number of times $K_1[d], ..., K_1[d+t+1](K_2[d], ..., K_2[d+t+1])$ appear during the KSA. $n = \lfloor \frac{256+k-1-d}{k} \rfloor$

The colliding key pairs' behavior is influenced by three parameters d, t, n. In order to understand how a key pair with the relation described before can achieve

collision after KSA, we explain by illustrating three examples with different d, t, n. In the simple case where $n = 1$ (different positions appear only once during KSA), example 1 and 2 are given to show that collision can be achieved by various d and t. Example 3 shows that in more complex cases where $n > 1$, collisions are still possible. We analyze the key scheduling process for key pairs with the above relation that can lead to collisions. We summarize the transition pattern in the Table 1,2 and 3, which helps in tracking the internal state of RC4 when i touches the different positions of the two keys. The internal state part of each table tracks the information of i, j, K and S-BOX generated by the two keys. Notice that the value of S-BOX denotes the state after the swap. The "Difference" part of each table tells us the difference between two internal states at time i.

3.2 Example 1 ($d = 2, t = 2, n = 1(k = 256)$)

Table 1. $d = 2, t = 2, n = 1(k = 256)$

Internal State									Difference
i	$K_1[i]/K_2[i]$	$j_{1,i}/j_{2,i}$	1	2	3	4	5	6	
2	$256 - j_1$	2	*	2	3	4	5	*	$K_1[2] = K_2[2] - 2$
	$256 - j_1 + 2$	4	*	4	3	2	5	*	$j_{1,2} = j_{2,2} - 2, S_1 \neq S_2$
3	$X + 2$	$X + 7$	*	2	$S[X + 7]$	4	5	*	$K_1[3] = K_2[3] + 2$
	X	$X + 7$	*	4	$S[X + 7]$	2	5	*	$j_{1,3} = j_{2,3}, S_1 \neq S_2$
4	$256 - X - 7$	4	*	2	$S[X + 7]$	4	5	*	$K_1[4] = K_2[4]$
	$256 - X - 7$	2	*	2	$S[X + 7]$	4	5	*	$j_{1,4} = j_{2,4} + 2, S_1 = S_2$
5	Y	$Y + 9$	*	2	$S[X + 7]$	4	$S[Y + 9]$	*	$K_1[5] = K_2[5] - 2$
	$Y + 2$	$Y + 9$	*	2	$S[X + 7]$	4	$S[Y + 9]$	*	$j_{1,5} = j_{2,5}, S_1 = S_2$

Let's demonstrate the case in which the key length is $k = 256$ byte, $d = 2$ and $t = 2$. Our key pattern requires that $K_1[d] = K_2[d] - t$, $K_1[d+1] = K_2[d+1] + t$, $K_1[d + t + 1] = K_1[d + t + 1] - t$ and all other parts of the key must be the same. Now we describe how the new transition pattern works. First, before i touches index d, in this example we assume $j_1(j_2)$ does not touch d or $d + 2$. When $i = d$, we require $j_{1,2} = d(j_{1,2} = 2)$ and $j_{2,2} = d + 2(j_{2,2} = 4)$. Since $j_{1,2} = j_1 + 2 + K_1[2]$ and $j_{2,2} = j_1 + 4 + K_2[2]$ then according to the key pattern, we know that $K_1[2] = 256 - j_1$ and $K_2[2] = 256 - j_1 + 2$. Therefore, at step $i = d$, $S_1[d]$ will not be swapped and $S_2[d]$ will be swapped to index $d + t$. When i increases to $d + 1$, we need $K_1[d + 1]$ and $K_2[d + 1]$ to absorb the difference between j_1 and j_2. In this example, we let $K_1[3] = X + t$ and $K_2[3] = X$ for some X. When i touches $d + t$, namely, $i = 4$, we need $j_{1,4} = d + t$ and $j_{2,4} = d$, namely, $j_{1,4} = 4$ and $j_{2,4} = 2$ so that the difference between the two S-Boxes is absorbed. Now the internal difference exists only between $j_{1,4}$ and $j_{2,4}$. Finally when $i = d + t + 1$, namely, $i = 5$, then $K_1[d + t + 1](\mathrm{Y})$ and $K_2[d + t + 1])(\mathrm{Y}+2)$

are there to absorb the difference between $j_{1,4}$ and $j_{2,4}$. For the rest of the steps($i > d + t + 1$), the internal state remains the same, until the end of KSA. Table 1 describes the above transition procedure.

3.3 Example 2 ($d = 3, t = 3, n = 1(k = 256)$)

In Example 2, we change the values d and t to demonstrate that parameters d, t need not be fixed in order to achieve collisions. The transition pattern is summarized in Table 2. It is easy to understand by following Example 1.

Table 2. $d = 3, t = 3, n = 1(k = 256)$

Internal State								Difference
i	$K_1[i]/K_2[i]$	$j_{1,i}/j_{2,i}$	2	3	4	5	6	7	
3	$256 - j_1$	3	*	3	4	5	6	7	$K_1[3] = K_2[3] - 3$
	$256 - j_1 + 3$	6	*	6	4	5	3	7	$j_{1,3} = j_{2,3} - 3, S_1 \neq S_2$
4	$X + 3$	$X + 10$	*	3	$S[X + 10]$	5	6	7	$K_1[4] = K_2[4] + 3$
	X	$X + 10$	*	6	$S[X + 10]$	5	3	7	$j_{1,4} = j_{2,4}, S_1 \neq S_2$
5	Y	$X + Y + 15$	*	3	$S[X + 10]$	$S[X + Y + 15]$	6	7	$K_1[5] = K_2[5]$
	Y	$X + Y + 15$	*	6	$S[X + 10]$	$S[X + Y + 15]$	3	7	$j_{1,5} = j_{2,5}, S_1 \neq S_2$
6	$256 - (X + Y + 15)$	6	*	3	$S[X + 10]$	$S[X + Y + 15]$	6	7	$K_1[6] = K_2[6]$
	$256 - (X + Y + 15)$	3	*	3	$S[X + 10]$	$S[X + Y + 15]$	6	7	$j_{1,6} = j_{2,6} + 3, S_1 = S_2$
7	Z	$Z + 13$	*	3	$S[X + 10]$	$S[X + Y + 15]$	5	$S[Z + 13]$	$K_1[7] = K_2[7] - 3$
	$Z + 3$	$Z + 13$	*	3	$S[X + 10]$	$S[X + Y + 15]$	5	$S[Z + 13]$	$j_{1,7} = j_{2,7}, S_1 = S_2$

