RZ 3019 (# 93065) 05/04/98
Computer Science/Mathematics 13 pages

Research Report

On the Entropy of Arcfour Keys

Luke O'Connor

IBM Research Division
Zurich Research Laboratory
8803 Rüschlikon
Switzerland

IBM Research Division
Almaden · Austin · Beijing · Haifa · T.J. Watson · Tokyo · Zurich

On the Entropy of Arcfour Keys

Luke O'Connor

IBM Research Division, Zurich Research Laboratory, 8803 Rüschlikon, Switzerland

Abstract

Arcfour is a stream cipher that produces a byte keystream $B = \{b_i\}_{i=0}^{\infty}$, where a key K is used to select the initial state S_0, and the b_i are produced by the state transition $\delta(S_i) = S_{i+1}$. Let the byte length of K be $|K|$, and let $S_0(K)$ be the initial state produced by K. Two keys K_1, K_2 are considered *equivalent* if $S_0(K_1) = S_0(K_2)$, and further K_2 is *weak* if $|K_1| < |K_2|$. We show that there is a class of weak keys based on the notion of string periodicity which contains 256 weak 40-bit keys and 2^{64} weak 128-bit keys. We exhibit 128-bit keys whose entropy is no more than a byte.

We also present an algorithm for constructing the initial contents of the Arcfour state machine based on observing $B_{256} = \{b_i\}_{i=0}^{255}$. The method is significantly faster than exhaustive search for initial the state S_0, and shows that no additional security against brute-force attacks is expected to be achieved by selecting keys K for which $|K| \geq 57$. Also it shows that if Arcfour is scaled down to operate on 4-bit values with 64-bit keys, say suitable for smart card environments, the state contents can be recovered in approximately 10^7 operations.

1 Introduction

Arcfour is a stream cipher recently proposed to the Internet Engineering Task Force [3], and has been included in several other IETF documents, including an IPSEC Internet Draft [4]. Arcfour operates on variable-length keys, to produce a byte keystream $B = \{b_i\}_{i=0}^{\infty}$, and is implemented as a finite state machine. The current state is given by a permutation S of $0, 1, \ldots, 255$ and by two byte pointers x, y into S, which are used in the generation of the initial state and the next-state function. The state transition function δ outputs a byte b_i during each update, and the next message byte m_i is encrypted as as $c_i = m_i \oplus b_i$. The initial state S_0 is generated under the action of a user supplied d-byte key K, $|K| = d$, $1 \leq d \leq 256$, and let this state be $S_0(K)$. We will say that K_1 and K_2 are *equivalent*, denoted $K_1 \equiv K_2$, if $S_0(K_1) = S_0(K_2)$. Equivalent keys produce the same keystream sequence B as δ is independent of the key. A potential problem with the variable-length keys of Arcfour is that if $K_1 \equiv K_2$ and $|K_1| \ll |K_2|$, then the entropy of K_2 is only $8 \cdot |K_1|$ bits rather than the expected $8 \cdot |K_2|$ bits. When Arcfour keys are used with a length of at least 211 bytes, key equivalences must be present since the number of possible keys exceeds the number of possible initial states $(256! < 256^{211})$. A more interesting question is to determine if equivalences can be detected for much shorter keys, for say $d \in \{5, 16\}$, corresponding to 40-bit and 128-bit Arcfour keys respectively, which are recommended for use [3].

As explained below, Arcfour effectively takes the user supplied key K and creates an *expanded key* $\bar{K}$ of 256 bytes. If $|K| < 256$ then $\bar{K}$ is obtained by assigning K to consecutive d-byte blocks of $\bar{K}$ until all bytes of the expanded key are assigned. Expressed another way, K is concatenated with itself $\lceil 256/d \rceil$ times, with $\bar{K}$ being set to the first 256 bytes of this concatenation. This implies, for example, that the 1-byte key $K_1 = [1]$ and the 2-byte $K_2 = [1, 1]$ will both give the same expanded key, as will $K_1 = [13, 249]$ and $K_2 = [13, 249, 13, 249]$. In both cases the first $d/2$ bytes of K_2 is exactly K_1, and obviously the byte sequence produced by copying (concatenating) K_2 $256/d$ times is equivalent to copying K_1 $512/d$ times. We will call such equivalent keys *periodic*, a term borrowed from combinatorial pattern matching. In the sequel prove that for Arcfour there are exactly 256 periodic 40-bit keys, and exactly $2^{64} = 256^8$ periodic 128-bit keys. In particular, there exist 128-bit keys whose entropy is no more than a byte. Further, if a mode of Arcfour was introduced based on 64-bit keys, then the probability of selecting a key with only 32 bits of entropy is about one in 10 billion.

