cryptography

MDPI

Article

New Weak Keys with Parity Patterns in the RC4 Stream Cipher

Evaristo José Madarro-Capó [1], Carlos Miguel Legón-Pérez [1], Guillermo Sosa-Gómez [2,*] and Omar Rojas [2]

[1] Instituto de Criptografía, Facultad de Matemática y Computación, Universidad de la Habana, Habana 10400, Cuba; evaristo.madarro@matcom.uh.cu (E.J.M.-C.); clegon58@gmail.com (C.M.L.-P.)
[2] Facultad de Ciencias Económicas y Empresariales, Universidad Panamericana, Álvaro del Portillo 49, Zapopan 45010, Jalisco, Mexico; orojas@up.edu.mx
* Correspondence: gsosag@up.edu.mx; Tel.: +52-3313682200

Abstract: The RC4 cryptographic algorithm is the most extensively studied stream cipher of the past two decades. This extensive research has resulted in numerous publications, many of which identify various vulnerabilities. Although these vulnerabilities do not preclude the correct use of the algorithm, they complicate its practical implementation. In this paper, we present a novel weakness in the RC4 cipher. Our findings indicate that, for input keys exhibiting certain patterns, the parity of the values in the output permutation of the KSA can be determined with high probability from the parity of its position in the output permutation. Furthermore, the use of keys with these specific patterns leads to noticeable distortions in several bytes of the RC4 output.

Keywords: RC4; weak keys; stream cipher; cryptography; cybersecurity

Citation: Madarro-Capó, E.J.; Legón-Pérez, C.M.; Sosa-Gómez, G.; Rojas, O. New Weak Keys with Parity Patterns in the RC4 Stream Cipher. *Cryptography* **2024**, *8*, 54. https://doi.org/10.3390/cryptography8040054

Academic Editor: Josef Pieprzyk

Received: 15 October 2024
Revised: 20 November 2024
Accepted: 25 November 2024
Published: 27 November 2024

1. Introduction

With the rapid advancement of information technologies, encryption methods play a crucial role in ensuring information security [1]. Stream ciphers are a type of symmetric encryption that are essential for high-volume data transmission, storage, and low-resource applications [2,3]. The construction of stream ciphers is currently a prominent research topic in the field of cryptography [2,3]. This interest is primarily driven by the increasing complexity of application environments for stream ciphers, which demand enhanced performance in terms of speed, data handling, and memory usage [2]. Consequently, developing new designs that are well suited for these environments presents a continuous challenge. Examples of such environments include electronic commerce [4], cloud computing [5,6], big data [7,8], instant messaging [9], wireless data connections [10], streaming services [11,12], and the Internet of Things [13].

Nowadays, stream ciphers are distinguished by their straightforward hardware and software implementation, as well as their rapid encryption and decryption capabilities [2]. Among contemporary designs, the RC4 stream cipher is notable for its exceptional speed and ease of implementation [14]. These attributes make it an ideal choice for low-power devices and applications that necessitate high-speed data transmission. The RC4 cipher is one of the most widely used stream ciphers, nowadays supported by the TLS 1.2 protocol [15], Oracle Advanced Security 11g [16], and Microsoft [17] systems, among others, and has garnered significant attention in the cryptological literature over the past two decades [14]. Despite years of cryptanalysis, the strengths and weaknesses of the RC4 remain a topic of great interest within the community [3]. Although several stream ciphers have been proposed since the introduction of the RC4, it has been the most extensively studied stream cipher in recent years due to its simplicity, ease of implementation, byte-oriented structure, speed, and efficiency, leading to numerous publications [18–28]. Many of these studies have identified vulnerabilities that complicate its practical use. However, none of the attacks against the RC4 has managed to compromise the correct implementation of the cipher, and it remains a focus of interest for the cryptographic community [3,14,29].

In general, weaknesses in the RC4 can be categorized into three major groups: weak keys, methods for recovering the internal state (including the key and permutation) from output bits, and distinguishing distortions in the output bits of the RC4 [14]. The presence of weak keys in the RC4 has prompted the publication of several studies in the literature [30–36]. In this paper, we describe the existence of new weak input keys that follow a specific pattern, which allows for the determination of the parity of the permutation values in the RC4 with high probability. Furthermore, the use of these keys results in distortions in the initial output bytes, which is an undesirable characteristic.

In this paper, a new weakness in the RC4 cipher is presented and its cryptanalytic significance is described. Through the analysis of the RC4 key scheme, a new set of weak keys was identified that allows the parity bits of each permutation component to be determined with high probability. This probability was estimated experimentally and compared with a theoretical approximation based on the conditions derived from the operation of the RC4 through the use of these keys.

This undesirable operation causes a bias in the parity bits of the first output bytes of the RC4. The probability of the occurrence of this bias in the first output bytes of the RC4 was estimated experimentally. In addition, a theoretical approximation was made based on the distribution of the permutation components after the execution of the RC4 key scheme.

2. Motivation

The RC4 is one of the most widely used stream cipher algorithms in the history of cryptography. Today, it continues to be studied despite its vulnerabilities and age, mainly because studying it allows us to understand how the first stream ciphers were designed and their impact on the evolution of modern cryptography. Many of these vulnerabilities in the RC4 can be avoided if the first bytes of the output are omitted and not used, while others, such as the one presented in this work, can be overcome by avoiding the use of weak keys [14,37,38].

On the other hand, modifications have been proposed on many occasions that have resulted in new cipher variants, such as the Spritz [39] and others [40,41]. In [41], it was experimentally illustrated that these modifications, far from increasing security, can decrease it. In this way, it is evident that the study of the RC4 allows us to evaluate new vulnerabilities in its modifications or other ciphers that base their design on the structure of this cipher. The study of these vulnerabilities provides important lessons on how to identify and mitigate potential flaws in modern cryptographic systems.

The RC4 continues to motivate the creation of new lines of research today [42–44]. Furthermore, this cipher is a good choice for measuring the effectiveness of cryptanalysis methods that look for correlations between the bits of the internal state and the bits of the outputs [45,46], or for verifying the performance of hardware or software schemes that make use of cryptography [47–50].

In summary, the RC4 remains a topic of interest due to its academic and cryptographic value, its role in the development of cryptography, and its presence in legacy systems, which makes it a useful reference for those studying the security and design of cryptographic algorithms.

3. Description of the RC4

The RC4 is extremely fast and exceptionally simple, making it ideal for protecting large numbers of data both at rest and in transit. The internal state of the RC4 consists of a permutation of dimension $N = 2^n$, which allows it to store all possible elements of n bits, and two indices i and j, which act on this permutation. Its operation is based on the principle of random shuffling [51]. In practical applications, $n = 8$, which provides the RC4 with a sufficiently large internal state of $\log_2((2^8)! \cdot (2^8)^2) \approx 1700$ bits [14].

The RC4 comprises two components, as illustrated in Algorithms 1 and 2. The first component is the Key Scheme Algorithm (KSA), which accepts a secret key, denoted by K, as input and combines it with a permutation, S, of order N. The second component is the Pseudorandom Generation Algorithm (PRGA), which utilizes the output permutation

from the KSA to generate a sequence of output pseudorandom numbers. The KSA consists of N steps. S is initialized with the identity permutation and the indices i and j at 0. Then, using a random shuffling mechanism, the algorithm modifies the initial permutation by moving the indices i and j through the permutation, updating j based on the values of the secret key K.

The PRGA reinitializes the indices i and j to 0 and then iterates q times, performing four straightforward operations. These operations are as follows: increment i by one in each iteration up to q, update j pseudorandomly, swap the values of the permutation indexed by i and j, and return as output the value of the permutation corresponding to the position $S[i] + S[j]$. The value corresponds to the number of bytes to be generated.

