(Not So) Random Shuffles of RC4

Ilya Mironov[⋆]
mironov@stanford.edu

Computer Science Department, Stanford University

Abstract. Most guidelines for implementation of the RC4 stream cipher recommend discarding the first 256 bytes of its output. This recommendation is based on the empirical fact that known attacks can either cryptanalyze RC4 starting at any point, or become harmless after these initial bytes are dumped. The motivation for this paper is to find a conservative estimate for the number of bytes that should be discarded in order to be safe. To this end we propose an idealized model of RC4 and analyze it applying the theory of random shuffles. Based on our analysis of the model we recommend dumping at least 512 bytes.

1 Introduction

RC4 is a stream cipher designed by Ron Rivest in 1987. This cipher is extremely fast and exceptionally simple, which makes it ideal for protecting network traffic. In particular, RC4 is part of SSL and WEP implementations, which probably makes it, in the words of its author, "the most widely-used stream cipher in the world" [Riv01]. The design of the cipher was kept as a trade secret until 1994, when it was leaked to the cypherpunks mailing list.

Several vulnerabilities and possible attacks, surveyed in Section 3, have appeared in the open literature. Most of these attacks revolve around the concept of a *distinguisher*, which is an algorithm that can reliably distinguish a pseudo-random number generator from a truly random source. We separate *weak* and *strong* distinguishers. Weak distinguishers may be applied to any continuous segment of the RC4 output stream. A strong distinguisher can only detect a bias at the beginning of the output stream and usually requires access to several streams generated for different keys. The most staggering discovery that falls in the second category was a statistical anomaly in the second byte of the RC4 output, which is zero with probability twice as much as it should [FMS01]. To thwart this and other strong attacks, researchers recommend discarding the first 256 bytes of the output. This proposal can be traced back to 1995, when a Usenet post suggested it as a hedge against a weak-key attack [Roo95]. RSA Security, Inc. maintains that it has been its routine recommendation.

We propose a novel idealized model for studying RC4. We use this abstract model to estimate the number of initial bytes that ought to be dumped from the output stream. In other words, we want to know where the beginning of RC4

[⋆] Supported by NSF contract #CCR-9732754.

ends. We conclude that this number must be more than previously thought. As a practical corollary we describe and explain a bias in the *first* byte of RC4 output. This phenomenon is almost as persistent and reliable as the bias in the second byte, even though they are completely unrelated.

2 The RC4 Cipher

RC4 is a stream cipher in which the keystream is independent of the plaintext. The internal state consists of a permutation on numbers 0. . . 255 represented as an array of length 256 and two indices in this array. We parameterize the length of the permutation by a variable n that may take any integer values (not necessarily powers of 2). The high-level view of the encryption algorithm is as follows:

```
Input: $L$-byte message $m_1, \ldots, m_L$, key $K$
Output: ciphertext $c_1, \ldots, c_L$
$\mathsf{state}_0 \leftarrow \mathsf{KeySched}(K)$
for $i := 1$ to $L$ do
    $\langle \mathsf{state}_i, z_i \rangle \leftarrow \mathsf{PseudoRand}(\mathsf{state}_{i-1})$
    $c_i \leftarrow z_i \oplus m_i$
```

The size of the key K is variable and typically ranges from 40 to 256 bits.

The internals of the algorithm (functions KeySched and PseudoRand) are shown in Fig. 1. Throughout the paper we denote the permutation as S, the two indices of the internal state as i and j and assume that all arrays are indexed starting at 0. All arithmetic is done modulo n.

$\mathsf{KeySched}(K)$

```
for $i := 0$ to $n - 1$ do
  $S[i] := i$

$j := 0$
for $i := 0$ to $n - 1$ do
  $j := j + S[i] + K[i \bmod \ell]$
  swap$(S[i], S[j])$

$i, j := 0$
```

$\mathsf{PseudoRand}(i, j, S)$

```
$i := i + 1$
$j := j + S[i]$
swap$(S[i], S[j])$

output $z := S[S[i] + S[j]]$
```

Fig. 1. The RC4 stream cipher.

3 Existing Attacks

Since the encryption algorithm, namely XORing the plaintext with the output stream, is so simple, any algorithm that predicts a bit in the RC4 output can be used to launch an attack. Consequently, any statistical anomaly in the output stream is a potential vulnerability.

The first weak distinguisher was Golić's [Goli97] that exploited a correlation between z_i and z_{i+2}. Later, twelve much stronger correlations between consecutive bytes of the output stream were discovered by Fluhrer and McGrew [FM00]. Two backtracking algorithms were independently proposed in [MT98] and [K+98]. In this approach the attacker tries to guess the internal state of RC4 by emulating its execution and checking it for consistency with the known output. Both attacks are not practical because of their enormous computation complexity.

Another approach has been taken by Roos and Wagner [Roo95,Wag95]. They described several classes of weak keys that either generate predictable output or, when these keys are used, several bytes of the key can be extracted from the output stream. Ironically, according to our definitions, these classes of weak keys lead to strong distinguishers since they can only be applied to the beginning of the output stream.