Since the byte length of Arcfour keys is between 1 and 256, it is possible to select a 2048-bit key but it will not be the case that the effort of recovering such keys is on the order of 2^{2048} operations. As mentioned above, there are less than $256^{211} = 2^{1688}$ initial states, so no additional security against brute-force attacks can be achieved by increasing key lengths beyond 1688 bits. However it is possible to recover the contents of the Arcfour state machine in considerably less time. The obvious brute force strategy is to guess an initial state S_0^* and then perform state transitions, producing the keystream $B^* = \{b_i^*\}_{i=0}^{\infty}$, until $b_j^* \neq b_j$ or $j = n$, where incorrect initial states will be identified with high probability before j reaches n. An important observation is that if $\delta(S_i^*) = S_{i+1}^*$ then S_i^* and S_{i+1}^* *differ in at most 2 bytes* . Thus, rather than guess the entire contents of S_0, we assign a few values to S_0^* so that $b_0^* = b_0$; then assign more values to S_1^* so that $b_1^* = b_1$, and so on, where we require at most 3 assignments at each transition (two for the state pointers and one for the key byte). If at any point $b_i^* \neq b_i$, we backtrack to the greatest $j < i$ for which $b_j^* = b_j$ and make new assignments. If $B_n = B_n^*$ then all bytes in S^* have been assigned and $S_n^* = S_n$ with high probability.

Because of the backtrack property, this method of finding S_n is considerably more efficient than searching through the initial states explicitly. Our experiments have shown that Arcfour initial states can be recovered by backtrack methods that are expected to examine less than

2^{455} search nodes. This result indicates that the maximal entropy of any Arcfour key is expected to be less than 455 bits, or $d = 57$ bytes. This 455-bit value is derived from the standard Monte Carlo method [5, p.112] for estimating the size of a backtrack tree. Note that the description of Arcfour is sufficiently general to be adapted to operate on values other than bytes. For example, let Arcfour-16 be the scaling of Arcfour (or Arcfour-256) to operate on 4-bit values, with computations reduced modulo 16. Arcfour-16 could be used in smart cards environments where the 259 bytes needed to represent Arcfour-256 (256 bytes for S and 3 bytes for pointers) is not always available. Arcfour-16 requires less than 10 bytes of storage and can be operated using 64-bit keys, yielding a nontrivial level of security against exhaustive search of the keyspace (approximately 10^{20} keys would have to be searched). However, using the backtrack algorithm given above, the state contents of Arcfour-16 can be recovered in time proportional to 10^7 operations.

The paper is outlined as follows. In section 2 we review the Arcfour algorithms for producing the initial state, and the keystream sequence B. In section 3 we make some relevant definitions concerning strings, and then present our weak keys based on the periodicity induced by the expanded key. In §4 we describe the backtrack algorithm for recovering the contents of the Arcfour state machine. In §5 we suggest some improvements to the backtrack search algorithm based on the distribution of the elements in the initial state when random keys are used.

2 Arcfour-n and its Expanded Key

The initial state generation algorithm (IGSA) of Arcfour is shown in Figure 1, and it maps the identity permutation to S_0 under the action of a key. The IGSA is expressed using the parameter n which indicates that S is a permutation of $\{0, 1, \ldots, n-1\}$, and K is an array of d values from Z_n. In this case we will denote the cipher for a given value of n as Arcfour-n, and standard Arcfour (as described in the introduction) is equivalent to Arcfour-256. We use this notation since it will be convenient in later sections to consider versions of Arcfour that operate on values other than bytes. For generality we will refer to $S[i]$ as a *state word* , and refer to key K as a string of d *key words* $K[0], K[1], \ldots, K[d-1]$.

The ISGA proceeds by performing n swaps on the state array with the i-th swap[1] interchanging state words $S[x]$ and $S[y]$. In each of the n iterations of the main loop in the ISGA, corresponding to the x-th swap, y is updated as a function of itself, a state word and a key word. The expanded key $\bar{K} = \bar{K}[0], \bar{K}[1], \ldots, \bar{K}[n-1]$ of length n is formed from the d-word key K by concatenating K with itself $\lceil \frac{n}{d} \rceil$ times, and letting $\bar{K}$ be the first n words of this operation. The algorithm for producing $B = \{b_i\}_{i=0}^{\infty}$ is shown in Figure 2, which is similar to the ISGA, and again it is stated in terms of the parameter n. Note that the change in S between the output of successive words $S[X]$ is minimal: the state words $S[x]$ and $S[y]$ are swapped, and if $x = y$ then there is no change. We will use this observation in section §4 to construct a backtrack algorithm for finding the state contents of Arcfour-n.

In the analysis of later section we will use the following model for the key.

Definition 2.1 The *random key model* for Arcfour-n is defined as the distribution of initial states induced by running the ISGA using a key $K = K[0], K[1], \ldots, K[n-1]$ of length n, such that each word $K[i]$ is independently and uniformly selected from Z_n. □

[1] As x is bound as $0 \leq x < n$, the swaps are indexed as the 0-th swap, the 1st-swap, and so on.

```
for  x ← 0 to (n − 1) do
         S[x] ← x ;
         K̄[x] ← K[x mod d];
od
y ← 0 ;
for  x ← 0 to (n − 1) do
         y ← (K̄[x] + S[x] + y) mod n ;
         swap  (S[x], S[y]);
od
```

Figure 1: The ISGA for Arcfour-n.

```
x ← y ← 0 ;
while  keystream bytes are required
         x ← (x + 1) mod n;
         y ← (S[x] + y) mod n;
         swap  (S[x], S[y]);
         X ← (S[x] + S[y]) mod n ;
         output  S[X] ;
od
```

Figure 2: Keystream generation in Arcfour-n.

