Glimpses are Forever in RC4 amidst the Spectre of Biases

Chandratop Chakraborty[1], Pranab Chakraborty[2], Subhamoy Maitra[3]

[1] Department of Computer Science and Engineering, PES University, 100 Feet Ring Road, BSK III Stage, Bangalore 560085, India. chandratop@protonmail.ch

[2] Learning and Development, Human Resources, Wipro Limited, Doddakannelli, Sarjapur Road, Bangalore 560035, India. kojagori@gmail.com

[3] Applied Statistics Unit, Indian Statistical Institute, 203 B T Road, Kolkata 700108, India, subho@isical.ac.in

Abstract. In this paper we exploit elementary combinatorial techniques to settle different cryptanalytic observations on RC4 that remained unproved for more than two decades. At the same time, we present new observations with theoretical proofs. We first prove the biases (non-randomness) presented by Fluhrer and McGrew (FSE 2000) two decades ago. It is surprising that though the biases have been published long back, and there are many applications of them in cryptanalysis till recent days as well, the proofs have never been presented. In this paper, we complete that task and also show that any such bias immediately provides a glimpse of hidden variables in RC4. Further, we take up the biases of two non-consecutive key-stream bytes skipping one byte in between. We show the incompleteness of such a result presented by SenGupta et al (JoC, 2013) and provide new observations and proofs in this direction relating the key-stream bytes and glimpses. Similarly, we streamline certain missed observation in the famous Glimpse theorem presented by Jenkins in 1996. Our results point out how biases of RC4 key-stream and the Glimpses of the RC4 hidden variables are related. It is evident from our results that the biases and glimpses are everywhere in RC4 and it needs further investigation as we provide very high magnitude of glimpses that were not known earlier. The new glimpses and biases that we identify in this paper may be exploited in improving practical attacks against the protocols that use RC4.

Keywords: Biases, Elementary Combinatorics, Cryptanalysis, Glimpses, Non-randomness, RC4, Stream Cipher.

1 Introduction

Until recently, RC4 was one of the most widely deployed stream ciphers, both in the public domain as well as in the commercial software products. However, a number of cryptanalytic attacks have proved that it is not secure enough and hence the usage of the cipher has been deprecated across the board. While there is a possibility that it may still be present in some legacy servers and software products, the Internet Engineering Task Force (IETF) has prohibited its usage in TLS (Transport Layer Security) through RFC 7465 [12]. At the same time, replacing the most popular stream cipher from the existing computational equipments will require more time. That is, RC4 is still in use in many applications and thus the analysis of the cipher is indeed relevant in the present context. From the motivation of fundamental cryptology research, there are many ciphers which are also known to be weak, but the cryptanalytic results are considered important in the overall development of the subject.

For nearly three decades cryptologists have studied this cipher and uncovered its various weaknesses in the form of short-term and long-term biases, as documented in several papers [1, 2, 3, 4, 5, 8, 9, 10, 11, 13, 14, 15] and the references therein. The short-term key dependent biases can generally be avoided in the truncated mode of RC4, by throwing out the initial 768 or more key-stream bytes. However, some recent and most prominent cryptanalysis, such as [1, 2, 15], have exploited the long-term biases in the key-stream that are present even after such truncation. Since these attacks have utilized robust statistical frameworks and have carried out "fine-grained analysis" of certain biases, it is generally considered that the nature of anomalies present in the long-term key-stream of RC4 have been well-understood by now. However, that is not correct as evident from several missing proofs. In the process of revisiting, we also uncover surprising elements of long-term biases in RC4 that so far remained undetected and unexplored. At this point, it is essential to first describe the RC4 as we would be referring to the algorithm, its internal elements and the output key-stream while summarizing the contributions of this paper.

The RC4 stream cipher uses two simple operations, swap for state evolution and double indirection for producing the key-stream output byte. In RC4, we have an $N = 256$ length array of $n = 8$ bit integers 0 to $N - 1$, that works as a permutation. That is $N = 2^n$. There is also an l length array of bytes K (the secret key), where l may vary from 5 to 32, depending on the key length. There are also two bytes i, j, where i is the deterministic index that increases by 1 in each step and j is updated in a manner so that it behaves pseudo-randomly. The Key Scheduling Algorithm (KSA) of RC4 is as follows:

- $j = 0$; for $i = 0$ to $N - 1$: $S[i] = i$;
- for $i = 0$ to $N - 1$:

$$j = j + S[i] + K[i \bmod l];\ \text{swap}(S[i], S[j]);$$

Next, the pseudo-random bytes z are generated during the Pseudo Random Generator Algorithm (PRGA) as follows:

- $i = j = 0$;
- while key-stream is required:

$$i = i + 1;\ j = j + S[i];\ \text{swap}(S[i], S[j]);\ z = S[S[i] + S[j]];$$

All the additions here are modulo N. Note that the subscript r is used to identify the round r and $S_r[x]$ means the value at x-th location after the swap operation in the said round. The index j_r is the value of j after the updation $j + S[i]$ in the round r. That way, z_r is the key-stream output byte in round r.

1.1 Organization and Contribution

As in most of the previous efforts (see [13], and the references therein), we explore non-randomness in the RC4 stream cipher in this paper. Here we first concentrate on the work of Fluhrer and McGrew [3], two decades ago. These are biases relating to two consecutive bytes z_r, z_{r+1}. The biases are still relevant as evident from the recent cryptanalytic work [15]. In [3, Section 4], there was some effort towards providing theoretical justification for the twelve identified biases in [3, Table 3]. However, that effort was incomplete in terms of proofs and identifying the consequences. In Section 2, we explain these proofs in detail and show how such biases of two consecutive bytes reveal state information.

Then we concentrate on the biases between two non-consecutive bytes z_r, z_{r+2}, for $r \equiv 0 \bmod N$, leaving one key-stream byte between them. In [3], such a pair of key-stream bytes is referred as "lag-one digraph". In Section 3, we study such biases. The byte-wise correlation for non-consecutive key-stream pair $z_r = 0$ and $z_{r+2} = 0$ has been reported by Sen Gupta et. al. (in [13, Section 4.3, Theorem 16]). Subsequently, Vanhoef-Piessens [15] experimentally identified the bias for key-stream pair $z_r = 128$ and $z_{r+2} = 0$, for $r \equiv 0 \bmod N$. However, Vanhoef et. al. [15] did not provide any theoretical explanation to the source of the bias. Later in [6], a proof of this has been presented. In this paper, we report two new non-consecutive double-byte biases in lag-one digraph. We present a generic approach to theoretically prove all these biases. This approach is different and more accurate than the proof of [13, Theorem 16]. In the technique followed by Sen Gupta et. al. (in [13]), the bias value is proved by assuming independence of two events. However, those are closely dependent upon each other. Moreover, the proof of [13] was partial and we present a sharper value for the bias corresponding to the lag-one digraph that was reported in [13]. As we have pointed out several times, these biases also lead to the glimpses of the pseudo-random index j_r and of certain bytes of the state. We show that in some cases, we can achieve a bias of the order of $\frac{4}{N}$ that could never be discovered earlier. In fact, in [13, Section 4.2.1], the idea of "Guessing State Information Using the Bias in z_r" has been presented and we study that concept with far more insight in this work. In the conclusion of [13], the following question has been posed: "How does one generalize the concept of digraph biases to related bytes with arbitrary gaps in between? Are there more long-term biases of this kind in the RC4 key-stream?" Our work advances the research in this direction.

It is expected that any information about the state should not be revealed from the key-stream of the stream cipher. It is well known that this is not the case for RC4. Robert J. Jenkins Jr., around quarter century ago, first mentioned about correlations that exist among certain permutation array bytes, index values and the key-stream byte in his web-site ([5]). This can be clearly mentioned as follows.

Glimpse: We have the complete knowledge about the public index i and the key-stream byte z at any round of RC4. From these two bytes, is it possible to obtain additional information regarding the hidden variables $j, S[i]$ or $S[j]$ other than the random association?

Following [5], the well known Glimpse of RC4 has been presented (for more details see Theorem 1 in Section 4). This glimpse is dependent upon the way the output key-stream byte, $z_r = S_r[S_r[i_r] + S_r[j_r]]$, is formed based on the two indices and are independent of the swap step. In this regard, let us reiterate that the source of this bias and explain that this bias is not dependent on the way RC4 permutation array evolves; instead it is only dependent upon the way key-stream byte is derived in the last step of the algorithm (which is $z = S[S[i] + S[j]]$).

In Section 4, we explore several results related to Glimpse that are generated due to the output function $z = S[S[i] + S[j]]$. In Section 4.1, we embark upon a fine-grained analysis of the Glimpse theorem (as in Theorem 1) and show that although the results are approximately valid at an aggregate level, there can be deviations when we constrain the values of indices (i and j). For example, if $i = j = 3$, the probability that $S_r[3] = (3 - z_r)$ is around $\frac{1}{N}$ instead of the expected value of $\frac{2}{N}$ (ignoring order $\frac{1}{N^2}$ terms) as given in Theorem 1. Similarly, if $i = j = 4$, the probability that $S_r[4] = (4 - z_r)$ is almost $\frac{3}{N}$ instead of $\frac{2}{N}$. In fact, we prove that under specific conditions (e.g., when $z_r = 2 * i_r$), one can get a glimpse of the pseudo-random index j and this bias is also independent of the way the RC4 permutation array evolves. Similar kinds of new glimpses are presented in Section 4.2 too. In Section 4.3, we note that the swap operation is also involved in addition to the output function format of RC4 in providing glimpses. We can show that it is possible to obtain glimpses of the order of $\frac{5}{N}$ exist, which is the best known so far. Our results substantially improve the work of [7] that could only provide an example of order $\frac{3}{N}$.

In summary, we analyse several non-randomness results of RC4 that remained unclear for several decades. In the process we provide new biases and glimpses that were never known earlier. These new non-randomness results can be exploited to improve the practical attacks as explored in [1, 4, 11, 14, 15]. Section 5 concludes this paper summarizing all these details and providing future directions.

2 Glimpses corresponding to Fluhrer-McGrew biases

Fluhrer-McGrew digraph biases are widely referred in RC4 literature and these have been utilized in a number of cryptanalytic attacks. Surprisingly, the root causes behind these biases do not appear to have been studied in a methodical manner. Vanhoef and Piessens [15], while combining Fluhrer-McGrew biases [3] with Mantin's $AB\mathcal{T}AB$ biases [9], to cryptanalyze RC4 in WPA-TKIP and TLS, have observed the following.

> "We will combine multiple types of biases by multiplying their individual likelihood estimates …. An open problem is determining which biases can be combined under a single likelihood estimation …"

That is, there is a scope to uncover the underlying mechanisms that cause Fluhrer-McGrew biases [3] and that can eventually lead to a better understanding of how one can combine different types of biases in a single framework of likelihood estimation. In another work [2, Section 3.3], Bricout et. al. have commented the following on "Double-byte bias correction".

> "Hence, based on the Fluhrer-McGrew biases alone, and assuming that occurrences of these biases are pair-wise independent, we would expect the pattern $00\mathcal{S}00$ (for any size of S) to occur with probability $2^{-32}(1+2^{-8})^2 \approx 2^{-32}(1+2^{-7})$."

The above justification is based on the assumption that the occurrences of the Fluhrer-McGrew biases [3] are pair-wise independent. This is not exactly correct. In fact, by experimenting on the pattern $01\mathcal{S}01$ for different lengths of $\mathcal{S}$, one can see that the occurrence probability reduces with the reduction of length of $\mathcal{S}$ which establishes the fact that Fluhrer-McGrew biases are not pair-wise independent and hence when $\mathcal{S}$ is small (e.g., empty), the source mechanism of one double-byte (01) would adversely affect the source mechanism of the other one. As per the fine-grained analysis of Bricout et. al.[2], it has been proved that for a pattern like $01\mathcal{S}01$, Mantin biases [9] would not be present and hence the variation of probability of occurrence of the pattern due to the length of $\mathcal{S}$ can't be attributed to Mantin biases.

Fluhrer and McGrew [3, Section 4] provided a direction of analysis to understand the mechanisms behind the digraph biases. They have specifically considered the positive biases for $(0,1)$ and $(0,0)$ digraphs and the negative bias for $(N-1, N-1)$ digraph. However, the reasoning given in their paper do not adequately substantiate the increase of probability for $(0,0)$ digraph when $i_r = 1$ and the decrease of probability for $(N-1, N-1)$. In fact, to the best of our knowledge, after two decades of the results presented in [3], there is no disciplined effort towards proving these biases. In this section we illustrate the source configurations corresponding to all Fluhrer-McGrew digraph scenarios, present the approach to calculate the probabilities and explain how these source configurations lead to a series of glimpse results. We make the following two key observations based on the results of this section.

1. Biases and glimpses are the two sides of the same coin. Both are either based upon non-randomness in the way key-streams are formed or due to non-randomness in the underlying data structure (e.g., in the permutation array). So biases are like the *spectre* behind the glimpses and presence of biases would surely reveal glimpses.