3.4 Example 3 ($d = 4, t = 2, n = 2(k = 128)$)

Notice that Examples 1 and 2 directly apply to the situations where the three different positions of the key appear only once during the KSA, such as the 256-byte key. However, when the three different positions of the key appear more than once, in other words, when the key becomes short, extra conditions need to be satisfied in order to achieve collision. Table 3 is one example for $k = 128, d = 4, t = 2$; in this case, the different positions of the key will appear twice during KSA. This example will give us all the techniques for even shorter keys. In order to extend the pattern to normal situations, we use abstract values such as a, b instead of specific values. The transition pattern is summarized in table 3.

In this example, index $4, 5, 7$ is the first appearance ($n = 1$) of the different positions of the key, while index $132, 133, 135$ is the second appearance ($n = 2$). After i touches index 135, the two internal states become same for the rest of KSA. Because the key pairs are determined during the first appearance (index $4, 5, 7$), index $132, 133, 135$ have to use the same keys as index $4, 5, 7$. So it is very

Table 3. $d = 4, t = 2, n = 2 (k = 128)$

Internal State							Difference
i	$K_1[i]/K_2[i]$	$j_{1,i}/j_{2,i}$	4	5	6	7	
4	$d-a-j_3$	d	a	b	$a+2$	$S[7]$	$K_1[4] = K_2[4] - 2$
	$d-a-j_3+2$	$d+2$	$a+2$	b	a	$S[7]$	$j_{1,4} = j_{2,4} - 2, S_1 \neq S_2$
5	$X+2$	$X+b+d+2$	a	$S_1[X+b+d+2]$	$a+2$	$S[7]$	$K_1[5] = K_2[5] + 2$
	X	$X+b+d+2$	$a+2$	$S_2[X+b+d+2]$	a	$S[7]$	$j_{1,5} = j_{2,5}, S_1 \neq S_2$
6	$254-X-a-b$	$d+2$	a	$S_1[X+b+d+2]$	$a+2$	$S[7]$	$K_1[6] = K_2[6]$
	$254-X-a-b$	d	a	$S_2[X+b+d+2]$	$a+2$	$S[7]$	$j_{1,6} = j_{2,6} + 2, S_1 = S_2$
7	Y	$Y+d+2+S[7]$	a	$S_1[X+b+d+2]$	$a+2$	$S_1[Y+d+2+S[7]]$	$K_1[7] = K_2[7] - 2$
	$Y+2$	$Y+d+2+S[7]$	a	$S_2[X+b+d+2]$	$a+2$	$S_2[Y+d+2+S[7]]$	$j_{1,7} = j_{2,7}, S_1 = S_2$
i	$K_1[i]/K_2[i]$	$j_{1,i}/j_{2,i}$	132	133	134	135	
132	$d-a-j_3$	132	$j_3-j_{131}+132+a-d$	$S[133]$	$j_3-j_{131}+134+a-d$	$S[135]$	$K_1[132] = K_2[132] - 2$
	$d-a-j_3+2$	134	$j_3-j_{131}+134+a-d$	$S[133]$	$j_3-j_{131}+132+a-d$	$S[135]$	$j_{1,132} = j_{2,132} - 2, S_1 \neq S_2$
133	$X+2$	$X+S[133]+134$	$j_3-j_{131}+132+a-d$	$S[X+S[133]+134]$	$j_3-j_{131}+134+a-d$	$S[135]$	$K_1[133] = K_2[133] + 2$
	X	$X+S[133]+134$	$j_3-j_{131}+134+a-d$	$S[X+S[133]+134]$	$j_3-j_{131}+132+a-d$	$S[135]$	$j_{1,133} = j_{2,133}, S_1 \neq S_2$
134	$254-X-a-b$	134	$j_3-j_{131}+132+a-d$	$S[X+S[133]+134]$	$j_3-j_{131}+134+a-d$	$S[135]$	$K_1[134] = K_2[134]$
	$254-X-a-b$	132	$j_3-j_{131}+132+a-d$	$S[X+S[133]+134]$	$j_3-j_{131}+134+a-d$	$S[135]$	$j_{1,134} = j_{2,134} + 2, S_1 = S_2$
135	Y	$Y+134+S[135]$	$j_3-j_{131}+132+a-d$	$S[X+S[133]+134]$	$j_3-j_{131}+134+a-d$	$S[Y+134+S[135]]$	$K_1[135] = K_2[135] - 2$
	$Y+2$	$Y+134+S[135]$	$j_3-j_{131}+132+a-d$	$S[X+S[133]+134]$	$j_3-j_{131}+134+a-d$	$S[Y+134+S[135]]$	$j_{1,135} = j_{2,135}, S_1 = S_2$

natural to think that in order for the transition pattern to work for shorter keys, there are some extra conditions that need to be satisfied and the requirement for those extra conditions come from the fact that keys are determined in the first appearance and could not be changed from that point. According to the transition table 3, we can derive $j_{1,134} = j_{1,133} + K_1[6] + S_{1,133}[134] = (X + S[133] + 134) + (254 - X - a - b) + (j_3 - j_{131} + 134 + a - d)$, thus $S[133] = a + b + j_{131} - j_3 - a + d - 132$. We also know from the table that $S[132] = j_3 - j_{131} + 132 + a - d$. This gives us $S[132] + S[133] = a + b$. This is exactly the extra condition that our transition sequence should satisfy in the case of $n = 2$. In summarizing, the following conditions should be satisfied in order to achieve a collision for case $d = 4, t = 2, n = 2(k = 128)$. All the conditions could be derived from the transition table 3 except for the little tricky one $S[132]+S[133] = a+b$, which we have just calculated.