3 Periodic Weak Keys

Recall that two keys K_1, K_2 are equivalent if $S_0(K_1) = S_0(K_2)$, and clearly this will be the case when $\bar{K}_1 = \bar{K}_2$. We may consider K as a string X_K of length d over an n-ary alphabet $\mathcal{A}_n = \{a_0, a_2, , \ldots, a_{n-1}\}$ where $K[i]$ is represented as a_i if $K[i] = i$. Let $XX = X^2$ denote the concatenation of X with itself, and let $X^t = XX^{t-1}$. The expanded key $\bar{K}$ for K is then clearly equal to the first n letters of the string $(X_K)^t$ where $t = \lceil \frac{n}{d} \rceil$. For two keys K_1 and K_2, let $|K_1| = d_1$ and $|K_2| = d_2$. Our goal is then to determine for which values of K_1 and K_2 we have equality in the first n letters of $(X_{K_1})^{t_1}$ and $(X_{K_2})^{t_2}$, where $t_1 = \lceil \frac{n}{d_1} \rceil$, $t_2 = \lceil \frac{n}{d_2} \rceil$.

The following definition can be found in [2], for example.

Definition 3.1 A string X of length d has period b if X can be written as $X = YY \cdots YV$ where $|Y| = b$, and V is a prefix of Y. If Y is such that $|Y| < d$ then X is *periodic* and Y is said to *generate* X. When no such Y exists then X is said to be *aperiodic*. □

Thus if X_K is periodic then K has an equivalent key of shorter length. When this is the case we will say that K is a *periodic key*, otherwise K is said to be *aperiodic*. For example, the key $K = [2, 4, 2, 4]$ is periodic while the key $K = [1, 3]$ is aperiodic.

Let $X = x_0, x_1, x_2$ and $Y = y_0, y_1$ be strings of lengths 3 and two respectively, and consider determining which assignments to X and Y yield the equality $X^2 = Y^3$. Aligning the indices of X and Y in X^2 and Y^3, respectively, we have that

$$\begin{aligned} X^2 &= 012012 \\ Y^3 &= 010101 \end{aligned}$$

This alignment states, for example, that the first, third and fifth letters of Y^3 are all equal to y_0, and if $X^2 = Y^3$ then $x_0 = x_1 = x_2 = y_0$. Similarly we must have $x_0 = x_1 = x_2 = y_1$ which implies X^2 can only equal Y^3 if $X = c^3$ and $Y = c^2$. A string of the form c^t will be called a *constant string*.

Definition 3.2 Let $X = x_0, x_1, \ldots, x_{d-1}$ and let $Y = y_0, y_1, \ldots, y_{b-1}$, $b < d$, and consider the alignments between the strings X^b and Y^d, both of length db. Then the *alignment set* $A(X, Y, i)$ of y_i is defined as

$$A(X,Y,i) \quad = \quad \{ j \mid k \equiv i \bmod b,\ k \equiv j \bmod d,\ 0 \le k < db \}. \tag{1}$$

□

We now show that the alignment sets are determined by the subgroup structure of $(Z_d, +)$.

Lemma 3.1 Let $|X| = d$ and $|Y| = b < d$. Let $H =< b >$ be the subgroup of $(Z_d, +)$ generated by b. Then $A(X, Y, i) = iH = \{ih \mid h \in H\}$ for i, $0 \le i < b$, and X^t has $n^{d/|H|}$ generators Y such that $Y^d = X^b$.

Proof. It is easily seen that $H =< b >$ is equal to $A(X, Y, 0)$. Further, it follows that $A(X,Y,i) = \{(i+h) \bmod d \mid h \in H\}$, which is by definition iH. Then for Y to generate X^t we must have that for each y_i, $x_j = y_i$ for all $j \in iH$. Thus the number of distinct assignment to the letters of Y over an n-ary alphabet to generate some X^b is $n^{d/|H|}$ since there are $d/|H|$ distinct cosets iH. □

Corollary 3.1 There are 256 periodic 40-bit Arcfour keys.

Proof. In this case $d = 5$, and note that $< b > = Z_5$ for $1 \le b < 5$. Thus there are $256^{5/5}$ generators Y of strings corresponding to periodic 40-bit Arcfour keys. □

Corollary 3.2 There are 2^{64} periodic 128-bit Arcfour keys.

Proof. In this case $d = 16$, and note that $bd < 256$ for all $b < d$. Thus if $< b > = Z_{16}$, which is the case for all b, $\gcd(b, 16) = 1$, then the only periodic key is the constant string c_0, Apart from $\{0\}$, the only other proper subgroups of Z_{16} are $\{0,2,4,6,8,10,12,14\}$, $\{0,4,8,12\}$ and $\{0,8\}$, which respectively correspond to periodic keys of the form $(c_0c_1)^8$, $(c_0c_1c_2c_3)^4$, $(c_0, c_1, c_2, c_3, c_4, c_5, c_6, c_7)^2$. Since the periodic keys corresponding to $\{0,8\} =< 8 >$ contain the other two classes of periodic keys, there are $256^{16/2}$ periodic 128-bit Arcfour keys. □

Table 1 lists the forms of the periodic keys for 40- and 128-bit Arcfour keys, and gives the probability of such a key being selected randomly. The largest set of periodic keys are those 128-bit keys for which the first 8 bytes equals the second 8 bytes, yielding an effective entropy of 8 bytes. If a mode of Arcfour was introduced that used 64-bit keys then there would exist keys with only 4 bytes of entropy that would be randomly selected with probability 2^{-32}. Thus this key length should be avoided.