2. Glimpses can be detected for individual events as well as for joint events. Since the joint events can be formed in a variety of ways by combining the source configurations

corresponding to different types biases, it appears that within the construct of RC4, *glimpses are forever!*

In Table 1, we explicitly show the source configurations of Fluhrer-McGrew digraph biases. These source configurations are the root causes behind the Fluhrer-McGrew biases [3]. We

Table 1: The source configurations for Fluhrer-McGrew Biases [3]

Scenario	$(z_r,\ z_{r+1})$	i_r	**Source configuration(s)**
1.	(0, 0)	$i_r = 1$	$S_r[1] = (2 - j_r),\ S_r[2] = 0,\ (S_r[j_r] = j_r$ or $j_r = 1)$
2.	(0, 0)	$i_r \notin \{1, 255\}$	$S_r[i_r] = (i_r + 1 - j_r),\ S_r[i_r + 1] = 0,\ S_r[j_r] = j_r$
3.	(0, 1)	$i_r \notin \{0, 1\}$	$S_r[i_r] = 1,\ S_r[i_r + 1] = 0,\ S_r[j_r] = i_r$
4.	$(i_r + 1,\ 255)$	$i_r \neq 254$	$S_r[i_r] = i_r + 1,\ j_r = (i_r + 1),\ S_r[j_r] = 255$
5.	$(255,\ i_r + 1)$	$i_r \notin \{1, 254\}$	$S_r[i_r] = 255,\ S_r[j_r] = (i_r + 1),\ S_r[i_r + 1] = 0$
6.	$(255,\ i_r + 2)$	$i_r \notin \{0, 253, 254, 255\}$	$S_r[i_r] = (i_r + 2),\ S_r[j_r] = 255,\ j_r = (i_r + 1)$
7.	(255, 0)	$i_r = 254$	$S_r[254] = 0,\ S_r[255] = 255,\ j_r = 255$
8.	(255, 1)	$i_r = 255$	$S_r[255] = 1,\ S_r[0] = 255,\ j_r = 0$
9.	(255, 2)	$i_r = 0$	$S_r[0] = 2,\ S_r[1] = 255,\ j_r = 1$
10.	(129, 129)	$i_r = 2$	$S_r[2] = 129,\ S_r[3] = 1,\ j_r = 2$
11.	(255, 255)	$i_r \neq 254$	$S_r[i_r + 1] = 0$ is an impossible event
12.	$(0,\ i_r + 1)$	$i_r \notin \{0, 255\}$	$S_{r+1}[i_{r+1}] = i_{r+1}$ is an impossible event

now demonstrate how the source configurations lead to the biases for each of the scenarios.

Scenario 1: Let us first consider the following set of source configurations (corresponding to Scenario 1):

(1) $S_r[1] = (2 - j_r)$, $S_r[2] = 0$ and $S_r[j_r] = j_r$ where $i_r = 1$

In round r: $z_r = S_r[2 - j_r + j_r] = S_r[2] = 0$
In round $r + 1$: $j_{r+1} = j_r$ as $S_r[2] = 0$ and after the swap step $S_r[2] = j_r$ and $S_r[j_r] = 0$. Hence, $z_{r+1} = S_{r+1}[j_r + 0] = S_{r+1}[j_r] = 0$.

Let us next consider the other set of source configurations (corresponding to Scenario 1):

(2) $S_r[1] = (2 - j_r)$, $S_r[2] = 0$ and $j_r = 1$ where $i_r = 1$

In round r: $z_r = S_r[2 - 1 + 1] = S_r[2] = 0$
In round $r + 1$: $j_{r+1} = j_r$ as $S_r[2] = 0$ and after the swap step $S_r[2] = 1$ and $S_r[1] = 0$. Hence, $z_{r+1} = S_{r+1}[1 + 0] = S_{r+1}[1] = 0$.

Therefore both the source configurations corresponding to Scenario 1, lead to the desired digraph (0, 0). Since in each set, we assume three separate conditions, by considering independence of these configurations, we get the probability $Pr((z_r, z_{r+1}) = (0, 0) | i_r = 1) = \frac{2}{N^3} + (1 - \frac{2}{N^3}) \cdot \frac{1}{N^2} \approx \frac{1}{N^2} + \frac{2}{N^3}$. It must be pointed out here that in [3, Section 4], Fluhrer and McGrew talked about one of these two source configurations (namely, configuration (1)) and hence the justification was incomplete for the Scenario 1 where $i_r = 1$.

One should note that these two sets of configurations can be independently used to generate the desired double byte in the output sequence. For example, fix $j_r = 1, S_r[1] = 1, S_r[2] = 0$; these would lead to the desired double bytes. On the other hand, assume j_r is in any random position when $i_r = 1$ (of course we have to exclude the two cases of $j_r = 1$ or $j_r = 2$, i.e., we are talking of 254 out of 256 choices). Fix $S_r[j_r] = j_r, S_r[1] = (2 - j_r), S_r[2] = 0$; with these three settings we would get the desired double byte condition. Hence, these are two independent sets of configurations.

These source configurations immediately lead to the following glimpse result as well.

Corollary 1. $Pr((S_r[1] = (2 - j_r)) \wedge (S_r[2] = 0) \wedge (S_r[j_r] = j_r) | (z_r, z_{r+1}) = (0, 0), i_r = 1) \approx \frac{1}{N}$.

Proof. By referring to the proof of (0,0) bias (as given above), we find $Pr((S_r[1] = (2 - j_r)) \wedge (S_r[2] = 0) \wedge (S_r[j_r] = j_r) | (z_r, z_{r+1}) = (0, 0), i_r = 1) \approx \frac{1}{N^3} / [\frac{1}{N^2} + \frac{2}{N^3}] \approx \frac{1}{N}$. $\square$

The other source configuration (for Scenario 1) would also have the same glimpse value. For each of the scenarios, the source configurations given in the table, reveal glimpses of permutation array bytes as well as the pseudo-random index j_r. We have observed that many of these glimpse results (taken either as a joint probability or separately on independent array byte values or the j_r value) are much sharper than their corresponding fair chance probabilities. Towards the end of this section, we have presented a table to capture the glimpses for the sub-events under each source configuration.

Scenario 2: The source configuration is as follows.

- $S_r[i_r] = (i_r + 1 - j_r)$,$S_r[i_r + 1] = 0$ and $S_r[j_r] = j_r$ where $i_r \notin \{1, 255\}$

In round r: $z_r = S_r[i_r + 1 - j_r + j_r] = S_r[i_r + 1] = 0$
In round $r + 1$: $j_{r+1} = j_r$ as $S_r[i_r + 1] = 0$ and after the swap step $S_r[i_r + 1] = j_r$ and $S_r[j_r] = 0$. Hence, $z_{r+1} = S_{r+1}[j_r + 0] = S_{r+1}[j_r] = 0$.

Since we assume three separate conditions, by considering independence of these configurations, we get the probability $Pr((z_r, z_{r+1}) = (0, 0)|i_r \notin \{1, 255\}) = \frac{1}{N^3} + (1 - \frac{2}{N^3}) \cdot \frac{1}{N^2} \approx \frac{1}{N^2} + \frac{1}{N^3}$.

Here, we have ignored the two cases of $j_r = i_r$ and $j_r = (i_r + 1)$, where the three values cannot be independently assigned. If we account for the two values of j that are not permissible, the probability would be slightly lower due to the multiplicative factor of $\frac{N-2}{N}$.

Scenario 3 to Scenario 10 are all similar to Scenario 2 and the approach that we followed to prove Scenario 2, can be used to derive the source configurations for Scenario 3–10 as well, including the way the we have calculated the bias. In each of those Scenarios, we can come up with a glimpse result (e.g., see Corollary 1) for the joint event corresponding to the source configuration set as well as separately for each of the sub-parts of a source configuration (presented in a table towards the end of this section).

Scenario 11: In this scenario a negative bias is observed. Hence the interpretation of the term “source configuration" is different here and it signifies an “impossible event". The source configuration is as follows.

- $S_r[i_r + 1] = 0$ where $i_r \neq 254$

To demonstrate why this is an “impossible event", let us assume $S_r[i_r] = p$ and $S_r[j_r] = q$, where p and q are two arbitrary byte values. Hence, $z_r = S_r[p + q] = (N - 1)$. In case j_r coincides with i_r, $p = q$. Based on the given condition $S_r[i_r + 1] = 0$, it's evident that $p \neq 0$. We now investigate what happens in round $r + 1$.

- **Before swap:** Since $S_r[i_r + 1] = 0$, $j_{r+1} = j_r$. So $S_{r+1}[i_{r+1}] = 0$ and $S_{r+1}[j_{r+1}] = q$.
- **After swap:** $S_{r+1}[i_{r+1}] = q$ and $S_{r+1}[j_{r+1}] = 0$.

Therefore, $z_{r+1} = S_{r+1}[q + 0] = S_{r+1}[q]$. As $p \neq 0$, $(p + q) \neq q$. So $(p + q)$ and q must point to two different array byte positions of S. The expected condition of $z_r = z_{r+1} = (N - 1)$ can be achieved only if $z_r (= S_r[p + q])$ gets swapped in round $r + 1$ and moves to a new position pointed to by q.

We now identify the three events that lead to the desired condition of $z_r = z_{r+1}$. For no other configuration the desired condition would hold.

In Event 1 and Event 3 $z_r = z_{r+1} = 0$ instead of the desired value of 255. In Event 2, $z_r = z_{r+1} = q$, however it's not possible to have the value of q as 255 since the event condition demands $i_{r+1} = q$ and the given condition for Scenario 11 excludes that possibility.

Event	i_r	**Event conditions**	$z_r = S_r[p+q]$	$z_{r+1} = S_{r+1}[q]$
1	All	$i_{r+1} = (p+q)$ and $j_{r+1} = q$	0	0
2	All	$i_{r+1} = q$ and $j_{r+1} = (p+q)$	q	q
3	1	$j_r = i_r$ and $p = q = 1$	0	0

Hence, with $S_r[i_r + 1] = 0$, we can't satisfy $(z_r, z_{r+1}) = (255, 255)$ for $i_r \neq 254$. Therefore, we get the probability $Pr((z_r, z_{r+1}) = (255, 255)|i_r \neq 254) \approx \frac{1}{N} \cdot (0) + (1 - \frac{1}{N}) \cdot \frac{1}{N^2} = \frac{1}{N^2} - \frac{1}{N^3}$ which matches with experimentally observed bias value for this digraph.

We would like to mention here that in [3, Section 4], Fluhrer and McGrew talked about the absence of an event where $S_r[i_r + 1] = (N - 1)$ and $j_r = i_r + 1$ to be the root cause behind the negative bias of $(N - 1, N - 1)$. However, since that event includes two sub-events (with a probability of occurrence of $\frac{1}{N^2}$ considering their independence) - that would have led to a probability value for Scenario 11 as $\frac{1}{N^2} \cdot (0) + (1 - \frac{1}{N^2}) \cdot \frac{1}{N^2} = \frac{1}{N^2} - \frac{1}{N^4}$ which is higher than the experimentally observed value of $\frac{1}{N^2} - \frac{1}{N^3}$. Hence, the justification given was incomplete which we have now solved.

The source configuration for this Scenario immediately lead to the following glimpse result as well.

Corollary 2. *$Pr(S_r[i_r + 1] = 0|(z_r, z_{r+1}) = (255, 255), i_r \neq 254) = 0$*

Proof. This proof directly follows from the discussion above where we have shown that under the condition of $S_r[i_r + 1] = 0$, we can't have the digraph $(z_r, z_{r+1}) = (255, 255)$ for $i_r \neq 254$. □

Scenario 12: The negative bias and glimpse result corresponding to Scenario 12 can be shown in the same fashion as we have demonstrated it for Scenario 11. The fundamental observation in this Scenario is that if $S_{r+1}[i_{r+1}]$ happens to be i_{r+1} then to get $z_{r+1} = i_r + 1$, one must have $S_{r+1}[j_{r+1}] = 0$ which can happen only if $S_r[i_r + 1] = 0$ implying $j_r = j_{r+1}$. But in that case z_r can never be 0. If one would like to interpret this in terms of source configuration in round r, we would require to have $S_r[j_r + S_r[i_r + 1]] = i_{r+1}$ in order to prevent a digraph of $(0, i_r + 1)$ in the key-stream corresponding to the positions (i_r, i_{r+1}). Since here we assume only one event, the reduction in probability for the configuration would be equal to $\frac{1}{N^3}$ as desired.

We now present the probabilities for the sub-events under each of the source configurations corresponding to the Fluhrer-McGrew digraph scenarios. Let us take one example to explain what we mean by the sub-events. In Scenario 1 corresponding to digraph $(0, 0)$, source configuration (1) has three sub-events: $S_r[1] = (2 - j_r)$,$S_r[2] = 0$ and $S_r[j_r] = j_r$. Let us call these events as e_1, e_2 and e_3 respectively. The following table shows the probabilities for these three events individually and by taking two at a time under the given condition that $(0, 0)$ has occurred in the key-stream at the position pair of $(i_r = 1, i_{r+1} = 2)$. These probability values are approximate to the extent that any term of the order of $\mathcal{O}(\frac{1}{N^2})$ or smaller have been ignored for the sake of simplicity. One should also note that strictly speaking, $\Pr(e_1)$ should be written as $\Pr(e_1|(z_r, z_{r+1}) = (0, 0))$. However, we omit the condition in the following tables for brevity.

Scenario	**Configuration**	$Pr(e_1)$	$Pr(e_2)$	$Pr(e_3)$	$Pr(e_1 \wedge e_2)$	$Pr(e_2 \wedge e_3)$	$Pr(e_3 \wedge e_1)$
1.	(1)	$2.5/N$	$2/N$	$2.5/N$	$2/N$	$2/N$	$2/N$

Similarly, in Scenario 1 corresponding to digraph $(0, 0)$, source configuration (2) has three sub-events: $S_r[1] = (2 - j_r)$,$S_r[2] = 0$ and $j_r = 1$. The following table shows the probabilities for these three events individually and by taking two at a time under the given condition that $(0, 0)$ has occurred in the key-stream at the position pair of $(i_r = 1, i_{r+1} = 2)$.

Scenario	**Configuration**	$Pr(e_1)$	$Pr(e_2)$	$Pr(e_3)$	$Pr(e_1 \wedge e_2)$	$Pr(e_2 \wedge e_3)$	$Pr(e_3 \wedge e_1)$
1.	(2)	$2.5/N$	$2/N$	$2/N$	$2/N$	$1/N$	$1/N$

The next table shows the probabilities for the sub-events corresponding to Scenario 2–10. For Scenario 11 and Scenario 12, the source condition has only one impossible event and therefore we have not mentioned probabilities for sub-events.