Algorithm 1 RC4 Key Scheme Algorithm (KSA)

for $i = 0 \rightarrow N - 1$ **do**
    $S[i] \leftarrow i$
end for
$j \leftarrow 0$
for $i = 0 \rightarrow N - 1$ **do**
    $j \leftarrow (j + S[i] + K[i \mod l]) \mod N$
    Swap $S[i]$ and $S[j]$
end for

For each step $r = 0, 1, 2, \ldots, N - 1$ of the KSA in the RC4, the indices i_r and j_r are denoted. The permutation before and after the swap in the KSA is denoted by S_{r-1} and S_r, respectively. In the case of the PRGA, at each step $r = 0, 1, 2, \ldots, q$, the permutation before and after the swap is denoted as S^N_{r-1} and S^N_r, respectively. The output element of the PRGA is denoted as $Z_r = S_r[t_r]$, where $t_r = S_r[i_r] + S_r[j_r]$. The permutation after the KSA but before the PRGA is denoted as S_N. It is denoted by S_0, the identity permutation, and S^N_0, the permutation at the beginning of the PRGA. It is also denoted by l, the length of the secret key K. All arithmetic operations in the context of the RC4 are considered modulo N unless otherwise specified.

Algorithm 2 RC4 Pseudorandom Generator Algorithm (PRGA)

$i \leftarrow 0$
$j \leftarrow 0$
while Generating Output **do**
    $i \leftarrow (i + 1) \mod N$
    $j \leftarrow (j + S[i]) \mod N$
    Swap $S[i]$ and $S[j]$
    Output $S[(S[i] + S[j]) \mod N]$
end while

4. Previous Results on Weak Keys in the RC4

There is a substantial body of literature on the RC4 that identifies several weaknesses [14]. While these weaknesses do not compromise the integrity of the cipher, they do restrict its practical applications. Among these vulnerabilities are those associated with the classes of weak keys that propagate statistical patterns into the bits of the output sequences. The following are some studies related to weak keys in the RC4.

In [52], a type of weak keys in the RC4 stream cipher, characterized by repeated patterns that remain invariant regardless of key length, is described. The authors suggest that the root of this issue lies in the simplicity of the key-dependent permutation initialization process. In [31], several vulnerabilities in the RC4 key scheme are discussed, and a new category of weak keys is identified. Knowledge of a limited number of bits of the key allows for the determination of several bits of the internal state and some output bits with a non-negligible probability. These weak keys are utilized in the development of

novel distinguishers for the RC4, thereby enabling the execution of related-key attacks with practical applicability. In reference to the cited source [53], it is demonstrated that, when a key is defined as a sequence of values, represented by the notation $K[0], \ldots, K[N]$, with the condition that $K[0] + K[1] = 0 (\text{mod } 256)$, the initial output byte generated by the RC4 is $K[2] + 3 (\text{mod } 256)$, with a probability that exceeds the expected value under a random distribution.

In [32], a distinctive internal state is utilized to generalize a class of weak keys, also referred to as predictive states. Among the findings presented is the probability that the use of weak keys within this generalized class will disclose information about the bytes of the secret key. The authors demonstrate that, compared to the keys defined by Roos [10], this class of weak keys offers a greater amount of information. In their paper, Nagao and colleagues [33] introduced a novel class of weak keys, expanding upon the findings of Teramura [32]. The weak key space they defined is approximately eight times larger than the one presented by Teramura [32]. In [37,38], a theoretical estimate was provided regarding the number of initial bytes that must be discarded to avoid the distinguishers presented in these studies. Based on the findings from these two works, it is recommended that the first 512 bytes be discarded.

5. Parity Pattern in Weak Keys

The majority of the research on weak keys focuses on the existence of inputs that exhibit probabilistic patterns in the output bits of the RC4. Furthermore, they outline the implications of utilizing weak keys in the distribution of the permutation elements throughout and following the KSA. In a similar manner, a new set of weak keys that adhere to a specific pattern, designated as the parity pattern, is introduced.

Definition 1. *A key, K, has a parity pattern if its length, l, is greater than two bytes and l is even, and it is expressed as* $K = \{1, 0, I, P, I, P, \ldots\}$*, where I means that the corresponding value of the key is an odd number and P means that it is an even number.*

This specific type of defined weak key allows for a high-probability determination of the parity of the elements in the permutation following the KSA and causes a significant distortion in the parity of the initial output bytes, as will be demonstrated below.

6. Biases in the Output Permutation of the KSA Due to the Use of Keys with Parity Patterns

In the KSA, operations dependent on the input key are performed to provide randomness to the initial permutation S_0 using the *random shuffles* principle [51]. Initially, it holds that

$$S_0[i] \to P \text{ if } i \to P \text{ and } S_0[i] \to I \text{ if } i \to I \ \ \forall i < N,$$

where $x \to P$ means that x is even and $x \to I$ means that x is odd.

A notable outcome is the identification of the parity of the values associated with the index j_r if the key used has the pattern established in Definition 1, across the r rounds of the KSA. During the KSA, it is holds that

$$j_r \to P \text{ if } i_r \to P \text{ and } j_r \to I \text{ if } i_r \to I, \text{ with } 1 < r < N,$$

which is formalized below.

Proposition 1. *Given the use of a key in the RC4 with the parity pattern established in Definition 1, it follows that* $j_0 = j_1 = 1$*, and, for* $r \geq 2$*,* $j_r \to P$ *if* $i_r \to P$ *and* $j_r \to I$ *if* $i_r \to I$*.*

Proof. Since $K[0] = 1$ and $K[1] = 0$, in the first round of the KSA with $r = 0$, we have $j_0 = 0 + K[0] + S_0[0] = 1$. In the second round with $r = 1$, we obtain $j_1 = 1 + K[1] + S_1[1] = 1$ since, after the swap in the first round, the value of the second position of the permutation becomes $S_1[1] = 0$. Therefore, in the first two steps of the KSA, it is required that $j_0 = j_1 = 1$.

Then, for $r \geq 2$, there are two cases based on the parity of i_r. If $i_r \to P$, then $K[i_r \mod l] \to I$ and $S_r[i_r] \to P$. On the other hand, if $i_r \to I$, then $K[i_r \mod l] \to P$ and $S_r[i_r] \to I$. Thus, as $j_r = j_{r-1} + K[i_r \mod l] + S_r[i_r]$, then $j_r \to P$ if $i_r \to P$ and $j_{r-1} \to I$, while $j_r \to I$ if $i_r \to I$ and $j_{r-1} \to P$.

Since for $r = 0$ and $r = 1$ it holds that $j_0 = j_1 = 1$, then for $r \geq 2$ it follows that $j_r \to P$ if i_r is even and $j_r \to I$ if i_r is odd. It is worth noting that, in the case of the KSA, in each round r, it holds that $r = i$. □

The parity of each value of the KSA output permutation is influenced by the parity behavior of the j_r index values. This result is formalized below.

Proposition 2. *If the key used in the RC4 has the pattern specified in Definition 1, then the permutation S_N after the KSA satisfies $S_N[x] \to P$ or $S_N[x] = 1$ if $x \to P$ and $S_N[x] \to I$ or $S_N[x] = 0$ if $x \to I$, with $0 \leq x < 256$.*

Proof. In the first step of the KSA, it is necessary that $i = 0$; thus, the value in $S_0[0]$ is swapped for the value in $S_0[1]$ because $i_0 = 0$ and $j_0 = 1$. This results in the permutation $S_1 = \{1, 0, 2, 3, \ldots, 255\}$. In the second step, no swap is made on the permutation since $i_1 = 1$ and $j_1 = 1$. In the third step, $i_2 = 2$ and $j_1 = 1$. Moreover, $S_2[i_2]$ is even and $K[i_2]$ is odd. In this way, there are two cases: j_2 will be 0 or it is an even number.

- Case $j_2 = 0$: They swap $S_2[i_2 = 2]$ and $S_2[j_2 = 0]$ and the permutation becomes $S_2 = \{2, 0, 1, 3, 4, \ldots, 255\}$.
- Case $j_2 \neq 0$: $S_2[i_2 = 0]$, which is an even number, is swapped for the even number $S_2[j_2 \to P]$.

It can be noted that whenever a key with the described pattern is used, it holds that $S_0[1] = S_1[0] = S_2[0] = 1$ and $S_0[0] = S_1[1] = S_2[1] = 0$.