The probability of selecting a periodic key in Arcfour-n is high when full length ($d = n$) keys are used, as this probability is at least $\Pr(K[0] = K[n-1]) = 1/n$). Guibas and Odlyzko [2] have proven that a string over an n-ary alphabet is aperiodic with probability tending towards $\frac{n-2}{n-1} + O(n^{-3})$.

Bit Length	Periodic String	Probability
40	$(c_0)^5$	2^{-32}
128	$(c_0)^{16}$	2^{-120}
128	$(c_0, c_1)^8$	2^{-112}
128	$(c_0, c_1, c_2, c_3)^4$	2^{-96}
128	$(c_0, c_1, c_2, c_3, c_4, c_5, c_6, c_7)^2$	2^{-64}

Table 1: Periodic 40- and 128-bit keys for Arcfour-8 = Arcfour.

$S^*[k] \leftarrow -1, \quad 0 \leq k < n$;
$P[0] \leftarrow 0$; $i \leftarrow 0$; $j \leftarrow 0$
backtrack $\leftarrow$ false ;

while more permutations **do**
    $P \leftarrow$ NEXT-PERM-ELEMENT(P, j) ; // j increases by one
    backtrack $\leftarrow$ CONSISTENT(S_i^*, P, i, j) ; // i and j may increase

    while backtrack == true **do**
        $P \leftarrow$ BACK-PERM-ELEMENT$(P,\ j)$ // j decreases by at least one ;
        backtrack $\leftarrow$ CONSISTENT(S_i^*, P, i, j) ;
    od
od

Figure 3: Pseudo-code for the backtrack recovery of the state contents of Arcfour-n.

4 Recovering State Contents Through Backtrack

Our goal in this section is to obtain an upper bound on the work factor T_n of recovering Arcfour-n keys, from which we conclude that the entropy of an Arcfour-n key is at most $\log T_n$ bits. Since determining the initial state $S_0(K) = S_0$ for a key K is equivalent to knowing K itself for the cryptanalyst, it follows that $T_n \leq (256!) \approx 2^{1684}$, since the set of initial states can be searched exhaustively. However we show that a backtrack solution for the recovery of S_0 runs in time much faster than exhaustive search, and in fact, $T_n \leq 2^{455}$. To perform the backtrack solution we assume that the cryptanalyst has access to the first n bytes of the keystream B, denoted B_n.

Consider the set of state transitions

$$(S_0, x_0, y_0) \rightarrow (S_1, x_1, y_1) \rightarrow (S_2, x_2, y_2) \rightarrow \cdots$$

where the current state is represented by a triple consisting of the current permutation S_i of $\{0, 1, \ldots, n-1\}$ and two pointers $x_i, y_i \in Z_n$. At the i-th state transition x_i and y_i are updated, $S_i[x_i]$ and $S_i[y_i]$ are swapped, and a byte b_i is emitted. Note that S_i and S_{i+1} will only differ in at most two positions, so that the change in the state permutation from one transition to the next is small. Consequently, to 'simulate' the transition from S_i to S_{i+1} we need only have knowledge of $S_i[x_i], S_i[y_i]$ and the state pointers.

The general approach is to construct another state machine (S_i^*, x_i^*, y_i^*) whose operation mimics that of S given B_k for some $k < n$. Each state word in $S_i^*[j]$ is either -1, meaning it is undefined, or is set to a value in Z_n distinct from other defined state words. The search begins by assigning all the words of S_0^* to be -1, thus marking them all as undefined. Then set $x_0^* = y_0^* = 0$, and guess the value of $S_0^*[x_0]$. If $S_0^*[y_0]$ is defined, then compute the position of the key byte b_0 relative to $S_0^*[x_0]$ and $S_0^*[y_0]$ which is $X_0^* = S_0^*[(S_0^*[x_0] + S_0^*[y_0]) \bmod n]$. If $S_0^*[y_0]$ is not defined, then guess its value and determine X_0^* as above. If either (1) b_0 is assigned to a state word that $S_0^*[X_0^*]$ does not point to, or (2) $S_0^*[X]$ is defined but not equal to b_0, then one or both of the choices for $S_0^*[x_0^*], S_0^*[y_0^*]$ is incorrect and we backtrack to make new choices. If there was no error we make the assignment $S_0^*[X_0^*] = b_0$. In this case we have constructed an initial state (S_0^*, x_0^*, y_0^*) that is *consistent* with the output byte b_0 even though most word values $S_0^*[i]$ values of are undefined. We may now proceed to make assignments for (S_1^*, x_1^*, y_1^*) relative to (S_0^*, x_0^*, y_0^*) and b_1. If the assignment is consistent with b_1 we proceed to make assignments for (S_2^*, x_2^*, y_2^*), otherwise we backtrack and choose new assignments for (S_1^*, x_1^*, y_1^*), and possibly new assignments for (S_0^*, x_0^*, y_0^*).