Scenario	$Pr(e_1)$	$Pr(e_2)$	$Pr(e_3)$	$Pr(e_1 \wedge e_2)$	$Pr(e_2 \wedge e_3)$	$Pr(e_3 \wedge e_1)$
2.	$2/N$	$1/N$	$2/N$	$1/N$	$1/N$	$1/N$
3.	$2/N$	$2/N$	$3/N$	$1/N$	$1/N$	$1/N$
4.	$2/N$	$2/N$	$3/N$	$1/N$	$1/N$	$2/N$
5.	$2/N$	$4/N$	$3/N$	$2/N$	$2/N$	$1/N$
6.	$2/N$	$2/N$	$2/N$	$1/N$	$1/N$	$1/N$
7.	$2/N$	$3/N$	$2/N$	$1/N$	$1/N$	$1/N$
8.	$2/N$	$3/N$	$2/N$	$1/N$	$1/N$	$1/N$
9.	$2/N$	$3/N$	$2/N$	$1/N$	$1/N$	$1/N$
10.	$2/N$	$2/N$	$3/N$	$1/N$	$1/N$	$2/N$

To illustrate the mechanism behind the probabilities of the sub-events, let us take the example of Scenario 2. We start by assuming that $(z_r, z_{r+1}) = (0, 0)$ is given for $i_r \notin \{1, (N-1)\}$. Now, for some arbitrarily chosen positions of indices (i_r, j_r), once we fix the source configuration of event e_1, two different possibilities may exist:

(1) we fix events e_2 and e_3 as well - leading to $(N-3)$ degree of freedom for rest of the permutation array bytes that result in $(N-3)!$ possibilities (let us call this value as $\mathcal{X}$).

(2) we allow arbitrary value for $S_r[j_r]$ which fixes the position of 0 in the permutation array. Also we allow arbitrary value for $S_r[i_r+1]$ and thus fix the value of $S_r[j_{r+1}]$ to ensure that $z_{r+1} = 0$. This would also have $(N-3)!$ possibilities.

Now the total number of permutations that satisfy the given condition of $(z_r, z_{r+1}) = (0, 0)$ can be shown to be approximately equal to $(N-2)! = (N-2) \cdot (N-3)!$. By taking the ratio of total number of events that satisfy the desired condition of e_1 event to the total number of possible events, we find $Pr(e_1) = 2\mathcal{X}/[(N-2)\mathcal{X}] \approx \frac{2}{N}$. In an identical way, we can also prove that $Pr(e_3) = \frac{2}{N}$. However, once we assume event e_2 to be true, e_3 must be true to ensure $z_{r+1} = 0$ and this leads to the validity of e_1 as well so that we get $z_r = 0$. Thus $Pr(e_2) = \frac{1}{N}$. The combined probabilities of $Pr(e_1 \wedge e_2)$ and $Pr(e_2 \wedge e_3)$ as a result of derivation above, become $\frac{1}{N}$. $Pr(e_1 \wedge e_3) = \frac{1}{N}$ because once we fix the events e_1 and e_3, e_2 gets fixed automatically.

It is important to note that the events e_1, e_2 and e_3 could have been defined with respect to round $r+1$ instead of round r and there could be different probability values corresponding to such events as observed in round $r+1$. This shows that for any key-stream bias one can always examine a variety of glimpses in the form of individual events as well as joint events and hence it is virtually impossible to list down all possible glimpse results associated with the long-term evolution of RC4 PRGA.

3 Lag-one digraph biases and glimpses

The well-known Fluhrer-McGrew biases (also referred as digraph probabilities in [3]) represent the most dominant ones that have been identified for consecutive key-stream pairs at different index (i) values (modulo N) in the long-term evolution of RC4. To the best of our knowledge, no such systematic study has so far been reported for non-consecutive alternate key-stream pairs. The first byte-wise correlation for non-consecutive key-stream pair $z_r = 0$ and $z_{r+2} = 0$ (corresponding to the indices $i_r = 0$ and $i_{r+2} = 2$ respectively) was reported by Sen Gupta et. al. (in [13, Theorem 16]). Subsequently, Vanhoef-Piessens (in [15]) experimentally identified the bias for key-stream pair $z_r = 128$ and $z_{r+2} = 0$ (corresponding to the indices $i_r = 0$ and $i_{r+2} = 2$ respectively). However, Vanhoef et. al. [15] did not provide any theoretical explanation to the source of the bias.

In this paper, we report two new non-consecutive double-byte biases (one as positive and the other one as negative). By following the terminology of Fluhrer-McGrew ([3]), we would henceforth call these pairs - “lag-one digraphs". We also present a generic approach to theoretically prove all these biases. This approach is different and more accurate as compared to the method followed by Sen Gupta et. al. (in [13, Theorem 16]) where the bias value is proved by assuming independence of two events that are actually closely dependent upon each other. Also, here we prove a sharper value for the bias corresponding to the lag-one digraph that was first reported in [13, Theorem 16]. In addition, we show how these biases lead to the glimpses of the pseudo-random index j_r and of certain key-stream bytes.

The following table captures all the four known lag-one digraphs and their corresponding probabilities. Apart from these four pairs, we have not come across any other dominant lag-one digraph bias during experimentation. In each of the following results, we will use the value N or 2^n interchangeably, i.e., $N = 2^n$.

Lag-one digraph	**Value of i_r**	**Probability**
$(z_r,\ z_{r+2})$=(0, 0)	$i_r = 0$	$2^{-2n}(1 + 2^{-n+1})$
$(z_r,\ z_{r+2})$=(2^{n-1}, 0)	$i_r = 0$	$2^{-2n}(1 + 2^{-n})$
$(z_r,\ z_{r+2})$=($2^n - 2$, 0)	$i_r = (2^n - 2)$	$2^{-2n}(1 + 2^{-n})$
$(z_r,\ z_{r+2})$=(0, 0)	$i_r = (2^n - 2)$	$2^{-2n}(1 - 2^{-n})$

The approach taken for proving these biases is as follows. We present the scenario(s) that ensure presence or absence of one of the desired lag-one digraphs in the RC4 key-stream in most of the circumstances. Each of the scenarios correspond to a partially specified state of RC4 by fixing the values of pseudo-random index j_r together with certain permutation array bytes. The probability of occurrence of a scenario is dependent upon the number of specific values assigned in that process. In all other scenarios we assume that the target digraph appears in RC4 key-stream with a fair chance probability.

The scenarios corresponding the results (Lemma 1 – 4) are as follows.

Scenario	**Result**	**Value of i_r**	**Source condition(s)**
1	Lemma 1	$i_r = 0$	$j_r = 0$, $S_r[0] = 1$ and $S_r[2] = 0$
2	Lemma 1	$i_r = 0$	$j_r = 0$, $S_r[0] = (2^{n-1} + 1)$ and $S_r[2] = 0$
3	Lemma 2	$i_r = 0$	$j_r = 0$, $S_r[0] = 2^{n-1}$ and $S_r[2] = 0$
4	Lemma 3	$i_r = (2^n - 2)$	$j_r = 0$, $S_r[2^n - 2] = 2^n - 2$ and $S_r[0] = 0$
5	Lemma 4	$i_r = (2^n - 2)$	$S_r[0] = 0$

Lemma 1. *During RC4 PRGA, if z_r and z_{r+2} are the key-stream bytes for rounds r and $r + 2$ respectively, then $Pr(z_r = 0 \wedge z_{r+2} = 0 | i_r = 0) \approx \frac{1}{N^2} + \frac{2}{N^3}$.*

Proof. We prove this result by referring to the two scenarios (Scenario 1 and Scenario 2) given in the table above.

[**Scenario 1:**] The conditions corresponding to this scenario are $S_r[0] = 1$, $S_r[2] = 0$ and $j_r = 0$. Since $i_r = j_r = 0$ and $S_r[0] = 1$, $z_r = S_r[1+1] = S_r[2] = 0$. Similarly, using the result of Mantin-Shamir (Theorem 1 in [8]) we know that unless $S_r[1] = 2$, z_{r+2} equals 0 (since $i_r = j_r = 0$ and $S_r[2] = 0$). Therefore, the probability associated with this scenario corresponding to the desired lag-one digraph is $\frac{1}{N^3} \cdot (1 - \frac{2}{N-2}) \approx \frac{1}{N^3}$.

[**Scenario 2:**] The conditions corresponding to this scenario are $S_r[0] = (\frac{N}{2}+1)$, $S_r[2] = 0$ and $j_r = 0$. Since $i_r = j_r = 0$ and $S_r[0] = (\frac{N}{2}+1)$, $z_r = S_r[\frac{N}{2}+1+\frac{N}{2}+1] = S_r[2] = 0$. Again, using the result of Mantin-Shamir (Theorem 1 in [8]) we know that unless $S_r[1] = 2$, z_{r+2} equals 0. Therefore, the probability associated with this scenario corresponding to the desired lag-one digraph is $\frac{1}{N^3} \cdot (1 - \frac{2}{N-2}) \approx \frac{1}{N^3}$.

In rest of the configurations, z_r and z_{r+2} can have the values of $(0,0)$ with equal chance as of any other value pair. Hence, the probability for each of these configurations would be $\frac{1}{N^2}$. Therefore, the probability associated with the complimentary cases is $(1 - \frac{2}{N^3}) \cdot \frac{1}{N^2}$.

By combining the probabilities for Scenario 1, Scenario 2 and the rest of the cases, we get the desired result $Pr(z_r = 0 \wedge z_{r+2} = 0 | i_r = 0) \approx (\frac{1}{N^2} + \frac{2}{N^3} - \frac{2}{N^5}) \approx (\frac{1}{N^2} + \frac{2}{N^3})$. □

The following result was also proved in [6, Section 4.4, Theorem 13, Page 319], but we present an independent proof here according to our path of analysis.

Lemma 2. *During RC4 PRGA, if z_r and z_{r+2} are the key-stream bytes for rounds r and $r+2$ respectively, then $Pr(z_r = \frac{N}{2} \wedge z_{r+2} = 0 | i_r = 0) \approx \frac{1}{N^2} + \frac{1}{N^3}$.*

Proof. We prove this result by referring to the Scenario 3.

[**Scenario 3:**] The conditions corresponding to this scenario are $S_r[0] = \frac{N}{2}$, $S_r[2] = 0$ and $j_r = 0$. Since $i_r = j_r = 0$ and $S_r[0] = \frac{N}{2}$, $z_r = S_r[\frac{N}{2} + \frac{N}{2}] = S_r[N] = \frac{N}{2}$. Similarly, using the result of Mantin-Shamir (Theorem 1 in [8]) we know that unless $S_r[1] = 2$, z_{r+2} equals 0. Therefore, the probability associated with this scenario corresponding to the desired lag-one digraph is $\frac{1}{N^3} \cdot (1 - \frac{2}{N-2}) \approx \frac{1}{N^3}$.

In rest of the configurations, z_r and z_{r+2} can have the values of $(\frac{N}{2}, 0)$ with equal chance as of any other value pair. Hence, the probability for each of these configurations would be $\frac{1}{N^2}$. Therefore, the probability associated with the complimentary cases is $(1 - \frac{1}{N^3}) \cdot \frac{1}{N^2}$.

By combining the probabilities for Scenario 3 and rest of the cases, we get the desired result $Pr(z_r = \frac{N}{2} \wedge z_{r+2} = 0 | i_r = 0) \approx (\frac{1}{N^2} + \frac{1}{N^3} - \frac{1}{N^5}) \approx (\frac{1}{N^2} + \frac{1}{N^3})$. □

Lemma 3. *During RC4 PRGA, if z_r and z_{r+2} are the key-stream bytes for rounds r and $r+2$ respectively, then $Pr((z_r = (N-2)) \wedge (z_{r+2} = 0) | i_r = (N-2)) \approx \frac{1}{N^2} + \frac{1}{N^3}$.*

Proof. We prove this result by referring to the Scenario 4.

[**Scenario 4:**] The conditions corresponding to this scenario are $S_r[N-2] = (N-2)$, $S_r[0] = 0$ and $j_r = 0$. Since $i_r = (N-2)$, $j_r = 0$ and $S_r[N-2] = (N-2)$, $z_r = S_r[N-2+0] = S_r[N-2] = (N-2)$. Again, using the result of Mantin-Shamir (Theorem 1 in [8]) we know that unless $S_r[N-1] = 2$, z_{r+2} equals 0. Therefore, the probability associated with this scenario corresponding to the desired lag-one digraph is $\frac{1}{N^3} \cdot (1 - \frac{2}{N-2}) \approx \frac{1}{N^3}$.

In rest of the configurations, z_r and z_{r+2} can have the values of $(N-2, 0)$ with equal chance as of any other value pair. Hence, the probability for each of these configurations would be $\frac{1}{N^2}$. Therefore, the probability associated with the complimentary cases is $(1 - \frac{1}{N^3}) \cdot \frac{1}{N^2}$.

By combining the probabilities for Scenario 4 and rest of the cases, we get the desired result $Pr(z_r = (N-2) \land z_{r+2} = 0 | i_r = (N-2)) \approx (\frac{1}{N^2} + \frac{1}{N^3} - \frac{1}{N^5}) \approx (\frac{1}{N^2} + \frac{1}{N^3})$. □

Lemma 4. *During RC4 PRGA, if z_r and z_{r+2} are the key-stream bytes for rounds r and $r+2$ respectively, then $Pr((z_r = 0) \land (z_{r+2} = 0) | i_r = (N-2)) \approx \frac{1}{N^2} - \frac{1}{N^3}$.*

Proof. We prove this result by referring to the Scenario 5.

[**Scenario 5:**] In this scenario, the given condition is $S_r[0] = 0$ which can occur with a probability of $\frac{1}{N}$. We now investigate this scenario under two cases - one for $j_r = 0$ and the other one for $j_r \neq 0$.

When $j_r = 0$, $S_r[i_r]$ can't be 0 as $i_r = (N-2)$ and hence $S_r[S_r[i_r] + S_r[j_r]] = S_r[S_r[i_r]] \neq 0$. So $z_r \neq 0$. On the other hand, when $j_r \neq 0$ and $S_r[0] = 0$ (assumed condition for this scenario), by using the converse of Mantin-Shamir result (Theorem 1 in [8]), we know that z_{r+2} can't be 0. Therefore, in Scenario 5, the desired lag-one digraph corresponding to $z_r = 0$ and $z_{r+2} = 0$ can't appear in RC4 key-stream.