In the fourth step $r = 3$ of the KSA, an odd number is swapped for another odd number or an odd number for the number 0. That is, with $i_3 = 3$, j_3 will be odd, because $S_3[i_3 = 3]$ is odd, $K[i_3 = 3]$ is even, and j_2 is even. Then, based on the result regarding the distribution of the values of j obtained through Proposition 1, it follows that $S_N[x] \to P$ or $S_N[x] = 1$ if $x \to P$ and $S_N[x] \to I$ or $S_N[x] = 0$ if $x \to I$, with $0 \leq x < 256$. □

In the following example, this pattern is repeated consecutively. The output permutation of the KSA will fulfill $S_N[i] \to P$ or $S_N[i] = 1$ if i is even and $S_N[i] \to I$ or $S_N[i] = 0$ if i is odd.

Example 1. *Using the key $K = \{1, 0, 51, 124, 85, 140\}$, with $l = 6$, the first five steps of the RC4 KSA are outlined. Tables 1–6 show the permutation in the first five steps of the KSA.*

Permutation S_0

Table 1. First five elements of the initial permutation.

0	1	2	3	4	5	...	254	255

Step $r = 0$
$i_0 = 0$ and $j_0 = 0 + S_0[0] + K[0] = 1$
Swap ($S_0[0]$, $S_0[1]$)

Table 2. Permutation resulting from the first swap.

1	0	2	3	4	5	...	254	255

Step $r = 1$
$i_1 = 1$ *and* $j_1 = 1 + S_1[1] + K[1] = 1$
No swap is made.

Table 3. Permutation resulting from the second step (without swap).

1	0	2	3	4	5	...	254	255

Step $r = 2$
$i_2 = 2$ *and* $j_2 = 1 + S_2[2] + K[2] = 54$
Swap ($S_2[2]$, $S_2[54]$)

Table 4. Permutation resulting from the second swap.

1	0	54	3	4	5	...	254	255

Step $r = 3$
$i_3 = 3$ *and* $j_3 = 54 + S_3[3] + K[3] = 181$
Swap($S_3[3]$, $S_3[181]$)

Table 5. Permutation resulting from the third swap.

1	0	54	181	4	5	...	254	255

Step $r = 4$
$i_4 = 4$ *and* $j_4 = 181 + S_4[4] + K[4] = 14$
Swap($S_4[4]$, $S_4[14]$)

Table 6. Permutation resulting from the fourth swap.

1	0	54	181	14	5	...	254	255

In Table 7, the resulting permutation S_N at the end of the KSA is shown, and it can be observed that odd positions contain odd values and even positions correspond to even values, with the exception of the values 0 and 1, which are found in crossed columns.

Table 7. Permutation S_N after KSA.

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	17	4	227	14	89	166	173	106	121	190	95	28	133	32	63
1	164	61	84	103	176	83	48	119	130	127	10	197	152	13	26	187
2	154	237	34	209	30	75	116	77	148	105	46	139	254	167	160	229
3	170	175	172	91	228	165	208	223	74	43	142	85	238	129	24	57
4	96	177	92	135	230	221	88	15	86	137	58	185	78	253	128	225
5	156	151	60	113	132	191	94	97	206	79	8	87	22	131	192	99
6	108	205	98	249	200	143	50	169	252	199	76	241	100	123	234	145
7	250	49	138	251	182	93	174	39	144	19	194	7	126	233	224	159
8	40	3	20	9	114	125	226	55	210	53	44	23	178	111	242	29
9	124	101	158	181	82	217	186	235	212	157	72	25	246	59	244	255
10	6	155	218	161	216	231	52	193	38	201	68	141	150	67	62	153
11	118	69	102	183	188	207	110	239	248	11	64	203	16	211	120	45
12	180	115	196	0	202	51	56	73	140	179	240	219	36	37	42	65
13	12	215	122	195	112	247	204	107	54	163	134	21	220	189	214	33
14	162	81	198	245	66	5	104	117	232	213	222	149	18	243	146	109
15	70	41	236	27	184	147	80	31	90	171	168	71	2	47	136	35

In the previous table, each element a_{xy} of the matrix corresponds to the value $S[(x * 16) + y]$ of the permutation, taking x as the row and the value of y as the column.

According to these results, if i is even, then $S_N[i]$ takes values in the set $\{1, 2, 4, 6, 8, \ldots, 254\}$ of 128 elements, which groups the even elements less than $N = 256$ plus 1 and does not include zero. If i is odd, then the value of $S_N[i]$ is found in the set $\{0, 3, 5, 7, 9, \ldots, 255\}$ of 128 elements that groups the odd elements plus 0, excluding 1. The use of this type of keys allows the parity of any element of S_N to be determined with high probability.

7. Biases in the First Output Bytes of the PRGA from the Use of Keys with Parity Patterns

Once the KSA is completed, the distribution of elements in the output permutation S_N causes certain biases to appear in the output bytes of the PRGA. This bias allows the parity of the initial output bytes with high probability to be determined with high probability, thereby allowing the extraction of several of the first output bits with similarly high probability. The existence of a bias known as parity bias in the initial output bytes of the PRGA largely depends on the distribution of the values 0 and 1 within the permutation S_0^N. The positioning of these values in S_0^N directly influences the likelihood of bias occurrence in the first output bytes of the PRGA. The following experiments were conducted using input keys that exhibit the parity pattern.

To illustrate the influence on the initial output bytes, experiments were conducted using 1000 input keys with a specific parity pattern. Figure 1 displays the observed deviation of the number X of even bytes with respect to its expected value $E(X) = 500$.

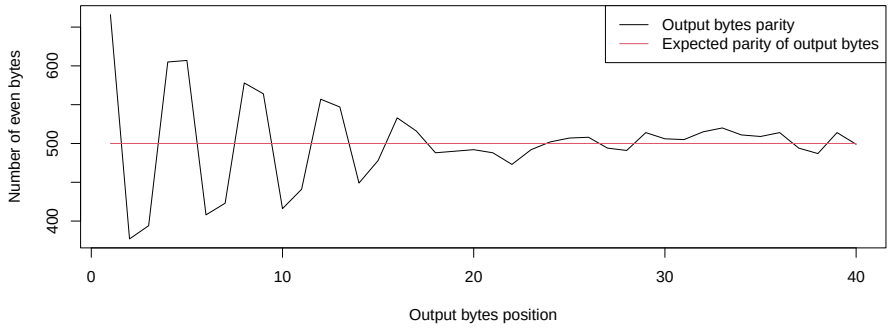

Figure 1. Parity distribution of the first 16 output bytes.

A significant factor influencing the deviation of the parity of the initial output bytes of the RC4, compared to what is expected under random conditions, is the high frequency of the events $S_0^N[0] = 1$ and $S_0^N[1] = 0$ relative to the rest of the possible events. Figure 2 illustrates the distribution of the value 1 over the possible 256 indices of the permutation S_0^N, highlighting the high frequency observed for the event $S_0^N[0] = 1$.

Something similar occurs with the distribution of the element 0 in the second position of the permutation; it appears with a significantly higher frequency than the rest of the values. Figure 3 illustrates the distribution of the value 0 across the 256 indices of the permutation S_0^N, with a notably high frequency observed for the event $S_0^N[0] = 1$.

As observed in both graphs, excluding the first two positions, the frequency of occurrence for both values tends to increase slightly as the value of the permutation index increases. This suggests a higher probability that the values 0 and 1 will appear in later positions, particularly above the index $N/2$, if they do not end up in these first two positions. These results suggest that the initial output bytes of the PRGA may be biased by the positions of the values 0 and 1 in the output permutation of the KSA. In this way, it is essential to consider the cases $S_0^N[0] = 1$, $S_0^N[0] \neq 1$, $S_0^N[1] = 0$ and $S_0^N[1] \neq 0$ when analyzing the PRGA.