The results due to Mantin and Shamir [MS01] and Fluhrer, Mantin, and Shamir [FMS01] are of most practical importance. The first work describes the bias in the second byte of RC4, which is zero with probability twice what you would expect. The second work presents an analysis of a broad class of weak keys based on some parity-preserving properties in the key scheduling.

The fullest study of RC4 to date with comprehensive description of attacks of [MS01] and [FMS01] can be found in [Man01]. [Dur01] proves exact bounds on the effectiveness of the distinguisher of [FMS01].

If cryptanalisys of [FMS01] favors short keys, [GW00] demonstrates a related-key attack that works better on very long keys. It is generally bad practice to use related keys in cryptographic applications. In the case of RC4 it has turned out to be disastrous: the second part of the [FMS01] paper cryptanalyzed the Wired Equivalent Privacy protocol (WEP) that did just that; the attack was implemented and shown to be feasible by [SIR02].

To contrast our work with known results, we stress that we do *not* exploit weak keys. We depart from the concept of modelling RC4 as a finite automaton (as in [Fin94]) and consider the cipher as a random walk on a symmetric group. We do not describe any practical distinguishers other than the one that detects a bias in the first byte. Instead we prove a necessary condition for existence of strong distinguishers in the idealized model.

4 Random Shuffles in RC4

In this section we describe our idealized model for RC4. We model KeySched and PseudoRand as a random shuffle and study the resulting distribution of the permutation S. After one application of KeySched this distribution is not uniform, which results in a detectable bias in the first byte of the output. This raises the question of convergence, i.e., how fast the distribution becomes indistinguishable from the uniform. In Section 5 we tackle this problem.

4.1 Motivational observation

It is often convenient to assume that the internal state is a random permutation. Sometimes it is a valid assumption, but it is hardly so when we look at the behavior of the cipher just after the key scheduling algorithm. The following observation explains why.

Claim. After execution of KeySched we may correctly guess the *sign* of the permutation S with probability 56%.

Before proceeding with a heuristic argument we recall the definition of the sign of a permutation. Here we use one of many equivalent definitions. For any representation of a permutation π as a product of non-trivial transpositions

$$\pi = (a_1b_1)(a_2b_2)\dots(a_mb_m),$$

the sign is defined as

$$\operatorname{sign}(\pi) = (-1)^m = \begin{cases} +1, & \text{if } 2 \mid m \\ -1, & \text{otherwise} \end{cases} .$$

The RC4 cipher initializes the permutation S with the identity permutation. Therefore $\operatorname{sign}(S) = +1$ before the main loop of KeySched. Each iteration of the loop transposes $S[i]$ and $S[j]$. Unless $i = j$, the transposition changes the sign of the permutation. Heuristically, $i \neq j$ with probability $1 - \frac{1}{n}$ and these events are independent for different i. Therefore, the probability that the sign changes every time, for a total of n times, is $(1 - \frac{1}{n})^n \approx e^{-1}$. Similarly, we may compute the probability of the sign changing $1, 3, 5, \dots$ times taking on -1 at the end and $2, 4, 6, \dots$ resulting in the $+1$ value. The limiting distribution for the two possible values of the sign of the permutation S after KeySched is

$$\begin{aligned}
\Pr[\operatorname{sign}(S) = (-1)^n] &= \left(1 - \frac{1}{n}\right)^n + \binom{n}{2}\left(1 - \frac{1}{n}\right)^{n-2}\frac{1}{n^2} + \dots + \frac{1}{n^n} \longrightarrow \\
&\qquad e^{-1}\left(1 + \frac{1}{2!} + \frac{1}{4!} + \dots\right) \xrightarrow[n\to\infty]{} \frac{1}{2}(1 + e^{-2}) \\
\Pr[\operatorname{sign}(S) = (-1)^{n-1}] &= 1 - \Pr[\operatorname{sign}(S_n) = (-1)^n] \qquad \xrightarrow[n\to\infty]{} \frac{1}{2}(1 - e^{-2}).
\end{aligned}$$

Therefore we may predict the sign of the permutation S after execution of the key scheduling algorithm with advantage $\frac{1}{2}e^{-2} \approx 6.7\%$ over a random guess.
Significance of this anomaly. First, this computation is asymptotic and only valid when n goes to infinity. However, $n = 256$ is large enough to make the exact probabilities sufficiently close to their limit values.

Second, and more importantly, the argument hinges on the heuristic assumption that $i = j$ with probability $\frac{1}{n}$ and that these events for different $i = 0 \dots n-1$ are independent of one another. When the key length is maximal, i.e., n bytes, and the key material is drawn from the uniform distribution, this argument is rigorously true. For the actual RC4 parameters it is rarely the case, but the table in Appendix B supports our heuristic.

Third, we do not suggest that this predictor can be useful for an attack on RC4. What we want to stress, however, is that **RC4's state is by no means random**, at least just after executing the key generation algorithm. We hope that the above argument proves this point convincingly. In the next sections we describe a model that captures some aspects of the permutation's behavior and explain other irregularities in its distribution.