In rest of the configurations, z_r and z_{r+2} can have the values of $(0, 0)$ with equal chance as of any other value pair. Hence, the probability for each of these configurations would be $\frac{1}{N^2}$. Therefore, the probability associated with the complimentary cases is $(1 - \frac{1}{N}) \cdot \frac{1}{N^2}$.

By combining the probabilities for Scenario 5 and rest of the configurations, we get the desired result $Pr(z_r = 0 \land z_{r+2} = 0 | i_r = (N-2)) \approx (0 + \frac{1}{N^2} - \frac{1}{N^3}) = (\frac{1}{N^2} - \frac{1}{N^3})$. □

Based on these results (Lemma 1 – Lemma 4), we derive the following corollaries. The corollaries reveal conditional biases on the pseudo-random index j and that of certain permutation array bytes. These results demonstrate that it's possible to get sharper glimpse correlations for the pseudo-random index and the permutation array bytes, by selecting appropriate conditions on key-stream values.

Corollary 3. *During RC4 PRGA, if z_r and z_{r+2} are the key-stream bytes for rounds r and $r+2$ respectively, then $Pr(j_r = 0 | z_r = 0, z_{r+2} = 0, i_r = 0) \approx \frac{4}{N}$.*

Proof. We prove this result by analyzing the following three cases.

[**Case 1:**] We first consider the case of $S_r[0] = 0$. Since $i_r = 0$ and $z_r = 0$ (as per the given conditions), $j_r = 0$. So, the desired condition is met by choosing $S_r[0] = 0$. Let us now consider the number of permutations that can be freely chosen under this case to ensure the given conditions ($z_r = 0$ and $z_{r+2} = 0$ for $i_r = 0$).

In round $(r+1)$, there is no constraint on the values of $S_{r+1}[i_{r+1}]$ and $S_{r+1}[j_{r+1}]$. Hence these two values can be chosen in $(N-1)$ and $(N-2)$ ways (since 0 is already taken in place of $S_r[0]$) in any order. In round $(r+2)$, depending on the value of $S_{r+2}[j_{r+2}]$ and the position of 0 in the permutation array (in that round), the value of $S_{r+1}[j_{r+2}]$ (which is same as $S_{r+2}[i_{r+2}]$ after the swap step) gets constrained to ensure $z_{r+2} = 0$. Hence, the number of ways one may select the permutation array bytes to satisfy the given conditions ($z_r = 0$ and $z_{r+2} = 0$ for $i_r = 0$) is $(N-1)(N-2)(N-3)(N-5)! \approx (N-2)!$ assuming $(N-1) \approx (N-4)$ for large N. Since in this case two array byte-values got constrained ($S_r[0]$ and $S_{r+2}[i_{r+2}]$), we may conceptually view it as a $(N-2)$ "degree-of-freedom" array that has $(N-2)!$ number of possible permutations to ensure the given conditions. We would use this approach in rest of the cases as well as in other corollaries. Let us designate the count $(N-2)!$ as $\mathcal{X}$.

[**Case 2:**] Let us next consider the case of $S_r[2] = 0$. Since $i_r = 0$ and $z_{r+2} = 0$, we must have $j_r = 0$. Let us now consider the number of permutations that can be freely chosen under this constraint to ensure the given configuration of $z_r = 0$ and $z_{r+2} = 0$ with $i_r = 0$.

In round r, in order to ensure $z_r = 0$, the value of $S_r[0]$ can be either 1 or $(\frac{N}{2}+1)$. In each of these two cases, there are $(N-2)$ array bytes that can be freely chosen with a total number of $(N-2)!$ permutations. Hence, the permissible count of permutations for this case would be $2\mathcal{X}$.

[**Case 3:**] In each of the remaining cases, array byte value 0 is present in a location other than $S_r[0]$ and $S_r[2]$. In this configuration $j_r = 0$ would constrain the byte position of 0 and depending on the value of $S_{r+2}[j_{r+2}]$, the value of $S_{r+2}[i_{r+2}]$ gets constrained to ensure $z_{r+2} = 0$. Therefore, the array has $(N-2)$ degrees of freedom leading to $(N-2)! = \mathcal{X}$ number of permutations.

In general, the configuration of $z_r = 0$ for $i_r = 0$ does not impose any constraint on the array byte values as j_r can be freely chosen for any random permutation. However, once a random permutation is chosen the position of 0 at round $(r+2)$ also gets pre-determined imposing a constraint on the value of $S_{r+2}[i_{r+2}]$. This implies the total number of allowed permutations would be $(N-1)! = (N-1).\mathcal{X}$.

Considering all three cases and the total number of possible permutations we arrive at the desired result $Pr(j_r = 0 | z_r = 0, z_{r+2} = 0, i_r = 0) \approx \frac{4\mathcal{X}}{(N-1)\cdot\mathcal{X}} \approx \frac{4}{N}$. □

Corollary 4. *During RC4 PRGA, if z_r and z_{r+2} are the key-stream bytes for rounds r and $r+2$ respectively, then $Pr((j_r = 0 \wedge S_r[0] = 0 | z_r = 0, z_{r+2} = 0, i_r = 0) \approx \frac{1}{N}$.*

Proof. This directly follows from the Case-1 of the Corollary 3 in which the number of allowed permutations is $\mathcal{X}$ and hence $Pr((j_r = 0 \wedge S_r[0] = 0 | z_r = 0, z_{r+2} = 0, i_r = 0) \approx \frac{\mathcal{X}}{(N-1)\cdot\mathcal{X}} \approx \frac{1}{N}$. □

Next corollary is similar to Corollary 3 and is related to Lemma 2.

Corollary 5. *During RC4 PRGA, if z_r and z_{r+2} are the key-stream bytes for rounds r and $r+2$ respectively, then $Pr(j_r = 0 | z_r = \frac{N}{2}, z_{r+2} = 0, i_r = 0) \approx \frac{4}{N}$.*

Proof. We prove this result by analyzing the following three cases.

[**Case 1:**] We first consider the case of $S_r[0] = \frac{N}{2}$. Since $i_r = 0$ and $z_r = \frac{N}{2}$, we must have $j_r = 0$.

Let us now consider a sub-case where $S_r[2] = 0$. Using the result of Mantin & Shamir we know that this configuration ensures $z_{r+2} = 0$ since $j_r = 0$. Hence, the number of permutations that can be freely chosen under this constraint where $S_r[0]$ and $S_r[2]$ has been assigned to ensure the given configuration of $z_r = \frac{N}{2}$ and $z_{r+2} = 0$ with $i_r = 0$, is $(N-2)!$. Borrowing the terminology as per Corollary 3, we designate this count as $\mathcal{X}$.

We now consider the sub-case where $S_r[2] \neq 0$. In this case to ensure $z_{r+2} = 0$, the permutation array byte $S_{r+2}[i_{r+2}]$ gets constrained so that $S_{r+2}[i_{r+2}] + S_{r+2}[j_{r+2}]$ points to the position corresponding to the byte-value of 0. Hence, in this sub-case the "degree of freedom" being $(N-2)$, the number of permutation bytes that can be freely chosen is around $(N-2)! = \mathcal{X}$.

[**Case 2:**] Let us next consider the case of $S_r[0] \neq \frac{N}{2}$ and $S_r[2] = 0$. Since $i_r = 0$ and $z_{r+2} = 0$, we must have $j_r = 0$. Depending upon the value of $S_r[0]$, the position of array byte-value of $\frac{N}{2}$ gets constrained. Hence, the number of permutations that can be freely chosen under this constraint to ensure the given configuration of $z_r = \frac{N}{2}$ and $z_{r+2} = 0$ with $i_r = 0$ is $\mathcal{X}$.

[**Case 3:**] In each of the remaining cases, $S_r[0] \neq \frac{N}{2}$ and $S_r[2] \neq 0$. Therefore $j_r = 0$ would constrain the byte position of $\frac{N}{2}$ and depending on the value of $S_{r+2}[j_{r+2}]$, the value of $S_{r+2}[i_{r+2}]$ gets constrained to ensure $z_{r+2} = 0$. Therefore, there would be $(N-2)$ array byte locations which can be freely chosen giving rise to $(N-2)! = \mathcal{X}$ permutations.

In general, the configuration of $z_r = \frac{N}{2}$ for $i_r = 0$ does not impose any constraint on the array byte values as j_r can be freely chosen for any random permutation. However, once a random permutation is chosen, the position of byte-value 0 at round $(r+2)$ also gets fixed thereby imposing a constraint on $S_{r+2}[i_{r+2}]$. This implies the total number of allowed permutations would be $(N-1)! = (N-1).\mathcal{X}$.

Considering all three cases and the total number of possible permutations we arrive at the desired result $Pr(j_r = 0 | z_r = \frac{N}{2}, z_{r+2} = 0, i_r = 0) \approx \frac{4\mathcal{X}}{(N-1)\cdot\mathcal{X}} \approx \frac{4}{N}$. □

Corollary 6. *During RC4 PRGA, if z_r and z_{r+2} are the key-stream bytes for rounds r and $r+2$ respectively, then $Pr((j_r = 0 \wedge S_r[0] = \frac{N}{2} | z_r = \frac{N}{2}, z_{r+2} = 0, i_r = 0) \approx \frac{2}{N}$.*

Proof. This directly follows from the Case-1 of the Corollary 5 in which the number of allowed permutations is $2 \cdot \mathcal{X}$ and hence $Pr((j_r = 0 \wedge S_r[0] = \frac{N}{2} | z_r = \frac{N}{2}, z_{r+2} = 0, i_r = 0) \approx \frac{2\cdot\mathcal{X}}{(N-1)\cdot\mathcal{X}} \approx \frac{2}{N}$. □

Next corollary is similar to Corollary 3 and Corollary 5 and is related to Lemma 3.

Corollary 7. *During RC4 PRGA, if z_r and z_{r+2} are the key-stream bytes for rounds r and $r+2$ respectively, then $Pr((j_r = 0 | z_r = (N-2), z_{r+2} = 0, i_r = (N-2)) \approx \frac{4}{N}$.*

Proof. We prove this result by analyzing the following three cases.

[**Case 1:**] We first consider the case of $S_r[0] = 0$. Since $i_r = (N-2)$ and $z_{r+2} = 0$, using the result of Mantin & Shamir we know that $j_r = 0$.

Let us now consider a sub-case where $S_r[N-2] = (N-2)$. Since $S_r[0] = 0$, $i_r = (N-2)$ and $j_r = 0$, z_r must be 0 which means this configuration satisfies the given and desired conditions. Hence, the number of permutations that can be freely chosen under this constraint where two permutation array bytes ($S_r[N-2]$ and $S_r[0]$) have been fixed is $(N-2)!$. Borrowing the terminology as per Corollary 3, we designate this count as $\mathcal{X}$.

We now consider the sub-case where $S_r[N-2] \neq (N-2)$. In this case to ensure $z_r = (N-2)$, the permutation array byte $S_r[i_r]$ gets constrained so that $S_r[N-2] + S_r[0] = S_r[N-2]$ points to the position corresponding to the byte-value of $(N-2)$. Hence, in this sub-case the "degree of freedom" being $(N-2)$, the number of permutation bytes that can be freely chosen is around $(N-2)! = \mathcal{X}$.

[**Case 2:**] Let us next consider the case of $S_r[N-1] = 0$. Since $i_r = (N-2)$ and $S_r[N-1] = 0$, then $j_r = 0$ would guarantee $j_{r+1} = j_{r+2} = j_r$ which would in turn ensure the given condition of $z_{r+2} = S_{r+2}[0+0] = 0$. Depending upon the position of array byte-value of $(N-2)$ one of the values among $S_r[i_r]$ or $S_r[j_r]$ gets constrained. Hence, the number of permutations that can be freely chosen under this constraint to ensure the given configuration of $z_r = (N-2)$ and $z_{r+2} = 0$ with $i_r = (N-2)$ and $j_r = 0$ is $\mathcal{X}$.

[**Case 3:**] In each of the remaining cases, $S_r[0] \neq 0$ and $S_r[N-1] \neq 0$. Therefore $j_r = 0$ would constrain the byte position of $(N-2)$ and depending on the value of $S_{r+2}[j_{r+2}]$, the value of $S_{r+2}[i_{r+2}]$ gets constrained to ensure $z_{r+2} = 0$. Therefore, there would be $(N-2)$ array byte locations which can be freely chosen giving rise to $(N-2)! = \mathcal{X}$ permutations.

In general, the configuration of $z_r = (N-2)$ for $i_r = 0$ does not impose any constraint on the array byte values as j_r can be freely chosen for any random permutation. However, once a random permutation is chosen, the position of byte-value 0 imposes a constraint on $S_{r+2}[i_{r+2}]$. This implies the total number of allowed permutations would be $(N-1)! = (N-1).\mathcal{X}$.

Considering all three cases and the total number of possible permutations we arrive at the desired result $Pr(j_r = 0 | z_r = (N-2), z_{r+2} = 0, i_r = (N-2)) \approx \frac{4\mathcal{X}}{(N-1)\cdot\mathcal{X}} \approx \frac{4}{N}$. □

Corollary 8. *During RC4 PRGA, if z_r and z_{r+2} are the key-stream bytes for rounds r and $r+2$ respectively, then $Pr((j_r = 0 \wedge S_r[0] = 0 | z_r = (N-2), z_{r+2} = 0, i_r = (N-2)) \approx \frac{2}{N}$.*

Proof. This directly follows from the Case-1 of the Corollary 7 in which the number of allowed permutations is $2 \cdot \mathcal{X}$ and hence $Pr((j_r = 0 \wedge S_r[0] = 0 | z_r = (N-2), z_{r+2} = 0, i_r = (N-2)) \approx \frac{2\cdot\mathcal{X}}{(N-1)\cdot\mathcal{X}} \approx \frac{2}{N}$. □

4 Further exploration of Glimpses

As we have already discussed, certain information about hidden state variables of RC4 can be exposed due to the choice of the output function $z = S[S[i] + S[j]]$.