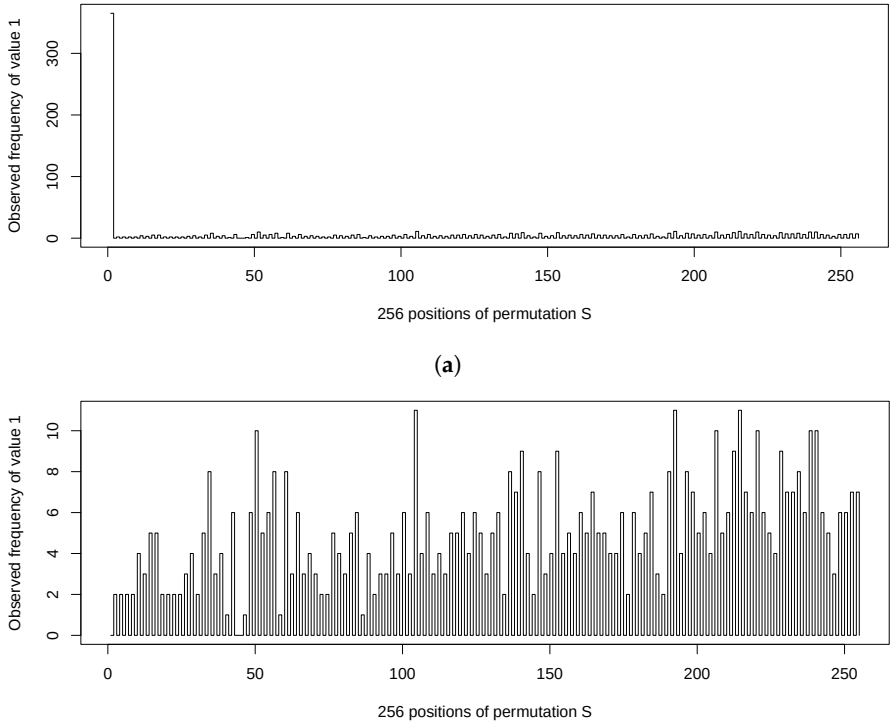

Figure 2. Distribution of the value 1 over the indices of the permutation. (**a**) All positions. (**b**) Without the first position.

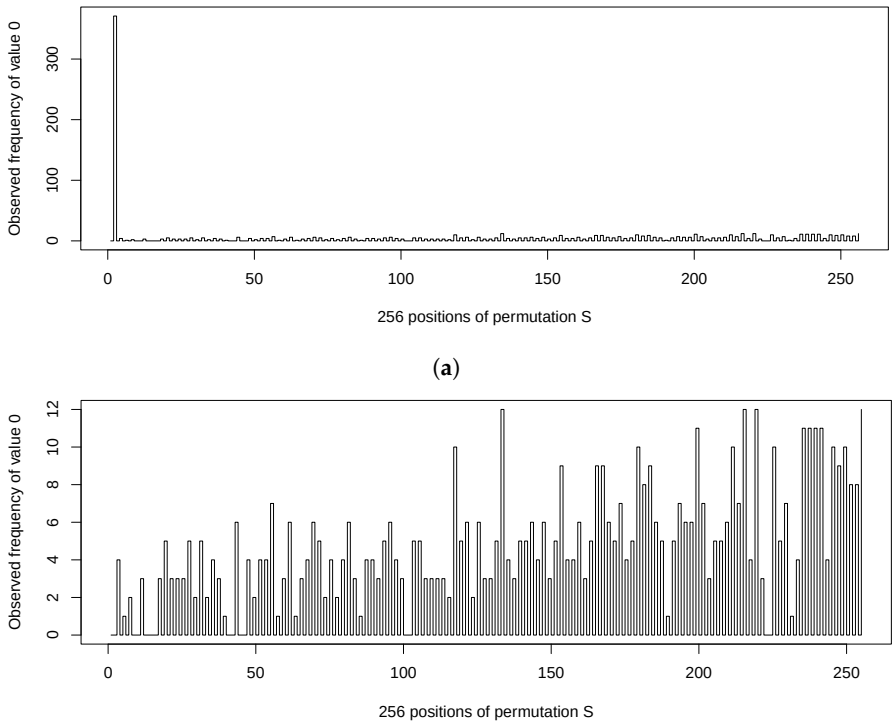

Figure 3. Distribution of element 0 over the permutation indices. (**a**) All positions. (**b**) Without the second position.

The probabilities $P\left(S_0^N[0] = 1\right)$ and $P\left(S_0^N[1] = 0\right)$ depend on j not taking the values 0 and 1 after the second step. Assuming that j is a variable that takes values randomly

and that a key with the defined parity pattern is used, a theoretical approximation for the value of the probability $P(j_r \neq x)$, that, in a round r of the KSA j_r, does not assume a predetermined value x, is

$$P(j_r \neq x) \approx \left(1 - \frac{1}{Q}\right),\ 0 \leq r, x < N, \tag{1}$$

where $Q = N/2$. Then,

$$P\left(S_0^N[0] = 1\right) \approx \prod_{r=2}^{N} P(j_r \neq 0) \approx \left(1 - \frac{1}{Q}\right)^{\frac{N-2}{2}} = \left(1 - \frac{1}{128}\right)^{126} = 0.369323. \tag{2}$$

On the other hand,

$$P\left(S_0^N[0] \neq 1\right) = 1 - P\left(S_0^N[0] = 1\right) = 0.63, \tag{3}$$

while

$$P\left(S_0^N[1] = 0\right) \approx P\left(S_0^N[0] = 1\right). \tag{4}$$

The probabilities $P(S_0^N[1] = 0)$ and $P(S_0^N[0] = 1)$ are affected by the parity of the index j during the KSA. From Proposition 1, it is known that the parity of index j depends on the parity of round r. The following shows for each case $S_0^N[1] = 0$ and $S_0^N \neq 0$ the impact of utilizing the defined weak keys on the first bytes of the RC4 output.

7.1. Case $S_0^N[1] = 0$

The initial values of the variables i and j in the PRGA are $j_0 = 0$ and $i_0 = 1$. When using a key with a parity pattern, it holds that $S_0^N[0] = 1$ or $S_0^N[0]$ is an even number. Before issuing the first output, the PRGA performs several operations. Among these is the swap of $(S_0^N[1], S_0^N[0])$, performed so that $S_1^N[1]$ will be 1 or an even number. Then, the first output byte will be $Z_0 = S_1^N[S_1^N[0] + S_1^N[1]]$ and, as $S_1^N[0] = 0$, then $S_1^N[0] + S_1^N[1] = S_1^N[1]$ and $Z_0 = S_1^N[S_1^N[1]]$. The first output of the PRGA will be 1 or an even number. It can be noted that if $S_0^N[0] = 1$, then after the swap it holds that $S_1^N[S_1^N[1]] = 1$ and therefore $Z_0 = 1$. If $S_0^N[0] \neq 1$, then Z_0 will be 1 if it is jointly satisfied that $S_0^N[0] = x$ and $S_0^N[x] = 1$, which implies that $S_0^N\left[S_0^N[0]\right] = 1$.

It is also necessary to take into account the cases $S_0^N[0] = 1$ and $S_0^N[0] \neq 1$. However, based on the results shown in Figure 1, it is known that the event $S_0^N[0] = 1$ has a high probability of occurrence.

Some Theoretical Approaches to Z_0 Given $S_0^N[1] = 0$

For the first output byte Z_0, the following approximation of its theoretical value is obtained:

$$P\left(Z_0 = 1 \mid S_0^N[1] = 0\right) \approx P\left(S_0^N[0] = 1 \mid S_0^N[1] = 0\right) + P\left(S_0^N\left[S_0^N[0]\right] = 1 \,\middle|\, S_0^N[1] = 0\right). \tag{5}$$

Starting from the premise of independence between the events $S_0^N[0] = 1$ and $S_0^N[1] = 0$, it follows that

$$P\left(S_0^N[0] = 1 \mid S_0^N[1] = 0\right) = P\left(S_0^N[0] = 1\right). \tag{6}$$

On the other hand, the event $S_0^N[S_0^N[0]] = 1$ is not influenced by the event $S_0^N[1] = 0$; thus,

$$P\left(S_0^N[S_0^N[0]] = 1 | S_0^N[1] = 0\right) = P\left(S_0^N\left[S_0^N[0]\right] = 1\right). \tag{7}$$

A theoretical approximation of the value of $P(S_0^N[S_0^N[0]] = 1)$ can be obtained through Theorem 4.3 in [34], yielding $P(S_0^N[S_0^N[0]] = 1) \approx 0.136776$. In Table 8, the results obtained experimentally for these probabilities are shown in the case of $S_0^N[1] = 0$.