4.2 Idealized model

In idealizing (in other words, simplifying) the inner workings of RC4 we take an approach which is seemingly consistent with the initial intent of the cipher's designer. Indeed, all arithmetic performed on the index j is supposed to randomize it, rendering j unpredictable. The only modification we make to RC4 is to explicitly assign to j a random value whenever j gets changed (see Fig. 2).

KeySched*

```
for $i := 0$ to $n-1$ do
    $S[i] := i$

for $i := 0$ to $n-1$ do
    $j :=$ random$(n)$
    swap$(S[i], S[j])$

$i := 0$
```

PseudoRand*

```
$i := i+1$
$j :=$ random$(n)$
swap$(S[i], S[j])$

output $z := S[S[i] + S[j]]$
```

Fig. 2. Idealized RC4 stream cipher.

A closer look at the two modified algorithms KeySched* and PseudoRand* comprising the idealized RC4 reveals that their actions on S are identical and can be unified in a single procedure (Fig. 3). This procedure reflects transpositions that occur in the permutation over the running time of the cipher, both in the key scheduling and the pseudo-random number generator.[1] We call this procedure the P_t **shuffle**, where t is the length of the output including the key scheduling phase or the time, for short. The focus of the rest of the paper is on understanding this shuffle and its connection with real RC4.

[1] There is a slight inaccuracy in this approximation. Index i points to $S[1]$ rather than $S[0]$ at the beginning of PseudoRand. We ignore this difference in our theoretical analysis but will take it into account in computations.

```
$S \leftarrow$ id
for $i := 0$ to $t-1$ do
    swap($S[i \bmod n], S[$random$(n)]$)
```

Fig. 3. P_t shuffle.

5 Exchange Shuffle

The seemingly innocuous problem of shuffling cards has become a subject in its own right, with books and Ph.D. theses devoted entirely to it. The richness and difficulty of the problem lies in the fact that all shuffles are not alike, and no general method exists for dealing with some of the most interesting cases.

Card shuffling is often disguised as a problem in the theory of random walks on groups. We refer the reader to a monograph by Diaconis [Dia88] for both introductory and advanced material and to a recently published survey [Sal01] of an even broader range of topics.

It is even more surprising that the P_t shuffle (Fig. 3) has attracted little attention compared to other shuffles. Examples of better studied shuffling schemes include the random transposition shuffle, when pairs of randomly chosen card are transposed, the riffle shuffle that is modelled after the way professional players shuffle cards, analyzed by Shannon, and several others.

Keeping up with the tradition of a large body of literature, we refer to the permuted elements as *cards* and to the permutation itself as a *deck*. For consistency with our discussion of RC4 we enumerate cards starting at 0.

5.1 Combinatorial results

To the best of our knowledge, the P_t shuffle first appeared in print as a problem in the American Mathematical Monthly [Tho65]. The problem asked for which n the P_n shuffle is truly random, i.e. induces the uniform distribution on permutations.[2] Later on, a series of papers [RB81,SS92,GM01] explored some combinatorial properties of the shuffle, whose summary is given below. The first paper due to Robbins and Bolker christened P_n the *exchange shuffle*. We expand usage of this term, calling P_t the exchange shuffle even when $t \neq n$. The shuffle P_n, as well as any other full sweep through the deck starting with the first card, is called a *pass*.

The simplest argument that demonstrates that the exchange shuffle cannot be truly random is the following. Any single execution of P_t is chosen randomly among n^t equally likely possibilities. When $n > 2$, it is impossible for this scheme to generate the uniform distribution on $n!$ possible decks of n cards since $n!$ does

[2] Several years later another problem in the same venue [Gro77] asked to compute the probability that i ended up in position j after the P_n shuffle. Apparently, that problem prompted combinatorialists Robbins and Bolker to study the shuffle deeper [RB81].

not divide n^n. It is then only natural to ask what are the most and the least likely permutations after one application of the exchange shuffle.

Let us first explain how to fix the exchange shuffle to make it truly random, resulting in a uniform distribution after n steps. Instead of swapping $S[i]$ with a random card, it must be transposed with a card randomly chosen from $S[i] \ldots S[n-1]$. This algorithm is discussed and a detailed proof is given in [Knu75, section 3.4.2].

We list several facts of varied difficulty known about the exchange shuffle P_n.

Fact 1 [RB81] *The probability of obtaining the right cycle (i.e. the final permutation being* $(1\ 2 \ldots n-1\ 0)$ *in cycle notation) at the end of* P_n *is* $2^{n-1}/n^n$.

Fact 2 [RB81] *The right cycle* $(1\ 2 \ldots n-1\ 0)$ *is the least likely permutation after one execution of* P_n.