Theorem 1 (Glimpse theorem). *[5] During RC4 PRGA, for any arbitrary round r, $Pr(S_r[i_r] = j_r - z_r) = \frac{2}{N} - \frac{1}{N^2}$ and $Pr(S_r[j_r] = i_r - z_r) = \frac{2}{N} - \frac{1}{N^2}$.*

Proof. This result follows from the fact that if $S_r[j_r]$ (respectively $S_r[i_r]$) happens to be same as z_r, then $(S_r[i_r] + S_r[j_r])$ which is the location of the output key-stream byte, must be same as j_r (resp. i_r). Therefore, $S_r[j_r] = z_r$ (resp. $S_r[i_r] = z_r$) configuration guarantees the desired condition of $S_r[i_r] = j_r - z_r$ (resp. $S_r[j_r] = j_r - z_r$). Under the assumption of uniform random distribution, if we consider $Pr(S_r[j_r] = z_r) = \frac{1}{N}$ and for the complimentary condition of $S_r[i_r] \neq z_r$, if we assume a fair chance of $\frac{1}{N}$ for the desired event of $S_r[i_r] = j_r - z_r$, the overall probability calculation comes out to be $Pr(S_r[i_r] = j_r - z_r) = \frac{1}{N} \cdot 1 + (1 - \frac{1}{N})\frac{1}{N} = \frac{2}{N} - \frac{1}{N^2}$. In the same way, one can also prove the other result, i.e., $Pr(S_r[j_r] = i_r - z_r) = \frac{2}{N} - \frac{1}{N^2}$. □

In this regard, let us first explain the limitation of the existing Glimpse theorem presented in [5] and as explained above in Theorem 1.

4.1 Deeper analysis of the Glimpse theorem [5]

This result needs to be studied in a more disciplined manner, and in that direction, we concentrate on the case when $i = j$. This is the case where the Glimpse is not uniform. If $(i_r = j_r)$, then $S_r[i_r] = S_r[j_r]$ and hence the conditions $S_r[i_r] = (j_r - z_r)$ and $S_r[j_r] = (i_r - z_r)$ essentially merge to one and the same condition. Interestingly, in this situation we lose the flexibility of choosing one of the permutation bytes (say $S_r[i_r]$) in accordance with the other byte $(S_r[j_r])$ to guarantee either $S_r[i_r] = z_r$ or $S_r[j_r] = z_r$. Thus the method of proving the Glimpse theorem fails in the case when $i_r = j_r$.

Lemma 5. *During RC4 PRGA, for any arbitrary round r,*

1. $Pr(S_r[i_r] = S_r[j_r] = i_r - z_r = j_r - z_r | i_r = j_r \equiv 1 \bmod 2) = \frac{1}{N-1} \approx \frac{1}{N}$ *and*
2. $Pr(S_r[i_r] = S_r[j_r] = i_r - z_r = j_r - z_r | i_r = j_r \equiv 0 \bmod 2) = \frac{3}{N} - \frac{1}{N(N-1)} \approx \frac{3}{N}$.

Proof. For the first case, we observe that when $i_r = j_r = (2*m+1), m \in \mathbf{N}$, it's impossible for $S_r[i_r] + S_r[j_r]$, which is an even number, to be equal to i_r (or j_r). So whatever may be the value of $S_r[i_r]$, it can't be same as z_r. Hence the possible number of values that z_r can take is $(N-1)$ and not N. Now, for every value of $S_r[i_r]$, there can be exactly one possible value of z_r, so that the Glimpse condition holds. Hence, the probability that $S_r[i_r] = (i_r - z_r) = \frac{1}{N-1}$.

Now we prove the second part. We observe that when $i_r = j_r = 2*m, m \in \mathbf{N}$, if the value of $S_r[i_r]$ happens to be m, z_r equals $S_r[i_r]$. Similarly, if $S_r[i_r] = m + \frac{N}{2}$, then also $z_r = S_r[i_r]$. Hence in these two cases, the Glimpse conditions holds. In rest of the $(N-2)$ cases, $S_r[i_r] + S_r[j_r]$ would have a value different from i_r (or j_r). Hence, the probability that $S_r[i_r] = (i_r - z_r) = \frac{1}{N-1}$. The overall probability can be computed as

$$\begin{aligned}
& Pr(S_r[i_r] = S_r[j_r] = i_r - z_r = j_r - z_r | i_r = j_r = 2*m, m \in \mathbf{N}) \\
= & \frac{N-2}{N} \cdot \frac{1}{N-1} + \frac{2}{N} \cdot 1 \\
= & \frac{N-2}{N(N-1)} + \frac{2}{N} = \frac{3N-4}{N(N-1)} = \frac{3}{N} - \frac{1}{N(N-1)}
\end{aligned}$$

□

From Lemma 5, it becomes apparent that if we combine the even and odd cases and consider the overall $i = j$ event, the probability of the Glimpse condition would converge towards $\frac{2}{N}$ for $i = j$. However, the cases for i even and odd are actually different and that we will show in Theorem 2. Before that, let us now consider the $i \neq j$ scenario in detail.

Lemma 6. *During RC4 PRGA, for any arbitrary round r,*

1. $Pr(S_r[j_r] = i_r - z_r | i_r \textit{ odd}, i_r \neq j_r) \approx \frac{2}{N} + \frac{1}{N(N-1)} - \frac{2}{N(N-1)(N-2)}$
2. $Pr(S_r[j_r] = i_r - z_r | i_r \textit{ even}, i_r \neq j_r) \approx \frac{2}{N} - \frac{1}{N(N-1)}$
3. $Pr(S_r[i_r] = j_r - z_r | i_r \textit{ odd}, i_r \neq j_r) \approx \frac{2}{N} - \frac{1}{N(N-1)(N-2)}$
4. $Pr(S_r[i_r] = j_r - z_r | i_r \textit{ even}, i_r \neq j_r) \approx \frac{2}{N} - \frac{1}{N(N-1)(N-2)}$

Proof. In RC4 PRGA, for any arbitrary round r, we assume the permutation array bytes are arranged in a uniform random distribution. Let us first prove the item 1. We consider the following three cases [(a)–(c)].

Case (a): In this case, the permutation array bytes are chosen in such a way that z_r becomes equal to $S_r[i_r]$. To achieve that, let us first arbitrarily choose $S_r[i_r] = x$ and this can be done in N ways. Next, we assign $S_r[j_r] = i_r - x$ (since i_r is an odd number, $i_r - x$ can't be equal to x). Rest of the $(N-2)$ bytes can be chosen in $(N-2)!$ ways. Hence, the total number of ways in which we can satisfy the desired condition for this case is $N(N-2)!$.

Case (b): We now investigate the possibility of $z_r = S_r[j_r]$. To satisfy the desired condition, this would imply $i_r = 2 \cdot z_r$. However, this is impossible since i_r is an odd number.

Case (c): Next, we consider the case where $z_r \neq S_r[i_r]$ and $z_r \neq S_r[j_r]$. Let us first choose $S_r[i_r] = x$ which can be done in N possible ways. Next, we choose $S_r[j_r] = y$. Now

y can't be equal to x (as $i_r \neq j_r$) or $i_r - x$ (as that's already included in Case (a)) or $j_r - x$ (as that's improbable as per Case (b)). Once, $S_r[i_r]$ and $S_r[j_r]$ are chosen, z_r gets defined to satisfy the desired condition and the position of z_r also gets decided as $x + y$. Rest of the $(N-3)$ bytes can be chosen in $(N-3)!$ ways. Hence, the total number of ways in which we can satisfy the desired condition for this case is $N(N-3)(N-3)!$.
Since the total number of ways the permutation byte array can be chosen is $N!$, the desired probability can be derived from the above three cases as $Pr(S_r[j_r] = i_r - z_r | i_r\ odd, i_r \neq j_r) \approx [N(N-2)! + 0 + N(N-3)(N-3)!]/N! \approx \frac{2}{N} + \frac{1}{N(N-1)} - \frac{2}{N(N-1)(N-2)}$.

Let us next prove the item 2. We consider the following four cases [(a)–(d)] here.

Case (a): In this case, the permutation array bytes are chosen in such a way that z_r becomes equal to $S_r[i_r]$. To achieve that, let us first arbitrarily choose $S_r[i_r] = x$ and this can be done in $(N-2)$ ways by excluding the two values $\frac{i_r}{2}$ and $\frac{(i_r+N)}{2}$ (which are handled in Case (b)). Next, we assign $S_r[j_r] = i_r - x$ (since $i_r \neq 2x$, $(i_r - x)$ can't be equal to x). Rest of the $(N-2)$ bytes can be chosen in $(N-2)!$ ways. Hence, the total number of ways in which we can satisfy the desired condition for this case is $(N-2)(N-2)!$.

Case (b): In this case we consider the two sub-cases $S_r[i_r] = \frac{i_r}{2}$ and $S_r[i_r] = \frac{(i_r+N)}{2}$. For each of these sub-cases $S_r[j_r]$ can be chosen in $(N-2)$ ways as $S_r[j_r] = y$ can't take any of these two values. Once, $S_r[i_r]$ and $S_r[j_r]$ are chosen, z_r gets defined to satisfy the desired condition and the position of z_r also gets decided as $x + y$. Rest of the $(N-3)$ bytes can be chosen in $(N-3)!$ ways. Hence, the total number of ways in which we can satisfy the desired condition for this case is $2(N-2)(N-3)!$.

Case (c): We now consider the two sub-cases corresponding to $S_r[j_r] = \frac{i_r}{2}$ and $S_r[j_r] = \frac{(i_r+N)}{2}$. These conditions lead to $z_r = S_r[j_r]$ as well. Hence, for each of these sub-cases $S_r[i_r]$ gets defined to ensure $(S_r[i_r] + S_r[j_r]) = j_r$. Rest of the $(N-2)$ bytes can be chosen in $(N-2)!$ ways. Hence, the total number of ways in which we can satisfy the desired condition for this case is $2(N-2)!$.

Case (d): Next, we consider the case where $z_r \neq S_r[i_r]$ and $z_r \neq S_r[j_r]$. Let us first choose $S_r[i_r] = x$ which can be done in $(N-2)$ possible ways (excluding the sub-cases under Case (b)). Next, we choose $S_r[j_r] = y$. Now y can't be equal to x (as $i_r \neq j_r$) or $(i_r - x)$ (as that's already included in Case (a)) or the two sub-cases considered in Case (c) or the value $(j_r - x)$ (as that doesn't satisfy the desired condition unless already subsumed by Case (c)). Once, $S_r[i_r]$ and $S_r[j_r]$ are chosen, z_r gets defined to satisfy the desired condition and the position of z_r also gets decided as $x + y$. Rest of the $(N-3)$ bytes can be chosen in $(N-3)!$ ways. Hence, the total number of ways in which we can satisfy the desired condition for this case is $(N-2)(N-5)(N-3)!$.

Since the total number of ways the permutation byte array can be chosen is $N!$, the desired probability can be derived from the above three cases as $Pr(S_r[j_r] = i_r - z_r | i_r\ even, i_r \neq j_r) \approx [(N-2)(N-2)! + 2(N-2)(N-3)! + 2(N-2)! + (N-2)(N-5)(N-3)!]/N! \approx \frac{2}{N} - \frac{1}{N(N-1)}$.

We now prove the rest of the two results together as the proof would be identical for each of those. We make an observation here that while proving either of the two results [(1) or (2)], we didn't require to differentiate whether j_r is even or odd. Moreover, the result (3) is a dual of result (1) where i_r and j_r have interchanged their positions in the desired condition's expression. Similarly, result (4) is a dual of result (2). However, as no constraint has been put on j_r, we can consider j_r to have equal probability to be even or odd. Therefore, result 3 (as well as result 4) can be derived by considering equal probability on both result 1 and result 2. Hence, we get $Pr(S_r[i_r] = j_r - z_r | i_r \text{ odd}, i_r \neq j_r) \approx \frac{1}{2} \cdot [\frac{2}{N} + \frac{1}{N(N-1)} - \frac{2}{N(N-1)(N-2)}] + \frac{1}{2} \cdot [\frac{2}{N} - \frac{1}{N(N-1)}] = \frac{2}{N} - \frac{1}{N(N-1)(N-2)}$. Result 4 can be proved in a similar manner. □

With this understanding, the revised Glimpse theorem works out to be as followed.

Theorem 2 (Revised Glimpse Theorem). *During RC4 PRGA, for any arbitrary round r,*

1. $Pr(S_r[i_r] = j_r - z_r | i_r \text{ even}) \approx \frac{2}{N} + \frac{1}{N^2}$
2. $Pr(S_r[i_r] = j_r - z_r | i_r \text{ odd}) \approx \frac{2}{N} - \frac{1}{N^2}$
3. $Pr(S_r[j_r] = i_r - z_r | i_r \text{ even}) \approx \frac{2}{N}$
4. $Pr(S_r[j_r] = i_r - z_r | i_r \text{ odd}) \approx \frac{2}{N}$

Proof. While proving this theorem, we use Lemma 5 and Lemma 6. Let us prove the first item. $Pr(S_r[i_r] = j_r - z_r | i_r \text{ even}) = Pr(i_r = j_r) \cdot Pr(S_r[i_r] = j_r - z_r | i_r \text{ even}, i_r = j_r) + Pr(i_r \neq j_r) \cdot Pr(S_r[i_r] = j_r - z_r | i_r \text{ even}, i_r \neq j_r) = \frac{1}{N}[\frac{3}{N} - \frac{1}{N(N-1)}] + \frac{(N-1)}{N}[\frac{2}{N} - \frac{1}{N(N-1)(N-2)}] \approx \frac{2}{N} + \frac{1}{N^2}$.