Table 8. Experimentally observed probabilities for $S_0^N[1] = 0$.

Event E	$P(E\|S_0^N[1] = 0)$	$P(E)$
$S_0^N[0] = 1$	0.36579	0.36736
$S_0^N[S_0^N[0]] = 1$	0.13645	0.13666
$Z_0 = 1$	0.497757	0.191

Based on the above, it follows that

$$P\left(Z_0 = 1 \,\middle|\, S_0^N[1] = 0\right) \approx$$
$$P\left(S_0^N[0] = 1\right) + P\left(S_0^N\left[S_0^N[0]\right] = 1\right) \approx$$
$$0.3686 + 0.136776 = 0.505376$$

and

$$P\left(Z_0 \neq 1 \,\middle|\, S_0^N[1] = 0\right) = 1 - P\left(Z_0 = 1 \,\middle|\, S_0^N[1] = 0\right) = 1 - 0.505376 = 0.4946.$$

Table 9 presents the experimentally observed probability of adhering to the fixed parity bias given the case $S_0^N[1] = 0$ for the first 10 output bytes of the RC4. As can be observed, the first output bytes of the PRGA are highly likely to contain the parity bias. It is important to specify that the actual value of the output bytes is not of interest, as was customary in [30–36], but rather the parity bias they follow.

Table 9. Observed probability of parity bias in the first 10 output bytes given.

r	Bias (X)	$P(Z_r \to X)$
0	P	0.49868
1	P	0.988935
2	P	0.987298
3	I	0.979284
4	I	0.96266
5	P	0.954551
6	P	0.938047
7	I	0.923937
8	I	0.899943
9	P	0.888108

Example 2. *If the key* $K = \{1, 0, 111, 124, 85, 140\}$ *is used, the first elements of the KSA output permutation are as follows:*

$$S_0^N = \{88, 0, 46, 241, 74, 125, 226, 141, 96, 39, 28, 5, 232, 245, \ldots\}.$$

After performing the PRGA, the output sequence has the following form:

$$Z = \{228, 64, 116, 117, 237, 112, 34, 139, 53, 44, \ldots\}.$$

The output sequence Z has the parity bias of the form $\{P, P, P, I, I, P, P, I, \ldots, \}$ *in the first output bytes.*

7.2. Case $S_0^N[1] \neq 0$

In the case that the second position is an odd number, after the swap and given the initial indices $i_0 = 1$ and $j_0 = 0 + S_0^N[1]$, the initial output byte of the PRGA is $Z_0 = S_1^N[S_1^N[j_0] + S_1^N[i_0]]$. Since $S_0^N[1] \neq 0$, after the swap, it must be that $S_1^N[i_0]$ is $S_0^N[j_0]$,

which is 0 or an odd number. If $S_1^N[i_0] \neq 0$, then $S_1^N[j_0] + S_1^N[i_0]$ will be even and Z_0 will be even or 1. On the other hand, if $S_1^N[i_0] = 0$, then $S_1^N[j_0] + S_1^N[i_0]$ will be odd and Z_0 will be an odd number. The event $S_1^N[i_0] \neq 0$ has a higher probability of occurrence than the event $S_1^N[i_0] = 0$. Moreover, the event $Z_0 \rightarrow P$ is more likely than $Z_0 = 1$, due to the distribution of values in the even indices of the permutation S_0^N at the end of the KSA. The experimental results obtained are presented in Table 10.

Table 10. Experimentally observed probabilities for $S_0^N[1] \neq 0$.

Event E	$P(E\|S_0^N[1] \neq 0)$	$P(E)$
$S_0^N[S_0^N[1]] = 0$	0.2178519	0.13652
$Z_0 \rightarrow P$	0. 776231	0.6732

The experiments suggest that there is no independence between each event E and $S_0^N[1] \neq 0$, which complicates a theoretical approximation of these results. Table 11 shows the experimentally observed probability that the output sequence of the PRGA exposes the fixed bias.

Table 11. Observed probability of parity bias in the first 10 bytes of output for $S_0^N[1] \neq 0$.

r	Parity (X)	$P(Z_r \rightarrow X)$
0	P	0. 776231
1	I	0.984627
2	I	0.956325
3	P	0.949407
4	P	0.933113
5	I	0.917132
6	I	0.89199
7	P	0.880034
8	P	0.859993
9	I	0.839828

Example 3. *If the key* $K = \{1, 0, 51, 124, 85, 140\}$ *is used, the first elements of the KSA output permutation are as follows:*

$$S_0^N = \{1, 17, 4, 227, 14, 89, 166, 173, 106, 121, 190, 95, 28, 133, 32, 63, \ldots\}.$$

After performing the PRGA, the output sequence has the following form:

$$Z = \{128, 97, 129, 188, 16, 53, 239, 220, 38, 53, \ldots\}.$$

The output sequence Z has the parity bias of the form $\{P, I, I, P, P, I, I, P, \ldots, \}$ *in the first output bytes.*

Some Theoretical Approaches to Z_0 Given $S_0^N[1] \neq 0$

After the first swap, it holds that $S_1^N[i_0 = 1] = S_0^N[j_0] = S_0^N[S_0^N[1]]$ and $S_1^N[j_0] = S_0^N[i_0 = 1]$. Then, setting $x = S_0^N[S_0^N[1]] + S_0^N[1]$, one obtains

$$P\left(Z_0 \rightarrow P \;\middle|\; S_0^N[1] \neq 0\right) \approx P\left(S_1^N[x] \rightarrow P \;\middle|\; x \rightarrow P \bigwedge S_0^N[1] \neq 0\right) P\left(x \rightarrow P \;\middle|\; S_0^N[1] \neq 0\right),$$

where

$$P\left(x \rightarrow P \;\middle|\; S_0^N[1] \neq 0\right) \approx P\left(S_0^N\left[S_0^N[1]\right] \neq 0 \;\middle|\; S_0^N[1] \neq 0\right) \approx 1 - P\left(S_0^N\left[S_0^N[1]\right] = 0 \;\middle|\; S_0^N[1] \neq 0\right).$$

Using Bayes' theorem [54], one arrives at

$$P\left(S_0^N\left[S_0^N[1]\right]=0 \;\middle|\; S_0^N[1]\neq 0\right)=\frac{P\left(S_0^N\left[S_0^N[1]\right]=0\wedge S_0^N[1]\neq 0\right)}{P\left(S_0^N[1]\neq 0\right)}. \tag{8}$$

The event $S_0^N\left[S_0^N[1]\right]=0$ can only occur if $S_0^N[1]\neq 0$. In this way, it is fulfilled that $P\left(S_0^N\left[S_0^N[1]\right]=0\wedge S_0^N[1]\neq 0\right)=P\left(S_0^N\left[S_0^N[1]\right]=0\right)$. Using an idea similar to Theorem 3.1.14 in [14] to calculate $P\left(S_0^N\left[S_0^N[1]\right]=0\right)$, considering that $S_0^N[1]\neq 0$, we propose the following:

Proposition 3. *Given an input key to the RC4 with the parity pattern described in Definition 1, and having* $Q=\frac{N}{2}$*, it follows that*

$$P\left(S_0^N\left[S_0^N[1]\right]=0\right)\approx\sum_{x=0}^{N-1}P(S_0^N[x]=0\wedge S_0^N[1]=x) \tag{9}$$

$$\approx\left(\frac{Q-1}{Q}\right)\left(\frac{Q-2}{Q}\right)^{\frac{N-4}{2}}\approx 0.1364. \tag{10}$$

Proof. Since $S_2[1]=0$, the value of 0 can remain in $S_0^N\left[S_0^N[1]\right]$ if the values $S_r[1]$ and $S_r[x]$ are swapped in the round $r>2$, provided that $S_r[x]=x$ before the swap and, in the remaining rounds, the values of neither of these two positions are involved.

This scenario can be represented starting from index j. In this way, for each possible value of x, the following three cases are considered:

1 $j_r\neq\{1,x\}$, with $r\in[3,x]$;
2 $j_{x+1}=1$;
3 $j_r\neq\{1,x\}$, with $r\in[x+2,N-1]$.