Fact 3 [RB81] *The probability of obtaining the identity permutation is* Q_n/n^n*, where* Q_n *is the number of involutions on* n *elements (an involution is a permutation which is its own inverse).*

Fact 4 [Knu75, section 5.1.4] *The number of involutions satisfies*

$$Q_n = \frac{1}{\sqrt{2}} n^{n/2} e^{-n/2+\sqrt{n}-1/4}(1 + O(n^{-1/2})).$$

Fact 5 [SS92] *The probability of obtaining a derangement (a permutation with no fixed points) using* P_n *is asymptotically* $\exp(3e^{-1} + e^{-2}/2 - 2) = 0.436\ldots$ *The expected number of fixed points is asymptotically* $2 - 3/e = 0.896\ldots$

For comparison, if a permutation is drawn from the uniform distribution, its probability of being a derangement is $1/e = 0.367\ldots$ and the expected number of fixed points is 1.

Fact 6 [GM01] *If* $n \geq 18$*, the identity permutation is the most likely result of the* P_n *shuffle.*

From the facts above we conclude that the probability of any deck π is contained within the following bounds:

$$\frac{1}{2}\left(\frac{2}{n}\right)^n \leq \Pr[\pi \text{ is generated by } P_n] \leq \frac{1}{\sqrt{2}} n^{-n/2} e^{-n/2+\sqrt{n}-1/4}.$$

The expected value for this probability is approximated using the Sterling formula

$$E[\Pr[\pi \text{ is generated by } P_n]] = \frac{1}{n!} \approx \frac{1}{\sqrt{2\pi n}}\left(\frac{e}{n}\right)^n.$$

There are asymptotically exponential gaps between the probabilities of the random, the most and the least likely permutations.

However instructive and interesting these combinatorial results can be, they are not very useful in analyzing RC4. Indeed, the chances of observing one of the extreme permutations are negligibly small. The only notable exception is Fact 5 that gives us a powerful distinguisher between random permutations and the ones generated by P_n. We show more examples of this approach in the next section.

5.2 Distinguishers

If there is a statistical anomaly in distributions of permutations that can be used to attack RC4, the most direct way to demonstrate existence of this anomaly is to design an algorithm that can reliably distinguish a deck shuffled by P_t from a truly random one. We already know one such distinguisher, namely computing the sign of the permutation (Section 4.1). A practical attack based on this distinguisher is quite unlikely, but due to its theoretical importance (unveiled in Section 5.4) we formally define it and complete the analysis.

Sign distinguisher. First, we describe the algorithm in pseudo-code (Fig. 4).

```
Input: $n$-card deck $S$, time $t$
Output: guessed bit $b$: true if $S$ is random and false if $S$ is output by $P_t$
if sign($S$) = $(-1)^t$
    then return false
    else return true
```

Fig. 4. Sign distinguisher.

A computation analogous to our first analysis of this algorithm in Section 4.1 proves that the *advantage* of guessing bit b correctly is approximately $\frac{1}{2}e^{-2t/n}$. Therefore, after two passes of the shuffle, which models discarding the first 256 bytes of the output of RC4, the sign is 1.83% more likely to be $+1$ than -1. With more passes the advantage of guessing the sign vanishes exponentially fast.

To correctly compute the sign, we must know exactly the entire permutation. More useful distinguishers should be more robust to partial or inaccurate information. Our next algorithm has precisely this property.

Position distinguisher. We observe that the likelihood of the ith card ending up in the jth slot after the P_t shuffle is not constant, as it would be in the case of a random deck, but instead depends on i, j, and t. Let

$$p_{i,j}^{(t)} = \Pr[S[j] = i \text{ after } P_t].$$

The following recurrence equation can be used to efficiently compute p for all t:

$$p_{i,j}^{(0)} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases} \qquad (1)$$

and for $t > 0$

$$p_{i,j}^{(t)} = \begin{cases} p_{i,j}^{(t-1)}(1 - \frac{1}{n}) + \frac{1}{n}p_{t_0,j}^{(t-1)}, & j \neq t_0 \\ \frac{1}{n}, & \text{otherwise} \end{cases}, \qquad (2)$$

where $t_0 = t \pmod n$. In the appendix (Section A) we solve this recurrence for the important case $t = n$.

Given the probabilities $p^{(t)}_{i,j}$, we assign to any permutation π a measure

$$p_t(\pi) = \prod_{i=0}^{n-1} p^{(t)}_{i,\pi(i)}$$

that approximates the likelihood of π after P_t. The approximation is rather crude, since the events $\pi(i) = a$ and $\pi(j) = b$ are not independent (in particular, $a \neq b$). In practice, instead of computing $p_t(\pi)$ directly, we evaluate $\log p_t(\pi) = \sum_{i=0}^{n-1} \log p^{(t)}_{i,\pi(i)}$. We also multiply the probabilities by a scaling factor n, which shifts the distribution toward zero.