$$\begin{cases} S_3[4] + 2 = S_3[6] & (1) \\ j_{1,4} = 4(j_{2,4} = 6) & (2) \\ j_5 \neq 4, 6 & (3) \\ j_{1,6} = 6(j_{2,6} = 4) & (4) \\ \\ S_{131}[132] + 2 = S_{131}[134] & (5) \\ j_{1,132} = 132(j_{2,132} = 134) & (6) \\ j_{133} \neq 132, 134 & (7) \\ S_{131}[132] + S_{131}[133] = a + b & (8) \\ j_{1,134} = 134(j_{2,134} = 132) & (9) \end{cases}$$

Notice that (5) and (8) will give (9), so actually we only need (1) to (8). From the above analysis, we could easily derive the sufficient and necessary conditions for our transition pattern, namely,

1. $j_{d+pk} = d + pk$ for $p = 0, ..., n - 1$. (corresponding to (2),(6))
2. $j_{d+pk+t} = d + pk + t$ for $p = 0$. (corresponding to (4),(9))
3. $S_{d+pk}[d + pk] + t = S_{d+pk}[d + pk + t]$ for $p = 0, ..., n - 1$. (corresponding to (1),(5))
4. $(S_{d+pk}[d+pk]+\cdots+S_{d+pk}[d+pk+t-1]) \bmod 256 = S_d[d]+\cdots+S_d[d+t-1]$ for $p = 1, ..., n - 1$. (corresponding to (8))
5. $j_{d+pk+1}, ..., j_{d+pk+t-1} \neq d+pk, d+pk+t$ for $p = 0, ..., n-1$. (corresponding to (3),(7))

4 Probability Analysis

In this section, we give probability estimation for our transition pattern.

Assume that K_1 and K_2 form a k byte key pair which has the following properties: $K_1[d] = K_2[d] - t$, $K_1[d + 1] = K_2[d + 1] + t$, $K_1[d + t + 1] = K_1[d + t + 1] - t$ for some t, d and n. From the previous section, we know the conditions for our transition pattern. Now we formalize some conditions into events for convenience in the analysis.

Event A: $j_{d+pk} = d + pk$ for $p = 0, 1, 2, ..., n-1$.
Event B: $S_{d+pk}[d+pk] + t = S_{d+pk}[d+pk+t]$ for $p = 0, 1, ..., n-1$
Event C: $(S_{d+pk}[d+pk] + \cdots + S_{d+pk}[d+pk+t-1]) \bmod 256 = S_d[d] + \cdots + S_d[d+t-1]$ for $p = 1, 2, ..., n-1$.
Event D: $j_{d+pk+1}, ..., j_{d+pk+t-1} \neq d+pk, d+pk+t$ for $p = 0, ..., n-1$.

Lemma 1. *The probability of Event A is*

$$P(A) = P(j_{d+pk} = d + pk) = \frac{1}{256} \text{ for } p = 0, 1, ..., n-1$$

Proof. Here we assume that j behaves randomly, thus $P(j_{d+pk} = p + dk) = \frac{1}{256}$.

Lemma 2. *The probability of Event B is*

$$P(B_{d,k,p}) = \frac{255}{256} \times (\frac{254}{256})^{d+pk-1} + \frac{1}{256} \text{ for } p = 0, ..., n-1$$

Proof. There are two cases that could lead to $S_{d+pk}[d+pk]+t = S_{d+pk}[d+pk+t]$.
Case 1(Event D): $S_{d+pk}[d+pk]$ and $S_{d+pk}[d+pk+t]$ have not been swapped before. The probability for this case(Event D) is $(\frac{254}{256})^{d+pk-1}$.
Case 2(Event E): $S_{d+pk}[d+pk]$ and $S_{d+pk}[d+pk+t]$ have been touched before i touches $d+pk$. The probability of Event E is the complement of Event D, namely, $1 - (\frac{254}{256})^{d+pk-1}$.
According to the law of total probability, we have

$$\begin{aligned} P(B) &= P(B|D)P(D) + P(B|E)P(E) \\ &= 1 \times (\tfrac{254}{256})^{d+pk-1} + \tfrac{1}{256} \times (1 - (\tfrac{254}{256})^{d+pk-1}) \\ &= \tfrac{255}{256} \times (\tfrac{254}{256})^{d+pk-1} + \tfrac{1}{256} \end{aligned}$$

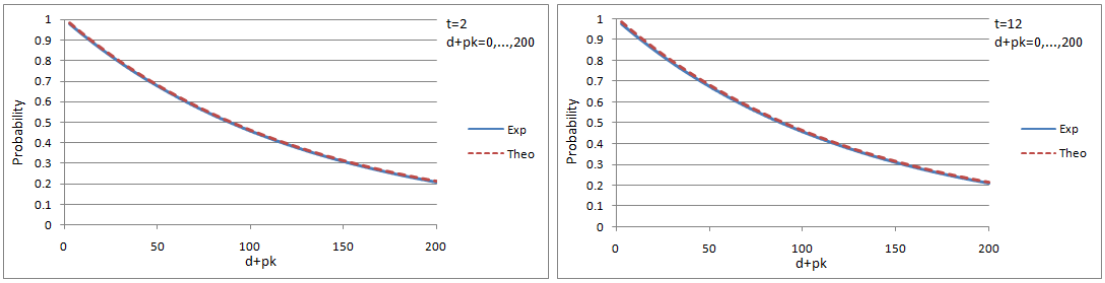

Fig. 1. Event B

Figure 1 shows the experimental and theoretical results with $d + pk$ from 3 to 200. From the figure, we can see that the theoretical result agrees with the experimental result very well. Notice that the probability of Event B is not affected by value t.