The pseudocode for the backtrack method to finding the state contents of Arcfour-n is given in Figure 3. Imagine that during the search j assignments to S^* have been made so that S^* is consistent with $b_0, b_1, \ldots, b_{i-1}$ and we thus consider S^* to be in state (S_i^*, x_i^*, y_i^*). We will store these j assignments in P as $P[0], P[1], \ldots, P[j-1]$. When we say that S^* is in state (S_i^*, x_i^*, y_i^*) we mean that one of $S_i^*[x_i]$ or $S_i^*[y_i]$ is undefined. Then to facilitate a transition, P must be extended by assigning values to $P[j]$ and possibly $P[j+1]$, corresponding to assignments to $S_i^*[x_i]$ and $S_i^*[y_i]$. The routine NEXT-PERM-ELEMENT assigns $P[j]$ and increments j by one, and extending P by k elements requires k calls to this routine. After each extension to P the routine CONSISTENT is called to determine if the assignments $P[0], P[1], \ldots, P[j-1]$ are consistent with B_i, for some i. If the assignment is not consistent then BACK-PERM-ELEMENT is called which decreases j until $P[0], P[1], \ldots, P[j-1]$ is consistent, and selects a new value of $P[j]$ according to a systematic backtracking strategy.

Both NEXT-PERM-ELEMENT and BACK-PERM-ELEMENT are based on an algorithm for generating permutations lexicographically. Given $P = P[0], P[1], \ldots, P[j-1]$, NEXT-PERM-ELEMENT selects $P[j]$ such that $P[0], P[1], \ldots, P[j-1], P[j]$ is a prefix of the next smallest permutation π not yet generated, while BACK-PERM-ELEMENT reduces j so that $P[0], P[1], \ldots, P[j-1], P[j]$ is the longest prefix of the smallest permutation π not yet generated. Thus the search for the initial state examines all permutations, and does not terminate when it finds the first π consistent with B_n.

Note that $P = P[0], P[1], \ldots, P[j-1]$ is the order in which values are assigned to state words. Since the set of state words is a permutation of $\{0, 1, \ldots, n-1\}$ then after n assignments P is also a permutation, which we will call an *assignment permutation.* In theory, to test all possible state contents, P would have to cycle through all $n!$ assignment permutations for n state words. However, the major gain in efficiency is that if $P = P[0], P[1], \ldots, P[j-1]$ is consistent and $P = P[0], P[1], \ldots, P[j-1], P[j]$ is not, then all $(n-j-1)!$ assignment permutations for which P is a prefix can be eliminated from consideration.

Table 2 shows the results of running the backtrack search method on Arcfour-n for various values of n, using the random key model. We performed experiments using shorter keys but found no significant deviation from the results presented in Table 2. Columns 2-4 present results for the Monte Carlo estimation [5, p.112] of the size of the backtrack tree, listed with the number of sample problems examined and the number of branches in each tree that were examined (the probes). Columns 5-6 are the results for the true running time of the backtrack algorithm. Here $\mathbf{E}[MCS]$ and $\mathbf{E}[ES]$ are the expected number of nodes in the backtrack tree as determined by the Monte Carlo method and by true exhaustive search, respectively. The '1 solution' column shows the fraction sample problems where only one permutation π was found to be consistent with B_n. In all cases, the maximum number of consistent permutations

Arcfour-n	#samples	#probe	$\mathbf{E}[MCS]$	$\mathbf{E}[ES]$	1 solution (%)	$\mathbf{E}[ES]/\mathbf{E}[MCS]$
4	1000	1000	21.1	23.3	77.6	1.12
4	2000	2000	21.3	23.6	77.6	1.11
5	1000	1000	47.1	55.4	96.8	1.17
5	2000	2000	47.0	55.1	95.8	1.17
6	1000	1000	112.1	133.9	97.2	1.19
6	2000	2000	112.3	135.1	95.9	1.2
7	1000	1000	291.1	348.2	98.7	1.19
7	2000	2000	289.4	344.7	98.4	1.19
8	1000	1000	758.1	907.4	99.4	1.19
8	2000	2000	756.6	908.9	99.3	1.2
9	1000	1000	2061.9	2471.3	99.9	1.19
9	2000	2000	2063.9	2477.8	99.6	1.2
10	1000	1000	5723.7	6829.9	99.8	1.19
10	1000	2000	5735.4	6835.3	99.8	1.18
11	1000	1000	17169.0	20228.3	100	1.17
11	1000	2000	17205.1	20304.0	99.9	1.18
12	1000	1000	51402.0	60177.5	100	1.17
12	1000	2000	51772.1	60818.2	99.9	1.17
13	1000	2000	1.6×10^5	1.8×10^5	100	1.16
14	1000	2000	5.5×10^5	6.5×10^5	99.9	1.18
15	1000	2000	1.5×10^6	-	-	-
16	1000	2000	5.9×10^6	-	-	-
16	2000	2000	5.8×10^6	-	-	-
32	1000	1000	6.4×10^{15}	-	-	-
64	1000	1000	1.1×10^{63}	-	-	-
128	1000	1000	7.1×10^{126}	-	-	-
256	1000	1000	2.2×10^{137}	-	-	-
256	1000	5000	5.1×10^{134}	-	-	-

Table 2: Summary of results for recovery of state contents in Arcfour-n using backtracking.

found was 2.