The expected value of $p_t(\pi)$ for π drawn from the uniform distribution can be written in the following closed form. It is $\frac{1}{n!} perm\left((p^{(t)}_{i,j})\right)$, where $(p^{(t)}_{i,j})$ is the $n \times n$ matrix of $p^{(t)}$-values and *perm* is the permanent of a matrix. However, in addition to the expected values we also need the distribution of this measure on random π, as well as its distribution on decks shuffled by the exchange shuffle. We resort to a numerical computation and experimentally find the threshold value A used in the distinguisher in Fig. 5.

```
Input: $n$-card deck $S$, time $t$, table $(p^{(t)}_{i,j})$, and the threshold value $A$
Output: guessed bit $b$: true if $S$ is random and false if $S$ is output by $P_t$
$p$ := 0
for $i := 0$ to $n-1$ do
    $p$ := $p + \log np^{(t)}_{i,S[i]}$
if $p < A$
    then return false
    else return true
```

Fig. 5. Position distinguisher. The table $(p^{(t)}_{i,j})$ and the threshold value A are precomputed.

The effectiveness of this measure for different t is summarized in Section B. The table demonstrates that for $t \leq 3n$ the measure p_t is a good distinguisher. Distributions of measure p_n on P_n outputs and on truly random decks are plotted in Fig. 7.

The irregularities in permutations' distribution exploited by this distinguisher are directly observable and show up in the first byte of the RC4 output stream. We analyze its bias in Section 6.

5.3 Variation distance

We have demonstrated two statistical tests that the decks shuffled by P_t fail to pass when $t \leq 3n$. Once it is shown that two distributions are sufficiently distant, existence of more potent distinguishers cannot be excluded. Our next problem is to determine what number of passes of the exchange shuffle is enough to rule out existence of *any* effective statistical tests. To put it another way, we are concerned about computational indistinguishability of two distributions: the uniform distribution and the one induced by P_t.

One technique to prove that two distributions can not be told apart is to show that they are *statistically* close. We define the distance between two distributions.

Definition 1 (variation distance). *Let P and Q be probability distributions on set F. The* variation distance *between P and Q is*

$$\|P - Q\| = \max_{G \subset F} |P(G) - Q(G)| = \max_{G \subset F} \left| \sum_{g \in G} P(g) - \sum_{g \in G} Q(g) \right| .$$

The following simple fact clarifies the relation between the variation distance and statistical tests:

Fact 7 *For two probability distributions P and Q on F and any probabilistic distinguisher $\mathcal{A}\colon F \mapsto \{0,1\}$, its probability of success is no more than the variation distance between P and Q. Formally,*

$$\left| \Pr_{f \leftarrow P(F)} [\mathcal{A}(f) = 1] - \Pr_{f \leftarrow Q(F)} [\mathcal{A}(f) = 1] \right| \leq \|P - Q\|,$$

where $f \leftarrow P(F)$ means that f is sampled from the set F according to the probability distribution P.

The converse is not true, i.e. computational indistinguishability does not imply statistical closeness [Gold01].

Let S_n be the symmetric group on n cards. Let $U(S_n)$ be the uniform distribution on this group and $P_t(S_n)$ be the probability distribution of decks shuffled by P_t. Then the main problem of this section can be formulated as follows:

> What is the minimal t, as a function of n and ε, such that $\|P_t(S_n) - U(S_n)\| < \varepsilon$?

The exchange shuffle can be modelled as a Markov chain (we organize the shuffle in passes, which have identical transition matrices on S_n). This chain is finite, aperiodic (a permutation may stay unchanged after one pass), irreducible (each permutation can be reached from any other within one pass) and therefore converges to the unique stationary distribution. Since the uniform distribution can be shown to be stationary, $P_t(S_n) \xrightarrow[t \to \infty]{} U(S_n)$ and the question posed above is correct when $\varepsilon > 0$. One technical step omitted from this argument is a proof that the variation distance between P_t and U monotonically decreases in t (see full version of the paper [Mir02]).

5.4 Upper and lower bounds

In this section we prove explicit bounds on the convergence rate of $P_t(S_n)$, i.e. its variation distance from $U(S_n)$.

Lower bound. For the lower bound we use the sign distinguisher (Fig. 4). Its probability of success in distinguishing $P_t(S_n)$ and the uniform distribution is known and asymptotically equal to $\frac{1}{2}e^{-2t/n}$. Combining it with Fact 7 we prove that

Theorem 8. *For $n \to \infty$ and $t/n \to \lambda$:*

$$\|P_{t+1}(S_n) - U(S_n)\| > \frac{1}{2}e^{-2\lambda}.$$

Corollary 1. *If the variation distance between $P_t(S_n)$ and $U(S_n)$ is to be made smaller than ε, it must hold that $t/n > \frac{1}{2}\ln\frac{1}{2\varepsilon}$.*

The conclusion is that the number of discarded shuffles must grow *at least* linearly in n. For example, taking $\varepsilon = 2^{-20}$ leads to $t \geq 6.5n$.
Upper bound. To prove an upper bound on the convergence rate we use a variant of an ingenious argument credited to Andrei Broder by [Dia88,Mat88]. We begin by introducing strong uniform times instrumental in the proof.