Before evaluating Event C, let's first look at the following Lemma 3.

Lemma 3. *The probability of $S_v[v^{'}]$ in round v equals to some value $u(u < v$ or $u = v^{'}, v^{'} \geq v)$ is as follows:*
(1) When $u = v^{'}$, then

$$P(S_v[v^{'}] = u) = (\frac{255}{256})^v$$

(2) When $u < v$, then

$$P(S_v[v^{'}] = u) = (\frac{255}{256})^{v-1} \times \frac{1}{256} + \sum_{t_1=u+1}^{v-1} (\frac{255}{256})^{v-2} \times (\frac{1}{256})^2 + \sum_{t_1=u+1}^{v-2} \sum_{t_2=t_1+1}^{v-1} (\frac{255}{256})^{v-3} \times (\frac{1}{256})^3 + \cdots + \sum_{t_1=u+1}^{v-(n-1)} \cdots \sum_{t_{n-1}=t_{n-2}+1}^{v-1} (\frac{255}{256})^{v-n} \times (\frac{1}{256})^n$$

where $n = v - u$.

Proof. (1) When $u = v^{'}$, $S_v[v^{'}] = v^{'}$ means after round v, $S[v^{'}]$ remains untouched. Thus the probability is $(\frac{255}{256})^v$.
(2) When $u < v$, u will be in index $v^{'}$ after v rounds(i touches $v-1$). In order for this to happen, first, we need u to remain untouched before i touches it, otherwise u will be swapped somewhere before i and the chance for it to be swapped to index $v^{'}$ will be lost. The probability of this is $(\frac{255}{256})^u$. Then when i touches u, we have several choices, either to swap u directly to index $v^{'}$(we call this one hop), or to swap u to an intermediate index t_1 between index u and v, and then when i touches t_1, u can be swapped to index $v^{'}$(we call this two hops), etc. As you may guess, there exist maximum $v - u$ hops. We analyze one-hop, two-hop and three-hop cases here.
One hop: When i touches u, u is swapped to index $v^{'}$ with probability $\frac{1}{256}$ and remains untouched for $v - u - 1$ rounds. Thus the probability is $(\frac{255}{256})^u \times \frac{1}{256} \times (\frac{255}{256})^{v-u-1} = (\frac{255}{256})^{v-1} \times \frac{1}{256}$
Two hops: When i touches u, u is first swapped to index t_1 between index u and v with probability $\frac{1}{256}$. For $t_1 - u - 1$ rounds, index t_1 remains untouched. Then when i touches t_1, u is swapped to index $v^{'}$ with probability $\frac{1}{256}$ and then index $v^{'}$ remains untouched for $v - t_1 - 1$ rounds. Notice that the intermediate index t_1 can vary between $u + 1$ and $v - 1$, thus the probability of the above is $\sum_{t_1=u+1}^{v-1} (\frac{255}{256})^u \times \frac{1}{256} \times (\frac{255}{256})^{t_1-u-1} \times \frac{1}{256} \times (\frac{255}{256})^{v-t_1-1} = \sum_{t_1=u+1}^{v-1} (\frac{255}{256})^{v-2} \times (\frac{1}{256})^2$
Three hops: The analysis is the same as for two hops. The only difference for three hops is that we have two intermediate indices t_1 and t_2, and t_1 can vary between index $u + 1$ and $v - 2$ and t_2 can vary between index $t_1 + 1$ and $v - 1$. Thus the probability is $\sum_{t_1=u+1}^{v-2} \sum_{t_2=t_1+1}^{v-1} (\frac{255}{256})^u \times \frac{1}{256} \times (\frac{255}{256})^{t_1-u-1} \times \frac{1}{256} \times (\frac{255}{256})^{t_2-t_1-1} \times \frac{1}{256} \times (\frac{255}{256})^{v-t_2-1} = \sum_{t_1=u+1}^{v-2} \sum_{t_2=t_1+1}^{v-1} (\frac{255}{256})^{v-3} \times (\frac{1}{256})^3$.
For cases of more than 4 hops, the above analysis works in the same way. By summing these results together, we get our final result.

We tested 4 sets of data where $(v, v^{'})$ equal to $(64, 64), (64, 74), (128, 128)$ or $(128, 138)$ respectively. The theoretical result agrees well with the experimental result, except for a few points as shown in Figure 2. This result is sufficient for our probability evaluation.