We draw two general conclusions from these results. First, the Monte Carlo estimation method gives a good indication (to within a factor of 2) of the expected number nodes examined by a full backtrack search. Second, the true initial state $S_0(K)$ is uniquely identified with high probability. We see from the table that for Arcfour-256 the search complexity is estimated as approximately $10^{137} \approx 2^{455}$ nodes. Then since $8^{57} = 2^{456}$, we expect the entropy of all Arcfour-256 keys to be less than 57 bytes.

5 On the distribution of initial states

The backtrack results of Table 2 are based on generating all permutations (initial states) in lexicographic order. In this section we examine another approach where the assignment permutation P is extended or contracted according to an ordering based on the distribution of elements induced by the IGSA of Arcfour in the random key model. For example, if it was known that the element 0 is very likely to occur at position i, where say i is even, we could attempt to encode this information into the search strategy. There exists some bias the distribution of elements in the initial state, as we show next.

Lemma 5.1 When $n > 2$, uniform initial state generation is impossible in Arcfour-n.

Proof. A necessary condition for uniform initial state generation is for d to be selected so that $n! | n^d$. But since $n! > n$ for $n > 2$, $n!$ has at least one prime divisor p that is not a prime divisor of n. $\square$

Thus for all practical choices of n, and all choices of key length, the initial state of Arcfour-n is distributed non-uniformly. This non-uniformity can be measured in the random key model, and we make the following definition.

Definition 5.1 Recall that the state vector is pre-initialised to $S[i] = i$, $0 \le i < n$. Let $\Pr(i \to j)$ be the probability that $S_0[j] = i$, under the random key model, $0 \le i, j < n$. $\square$

Intuitively the random key model should yield maximally random initial states as the ISGA operates with the maximum amount of random (key) input. Below we give exact formulas for $\Pr(i \to j)$, separated into the three cases $i = j$, $i > j$ and $i < j$. The formulas are valid for $n \ge 2$. We sketch the proof of the $i = j$ case but note that the other cases can be proved similarly.

Theorem 5.1 For i, $0 \le i < n$,

$$\Pr(i \to i) \quad = \quad \frac{1}{n}\left(\frac{n-1}{n}\right)^{n-i-1} \; + \; \left(\frac{n-1}{n}\right)^{i} \cdot \phi_n(i+1) \tag{2}$$

where $\phi_n(n) = n$ and for $1 \le i < n$,

$$\phi_n(i) \;\stackrel{\text{def}}{=}\; \frac{1}{n}\sum_{k=i+1}^{n}\left(\frac{n-1}{n}\right)^{k-i-1} \cdot \left[\frac{1}{n}\left(\frac{n-1}{n}\right)^{n-k} + \phi_n(k)\right].$$

Proof. (Sketch) We compute $\Pr(i \to i)$ by listing the cases where i is moved to $S[i]$ on swap k, and then not moved (swapped) again. Some observation reveals that there are only two ways that i can end up at position i: (a) either i is swapped to position i on the i-th swap, and is not involved in any of the remaining $n - i - 1$ swaps, or (b) i is not moved during the first $i - 1$ swaps, on the i-th swap it is moved to a position $j > i$, and is then swapped back to position i on the k-th swap for some k, $j \le k < n$. In this case we say that i was *moved forward* , and then returned to its desired position.

The probability of case (a) corresponds to the first term on the RHS of (2). For case (b), the $\left(\frac{n-1}{n}\right)^i$ term is the probability that i is not moved during the first i swaps, and we claim that $\phi_n(i+1)$ is the probability that it is moved forward and then returned to position i. The $(n - i - 1)$ summation terms of $\phi_n(i+1)$ represent the positions that i can be moved forward to. Then with probability $\frac{1}{n} \cdot \left(\frac{n-1}{n}\right)^{k-i-1}$, i was moved forward to position k and remained there till the k-th swap; on the k-th swap i is swapped to position i and not moved again with probability $\frac{1}{n}\left(\frac{n-1}{n}\right)^{n-k}$, or moved forward and eventually swapped back to position i with probability $\phi_n(k)$ (note that this probability is not $\phi_n(k+1)$ as the indices of k are bound as $1 \le k \le n$ while those of i are bound as $0 \le i < n$. $\square$

Corollary 5.1 For i, j, $0 \le j < i < n$,

$$\Pr(i \to j) \;=\; \frac{1}{n}\left(\frac{n-1}{n}\right)^{n-j-1} + \left(\frac{n-1}{n}\right)^{i} \cdot \left[\frac{1}{n}\left(\frac{n-1}{n}\right)^{n-i-1} + \phi_n(i+1)\right].$$