The formulation of these three cases allows for the following approximation:

$$P\left(S_0^N\left[S_0^N[1]\right]=0\right)\approx$$

$$\sum_{x=2}^{N-1}\left[\left(\prod_{r=3}^{x}P\left(j_r\neq 1\bigwedge j_r\neq x\right)\right)\cdot P(j_{x+1}=1)\cdot\left(\prod_{r=x+2}^{N-1}P\left(j_r\neq 1\bigwedge j_r\neq x\right)\right)\right]. \tag{11}$$

Ignoring the first two bytes of the key, each byte in the key can be any even or odd number depending on the parity of its position. Assuming that they are uniformly distributed, this distribution propagates to j_r in the KSA of the RC4. Given this and Proposition 1, $P(j_r=y)=1/128$ for r and y with the same parity and 0 otherwise (since there are 128 even and odd numbers in $[0,255]$ for $r\geq 2$).

In particular, $P(j_r=0)=0$ for all odd r values and $P(j_r=1)=0$ for all even numbers with $r\geq 2$. In general, both probabilities cancel out according to the parity of the round r. The above implies that the products involving the calculation of $P(j_r\neq 1)$ only make sense when $r\to I$ in the interval $[3,255]$. Thus,

$$P\left(S_0^N\left[S_0^N[1]\right]=0\right)\approx\sum_{x=2}^{N-1}\left[\left(\frac{Q-2}{Q}\right)^{\frac{(x+1)-3}{2}}\cdot\frac{1}{Q}\cdot\left(\frac{Q-2}{Q}\right)^{\frac{N-(x+2)}{2}}\right] \tag{12}$$

$$=\sum_{x=2}^{N-1}\left[\frac{1}{Q}\cdot\left(\frac{Q-2}{Q}\right)^{\frac{N-(x+2)+x-2)}{2}}\right] \tag{13}$$

$$=\sum_{x=2}^{N-1}\left[\frac{1}{Q}\cdot\left(\frac{Q-2}{Q}\right)^{\frac{N-4}{2}}\right]=\frac{1}{Q}\cdot\left(\frac{Q-2}{Q}\right)^{\frac{N-4}{2}}\cdot(Q-1) \tag{14}$$

$$= \frac{Q-1}{Q} \cdot \left(\frac{Q-2}{Q}\right)^{\frac{N-4}{2}}. \quad (15)$$

At each moment, j_r is assumed to be uniformly distributed, that is, random according to its parity. It can be verified that, in the interval $[2, 255]$, there are $\frac{N-2}{2} = Q-1$ even numbers and an equal number of odd numbers. □

The value obtained through this approximation is very similar to that obtained in [14,34] for the calculation of this nested probability. Likewise, this value is very close to the value experimentally observed, shown in Table 10.

Then, substituting this result into $P\left(S_0^N\left[S_0^N[1]\right] = 0 \mid S_0^N[1] \neq 0\right)$, one arrives at

$$P\left(S_0^N\left[S_0^N[1]\right] = 0 \;\middle|\; S_0^N[1] \neq 0\right) \approx \frac{0.1364}{0.63} \approx 0.21628, \quad (16)$$

and

$$P\left(x \to P \;\middle|\; S_0^N[1] \neq 0\right) \approx 1 - \frac{0.1364}{0.63} \approx 1 - 0.21628 \approx 0.7837185. \quad (17)$$

According to the distribution of the elements of the permutation, it is given that, if $x \to P$, then $S_0^N[x] \to P$ or $S_0^N[x] = 1$; therefore,

$$P\left(S_1^N[x] \to P \;\middle|\; x \to P \wedge S_0^N[1] \neq 0\right) \approx \frac{126}{128} = 0.984375, with\ x > 2, \quad (18)$$

and, finally,

$$P\left(Z_0 \to P \;\middle|\; S_0^N[1] \neq 0\right) \approx 0.78264 \cdot 0.984375 = 0.77147. \quad (19)$$

The theoretically approximated value is close to the observed value obtained through experimentation, shown in Table 11.

For the rest of the output bytes Z_i of the RC4, with $i > 0$, the probability of the occurrence of the parity bias can be approximated by the distribution of the values of j and $S[i]$. The use of keys with the defined parity pattern implies that approximately the first 20 bytes of output from the RC4 exhibit parity bias and are distinguishable from randomness. In Tables 12 and 13, the probabilities of parity bias in the first 35 output bytes are specified for each of the two described cases.

Table 12. Parity bias in the first 20 bytes of output for case $S[1] = 0$.

r	$P(Z_r \to P)$	$P(Z_r \to P)$	$P(Z_r \to P)$	$P(Z_r \to I)$	$P(Z_r \to I)$
0–4	0.49868	0.988935	0.987298	0.979284	0.96266
	$P(Z_r \to P)$	$P(Z_r \to P)$	$P(Z_r \to I)$	$P(Z_r \to I)$	$P(Z_r \to P)$
5–9	0.954551	0.938047	0.923937	0.899943	0.888108
	$P(Z_r \to P)$	$P(Z_r \to I)$	$P(Z_r \to I)$	$P(Z_r \to P)$	$P(Z_r \to P)$
10–14	0.867675	0.848573	0.825482	0.809062	0.788782
	$P(Z_r \to I)$	$P(Z_r \to I)$	$P(Z_r \to P)$	$P(Z_r \to P)$	$P(Z_r \to I)$
15–19	0.769703	0.742517	0.728079	0.709148	0.690938
	$P(Z_r \to I)$	$P(Z_r \to P)$	$P(Z_r \to P)$	$P(Z_r \to I)$	$P(Z_r \to I)$
20–24	0.671049	0.656026	0.641275	0.626411	0.608667
	$P(Z_r \to P)$	$P(Z_r \to P)$	$P(Z_r \to I)$	$P(Z_r \to I)$	$P(Z_r \to P)$
25–29	0.599902	0.588308	0.576229	0.564269	0.557196
	$P(Z_r \to P)$	$P(Z_r \to I)$	$P(Z_r \to I)$	$P(Z_r \to P)$	$P(Z_r \to P)$
30–34	0.551562	0.542651	0.534225	0.53139	0.528181

Table 13. Parity bias in the first 20 bytes of output for case $S[1] \neq 0$.

r	$P(Z_r \to P)$	$P(Z_r \to I)$	$P(Z_r \to I)$	$P(Z_r \to P)$	$P(Z_r \to P)$
0–4	0. 776231	0.984627	0.956325	0.949407	0.933113
	$P(Z_r \to I)$	$P(Z_r \to I)$	$P(Z_r \to P)$	$P(Z_r \to P)$	$P(Z_r \to I)$
5–9	0.917132	0.89199	0.880034	0.859993	0.839828
	$P(Z_r \to I)$	$P(Z_r \to P)$	$P(Z_r \to P)$	$P(Z_r \to I)$	$P(Z_r \to I)$
10–14	0.813741	0.798875	0.77931	0.758337	0.73474
	$P(Z_r \to P)$	$P(Z_r \to P)$	$P(Z_r \to I)$	$P(Z_r \to I)$	$P(Z_r \to P)$
15–19	0.720614	0.702598	0.681672	0.659861	0.648327
	$P(Z_r \to P)$	$P(Z_r \to I)$	$P(Z_r \to I)$	$P(Z_r \to P)$	$P(Z_r \to P)$
20–24	0.635211	0.618867	0.602507	0.594702	0.585093
	$P(Z_r \to I)$	$P(Z_r \to I)$	$P(Z_r \to P)$	$P(Z_r \to P)$	$P(Z_r \to I)$
25–29	0.57233	0.559377	0.554374	0.549781	0.539603
	$P(Z_r \to I)$	$P(Z_r \to P)$	$P(Z_r \to P)$	$P(Z_r \to I)$	$P(Z_r \to I)$
30–34	0.531062	0.529085	0.526842	0.519741	0.513501

After byte 35, the probability of occurrence of the parity bias behaves as expected in a random sequence.

The results reveal a new vulnerability in the RC4 concerning the distribution of the initial output bytes. This finding underscores the importance of omitting the initial output bytes during encryption. Additionally, it is essential to determine the presence of statistical dependence between the bits of the internal state or the key and the bits of the output sequence in stream ciphers. Identifying such weak keys is also critical to prevent the emergence of vulnerabilities in the output sequences.