Definition 2 (strong uniform time [Dia88]). *Suppose we have a randomized shuffling process Q_t on S_n, each execution of which can be unambiguously described by a sequence $q_0, \ldots, q_{t-1}$ in alphabet $\mathcal{Q}$. A* strong uniform time *is a function $T\colon \mathcal{Q}^* \mapsto \{0, 1, \ldots, \infty\}$ with the following properties:*

1. *If $T(q_0, \ldots, q_t) = r < \infty$, then $T(q_0, \ldots, q_s) = r$ for all $r \leq s \leq t$.*
2. *If $T(q_0, \ldots, q_t) = r < \infty$, then T takes the same value r on all sequences which are prefixed with $q_0, \ldots, q_t$.*
3. *Conditionally on T being finite, the shuffle is uniform:*

$$\Pr[\pi = P_t(\mathrm{id}) \mid T(q_0, \ldots, q_t) < \infty] = \frac{1}{n!}.$$

We say that T *happens* when it becomes finite (and equal to the current time). Intuitively, the first two properties mean that T cannot change its value after this has happened. The last property implies that if we look at the decks where T has happened, their distribution is uniform.

The following construction adapted to the exchange shuffle (Fig. 3) from Broder's original algorithm is crucial for this section:

> At the beginning all cards numbered $0 \ldots n-2$ are *unchecked*, the $(n-1)^{\text{th}}$ card is *checked*. Whenever the shuffling algorithm exchanges two cards, $S[i]$ and $S[j]$, one of the two rules may apply before the swap takes place:
>
> **a.** If $S[i]$ is unchecked and $i = j$, check $S[i]$.
> **b.** If $S[i]$ is unchecked and $S[j]$ is checked, check $S[i]$.
>
> The event T happens when all cards become checked.

Theorem 9. *The event T defined above is the strong uniform time for the P_t shuffle.*

Proof. At any given moment preceding t, let k be the number of checked cards, $A = \{a_1, \ldots, a_k\}$ be the set of their positions, $B = \{b_1, \ldots, b_k\}$ be the set of their labels, and $\Pi_k \colon A \mapsto B$ be the correspondence between the two.

Lemma 1. *For all t, the permutations Π_k are uniformly distributed conditional on k, A, and B.*

Proof. [Lemma] Intuitively, when a card is checked, it joins the permutation on checked card in a random position, keeping the set of these permutations uniformly distributed. All other transpositions preserve the conditional distribution of checked cards.

More formally, the proof is by double induction on k and t. When $k = 1$ or $k > 1$ but $t = 0$, the claim is vacuous. Suppose the claim is true for all k and $t < t_0$. Any permutation on k checked cards at time t_0 was either a permutation on the same cards at time $t_0 - 1$, or a new card was checked. If no new card is checked, then all possible transpositions act identically on the permutations with the same A and B sets. It is easy to see that when a new card gets checked, all k positions for this card in the new permutation Π_k are equally likely. Therefore, the distribution on Π_k conditional on k, A, and B is uniform for t_0. □[Lemma]

If the distribution of $\Pi_k|_{t,k,A,B}$ is uniform for all t, we can drop the dependency on t from the condition. When all cards are checked, $k = n$ and $A = B = \{0, \ldots, n-1\}$, the permutation on checked cards is the same as the permutation on all cards. Since the number of checked cards is monotone, the first two requirements of Definition 2 are automatically satisfied; the third requirement has just been proved. Therefore, the event T is the strong uniform time for the shuffle P_t. □

We want to point out that many simpler or similar checking rules would not define a strong uniform time. For instance, always checking $S[i]$ or $S[j]$, or checking $S[j]$ if $S[i]$ is unchecked do not result in the uniform distribution on permutations.

The following lemma demonstrates how a strong uniform time can be used to bound the convergence rate of a shuffling process.

Lemma 2. [Dia88] *Let Q_t be a shuffling process with a strong uniform time T_Q. Then for all t*

$$\|Q_t(S_n) - U(S_n)\| \le \Pr[T_Q > t].$$

The last step in proving an upper bound on the variation distance between P_t and the uniform distribution is the following fact, whose proof is given in the appendix (Theorem **??**):

There exists some constant c such that

$$\Pr[T > cn \log n] \to 0 \text{ when } n \to \infty.$$

From our proof of this result it follows that c may be chosen to be less than 30.

5.5 Experimental results

Since our estimate of the rate of growth of the strong uniform time T is quite loose and "too" asymptotic, we compute the distribution of T for $n = 256$ experimentally. The expected strong uniform time turns out to be

$$E[T] \approx 11.16n \approx 1.4n \log n, \qquad \text{where } n = 256.$$

The threshold $\varepsilon = 1/n$ was chosen as a reasonable goal. Experimentally,

$$\Pr[T > 2n \log n] < 1/n, \qquad \text{for } n = 256.$$

These results[3] are a far cry both from the provable upper and lower bounds on the convergence rate of P_t. There is an apparent gap between theoretical bounds and empirical evidence.

6 First Byte of RC4 Output

As was shown in Section 5.2, the permutation S is not random after the key scheduling algorithm, and quite noticeably so (Section **??** gives explicit formula for individual probabilities). The first byte of RC4 output is $z_0 = S[S[1]+S[S[1]]]$ (in some cases a swap would affect the result). If the permutation is not uniform, there is no reason to believe that z_0 will be uniformly distributed. It is not, and in Fig. 6 we may see the fluctuations in distribution of z_0 around its mean value (the picture is zoomed in, the actual difference in probabilities is 0.6%).