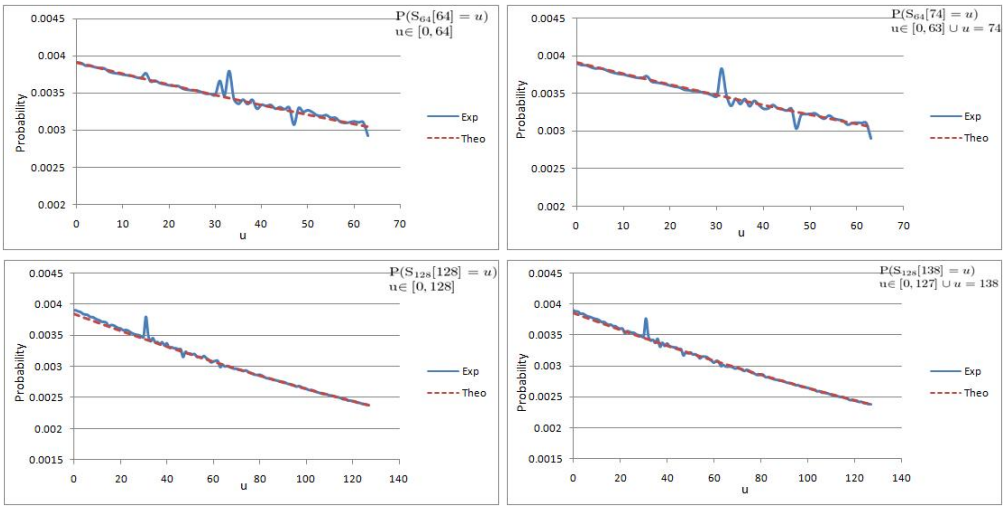

Fig. 2. Theorem 3

Now we are ready to calculate Event C by using the above Lemma and partition techniques. Refer to Appendix A for a more specific description of the integer partitioning algorithm.

Lemma 4. *The probability of Event C is*

$$P(C_{d,k,p}) = \sum_{i=0}^{\#(\mathcal{Q}(X)-\mathcal{I}(X))} \prod_{j=0}^{t-1} (P_{d+pk}[d+pk+j] = q_{ij})$$

where $\mathcal{Q}(X)$ denotes the partition function with input X the number to be partitioned, and the output is a set of t-element tuples. Each tuple represents a t distinct element partition of X. Let $PS = S_d[d] + \cdots + S_d[d+t-1]$ and $PD = (d+pk) + \cdots + (d+pk+t-1)$. $X \in \{q = \{0, ..., \lfloor \frac{PD-PS}{256} \rfloor\} | PS + 256 \times q\}$. $\mathcal{I}(X)$ denotes the subset of $\mathcal{Q}(X)$ which satisfies the following conditions: $S_{d+pk}[d+pk], ..., S_{d+pk}[d+pk+t-1] > d+pk$ and $S_{d+pk}[d+pk] \neq d+pk$,...,$S_{d+pk}[d+pk+t-1] \neq d+pk+t-1$. q_{ij} denotes the jth element in tuple i of set $\mathcal{Q}(X) - \mathcal{I}(X)$.

Proof. According to the definition of Event C, our goal is to partition the sum $S_d[d]+\cdots+S_d[d+t-1]$ and calculate the probability that the t-element partition will be in position $d+pk, ..., d+pk+t-1$. Here, one thing to notice is that when the difference between $(d+pk)+\cdots+(d+pk+t-1)$ and $S_d[d]+\cdots+S_d[d+t-1]$ is larger than $256*q$ for some $q \geq 0$, than the partition of $S_d[d]+\cdots+S_d[d+t-1]+256*q$ is also a possible partition for positions $d+pk, ..., d+pk+t-1$. Thus the input of partition function $\mathcal{Q}(X)$ can vary depending on the relation between PS and PD. The subset $\mathcal{I}(X)$ indicates partitions that will never occur in the

situation of RC4. This is easy to see because before i touches index $d+pk$, $S[d+pk], ..., S[d+pk+t-1]$ could never become a value greater than index $d+pk$ except for the value of the index itself, which is the case in which these values have never been swapped before. Thus $\mathcal{Q}(X) - \mathcal{I}(X)$ denotes all the legal partitions.
$\prod_{j=0}^{t-1}(P_{d+pk}[d+pk+i] = q_{ij})$ is the probability of each legal partition being in positions $d+pk, ..., d+pk+t-1$. The probability is calculated by using Lemma 3, and we have t elements for each partition, so we need to multiply them together. Since there are a total $\#(\mathcal{Q}(X) - \mathcal{I}(X))$ partitions, by summing them together, we get the probability of Event C.

Figure 3 represents the experimental and theoretical probability of $S_d[d] + S_d[d+1] + \cdots + S_d[d+t-1] \equiv S_{d+k}[d+k] + S_{d+k}[d+k+1] + \cdots + S_{d+k}[d+k+t-1] \pmod{256}$ for fixed k, t and various d.