□

Corollary 5.2 For i, j, $0 \le i < j < n$,

$$\Pr(i \to j) \;=\; \frac{1}{n}\left(\frac{n-1}{n}\right)^{n-j-1} + \left(\frac{n-1}{n}\right)^{i} \cdot \sum_{k=j+1}^{n-1} \lambda_n(i,j,k) \cdot \frac{1}{n}\left(\frac{n-1}{n}\right)^{n-k-1}$$

where

$$\lambda_n(i,j,k) \;\stackrel{\text{def}}{=}\; \frac{1}{n}\left(\frac{n-1}{n}\right)^{k-1-i} + \phi_n(n-(k-i)+1).$$

□

By evaluating $\Pr(i \to j)$ from the formulas given above for $3 \le n \le 8$, it was observed that the only element uniformly distributed under the random key model was $i = 0$. That is, $\Pr(0 \to j) = 1/n$ for all j, and $\Pr(i \to j) \ne 1/n$ for all $i \ne 0$, for the values of n tested, but we have not proved this algebraically. These probabilities can be conveniently represented in matrix form.

Definition 5.2 Let $P_n = [P_{n,i,j}]$ be the $n \times n$ matrix where $P_{n,i,j} = \Pr(i \to j)$ for $0 \le i, j < n$. The *row medians* of P_n are defined by the $1 \times n$ matrix $\sigma_n = [\sigma_{n,i}]$, $0 < i < n$, where

$$\sigma_n(i) \;=\; \min_{\substack{k \\ 0 \le k < n}} \left(\sum_{j=0}^{k} P_{n,i,j} \ge \frac{1}{2} \right) \qquad (3)$$

□

Thus $\sigma_n(i)$ gives the index k in S such that i has an approximately equal chance to be moved to $S[0], S[1], \ldots, S[k]$ or $S[k+1], S[k+2], \ldots, S[n-1]$ by the ISGA in the random key model. When S is uniformly distributed then $\sigma_n(i) = \lceil \frac{n-2}{2} \rceil$ for all i, and thus it appears that $\sigma_n(0) = \lceil \frac{n-2}{2} \rceil$ in the random key model. The row medians for Arcfour-256 are plotted in Figure 4, and we note that $\sigma_{256,i} \ge 135$ for all elements $i \ge 200$, where 127 is the row median for the uniform distribution on 256 elements. Also the explicit values for P_8 are given in Appendix. The P_n matrix can be used to order the initial state search, based on the M_n matrix defined next.

Definition 5.3 Let $M_n = [M_{n,i,j}]$ be the $n \times n$ matrix where $M_{n,i,j} = k$ if the j-th largest value in the i-th column of P_n is $P_{n,k,i}$. Equal j-th largest values are sorted in M_n according to their column position in P_n. □

Thus the j-th row of M_n represents the ordering of the probabilities $\Pr(i \to j)$ from most likely to least likely, according to the random key model. We use M_n to order the backtrack search as follows. The basic step in NEXT-PERM-ELEMENT is to select $P[j]$ as the least element from $\{0, 1, \ldots, n-1\}$ that is not included in $P[0], P[1], \ldots, P[j-1]$, and BACK-PERM-ELEMENT implements the inverse of this rule. Given M_n and $P[0], P[1], \ldots, P[j-1]$, NEXT-PERM-ELEMENT can be modified to select $P[j]$ such that it is the most likely element to position i in the intitial state under the random key model, where i the is next position in S to be assigned. We will call this ordering of examining the permutations the M-ordering.

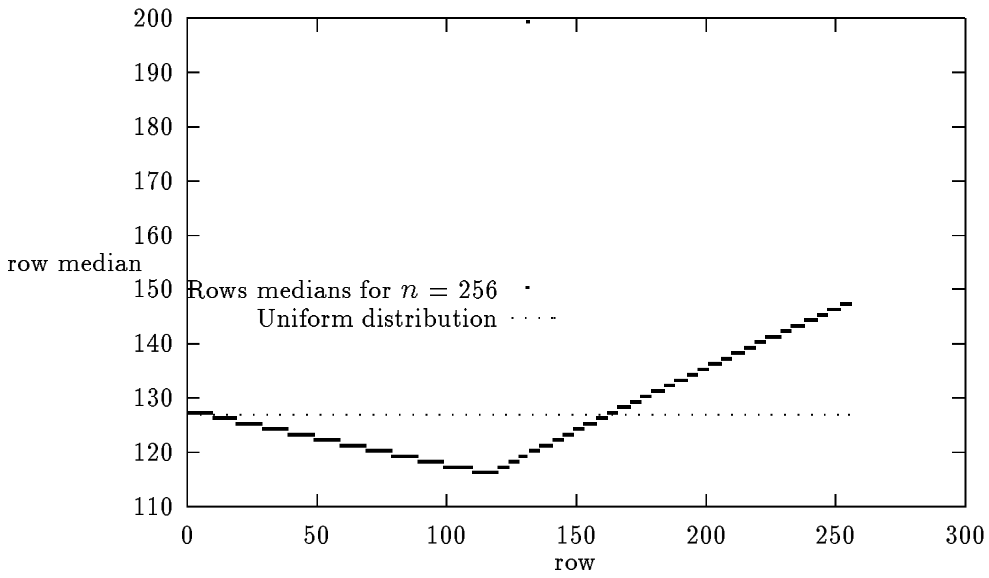

Figure 4: The row medians for Arcfour-256.