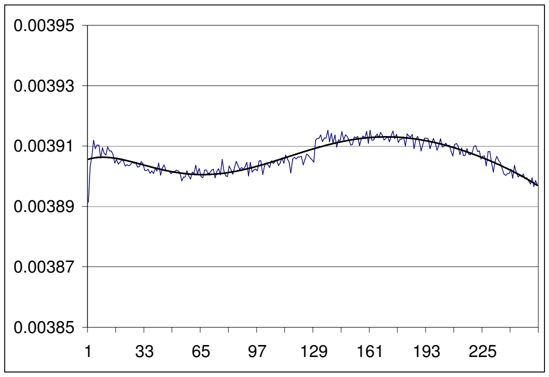

Fig. 6. Bias in the first byte.

Since the distribution of z_0 is different from the uniform distribution across the map, the difference between the two distributions is easily observable. Applying the information-theoretic bound from [FM00] on the number of necessary samples required for a reliable distinguisher, we have that about 1,700 first bytes are sufficient to recognize the distribution with 10% double-sided error.

[3] Tabulated for different n, the mean and the tail probability support the hypothesis that their asymptotic is $\Theta(n \log n)$.

We note that the exact probability distributions following from (1) and (2) are not sufficient to compute the bias in the first byte. The positions occupied by individual cards are not independent, and this must be taken into account.

7 Conclusion

We identified a weakness in RC4 stemming from an imperfect shuffling algorithm used in the key scheduling phase and the pseudo-random number generator. The weakness is noticeable in the first byte but does not disappear until *at least* the third or the fourth pass (512 or 768 bytes away from the beginning of the output). To find out when the nonuniformity vanishes completely we analyze an idealization of RC4 in the form of the exchange shuffle. There is an asymptotic gap between the upper and lower bounds on the convergence rate of the exchange shuffle, and there is no doubt that the constant factor in the upper bound can be improved.

Our most conservative recommendation is based on the experimental data on the tail probability of the strong uniform time T (Section 5.5). This means that discarding the initial $12 \cdot 256$ bytes most likely eliminates the possibility of a strong attack.

Dumping several times more than 256 bytes from the output stream (twice or three times this number) appears to be just as reasonable a precaution. We recommend doing so in most applications.

As a final remark we want to stress that the analysis of the idealized model of RC4 should on no account be accepted as a proof of its security. Many known vulnerabilities, such as weak attacks [Goli97,FM00] as well as results due to Fluhrer, Mantin, and Shamir are not captured by our model.

8 Acknowledgement

This work would have never started without excellent talks given at Stanford by Glenn Durfee and Scott Fluhrer. I am extremely grateful to Dan Boneh and Persi Diaconis for numerous valuable discussions and suggestions. It is my pleasure to thank Itsik Mantin for showing me [Man01]. I would also like to thank David Aldous, Philippe Golle, Ronald Rivest, Adi Shamir, David Wagner, and anonymous referees for their helpful comments.

References

[Dia88] P. Diaconis. *Group Representations in Probability and Statistics.* Lecture Notes-Monograph Series, vol. 11, IMS, Hayward, CA, 1988.

[Dur01] G. Durfee. Distinguishers for the RC4 stream cipher. Unpublished manuscript, 2001.

[Fin94] H. Finney. An RC4 cycle that can't happen. Post in `sci.crypt`, message-id `35hq1u$c72@news1.shell`, 18 September, 1994.

[FM00] S. Fluhrer and D. McGrew. Statistical analysis of the alleged RC4 keystream generator. In proceedings *Fast Software Encryption 2000*, pp. 19–30, Lecture Notes in Computer Science, vol. 1978, Springer-Verlag, 2000.

[FMS01] S. Fluhrer, I. Mantin, and A. Shamir. Weaknesses in the key scheduling algorithm of RC4. In proceedings *SAC 2001*, pp. 1–24, Eighth Annual Workshop on Selected Areas in Cryptography, August 2001.

[Gold01] O. Goldreich. *The Foundations of Cryptography. Basic tools.* Cambridge University Press, Cambridge, England, 2001.

[GM01] D. Goldstein and D. Moews. The identity is the most likely exchange shuffle for large n. `arXiv:math.co/0010066` available from `arXiv.org`.

[Goli97] J. Golić. Linear statistical weakness of alleged RC4 keystream generator. In proceedings *Eurocrypt '97*, Lecture Notes in Computer Science, vol. 1233, Springer-Verlag, 1997.

[Gro77] J. Grossman. Problem E 2645. Amererican Mathematical Monthly, vol. 84(3), p. 217, 1977.

[GW00] A. Grosul and D. Wallach. A related-key analysis of RC4. TR00-358, Rice University, 2000.