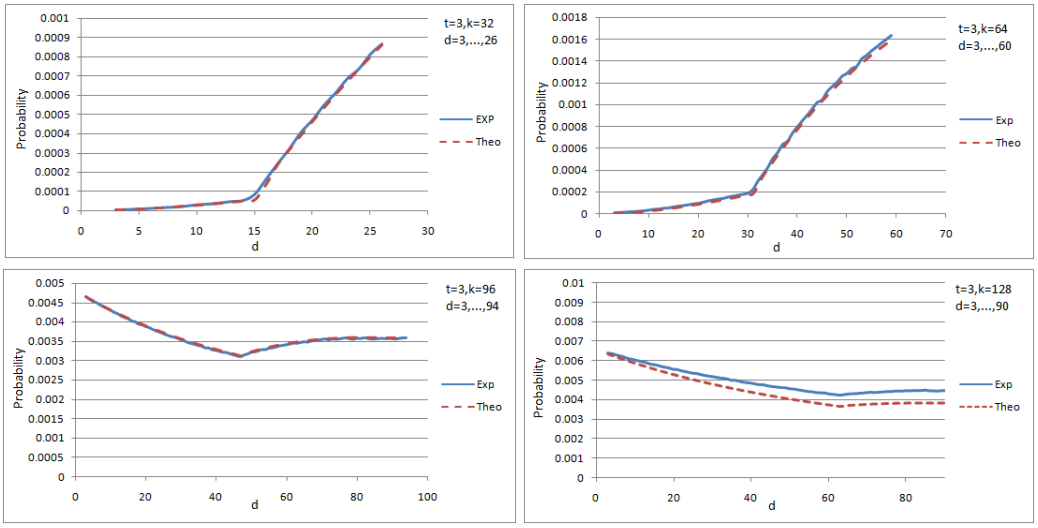

Fig. 3. Event C

Lemma 5. *The probability of Event D is*

$$P(D_t) = (\frac{254}{256})^{t-1}$$

Proof. There are $t-1$ places between index $d+pk$ and $d+pk+t$. When i touches these positions, j should not touch $d+pk$ and $d+pk+t$. This gives us the probability $(\frac{254}{256})^{t-1}$.

Theorem 1. *Assume that K_1 and K_2 form a k-byte key pair which has the following properties: $K_1[d] = K_2[d] - t$, $K_1[d+1] = K_2[d+1] + t$, $K_1[d+t+1] = K_1[d+t+1] - t$ for some fixed t, d and n. Then the probability that K_1 and K_2 are a colliding key pair is*

$$P_{col}(t,d,k) = (P(A))^{n+1}(P(D_t))^n \prod_{p=0}^{n-1} P(B_{d,k,p}) \prod_{p=1}^{n-1} P(C_{d,k,p})$$

Proof. Recall the conditions we summarized in the previous section. What we are doing here is just combining all the conditions. When $p = 0$, there is no restriction on Event C. The probability is $(P(A))^2 P(B)P(D)$. When $p \geq 1$, according to the previous analysis, we have $\prod_{p=1}^{n-1} P(A)P(B)P(C)P(D)$. By multiplying these two cases together, we get our result.

Since parameters d and t are not required to be fixed in our pattern, the total probability should include various d and t. This gives us the following Corollary.

Corollary 1. *Assume that K_1 and K_2 form a k-byte key pair which has the following properties: $K_1[d] = K_2[d] - t$, $K_1[d+1] = K_2[d+1] + t$, $K_1[d+t+1] = K_1[d+t+1] - t$. Then the probability that K_1 and K_2 are a colliding key pair is*

$$P_{col}(t,d,k) = \sum_{d=0}^{k-4} \sum_{t=2}^{k-d-2} (P(A))^{n+1}(P(D_t))^n \prod_{p=0}^{n-1} P(B_{d,k,p}) \prod_{p=1}^{n-1} P(C_{d,k,p})$$

Proof. Let's look at the bounds for d and t. First, for every fixed d, the max value t has to satisfy $d+t = k-2$, and the smallest value of t is 2 in our pattern. Thus $2 \leq t \leq k-d-2$. Then we look at d. d could start from 0. The max d is such d that $t = 2$, so we have $d + 2 = k - 2$ which gives us $0 \leq d \leq k - 4$.

Corollary 1 gives us the total collision probability of two keys with length k satisfying conditions $K_1[d] = K_2[d] - t$, $K_1[d+1] = K_2[d+1] + t$ and $K_1[d+t+1] = K_1[d+t+1] - t$. Thus the number of colliding key pairs could be calculated by multiplying 2^{8*k}. Since calculating Event C is time consuming and involves exponential time partition algorithm, to evaluate all d and t is a time-consuming job. We approximate by evaluating some fixed d and t. We choose d, t such that for a given k, they make the different positions of the key pair appear least often in the KSA. For example, for $k = 24$, we choose $d = 17, t = 2$ so that the different positions of the key pair appear 10 times not 11 times during KSA. The probability of the 10 times dominates, so we can ignore the 11 times d, t. Our results can be seen as an approximate lower bound of the total number of this new kind of colliding key pairs, and also we compare our work with the results in [2]. Future work is required for a more precise evaluation. Please refer to Appendix B for the detailed results. The values in parentheses in our result denote the chosen values of d and t.

5 An Efficient Search Technique

Searching for such colliding key pairs randomly will hardly generate any results. Finding the colliding key pairs we found involved some detection tricks. Recall the necessary conditions of the transition pattern we have to satisfy, namely, Event A, Event B, Event C and Event D. For Event B and Event C, we can control them to let them occur with much higher probability. We do this as follows: assume when i touches index $d-1$, $S[d], ..., S[d+t-1]$ have not been swapped, thus we can pre-compute the sum $S_d[d] + \cdots + S_d[d+t-1] = d + \cdots + (d+t-1)$. Then we compute $n-1$ partitions of the sum, namely, $(q_{i,1}, ..., q_{i,t}), i \in [1, n-1], d$ for Event C. Also for Event B we can calculate $p_i = q_{i,1} + t$ according to the partitions we calculated. Now we need to adjust two secret keys during KSA so that the values of p and q will be swapped to the corresponding positions $S[d+pk], ..., S[d+pk+t]$ during the first appearance of the secret key. After the secret key begins to duplicate during KSA, we only hope that those positions $d + pk, ..., d + pk + t$ will not be touched before i touches them and Event A occurs.

The above algorithm gives us a much more efficient way to search for this new class of colliding key pairs. The key point here is that we can control the behavior of the keys to satisfy our conditions in the process of the first key appearance in KSA. However, as the length of the key becomes shorter, in other words, as the number of appearances of the key increases, we'll have less control over the behavior during the KSA process. By using our algorithm, we are able to find a 55-byte colliding key pair within one hour. Refer to Appendix C for some concrete colliding key pairs we found.