Next, we prove the second one. $Pr(S_r[i_r] = j_r - z_r | i_r \text{ odd}) = Pr(i_r = j_r) \cdot Pr(S_r[i_r] = j_r - z_r | i_r \text{ odd}, i_r = j_r) + Pr(i_r \neq j_r) \cdot Pr(S_r[i_r] = j_r - z_r | i_r \text{ odd}, i_r \neq j_r) = \frac{1}{N}[\frac{1}{(N-1)}] + \frac{(N-1)}{N}[\frac{2}{N} - \frac{1}{N(N-1)(N-2)}] \approx \frac{2}{N} - \frac{1}{N^2}$.

For the third one, $Pr(S_r[j_r] = i_r - z_r | i_r \text{ even}) = Pr(i_r = j_r) \cdot Pr(S_r[j_r] = i_r - z_r | i_r \text{ even}, i_r = j_r) + Pr(i_r \neq j_r) \cdot Pr(S_r[j_r] = i_r - z_r | i_r \text{ even}, i_r \neq j_r) = \frac{1}{N}[\frac{3}{N} - \frac{1}{N(N-1)}] + \frac{(N-1)}{N}[\frac{2}{N} - \frac{1}{N(N-1)}] \approx \frac{2}{N}$.

Finally, $Pr(S_r[j_r] = i_r - z_r | i_r \text{ odd}) = Pr(i_r = j_r) \cdot Pr(S_r[j_r] = i_r - z_r | i_r \text{ odd}, i_r = j_r) + Pr(i_r \neq j_r) \cdot Pr(S_r[j_r] = i_r - z_r | i_r \text{ odd}, i_r \neq j_r) = \frac{1}{N}[\frac{1}{(N-1)}] + \frac{(N-1)}{N}[\frac{2}{N} + \frac{1}{N(N-1)}] \approx \frac{2}{N}$.

Note that in the above results we have ignored any term of the order of $\frac{1}{N^3}$ or smaller. □

4.2 Further Glimpses based on the output function

Here we start with a conditional glimpse on the pseudo-random hidden index j_r and this result remains valid irrespective of the way the RC4 permutation array evolves. This means that such a glimpse is generated due to the output function only and even if the evolution of j_r happens in a different fashion or the permutation array is shuffled in each step in an arbitrary way, the following non-randomness can still be observed.

Lemma 7. *During the RC4 PRGA, for any arbitrary round r, $\Pr(j_r = i_r | i_r = 2 \cdot z_r) = \frac{2}{N} - \frac{1}{N^2}$, under the usual assumptions.*

Proof. We prove this considering two cases. First we consider $z_r < \frac{N}{2}$. If $S_r[i_r] = z_r$, then under the given condition of $i_r = 2 \cdot z_r$, it is imperative for j_r to be same as i_r. Thus, we have the following result.

$$\begin{aligned}
& Pr(j_r = i_r | i_r = 2 \cdot z_r) \\
= \; & \Pr(j_r = i_r, S_r[i_r] = z_r | i_r = 2 \cdot z_r) + \Pr(j_r = i_r, S_r[i_r] \neq z_r | i_r = 2 \cdot z_r) \\
= \; & \Pr(j_r = i_r | S_r[i_r] = z_r, i_r = 2 \cdot z_r) \cdot \Pr(S_r[i_r] = z_r) \\
+ \; & \Pr(j_r = i_r | S_r[i_r] \neq z_r, i_r = 2 \cdot z_r) \cdot \Pr(S_r[i_r] \neq z_r) \\
= \; & \frac{1}{N} + \frac{1}{N} \cdot (1 - \frac{1}{N}) = \frac{2}{N} - \frac{1}{N^2}
\end{aligned}$$

Here we consider $\Pr(j_r = i_r | S_r[i_r] \neq z_r, i_r = 2 \cdot z_r) = \frac{1}{N}$ under the usual randomness assumption. Further, considering S as a random permutation, we get $\Pr(S_r[i_r] = z_r) = \frac{1}{N}$ as well. These have also been checked experimentally.

Now let us consider $z_r \geq \frac{N}{2}$. In this case, if $S_r[i_r] = z_r$, then under the given condition of $i_r = 2 \cdot z_r \bmod N$, we can deduce that $i_r = 2 \cdot (z_r - \frac{N}{2})$ (without modulo N). Hence, j_r must be same as i_r. The probabilistic result can be deduced in an identical fashion as has been done in the earlier case. This completes the proof. □

Lemma 7, in addition to providing a glimpse of j, also provides a glimpse to the permutation array byte $S_r[i_r]$ or $S_r[j_r]$. This is shown in the following result.

Corollary 9. $\Pr((j_r = i_r) \wedge (j_{r-1} = z_r) | (i_r = 2 \cdot z_r)) \approx \frac{1}{N}$.

Proof. We observe that when $S_r[i_r] = z_r$, then under the given condition of $(i_r = 2 \cdot z_r)$, we get $j_r = i_r$ and $j_{r-1} = j_r - S_r[i_r] = j_r - z_r = 2 \cdot z_r - z_r = z_r$. On the other hand, if $S_r[i_r] \neq z_r$, it's not possible to simultaneously have $(j_r = i_r)$ and $(j_{r-1} = z_r)$, because if $(j_r = i_r)$, the $(j_{r-1} = j_r - S_r[i_r] = 2 \cdot z_r - S_r[i_r])$ can't be equal to z_r.

$$\begin{aligned}
& \Pr((j_r = i_r) \wedge (j_{r-1} = z_r) | (i_r = 2 \cdot z_r)) \\
= \ & \Pr((j_r = i_r) \wedge (j_{r-1} = z_r) \wedge S_r[i_r] = z_r | (i_r = 2 \cdot z_r)) \\
+ \ & \Pr((j_r = i_r) \wedge (j_{r-1} = z_r) \wedge S_r[i_r] \neq z_r | (i_r = 2 \cdot z_r)) \\
= \ & \Pr((j_r = i_r) \wedge (j_{r-1} = z_r) | (i_r = 2 \cdot z_r) \wedge S_r[i_r] = z_r) \cdot \Pr(S_r[i_r] = z_r) \\
+ \ & \Pr((j_r = i_r) \wedge (j_{r-1} = z_r) | (i_r = 2 \cdot z_r) \wedge S_r[i_r] \neq z_r) \cdot \Pr(S_r[i_r] \neq z_r) \\
= \ & \frac{1}{N} + 0 = \frac{1}{N}
\end{aligned}$$

□

Corollary 9 demonstrates that Lemma 6 can be used to predict consecutive two j values with a probability $\approx \frac{1}{N}$ instead of the random probability of $\frac{1}{N^2}$ based on a relationship between the index i and the output key-stream byte z.

4.3 Glimpses based on swap as well as the output function

Maitra and SenGupta [7] had proved a new long-term conditional bias of the order of $\frac{3}{N}$ for a permutation array-byte that can also be referred as a new glimpse result. Here we note that it is only a single results and we can have many results like this. Already in Section 3, we have shown such results of the order of $\frac{4}{N}$ in Corollaries 3, 5, 7. In the same spirit, we prove a set of new results that reveal glimpses of permutation array bytes and/or that of pseudo-random index given a specific pair of values of key-stream bytes. We obtain a bias of the order of $\frac{5}{N}$ in this initiative.

Let us now start with some technical results.

Lemma 8. *During RC4 PRGA, if z_{r-1} and z_r are the key-stream bytes for two consecutive rounds $r-1$ and r respectively, then $Pr((S_r[i_r] = N-1) \wedge (S_r[i_r - 1] = 0) | z_r = z_{r-1} = N-1) \approx \frac{2}{N^2} + \frac{1}{N^3}$.*

Proof. To prove this lemma, let us consider the following four possible alternative cases:

Case 1: $i_{r-1} = j_{r-1}$ and $S_{r-1}[i_{r-1} + 1] = 0$

Case 2: $i_{r-1} = j_{r-1}$ and $S_{r-1}[i_{r-1} + 1] \neq 0$

Case 3: $i_{r-1} \neq j_{r-1}$ and $S_{r-1}[i_{r-1} + 1] = 0$

Case 4: $i_{r-1} \neq j_{r-1}$ and $S_{r-1}[i_{r-1} + 1] \neq 0$

(Case 1) In this case we first prove that $S_{r-1}[i_{r-1}] = N - 1$. Once that is proved, it is easy to observe that in round r the two permutation array bytes would get swapped to reach the desired state of $S_r[i_r] = N - 1$ and $S_r[i_r - 1] = 0$.

Let us assume $S_{r-1}[i_{r-1}] = x$ and $x \neq N - 1$ (clearly $x \neq 0$ as $S_{r-1}[i_{r-1} + 1] = 0$). Since $z_{r-1} = N - 1$ and $i_{r-1} = j_{r-1}$ then $S_{r-1}[2x]$ must be $N - 1$. In round r, after the swap operation $S_r[i_r] = x$ and $S_r[j_r] = 0$. Hence z_r must come from the permutation array byte $S_r[x]$. Since it was assumed that $x \neq N - 1$, neither x nor $2x$ can overlap with any of the array positions that got swapped in round r. As $z_{r-1} = z_r = N-1$, x must be same as $2x$ implying $x = 0$. But that's impossible. Hence our assumption must be wrong. That implies $x = N-1$. In that case, to satisfy the condition that $z_{r-1} = N-1$, it is imperative to have $i_{r-1} = j_{r-1} = N-2$. Such a configuration would also satisfy the $z_r = N-1$ in round r and post-swap the desired state of $S_r[i_r] = N-1$ and $S_r[i_r-1] = 0$ can be reached. Therefore, in this case the expected combination of states is reached by assuming two conditions $i_{r-1} = j_{r-1}$ and $S_{r-1}[i_{r-1}+1] = 0$ with a probability of $\frac{1}{N^2}$.

(Case 2) In this case we set a configuration of $S_{r-1}[i_{r-1}] = 0$ and $S_{r-1}[0] = N-1$. Since $i_{r-1} = j_{r-1}$, this satisfies the given key-stream value of $z_{r-1} = N-1$. As $S_{r-1}[i_{r-1}+1] \neq 0$, it is evident that the permutation array byte $S_{r-1}[i_{r-1}] = 0$ remains unaffected even in round r and hence $S_r[i_r-1] = 0$. Therefore by fixing two values $S_{r-1}[i_{r-1}] = 0$ and $S_{r-1}[0] = N-1$, we get $S_r[i_r-1] = 0$ with a probability of $\frac{1}{N^2}$.

However, we have proved only one part of the desired pair of permutation array bytes and we still need to prove the condition under which $S_r[i_r] = N-1$.

The only way we can arrive at that configuration is to have $j_r = 0$ so that the permutation array byte from that position gets swapped to the position indexed by i_r satisfying the desired state of $S_r[i_r] = N-1$. Let us assume that $S_{r-1}[i_r] = p$. Hence, $j_{r-1} + p = 0$ implying $p = N - j_{r-1} = N - i_r + 1$. Since $z_r = N-1$, it is required to have $p + (N-1) = i_r$ which means $N - i_r + 1 + N - 1 = i_r$ resulting in $i_r = \frac{N}{2}$ and $p = \frac{N}{2} + 1$. Since $S_r[j_r] = p = \frac{N}{2} + 1$ and $i_r - z_r$ also equates to the same value p, the desired condition can be achieved with approximately $\frac{2}{N}$ probability as per the Glimpse theorem.

To sum up, if we set a configuration of $S_{r-1}[i_{r-1}] = 0$ and $S_{r-1}[0] = N-1$, we get the desired pair of permutations $S_r[i_r] = N-1$ and $S_r[i_r-1] = 0$ with a probability of $\frac{2}{N^3}$.

(Case 3) In this case $S_{r-1}[i_{r-1}+1] = 0$ implying $j_{r-1} = j_r$. Hence the value of $S_{r-1}[i_{r-1}]$ remains unchanged in round r. Since $S_{r-1}[i_{r-1}] \neq 0$ the desired pair of permutations can't be achieved in this case. This condition corresponds to the probability of $\frac{1}{N} \cdot (1 - \frac{1}{N})$.

(Case 4) This is the case that remains after eliminating all the combinations that fall under rest of the three cases discussed above (Case 1-3). The probability of arriving at any of these cases would therefore be equal to $(1 - \frac{1}{N^2} - \frac{2}{N^3} - (\frac{1}{N} - \frac{1}{N^2})) = (1 - \frac{1}{N} - \frac{2}{N^3})$. For each of these cases the probability of getting the desired pair of permutation bytes $S_{r-1}[i_{r-1}] = 0$ and $S_{r-1}[0] = N-1$ would be $\frac{1}{N^2}$ due to random choice. Hence, the overall probability for this case would be approximately equal to $(\frac{1}{N^2} - \frac{1}{N^3})$.

We can now combine all four cases and declare the probability for the desired event as $\frac{1}{N^2} + \frac{2}{N^3} + 0 + (\frac{1}{N^2} - \frac{1}{N^3}) = \frac{2}{N^2} + \frac{1}{N^3}$. □

The lemma also leads to the following corollary that provides a glimpse of j in round $r-1$. It is interesting to observe that while in case of three independent events the expected joint probability would have been of the order of $\frac{1}{N^3}$, the bias observed is one order higher (around $\frac{1}{N^2}$).

Corollary 10. *During RC4 PRGA, if z_{r-1} and z_r are the key-stream bytes for two consecutive rounds $r-1$ and r respectively, then $Pr((S_r[i_r] = N-1) \wedge (S_r[i_r-1] = 0) \wedge (i_{r-1} = j_{r-1}) | z_r = z_{r-1} = N-1) \approx \frac{1}{N^2} + \frac{2}{N^3}$.*

Proof. From the proof of Lemma 8, it is apparent that the first two cases correspond to the configuration of $i_{r-1} = j_{r-1}$ in addition to the desired values of two permutation bytes in round r. Hence, the probability associated with the combined event is approximately $\frac{1}{N^2} + \frac{2}{N^3}$. □

We now present another lemma that demonstrates glimpse of two consecutive permutation array bytes.

Lemma 9. *During RC4 PRGA, if z_{r-1} and z_r are the key-stream bytes for two consecutive rounds $r-1$ and r respectively, then $Pr((S_r[i_r] = 1) \wedge (S_r[i_r - 1] = (\frac{N}{2}+1))|z_r = z_{r-1} = (\frac{N}{2}+1)) \approx \frac{2}{N^2}$.*

Proof. We prove this lemma by first demonstrating two special cases and then by handling rest of the cases.