[K+98] L. Knudsen, W. Meier, B. Preneel, V. Rijmen, and S. Verdoolaege. Analysis methods for (alleged) RC4. In proceedings *Asiacrypt '98*, Lecture Notes in Computer Science, vol. 1514, Springer-Verlag, 1998.

[Knu75] D. Knuth. *The Art of Computer Programming. Second Edition.* Addison-Wesley, Reading, MA, 1975.

[Man01] I. Mantin. Analysis of the stream cipher RC4. Master's Thesis, Weizmann Insitute, Israel, 2001.

[MS01] I. Mantin and A. Shamir. A practical attack on broadcast RC4. In proceedings *Fast Software Encryption 2001*, Lecture Notes in Computer Science, Springer-Verlag, 2001.

[Mat88] P. Matthews. A strong uniform time for random transpositions. *Journal of Theoretical Probability,* vol. 1(4), 1988.

[Mir02] I. Mironov. (Not So) Random Shuffles of RC4. full version of this paper. Cryptology ePrint Archive, Report 2002/003, available form `http://eprint.iacr.org/`, 2001.

[Mis98] S. Mister. Cryptanalysis of RC4-like ciphers. Master's Thesis, Queen's University, Kingston, Ontario, Canada. May 1998.

[MT98] S. Mister and S. Tavares. Cryptanalysis of RC4-like ciphers. In proceedings *SAC '98*, Fifth Annual Workshop on Selected Areas in Cryptography, August 1998.

[Riv01] R. Rivest. RSA Security response to weaknesses in key scheduling algorithm of RC4. Technical note available from RSA Security, Inc. site. `http://www.rsasecurity.com/rsalabs/technotes/wep.html`, 2001.

[RB81] D. Robbins and E. Bolker. The bias of three pseudo-random shuffles. *Acquationes Mathematicae*, vol. 22, pp. 268–292, 1981.

[Roo95] A. Roos. Class of weak keys in the RC4 stream cipher. Two posts in `sci.crypt`, message-id `43u1eh$1j3@hermes.is.co.za` and `44ebge$llf@hermes.is.co.za`, 1995.

[Rue86] R. Rueppel. *Analysis and Design of Stream Ciphers.* Springer-Verlag, 1986.

[Sal01] L. Saloff-Coste, Probability on groups: random walks and invariant diffusions. *Notices of the American Mathemtatical Society*, vol. 48(9), pp. 968–977. 2001.

[SS92] F. Schmidt and R. Simion, Card shuffling and a transformation on S_n. *Acquationes Mathematicae*, vol. 44, pp. 11–34, 1992.

[SIR02] A. Stubblefield, J. Ioannidis, and A. Rubin. Using the Fluhrer, Mantin, and Shamir attack to break WEP. In proceedings *NDSS '02*. 2002.

[Tho65] E. Thorp. Problem E 1763. American Mathematical Monthly, vol. 72(2), p. 183, 1965.

[Wag95] D. Wagner. My RC4 weak keys. Post in `sci.crypt`, message-id `447o1l$cbj@cnn.princeton.EDU`, 26 September, 1995.

A A solution to the recurrence

Claim. Solution to the equations (1) and (2) at time $t = n$ is given by

$$p^{(n)}_{a,b} = \begin{cases} \frac{1}{n}\left[(1-\frac{1}{n})^{n-b-1} + (1-\frac{1}{n})^{a}\right], & \text{if } b < a \\ \frac{1}{n}\left[(1-\frac{1}{n})^{a} + \left(1-(1-\frac{1}{n})^{a}\right)(1-\frac{1}{n})^{n-b-1}\right], & \text{if } b \geq a. \end{cases}$$

Proof. Notice that the card that has been indexed by i at least once has equal chances of ending up in any slot at time n. With this in mind consider the case of $b < a$ (the card moves left). There are two possibilities for this outcome. First, it may happen if at time $i = b$ the ath card is indexed by j and after the swap, when $S[b] = a$, the bth slot is never visited again. The probability of this event is $\frac{1}{n}(1-\frac{1}{n})^{n-b-1}$. Second, when $i = a$ and $a = S[a]$, the ath card become "randomized" and has equal chances of being in any position at the end of the pass. In this case, the probability that the ath card ends up in the bth slot is $\frac{1}{n}(1-\frac{1}{n})^{a}$.

The case $a \leq b$ is treated analogously. □

B Experimental Data

The following table summarizes success probability of the position distinguisher for different t (Section 5.2) with $n = 256$.

t	**Cut-off** A	**Advantage**
n	0.0	60%
$2n$	0.0	3.4%
$3n$	0.0	0.1%
$4n$	—	unreliable

Figure 7 plots two distributions of the measure p_t (Fig. 5), one on the random decks and another on $P_n(S_n)$ (decks shuffled with one pass of the exchange shuffle).

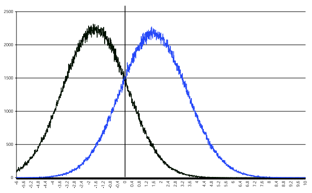

Fig. 7. Two distributions of measure p_t for $U(S_n)$ and $P_n(S_n)$, where $n = 256$.