6 Conclusion

We discovered a new class of colliding key pairs with properties differing from those discovered in [2]. Our new transition pattern requires relatively loose conditions on the key pairs, namely, our colliding key pairs can differ at more than one position and the value difference can also be more than one. We gave probability evaluations of our colliding key pairs and showed the scale of this new class of colliding key pairs. An efficient search algorithm was proposed which can allow us to find 55-byte colliding key pairs easily. We leave finding even shorter colliding key pairs and more precise probability evaluation as future work.

References

1. Grosul, A.L., Wallach, D.S.: A Related-Key Cryptanalysis of RC4. Technical Report TR-00-358, Department of Computer Science, Rice University (2000), http://cohesion.rice.edu/engineering/computerscience/tr/TR Download.cfm?SDID=126
2. Matsui, M.: Key Collisions of the RC4 Stream Cipher. In: Dunkelman, O., Preneel, B. (eds.) FSE 2009. LNCS, vol. 5665, pp. 1.24. Springer, Heidelberg (2009)

3. Anonymous: RC4 Source Code. CypherPunks mailing list (September 9, 1994), http://cypherpunks.venona.com/date/1994/09/msg00304.html, http://groups.google.com/group/sci.crypt/msg/10a300c9d21afca0
4. Roos, A.: A Class of Weak Keys in the RC4 Stream Cipher (1995), http://marcel.wanda.ch/Archive/WeakKeys
5. Mantin, I., Shamir, A.: A Practical Attack on Broadcast RC4. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 152.164. Springer, Heidelberg (2001)
6. Paul, S., Preneel, B.: A New Weakness in the RC4 Keystream Generator and an Approach to Improve Security of the Cipher. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 245.259. Springer, Heidelberg (2004)
7. Fluhrer, S., Mantin, I., Shamir, A.: Weaknesses in the Key Scheduling Algorithm of RC4. In: Vaudenay, S., Youssef, A.M. (eds.) SAC 2001. LNCS, vol. 2259, pp. 1.24. Springer, Heidelberg (2001)
8. Klein, A.: Attacks on the RC4 Stream Cipher. Designs, Codes and Cryptography 48(3), 269.286 (2008)
9. Tews, E., Weinmann, R.P., Pyshkin, A.: Breaking 104 Bit WEP in Less than 60 Seconds. In: Kim, S., Yung, M., Lee, H.-W. (eds.) WISA 2007. LNCS, vol. 4867, pp. 188.202. Springer, Heidelberg (2007)
10. Vaudenay, S., Vuagnoux, M.: Passive-Only Key Recovery Attacks on RC4. In: Adams, C., Miri, A., Wiener, M. (eds.) SAC 2007. LNCS, vol. 4876, pp. 344.359. Springer, Heidelberg (2007)

A Integer Partition Techniques

We refer to "The Art of Computer Programming Vol 4" regarding integer partition techniques. The partitions of an integer are the ways to write it as a sum of positive integers. The following Algorithm 3 calculates all partitions of integer n into a fixed number of parts. This algorithm was featured in C.F. Hindenburg's 18th-century dissertation [Infinitinomii Dignitatum Exponentis Indeterminati]. This algorithm generates all integer t-tuples $a_1...a_t$ such that $a_1 \geq \cdots \geq a_t \geq 1$ and $a_1 + \cdots + a_t = n$.

Algorithm 3. Partition n into t parts (Pa(n,t))

1: **[Initialize.]** Set $a_1 \leftarrow n - t + 1$ and $a_j \leftarrow 1$ for $1 < j \leq t$. Set $a_{t+1} \leftarrow -1$.
2: **[Visit.]** Visit the partition $a_1...a_t$. Then go to 4 if $a_2 \geq a_1 - 1$.
3: **[Tweak a_1 and a_2].** Set $a_1 \leftarrow a_1 - 1, a_2 \leftarrow a_2 + 1$, return to 2.
4: **[Find j.]** Set $j \leftarrow 3$ and $s \leftarrow a_1 + a_2 - 1$. Then, if $a_j \geq a_1 - 1$, set $s \leftarrow s + a_j, j \leftarrow j + 1$, repeat until $a_j < a_1 - 1$.
5: **[Increase a_j.]** Terminate if $j > m$. Otherwise set $x \leftarrow a_j + 1, a_j \leftarrow x$, $j \leftarrow j - 1$.
6: **[Tweak $a_1...a_j$.]** While $j > 1$, set $a_j \leftarrow x, s \leftarrow s - x$ and $j \leftarrow j - 1$. Finally set $a_1 \leftarrow s$ and return to 2.

For example, Pa(11,4) will generate the following partitions:

$$8111, 7211, 6311, 5411, 6221, 5321, 4421, 4331, 5222, 4322, 3332$$

Use it twice, namely, Pa(n,t) and Pa(n,t-1) to include 0 value cases in the partitions, and perform the permutation on each tuple $(a_1...a_t)$ to cover all the cases. Then this is our $\mathcal{Q}$ function.

B Number of colliding key pairs

Table 4 shows the number of newly discovered colliding key pairs, compared with [2]'s results. The values in the parentheses denote d and t. "-" means colliding key pairs do not exist theoretically.