8. Conclusions

In this work, a new set of weak keys was introduced for the RC4 cipher which allows information to be obtained about the internal state bits and the output bits of the cipher. These keys follow a defined parity pattern that causes an undesirable performance in the cipher. Through the use of these keys, the parity of each component of the permutation after the KSA is determined with a high probability. In addition, a parity bias is visible in the RC4 output bytes that allows the parity bit of the first 35 bytes of the RC4 output to be predicted with a high probability.

If the output bytes necessary to avoid the exposed parity bias are not ignored, then it is possible to distinguish the output from randomness. It is suggested that this result be taken into account in the proposal of new modifications to the RC4. In a later study, this result will be applied to several of the versions that have been proposed in the literature. The probabilities obtained by the theoretical approximation correspond to those observed experimentally. However, it is considered necessary to improve the calculation of theoretically approximate probabilities.

Author Contributions: Conceptualization, E.J.M.-C.; formal analysis, E.J.M.-C., C.M.L.-P. and G.S.-G.; investigation, E.J.M.-C., C.M.L.-P., G.S.-G. and O.R.; methodology, E.J.M.-C., C.M.L.-P. and G.S.-G.; project administration, O.R.; supervision, G.S.-G. and O.R.; validation, E.J.M.-C. and G.S.-G.; writing—original draft preparation, E.J.M.-C. and C.M.L.-P.; writing—review and editing, O.R. All authors have read and agreed to the published version of the manuscript.

Funding: This work was (partially) supported by the Ibero-American Program of Science and Technology for Development (CYTED) through red522RT0131.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in the study are included in the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Stallings, W. The principles and practice of cryptography and network security 7th edition, isbn-10: 0134444280. *Pearson Educ.* **2017**, *20*, 7.
2. Kuznetsov, A.A.; Potii, O.V.; Poluyanenko, N.A.; Gorbenko, Y.I.; Kryvinska, N. *Stream Ciphers in Modern Real-Time IT Systems: Analysis, Design and Comparative Studies*; Studies in Systems, Decision and Control; Springer International Publishing: Cham, Switzerland, 2022; Volume 375. [CrossRef]
3. Jiao, L.; Hao, Y.; Feng, D. Stream cipher designs: A review. *Sci. China Inf. Sci.* **2020**, *63*, 131101. [CrossRef]
4. Shyaa, G.S.; Al-Zubaidie, M. Utilizing Trusted Lightweight Ciphers to Support Electronic-Commerce Transaction Cryptography. *Appl. Sci.* **2023**, *13*, 7085. [CrossRef]
5. Gupta, K.; Gupta, D.; Prasad, S.K.; Johri, P. A Review on Cryptography based Data Security Techniques for the Cloud Computing. In Proceedings of the 2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), Noida, India, 4–5 March 2021; pp. 1039–1044. [CrossRef]
6. Ramesh, D.; Mishra, R.; Nayak, B.S. Cha-Cha 20: Stream Cipher Based Encryption for Cloud Data Centre. In Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies, Udaipur, India, 4–5 March 2016; pp. 1–6. [CrossRef]
7. Liu, S.; Jin, Z.; Li, Y. Research on Efficient Stream Cipher Design in Big Data Environment. In Proceedings of the 2024 16th International Conference on Machine Learning and Computing, Shenzhen, China, 2–5 February 2024; pp. 36–43. [CrossRef]
8. Wu, L.; Cai, H. Novel Stream Ciphering Algorithm for Big Data Images Using Zeckendorf Representation. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 4637876. [CrossRef]
9. Zhang, L.; Pan, G. Research on the Secure Communication Model of Instant Messaging. In Proceedings of the 6th International Conference on Computer Science and Application Engineering, Virtual Event, 21–23 October 2022; pp. 1–6. [CrossRef]
10. Li, C.; Chen, J.; Wang, L.; Shu, Y. Design of ZUC-256 Stream Cipher Coprocessor for 5G Communication Security. In Proceedings of the 2023 3rd International Symposium on Computer Technology and Information Science (ISCTIS), Chengdu, China, 16–18 June 2023; pp. 1122–1127. [CrossRef]
11. Faragallah, O.S.; Sallam, A.I.; Alajmi, M.; El-sayed, H.S. Efficient selective chaotic video stream cipher for SHVC bitstream. *Multimed. Tools Appl.* **2023**, *82*, 30689–30708. [CrossRef]
12. Achar, S.D.; C, S.S.; P, T.; Nandi, S. Secure Video Streaming Techniques: A Performance Overview. In Proceedings of the 2023 IEEE Guwahati Subsection Conference (GCON), Guwahati, India, 23–25 June 2023; pp. 1–6. [CrossRef]
13. Kumar, P.K.; Mondal, B. Lightweight Stream Cipher for Health Care IoT. In Proceedings of the 2023 IEEE 2nd International Conference on Industrial Electronics: Developments & Applications (ICIDeA), Imphal, India, 29–30 September 2023; pp. 444–449. [CrossRef]
14. Paul, G.; Maitra, S. *RC4 Stream Cipher and Its Variants*; CRC Press: Boca Raton, FL, USA, 2011.
15. Supported Load Balancer Ciphers. Available online: https://docs.oracle.com/en-us/iaas/Content/Balance/Tasks/managingciphersuites_topic-Supported_Ciphers.htm (accessed on 20 November 2024).
16. [MS-SAMR]: RC4 Cipher Usage. 2024. Available online: https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-samr/5fe3c4c4-e71b-440d-b2fd-8448bfaf6e04 (accessed on 20 November 2024).
17. Database Advanced Security Administrator's Guide. Available online: https://docs.oracle.com/cd/B28359_01/network.111/b28530/asoconfg.htm#BBJBIECD (accessed on 20 November 2024).
18. Doni, A.F.; Maria, O.A.H.; Hanif, S. Implementation of RC4 Cryptography Algorithm for Data File Security. *J. Phys. Conf. Ser.* **2020**, *1569*, 022080. [CrossRef]
19. Hanchinamani, G.; G, N.D.; Savakknavar, R. Design of S-Box Based on Chao Initialized RC4. In Proceedings of the 2021 International Conference on Computer Communication and Informatics (ICCCI), Rhodes, Greece, 29 September–1 October 2021; pp. 1–4. [CrossRef]
20. Gaffar, A.; Joshi, A.B.; Kumar, D. Securing Digital Images using Stream Cipher and MDS Matrix. *SN Comput. Sci.* **2021**, *2*, 462. [CrossRef]
21. Maniam, S.M.; Sasilatha, T. Area-efficient and high-speed hardware structure of hybrid cryptosystem (AES-RC4) for maximizing key lifetime using parallel subpipeline architecture. *Concurr. Comput. Pract. Exp.* **2021**, *33*, e5287. [CrossRef]
22. Munir, R. An Improved RC4 Algorithm Based on Multi Chaotic Map for Image Encryption. In Proceedings of the 2023 IEEE 8th International Conference on Recent Advances and Innovations in Engineering (ICRAIE), Kuala Lumpur, Malaysia, 2–3 December 2023; pp. 1–6. [CrossRef]
23. Noura, H.; Salman, O.; Chehab, A.; Couturier, R. Efficient and Secure Keyed Hash Function Scheme Based on RC4 Stream Cipher. In Proceedings of the 2020 IEEE Symposium on Computers and Communications (ISCC), Rennes, France, 7–10 July 2020; pp. 1–7. [CrossRef]
24. Rifki, R.; Septiarini, A.; Hatta, H.R. Cryptography using Random Rc4 Stream Cipher on SMS for Android-Based Smartphones. *Int. J. Adv. Comput. Sci. Appl.* **2018**, *9*, 89–93. [CrossRef]
25. Ullah, A.; Alam, T.; Khan, F.S.; Aznaoui, H. Parallel Implementation of RC4 Data Encryption Method for Cloud Computing. *Comput. Open* **2023**, *01*, 2350002. [CrossRef]
26. Sun, C.; Liu, W.; Cheng, J.; Sun, N.; Peng, Z.; Sha, H.; Yu, W. A high-speed and low-latency hardware implementation of RC4 cryptographic algorithm. *Int. J. Circuit Theory Appl.* **2023**, *51*, 5980–5996. [CrossRef]