[**Case 1:**] Let us consider the configuration of the permutation array along with the indices such that $i_{r-1} = j_{r-1}$ and $S_{r-1}[i_{r-1}+1] = 1$.

Based on this configuration, i_r must be equal to j_r. Since $z_r = (\frac{N}{2}+1)$, we can deduce that $z_r = S_r[2] = (\frac{N}{2}+1)$. As z_{r-1} is also equal to $(\frac{N}{2}+1)$ and $i_{r-1} = j_{r-1}$, it is imperative to have $S_{r-1}[i_{r-1}] = (\frac{N}{2}+1)$ which also means $i_{r-1} = 2$. Clearly in round r, there is no change of the byte values in the array positions at i_{r-1} and i_r. Hence in this case the desired condition of $S_r[i_r] = 1$ and $S_r[i_r - 1] = (\frac{N}{2}+1)$ are satisfied and the probability associated with this case would be $\frac{1}{N^2}$.

[**Case 2:**] We now consider the configuration $i_{r-1} = j_{r-1}$ and $S_{r-1}[i_{r-1}+1] \neq 1$.

Based on this configuration, to satisfy the given conditions of $z_r = z_{r-1} = (\frac{N}{2}+1)$ and one of the desired conditions of $(S_r[i_r - 1] = (\frac{N}{2}+1))$, one must have $i_{r-1} = 2$. With that value of i_{r-1}, the only possible choice for $S_{r-1}[i_{r-1}+1]$ is 1 else in round r there is no other way to satisfy the target position of the key-stream byte which is given by $S_r[i_r] + S_r[j_r]$. Hence, the probability associated with this case corresponding to the given and desired conditions is $\frac{1}{N}(1 - \frac{1}{N}) \cdot (0) = 0$.

[**Case 3:**] Let us next consider all the remaining configurations given by $i_{r-1} \neq j_{r-1}$.

In this case, irrespective of the value of $S_{r-1}[i_{r-1}+1]$, the given condition of $z_r = z_{r-1} = (\frac{N}{2}+1)$ can be satisfied with or without meeting the desired condition of $(S_r[i_r] = 1) \wedge (S_r[i_r - 1] = (\frac{N}{2}+1))$. Hence, there is a fair chance of satisfying the desired condition with the probability of $\frac{1}{N^2}$ within the probability corresponding to this case, which is $(1 - \frac{1}{N^2} - \frac{1}{N} + \frac{1}{N^2})$.

Combining the probability of all the three cases we get $Pr((S_r[i_r] = 1) \wedge (S_r[i_r - 1] = (\frac{N}{2}+1))|z_r = z_{r-1} = (\frac{N}{2}+1)) \approx (\frac{2}{N^2} - \frac{1}{N^3}) \approx \frac{2}{N^2}$. □

Similar to Lemma 8, this lemma also leads to a corollary that provides a glimpse of j in round $r-1$.

Corollary 11. *During RC4 PRGA, if z_{r-1} and z_r are the key-stream bytes for two consecutive rounds $r-1$ and r respectively, then $Pr((S_r[i_r] = 1) \wedge (S_r[i_r - 1] = (\frac{N}{2}+1)) \wedge (j_{r-1} = 2)|z_r = z_{r-1} = (\frac{N}{2}+1)) \approx \frac{1}{N^2}$.*

Proof. From the proof of Lemma 9, it is evident that Case 1 corresponds to the desired configuration of $(S_r[i_r] = 1) \wedge (S_r[i_r - 1] = (\frac{N}{2}+1)) \wedge (j_{r-1} = 2)$ in addition to satisfying the given condition on two successive key-stream bytes. Hence, the probability associated with the combined event is approximately $\frac{1}{N^2}$. Interestingly the probability would remain the same even when we associate the additional conditions of $(i_{r-1} = 2) \wedge (j_r = 3)$. □

We now present a significant result that provides a glimpse of a permutation array byte with a probability of around $\frac{5}{N}$ based on a specific condition for two consecutive key-stream byte values. The next lemma demonstrates the glimpse of two consecutive permutation array bytes based on another condition on the values of successive key-stream bytes.

Lemma 10. *During RC4 PRGA, if z_{r-1} and z_r are the key-stream bytes for two consecutive rounds $r-1$ and r respectively, then $Pr((S_r[i_r] = 0) \wedge (S_r[i_r - 1] = (N-1))|z_r = 0, z_{r-1} = (N-1)) \approx \frac{3}{N^2}$.*

Proof. We prove this lemma by first demonstrating two special cases and then by handling rest of the cases.

[**Case 1:**] Let us consider the configuration of the permutation array bytes as $S_{r-1}[i_{r-1}] = (N-1)$ and $S_{r-1}[i_{r-1}+1] = 0$.

We now prove that in this configuration j_{r-1} must be equal to $(i_{r-1}+1)$ by contradiction. If that's not true then $S_{r-1}[j_{r-1}] = p$ where p is an arbitrary byte value $(\neq 0)$. Since $z_{r-1} = (N-1)$ which is in the byte position of i_{r-1}, p must be equal to $(i_{r-1} - N + 1)$. Since $S_{r-1}[i_{r-1}+1] = 0$, j_r must be same as j_{r-1}. As $z_r = 0$, $S_{r-1}[j_{r-1}]$ must have the value of j_{r-1}. So $p = j_{r-1}$. This is not possible if $j_{r-1} \neq (i_{r-1}+1)$. Hence, we must have $j_{r-1} = (i_{r-1}+1)$ which would satisfy the desired condition of $(S_r[i_r] = 0) \wedge (S_r[i_r - 1] = (N-1))$ and the probability associated with this event would be $\frac{1}{N^2}$.

[**Case 2:**] Let us now consider the configuration of the permutation array bytes as $S_{r-1}[i_{r-1}] \neq (N-1)$ and $S_{r-1}[i_{r-1}+1] = 0$.

In this configuration, since $S_{r-1}[i_{r-1}+1] = 0$, in order to satisfy the given condition of $z_r = 0$ as well as the expected condition of $S_r[i_r] = 0$, the only possible choice is to have $j_r = j_{r-1} = i_{r-1} + 1 = 0$ which also means i_{r-1} must be equal to $(N-1)$. Hence, the value at $S_{r-1}[i_{r-1}]$ remains unaltered in round r making it impossible to satisfy one of the desired conditions, i.e., $S_r[i_r - 1] = (N-1)$. Hence the probability associated with this case would be $(1 - \frac{1}{N})\frac{1}{N} \cdot (0) = 0$.

[**Case 3:**] Let us now consider the configuration of the permutation array bytes as $S_{r-1}[i_{r-1}] = (N-1)$ and $S_{r-1}[i_{r-1}+1] \neq 0$.

In this configuration, since $S_{r-1}[i_{r-1}+1] \neq 0$, in order to satisfy the given condition of $z_r = 0$ as well as the expected condition of $S_r[i_r] = 0$, the only possible choice is to have $S_{r-1}[j_r] = 0$ so that the swap operation ensures $S_r[i_r] = 0$. Hence, the value at $S_{r-1}[i_{r-1}]$ remains unaltered in round r, i.e., $S_r[i_r - 1] = (N-1)$. Hence the probability associated with this case would also be $\frac{1}{N^2}$.

[**Case 4:**] We now consider a special configuration of the permutation array bytes as $S_{r-1}[i_{r-1}] = 0$ and $S_{r-1}[i_{r-1}+1] = (N-1)$.

In this configuration, the only way to satisfy the desired condition of $(S_r[i_r] = 0) \wedge (S_r[i_r - 1] = (N-1))$ is to swap the permutation array bytes in round r. That implies $i_{r-1} = (N-2)$ and $j_{r-1} = (N-1)$ which would satisfy the given as well as the desired conditions. Clearly, the probability associated with this case would be $\frac{1}{N^2}$. From this case it is also evident that in the rest of cases corresponding to $S_{r-1}[i_{r-1}] \notin \{0, (N-1)\}$ and $S_{r-1}[i_{r-1}+1] \notin \{0, (N-1)\}$, it is impossible to satisfy the desired conditions.

By combining all the 4 cases we arrive at the result that $Pr((S_r[i_r] = 0) \wedge (S_r[i_r - 1] = (N-1))|z_r = 0, z_{r-1} = (N-1)) \approx \frac{3}{N^2}$. □

Similar to Lemma 8 and Lemma 9, this lemma also leads to the following corollaries that provide glimpses of j.

Corollary 12. *During RC4 PRGA, if z_{r-1} and z_r are the key-stream bytes for two consecutive rounds $r-1$ and r respectively, then $Pr((S_r[i_r]=0) \wedge (S_r[i_r-1]=(N-1)) \wedge (j_{r-1}=j_r=0) | z_r=0, z_{r-1}=(N-1)) \approx \frac{1}{N^2}$.*

Proof. From the proof of Lemma 10, it is evident that Case 1 corresponds to the desired configuration of $(S_r[i_r]=0) \wedge (S_r[i_r-1]=(N-1)) \wedge (j_{r-1}=j_r=0)$ in addition to satisfying the given condition on two successive key-stream bytes. Hence, the probability associated with the combined event is approximately $\frac{1}{N^2}$. □

Corollary 13. *During RC4 PRGA, if z_{r-1} and z_r are the key-stream bytes for two consecutive rounds $r-1$ and r respectively, then $Pr((S_r[i_r]=0) \wedge (S_r[i_r-1]=(N-1)) \wedge (j_{r-1}=(N-1)) \wedge (j_r=(N-2)) | z_r=0, z_{r-1}=(N-1)) \approx \frac{1}{N^2}$.*

Proof. From the proof of Lemma 10, it is evident that Case 4 corresponds to the desired configuration of $(S_r[i_r]=0) \wedge (S_r[i_r-1]=(N-1)) \wedge (j_{r-1}=(N-1)) \wedge (j_r=(N-2))$ in addition to satisfying the given condition on two successive key-stream bytes. Hence, the probability associated with the combined event is approximately $\frac{1}{N^2}$. □

The following lemma is a straightforward application of the Glimpse correlation to derive a conditional bias of the pseudo-random index j across two rounds.

Lemma 11. *During RC4 PRGA, if j_{r-1} and j_r are the pseudo-random indices for two consecutive rounds $r-1$ and r respectively, then $Pr(j_r=(i_r-z_r) | j_r=0) \approx \frac{2}{N}$.*

Proof. Based on Jenkins' Glimpse correlation, it is known that $Pr(S_r[j_r]=(i_r-z_r)) \approx \frac{2}{N}$. If in round $(r-1)$, the value of pseudo-random index $j_{r-1}=0$ and $S_{r-1}[i_{r-1}+1]=p$, then the new value of $j_r=p$ and after the swap operation $S_r[j_r]=p$ as well. In other words, $S_r[j_r]=j_r$. The desired result immediately follows from this observation. Hence, $Pr(j_r=(i_r-z_r)|j_r=0) \approx \frac{2}{N}$. □ □

Lemma 12. *During RC4 PRGA, if z_{r-1} and z_r are the key-stream bytes for two consecutive rounds $r-1$ and r respectively, then $Pr((S_r[j_{r-1}]=0)|z_{r-1}=(N-1), z_r=0, i_r=0) \approx (\frac{5}{N+2})$.*

Proof. The events that lead to the condition connecting two consecutive key-stream bytes and the index i, have complex inter-dependencies. Due to that reason, the usual assumption of independence of Glimpse correlations arising out of the two events does not hold good. Hence, we prove this result from the first principle by counting events that satisfy the given conditions.

[**Case 1:**] Let us first consider $j_{r-1}=0$ and $S_{r-1}[i_{r-1}+1]=0$.

In this case two sub-cases may arise:

(a) $S_{r-1}[i_{r-1}]=(N-1)$

(b) $S_{r-1}[i_{r-1}] \neq (N-1)$

In sub-case 1(a), the configuration is sufficient to guarantee the given condition of $z_{r-1}=(N-1), z_r=0$ and $i_r=0$. In addition, the expected configuration of $S_r[j_{r-1}]=0$ is satisfied because $S_{r-1}[i_{r-1}+1]=0$ and therefore $j_r=j_{r-1}$ as well. Clearly, the possible number of events satisfying Case 1(a) can be derived uniformly from all possible $(N-2)!$ choices. Let us denote this number by $\mathcal{X}$.

In sub-case 1(b), let us assume $S_{r-1}[i_{r-1}]=p$ where p is any possible byte value for that array position other than $(N-1)$ and 0. If $S_{r-1}[p]=(N-1)$, the configuration is sufficient to guarantee the given condition of $z_{r-1}=(N-1), z_r=0, i_r=0$ and the expected configuration of $S_r[j_{r-1}] = 0$. In this case, the possible number of events satisfying Case 1(b) for an arbitrary byte value p can be derived uniformly from all possible $(N-3)!$ choices. Since p can be chosen in $(N-2)$ different ways, the possible number of choices is $(N-2)! = \mathcal{X}$.

[**Case 2:**] Let us now consider $j_{r-1} = 0$ and $S_{r-1}[i_{r-1}+1] \neq 0$. We consider three sub-cases.

(a) $S_{r-1}[i_{r-1}+1] = 1$

(b) $S_{r-1}[i_{r-1}+1] = (N-1)$ and

(c) $S_{r-1}[i_{r-1}+1] = p$, where $p \notin \{0, 1, (N-1)\}$ is any array byte.

Sub-case 2(a) can't happen as Finney cycles are excluded. Sub-case 2(b) can't satisfy the given condition of $z_{r-1} = (N-1), z_r = 0$ and $i_r = 0$, since to satisfy $z_{r-1} = (N-1)$, $S_{r-1}[i_{r-1}]$ must be 1 which forces z_r to be equal to 1.