Table 4. log_2 (number of colliding key pairs)

k	[2]	Ours(d,t)	k	[2]	Ours(d,t)	k	[2]	Ours(d,t)	k	[2]	Ours(d,t)	k	[2]	Ours(d,t)
15	-	-	25	106.9	9.2(7,2)	35	211.2	147.8(12,2)	45	306.4	267.9(32,2)	55	394.9	367.9(37,2)
16	-	-	26	120.7	13.5(3,2)	36	219.6	153.1(5,2)	46	314.9	275.7(27,2)	56	403.2	375.9(33,2)
17	2.7	-	27	130.5	45.8(14,2)	37	232.3	159.6(3,2)	47	323.3	283.4(22,2)	57	411.5	383.9(29,2)
18	18.5	-	28	139.2	49.6(5,2)	38	242.0	194.0(29,2)	48	331.6	291.1(17,2)	58	419.7	391.8(25,2)
19	34.0	-	29	153.0	83.8(25,2)	39	250.7	201.8(23,2)	49	339.9	298.3(12,2)	59	427.9	399.7(21,2)
20	48.7	-	30	162.5	91.0(17,2)	40	259.2	209.4(17,2)	50	348.1	305.4(7,2)	60	436.1	407.6(17,2)
21	58.8	-	31	171.2	97.2(9,2)	41	267.6	216.5(11,2)	51	356.3	311.7(3,2)	61	444.2	415.4(13,2)
22	73.7	-	32	179.7	108.8(3,2)	42	275.9	223.1(5,2)	52	368.7	343.7(49,2)	62	452.4	423.1(9,2)
23	83.0	-	33	193.7	133.7(26,2)	43	287.7	229.8(3,2)	53	377.8	351.8(45,2)	63	460.5	429.1(4,2)
24	97.7	4.7(17,2)	34	202.6	141.0(19.2)	44	297.6	259.7(37,2)	54	386.4	359.8(41,2)	64	468.6	444.9(3,2)

C Some Concrete Colliding Key Pairs

We demonstrate 4 colliding key pairs in this section with n from 1 to 4. "-" denotes that K_1 and K_2 have the same value.

C.1 $k = 132, d = 125, t = 5, n = 1$

$K_1[000] \sim K_1[023]$: d6 0d c0 2c ab d2 86 09 15 ec 7c d5 69 cd 3b 75 6e b2 34 b9 8f d1 fd b0
$K_2[000] \sim K_2[023]$: -
$K_1[024] \sim K_1[047]$: 65 42 ab 48 df fd 7b c9 79 0a 25 44 35 86 5c 9b 45 5a a7 e5 6b 2b 43 6c
$K_2[024] \sim K_2[047]$: -
$K_1[048] \sim K_1[071]$: 76 94 e8 41 f6 ec bb c9 e4 2e 62 b5 d8 11 17 e3 05 e7 c9 fd c1 36 3d 80
$K_2[048] \sim K_2[071]$: -
$K_1[072] \sim K_1[095]$: 53 00 38 f5 11 58 65 cd 73 43 2e 64 ea 25 d2 0e 75 26 8b a1 7a 10 a0 06
$K_2[072] \sim K_2[095]$: -
$K_1[096] \sim K_1[119]$: c6 2e 05 28 9e 60 23 9c 04 b1 ae 7e 58 f3 93 22 66 72 a4 f8 50 2f 22 1b
$K_2[096] \sim K_2[119]$: -
$K_1[120] \sim K_1[131]$: 39 09 b4 f6 03 a5 4e 08 c7 f3 b2 84
$K_2[120] \sim K_2[131]$: - - - - - - aa 49 - - - - 89

C.2 $k = 85, d = 5, t = 3, n = 2$

$K_1[00] \sim K_1[23]$: 5a 54 aa ff 51 4e e8 b6 50 bf d7 8e 35 0b 47 2e 17 a1 8c 86 db cf fb 3c
$K_2[00] \sim K_2[23]$: - - - - - - 51 e5 - - c2 - - - - - - - - - - - - - - -
$K_1[24] \sim K_1[47]$: 13 52 22 1a e0 5d 7d a3 91 d8 4c 45 21 c3 f2 f2 d6 ec c0 d9 ea c4 5f 95
$K_2[24] \sim K_2[47]$: -
$K_1[48] \sim K_1[71]$: 3e b6 6b 30 30 6d 78 9c df 79 7c 81 44 e7 65 10 af d6 95 38 9d dd 97 6f
$K_2[48] \sim K_2[71]$: -
$K_1[72] \sim K_1[84]$: 9f 5c 87 94 ce 80 4d 44 a9 6b b4 fc 3a
$K_2[72] \sim K_2[84]$: - - - - - - - - - - - - -

C.3 $k = 66, d = 58, t = 2, n = 3$

$K_1[00] \sim K_1[23]$: 83 db 06 ee 00 8d 7d cb e5 0c 3a b7 33 ec 8f 93 c5 7d 8d 95 64 c5 d9 19
$K_2[00] \sim K_2[23]$: -
$K_1[24] \sim K_1[47]$: 18 45 54 82 be e1 eb 03 4f 96 75 08 b5 6e c6 81 36 16 0d 15 77 8a a2 6b
$K_2[24] \sim K_2[47]$: -
$K_1[48] \sim K_1[65]$: 6f 49 53 96 67 6b cd 0b 88 09 40 db b0 7b 8d 7f 68 0d
$K_2[48] \sim K_2[65]$: - - - - - - - - - - 42 d9 - 7d - - - -

C.4 $k = 55, d = 36, t = 2, n = 4$

$K_1[00] \sim K_1[23]$: 89 2d b4 26 2c 12 3b 51 09 87 49 92 88 38 d9 3d e1 7d 4e 35 11 99 fc 76
$K_2[00] \sim K_2[23]$: -
$K_1[24] \sim K_1[47]$: f7 35 46 79 89 b3 00 dd 15 16 a9 14 35 05 b2 5f f6 a0 09 66 d9 08 e4 76
$K_2[24] \sim K_2[47]$: - - - - - - - - - - - - 37 03 - 61 - - - - - - - - -
$K_1[48] \sim K_1[54]$: 17 7e 2b 7a 69 5b 69
$K_2[48] \sim K_2[54]$: - - - - - - -