27. Khovayko, O.; Schelkunov, D. RC4OK. An Improvement of the RC4 Stream Cipher. (Cryptology ePrint Archive, Paper 2023/1486,2023). Available online: https://eprint.iacr.org/2023/1486 (accessed on 20 November 2024).
28. Kumari, M.; Gupta, S. A Novel Image Encryption Scheme Based on Intertwining Chaotic Maps and RC4 Stream Cipher. *3D Res.* **2018**, *9*, 10. [CrossRef]
29. Guo, T.; Feng, Y.; Fu, Y. A new form of initialization vectors in the FMS attack of RC4 in WEP. *Procedia Comput. Sci.* **2021**, *183*, 456–461. [CrossRef]
30. Matsui, M. Key Collisions of the RC4 Stream Cipher. In *Fast Software Encryption*; Dunkelman, O., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2009; pp. 38–50. [CrossRef]
31. Fluhrer, S.; Mantin, I.; Shamir, A. Weaknesses in the Key Scheduling Algorithm of RC4. In *Selected Areas in Cryptography*; Vaudenay, S., Youssef, A.M., Eds.; Springer: Berlin/Heidelberg, Germany, 2001; pp. 1–24. [CrossRef]
32. Teramura, R.; Ohigashi, T.; Kuwakado, H.; Morii, M. Generalized classes of weak keys on RC4 using predictive state. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **2011**, *94*, 10–18. [CrossRef]
33. Nagao, A.; Ohigashi, T.; Isobe, T.; Morii, M. Expanding Weak-key Space of RC4. *J. Inf. Process.* **2014**, *22*, 357–365. [CrossRef]
34. Dey, S.; Sarkar, S. Generalization of Roos bias in RC4 and some results on key-keystream relations. *J. Math. Cryptol.* **2018**, *12*, 43–56. [CrossRef]
35. Maitra, S.; Paul, G.; Sarkar, S.; Lehmann, M.; Meier, W. New Results on Generalization of Roos-Type Biases and Related Keystreams of RC4. In *Progress in Cryptology—AFRICACRYPT 2013*; Hutchison, D., Kanade, T., Kittler, J., Kleinberg, J.M., Mattern, F., Mitchell, J.C., Naor, M., Nierstrasz, O., Pandu Rangan, C., Steffen, B., et al., Eds.; Series Title: Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2013; Volume 7918, pp. 222–239. [CrossRef]
36. Sarkar, S.; Venkateswarlu, A. Revisiting Roos Bias in RC4 Key Scheduling Algorithm. In Proceedings of the WCC2015-9th International Workshop on Coding and Cryptography 2015, Paris, France, 13–17 April 2015.
37. Pudovkina, M. The Number of Initial States of the RC4 Cipher with the Same Cycle Structure. (Cryptology ePrint Archive, Paper 2003/012,2003). Available online: https://eprint.iacr.org/2003/012 (accessed on 20 November 2024).
38. Mironov, I. (Not So) Random Shuffles of RC4. In *Advances in Cryptology — CRYPTO 2002*; Goos, G., Hartmanis, J., Van Leeuwen, J., Yung, M., Eds.; Series Title: Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2002; Volume 2442, pp. 304–319. [CrossRef]
39. Banik, S.; Isobe, T. Cryptanalysis of the Full Spritz Stream Cipher. In *Fast Software Encryption*; Peyrin, T., Ed.; Series Title: Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2016; pp. 63–77. [CrossRef]
40. Alsharida, R.; Hammood, M.; Ahmed, M.A.; Thamer, B.; Shakir, M. RC4D: A New Development of RC4 Encryption Algorithm. In *Selected Papers from the 12th International Networking Conference*; Ghita, B., Shiaeles, S., Eds.; Springer International Publishing: Cham, Switzerland, 2021; pp. 19–30. [CrossRef]
41. Madarro-Capó, E.J.; Legón-Pérez, C.M.; Rojas, O.; Sosa-Gómez, G. Measuring Avalanche Properties on RC4 Stream Cipher Variants. *Appl. Sci.* **2021**, *11*, 9646. [CrossRef]
42. Jindal, P.; Makkar, S. Modified RC4 Variants and Their Performance Analysis. In *Microelectronics, Electromagnetics and Telecommunications*; Panda, G., Satapathy, S.C., Biswal, B., Bansal, R., Eds.; Lecture Notes in Electrical Engineering; Springer: Singapore, 2019; Volume 521, pp. 367–374. [CrossRef]
43. Parah, S.A.; Sheikh, J.A.; Akhoon, J.A.; Loan, N.A.; Bhat, G.M. Information hiding in edges: A high capacity information hiding technique using hybrid edge detection. *Multimed. Tools Appl.* **2018**, *77*, 185–207. [CrossRef]
44. Soundararajan, E.; Kumar, N.; Sivasankar, V.; Rajeswari, S. Performance Analysis of Security Algorithms. In *Advances in Communication Systems and Networks*; Jayakumari, J., Karagiannidis, G.K., Ma, M., Hossain, S.A., Eds.; Lecture Notes in Electrical Engineering; Springer: Singapore, 2020; Volume 656, pp. 465–476. [CrossRef]
45. Capó, E.J.M.; Cuellar, O.J.; Pérez, C.M.L.; Gómez, G.S. Evaluation of input—Output statistical dependence PRNGs by SAC. In Proceedings of the 2016 International Conference on Software Process Improvement (CIMPS), Aguascalientes, Mexico, 12–14 October 2016; pp. 1–6. [CrossRef]
46. Madarro-Capó, E.J.; Legón-Pérez, C.M.; Rojas, O.; Sosa-Gómez, G.; Socorro-Llanes, R. Bit Independence Criterion Extended to Stream Ciphers. *Appl. Sci.* **2020**, *10*, 7668. [CrossRef]
47. Dhiman, A.; Gupta, V.; Singh, D. Secure Portable Storage Drive: Secure Information Storage. In *Communication, Networks and Computing*; Verma, S., Tomar, R.S., Chaurasia, B.K., Singh, V., Abawajy, J., Eds.; Communications in Computer and Information Science; Springer: Singapore, 2019; Volume 839, pp. 308–316. [CrossRef]
48. Nita, S.L.; Mihailescu, M.I.; Pau, V.C. Security and Cryptographic Challenges for Authentication Based on Biometrics Data. *Cryptography* **2018**, *2*, 39. [CrossRef]
49. Tyagi, M.; Manoria, M.; Mishra, B. Effective Data Storage Security with Efficient Computing in Cloud. In *Communication, Networks and Computing*; Verma, S., Tomar, R.S., Chaurasia, B.K., Singh, V., Abawajy, J., Eds.; Communications in Computer and Information Science; Springer: Singapore, 2019; Volume 839, pp. 153–164. [CrossRef]
50. Zelenoritskaya, A.V.; Ivanov, M.A.; Salikov, E.A. Possible Modifications of RC4 Stream Cipher. In *Advanced Technologies in Robotics and Intelligent Systems*; Misyurin, S.Y., Arakelian, V., Avetisyan, A.I., Eds.; Mechanisms and Machine Science; Springer: Cham, Switzerland, 2020; Volume 80, pp. 335–341. [CrossRef]
51. Knuth, D.E. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*; Addison-Wesley Professional: Boston, MA, USA, 1969.

52. Strömbergson, J.; Josefsson, S. The Perils of Repeating Patterns: Observation of Some Weak Keys in RC4. (Cryptology ePrint Archive, Paper 2013/241,2013). Available online: https://eprint.iacr.org/2013/241 (accessed on 20 November 2024).
53. Roos, A. A Class of Weak Keys in the RC4 Stream Cipher. Vironix Software Laboratories. 1995. Available online: http://agreg.dnsalias.org/Luminy/WeakKeys-report.pdf (accessed on 20 November 2024).
54. Walpole, R.E.; Myers, R.H.; Myers, S.L.; Ye, K. *Probability and Statistics for Engineers and Scientists*; Macmillan: New York, NY, USA, 1993; Volume 5.