In sub-case 2(c), for any randomly chosen permutation of array bytes, depending on the position of output array byte $z_{r-1} = (N-1)$, $S_{r-1}[i_{r-1}]$ gets constrained. Similarly, depending on the position of output array byte $z_r = 0$, $S_r[j_r]$ gets fixed. This implies 3 array byte positions and values can't be chosen freely. Hence, the possible number of options can be at most $(N-3)!$. We already know that p can't have any of the 3 values $\{0, 1, (N-1)\}$. In addition, there would be two more values that p must avoid to satisfy the given condition of $z_{r-1} = (N-1), z_r = 0$ and $i_r = 0$. For example, p must not have the value for which $S_r[j_{r-1}+p]$ is either 0 or $(N-1)$ just before the swap operation. So total number of possible choices available in the case to satisfy the given condition is $(N-5) \cdot (N-3)! = \frac{N-5}{N-2} \cdot \mathcal{X} \approx \mathcal{X}$.

It's important to note that the desired condition of $S_r[j_{r-1}] = 0$ can't be satisfied in this case as $S_r[j_r] \neq 0$ before the swap and $i_r = j_{r-1}$.

[**Case 3:**] Let us next consider $j_{r-1} \neq 0$.

Let us first fix the position of j_{r-1} to an arbitrary position $t \neq 0$. In addition, let us assume $S_{r-1}[t] = p$ where p is one of the possible array byte values. The rest of the analysis of this case is similar to that of sub-case 2(c).

For any randomly chosen permutation of array bytes, depending on the position of output array byte $z_{r-1} = (N-1)$ and the assumed value of p, $S_{r-1}[i_{r-1}]$ gets constrained. Similarly, depending on the position of output array byte $z_r = 0$, $S_{r-1}[j_r]$ gets fixed. This implies 3 array byte positions and values can't be chosen freely. Hence, the possible number of options can be $(N-3)!$. So total number of possible choices available in the case to satisfy the given condition is $N \cdot (N-3)! = \frac{N}{N-2} \cdot \mathcal{X} \approx \mathcal{X}$. Since $t \neq 0$, it can assume in total $(N-1)$ values, resulting in approximately $(N-1) \cdot \mathcal{X}$ possible choices.

To investigate the number possible choices that satisfy the desired condition (i.e., $S_r[j_{r-1}]$) in Case 3 we observe the following 2 configurations.

(1) $S_{r-1}[i_{r-1}+1] = 0$ and

(2) $S_{r-1}[i_{r-1}+1] \neq 0$ and $S_{r-1}[j_{r-1}] = 0$

In configuration (1), $S_r[j_r]$ becomes 0 after the swap operation and since $j_{r-1} = j_r$, the desired condition is satisfied. In configuration (2), the position of the index j changes from round $r-1$ to r. Hence, the desired condition gets satisfied in this configuration as well.

In this configuration, to satisfy the condition $z_r = 0$, $S_r[j_r]$ must have the value of j_r which is also same as j_{r-1}. Also, depending on the position of output array byte $z_{r-1} = (N-1)$ and the required value of j_{r-1} for $S_{r-1}[j_{r-1}]$, the value of $S_{r-1}[i_{r-1}]$ gets constrained. This implies 3 array byte positions and values can't be chosen freely.

Hence, the possible number of options can be at most $(N-3)!$. Clearly j_r can't be 0 as per Case 3. It is also evident that $j_r = (N-1)$ can't satisfy the given condition of $z_{r-1} = (N-1), z_r = 0$ and $i_r = 0$. Hence there can be $(N-2)$ possible values of j_r resulting in $(N-2) \cdot (N-3)! = \mathcal{X}$ possible choices that satisfy the desired condition of $S_r[j_{r-1}] = 0$ in configuration (1).

We now investigate configuration (2). Let us first consider the situation where $S_{r-1}[i_{r-1}] = (N-1)$. Clearly the given condition of $z_{r-1} = (N-1)$ is readily satisfied and to satisfy the other condition $z_r = 0$, one needs to constrain an additional byte value resulting in $(N-3)!$ options and since j_r may assume $(N-2)$ feasible values, the possible number of choices is around $\mathcal{X}$. On the other hand if $S_{r-1}[i_{r-1}] \neq (N-1)$, then also we can use identical argument to arrive at an additional $\mathcal{X}$ choices.

The analysis of 3 possible cases can be now summarized as follows.

Case 1 Possible number of choices that ensure the given condition is $2 \cdot \mathcal{X}$ all of which satisfy the expected condition.

Case 2 Possible number of choices that meet the given condition is $\mathcal{X}$ out of which none satisfies the expected condition.

Case 3 Possible number of choices that ensure the given condition is $(N-1) \cdot \mathcal{X}$ out of which the number of choices that satisfy the expected condition is $3 \cdot \mathcal{X}$.

Therefore, the $Pr((S_r[j_{r-1}] = 0) | z_{r-1} = (N-1), z_r = 0, i_r = 0) \approx \frac{5\mathcal{X}}{(N+2)\mathcal{X}} = (\frac{5}{N+2})$. □

While the above result provides the glimpse of an array byte with a probability that is significantly biased (in the order of $\frac{5}{N}$), the position of the array byte is not immediately available since it is connected to the pseudo-random index j which is unknown. However, we have utilized the same result to uncover the bias on a joint event that provide the glimpse on the position as well as the value of the array byte.

We derive the following results based on Lemma 10 which illustrates that under the given condition, the probability of a joint event connecting the permutation array byte value and its position is much higher than the usual expected value in the range of $\frac{1}{N^2}$. The corollaries are presented in this regard, skipping the proof in a few cases.

Corollary 14. *During RC4 PRGA, if z_{r-1} and z_r are the key-stream bytes for two consecutive rounds $r-1$ and r respectively, then $Pr(S_r[j_{r-1}] = 0 | z_{r-1} = (r-1), z_r = r) \approx (\frac{4}{N})$.*

Corollary 15. *During RC4 PRGA, if z_{r-1} and z_r are the key-stream bytes for two consecutive rounds $r-1$ and r respectively, then $Pr(S_{r-1}[i_{r-1}+1] = 0 | z_{r-1} = (r-1), z_r = r, i_r = 0) \approx (\frac{3}{N})$. In this situation, $Pr(S_r[0] = 0 | z_{r-1} = (r-1), z_r = r, i_r = 0) \approx (\frac{2}{N})$ and $Pr(S_r[N-1] = (N-1) | z_{r-1} = (r-1), z_r = r, i_r = 0) \approx (\frac{2}{N})$*

Corollary 16. *During RC4 PRGA, if z_{r-1} and z_r are the key-stream bytes for two consecutive rounds $r-1$ and r respectively, then $Pr((S_r[j_{r-1}] = 0) \wedge (j_{r-1} = 0) | z_{r-1} = (N-1), z_r = 0, i_r = 0) \approx (\frac{2}{N})$.*

Proof. From the proof of Lemma 10, it is apparent that Case 1 and its two sub-cases 1(a) and 1(b) correspond to the desired configuration of $(S_r[i_{r-1}] = 0) \wedge (j_{r-1} = 0)$. Hence, the probability associated with the combined event is approximately $\frac{2}{N}$. □

From the proof of the above mentioned corollary it's evident that the value of the probability would remain unchanged even when we add a third condition resulting in a desired configuration of $(S_r[j_{r-1}] = 0) \wedge (j_{r-1} = 0) \wedge (j_r = 0)$.

5 Conclusion and future research

In this paper, we have investigated the close relationship between the key-stream biases and the glimpses of certain permutation array bytes as well as that of the pseudo-random index, in the long term evolution of RC4 PRGA. We have used three different lenses. First, we have thoroughly analyzed the Fluhrer-McGrew digraph biases [3] to identify the source configurations that are the root causes behind the observed positive and negative biases. Perhaps surprisingly, in spite of these biases being used in some of the celebrated attacks, the fundamental mechanisms were not studied in a structured manner so far, apart from the high level reasoning provided by Fluhrer and McGrew for only three scenarios in their original paper [3, Section 4]. We have also pointed out and solved the gaps that were present in their reasoning. Further, we establish a series of glimpse correlations based on the digraph biases.

In the second lens, we have theoretically proved four dominant “lag-one digraph" biases (i.e., biases of alternate key-stream bytes) out of which two are reported in this paper for the first time. For one of the “lag-one digraph" biases that was first reported by Sen Gupta et. al. [13], we have proved a sharper bias value and have given a proof that uncovers the underlying source configuration in a more accurate manner. Subsequently, we have proved a number of glimpse correlations based on the source configurations of these “lag-one digraph" biases.

Finally, our third lens focuses on two sub-areas. The first sub-area is about the original Glimpse correlations, discovered by Jenkins [5], in which we carry out a fine-grained analysis of the result to demonstrate that the results need significant revision for cases where the two indices are same $(i = j)$ and present a *Revised Glimpse Theorem* that more accurately describes the variations of the result for even and odd values of the sequential index i. Further, we also prove new glimpse results pertaining to the pseudo-random index j under certain given conditions. In the second sub-area, we derive a set of new glimpse results based on consecutive key-stream byte-values and for the first time prove a glimpse result on permutation array byte with a value of the order of $\frac{5}{N}$, which is to the best of our knowledge, sharper than any glimpse correlation that have been reported so far in RC4 literature.

Essentially, in our view, these three lenses now convincingly establish that the biases and glimpses are two sides of the same coin. While biases are like the *spectre* behind the glimpses, the omni-presence of glimpse correlations in the long-term evolution of RC4, make them appear as if the *glimpses are forever*. It is evident that these biases can be exploited to improve the practical attacks as mentioned in the works like [1, 4, 11, 14, 15]. Some of the areas that may be investigated further in the future include - creating a single statistical framework where multiple glimpses and biases can be combined together without introducing significant error, improving the existing cryptanalytic attacks by optimally utilizing the information from a known set of key-stream bytes and finally come up with a generic mathematical framework connecting the biases and glimpses based on the operations performed in each step of RC4 PRGA.

The scope of this paper does not include how the results may be exploited in exact improvements in cryptanalysis. Rather this is an effort to provide theoretical justifications of some very old and well known biases and providing some new and significant biases. In RC4 literature, the attacks are mostly based on the biases observed and there are several well known publications, for example, the very recent one [2] even after RC4 is deprecated, and the ones [10, 13] that actually helped in deprecating RC4 in that direction. That is, such works mainly concentrate on finding biases (but not providing attacks) that can be used in attacks later. This paper falls in that direction and at the same time settles theoretical issues using elementary combinatorial techniques that remained unproved for more than two decades. We believe that identification of the new biases and improved glimpses should reduce the complexity in the existing attacks. In fact, the lag-one digraph biases were never exploited in any earlier attack. Those can be combined with any of the earlier attacks (that were based on Fluhrer-McGrew [3] or Mantin's digraph repetition bias [9]) and the predictability of the key-stream bytes around certain index positions should improve. Additionally, stronger glimpse results combined with additional biases should sharpen the existing attacks. These may be considered in future direction of research.

References

[1] N. AlFardan, D. Bernstein, K. Paterson, B. Poettering and J. Schuldt. On the security of RC4 in TLS. USENIX 2013. Published online at: http://www.isg.rhul.ac.uk/tls/, pp. 305–320, 2013.

[2] R. Bricout, S. Murphy, K. G. Paterson, T. van der Merwe. Analysing and exploiting the Mantin biases in RC4. Designs Codes and Cryptography, 86:743-770, 2018.

[3] S. R. Fluhrer and D. A. McGrew. Statistical Analysis of the Alleged RC4 Keystream Generator. FSE 2000. LNCS, pp. 19-30, Vol. 1978, 2000.

[4] T. Isobe, T. Ohigashi, Y. Watanabe and M. Morii. Full plaintext recovery attack on broadcast RC4. FSE 2013. LNCS 8424, pp. 179–202.

[5] R. J. Jenkins Jr. ISAAC and RC4. Published online at: https://burtleburtle.net/bob/rand/isaac.html, 1993-1996.

[6] S. Jha, S. Banik, T. Isobe and T. Ohigashi. Some Proofs of Joint Distributions of Keystream Biases in RC4. INDOCRYPT 2016, LNCS 10095, pp. 305-321.

[7] S. Maitra and S. SenGupta. New Long-Term Glimpse of RC4 Stream Cipher. ICISS 2013, LNCS 8303, pp. 230-238, Springer-Verlag, 2013.

[8] I. Mantin and A. Shamir. A Practical Attack on Broadcast RC4. In proceedings of FSE 2001, Lecture Notes in Computer Science, Springer, Vol. 2355, pp. 152–164, 2001.

[9] I. Mantin. Predicting and Distinguishing Attacks on RC4 Keystream Generator. EUROCRYPT 2005, pages 491–506, vol. 3494, Lecture Notes in Computer Science, Springer.

[10] K. G. Paterson, B. Poettering and J. C. N. Schuldt. Big Bias Hunting in Amazonia: Large-scale Computation and Exploitation of RC4 Biases. ASIACRYPT 2014. LNCS, Part 1, pp. 398–419, Vol. 8873, 2014.

[11] K. G. Paterson, J. Schuldt and B. Poettering. Plaintext Recovery Attacks Against WPA/TKIP. FSE 2014, LNCS 8540, pp. 325–349, 2014.

[12] A. Popov. Prohibiting RC4 Cipher Suites. Request for Comments: 7465. ISSN: 2070-1721. 2015. Available at https://tools.ietf.org/html/rfc7465.

[13] S. SenGupta, S. Maitra, G. Paul, S. Sarkar. (Non–)Random Sequences from (Non–)Random Permutations – Analysis of RC4 stream cipher. Journal of Cryptology, 27(1):67–108, 2014

[14] P. Sepehrdad, S. Vaudenay, and M. Vuagnoux. Statistical Attack on RC4 - Distinguishing WPA. EUROCRYPT 2011. LNCS pp. 343–363, Vol. 6632, 2011.

[15] M. Vanhoef and F. Piessens. All Your Biases Belong to Us: Breaking RC4 in WPA-TKIP and TLS. USENIX 2016, pp. 1–16, 2016. Available at https://www.rc4nomore.com/vanhoef-usenix2015.pdf