Example 5.1 For $n = 4$, P_4 and M_4 are given as

$$P_4 = \begin{bmatrix} 0.250 & 0.250 & 0.250 & 0.250 \\ 0.292 & 0.222 & 0.234 & 0.250 \\ 0.246 & 0.281 & 0.222 & 0.250 \\ 0.211 & 0.246 & 0.292 & 0.250 \end{bmatrix}, \quad M_4 = \begin{bmatrix} 1 & 0 & 2 & 3 \\ 2 & 0 & 3 & 1 \\ 3 & 0 & 1 & 2 \\ 0 & 1 & 2 & 3 \end{bmatrix}. \qquad (4)$$

The M-ordering for examining the 24 permutations on 4 elements is, from left to right, top to bottom

$$\begin{array}{cccccccccccc} (1230) & (1203) & (1032) & (1023) & (1302) & (1320) & (0231) & (0213) & (0312) & (0321) & (0132) & (0123) \\ (2031) & (2013) & (2301) & (2310) & (2130) & (2103) & (3201) & (3210) & (3012) & (3021) & (3102) & (3120) \end{array}.$$

□

Note that the M-ordering and lexicographic ordering will generate backtrack trees with the same number of nodes when all permutations are to be tested. On the other hand, our preliminary results indicate that if the backtrack search terminates when the first permutation that is consistent with B_n is found, then the M-ordering examines $4 - 5\%$ less nodes than the lexicographic ordering. Table 3 shows results for experiments for several small n, where $\mathbf{E}[L]$ is the expected number of nodes in the lexicographic ordering, and $\mathbf{E}[M]$ is the expected number nodes in the M-ordering, both when the search terminates on the first permutation found. Note that in comparison to $\mathbf{E}[ES]$, which tests all permutations in lexicographic ordering, the M-ordering searches approximately 53% less nodes. We are currently extending our computational results to larger n.

6 Conclusion

For many symmetric key ciphers, the entropy or uncertainty associated with a key K is proportional to $2^{|K|}$, where $|K|$ is its bit length. The entropy is then a measure of the time

Arcfour-n	#samples	$\mathbf{E}[L]$	$\mathbf{E}[M]$	$\mathbf{E}[M]/\mathbf{E}[L]$	$\mathbf{E}[M]/\mathbf{E}[ES]$
8	10000	468.2	449.7	0.96	0.495
9	10000	1251.9	1190.1	0.96	0.482
10	3000	3398.9	3227.8	0.95	0.472
11	3000	9547.2	9170.7	0.96	0.452
12	3000	29073.1	27500.1	0.947	0.452

Table 3: Summary of results for recovery of state contents in Arcfour-n using backtracking with M-ordering and lexicographic ordering, exiting on the first consistent permutation found.

to recover keys exhaustively. The main point of this paper has been to show that the length of an Arcfour key does not necessarily reflect the time required to recovery it (or equivalent information) exhaustively. Since each K gives rise to exactly one initial state $S_0(K)$, and the state machine is independent of K, then determining $S_0(K)$ is equivalent to determining K from the viewpoint of cryptanalysis.

We have defined a class of weak keys for Arcfour based on the notion of string periodicity, which induces collisions in the initial state for distinct keys. Arcfour keys can be selected whose entropy is far less than their bit length, and in particular keys of length $2d$ bytes can be selected whose entropy is only d bytes. For 128-bit keys this implies a weak key class of 2^{64} keys with an entropy of at most 64 bits. We advise that key generation algorithms for Arcfour filter out periodic keys, but we also note that such keys are unlikely to be chosen at random.

Our backtrack analysis provides an upper bound on the time to recover arbitrary length Arcfour keys, and at the moment our analysis indicates that this bound is proportional to keys of length at most 57 bytes. Our analysis takes advantage of the small changes from one state S_i to the next S_{i+1}, which permits $S[x]$ and $S[y]$ to be guessed and the position of a key byte b_k is determined 'for free' ($S[x]$ and $S[y]$ determine the position of b_k uniquely). Our comparisons between the Monte Carlo method of estimating the size of a backtrack tree, and actual the search of the backtrack tree for values of n in the range $4 \leq n \leq 14$, suggest the approximation for the $n = 256$ case is accurate to within a constant. However we will continue to increase our sample sets.

In §5 we presented an improvement to the backtrack method based on the distribution $\Pr(i \to j)$ of elements in the initial state given the random key model. This distribution seems to describe the fundamental working of the IGSA, and we expect future research to exploit it beyond the application of backtrack speed-up. For example, since the distribution of initial states is not uniform, we may enquire if the distribution of elements can be made 'close to uniform', assuming this is a useful property for Arcfour to possess. An obvious approach would be to apply the IGSA several times, without resetting it to the identity permutation between applications. It can be shown that P_n is a doubly stochastic matrix whose m-th power describes the distribution of elements after applying the IGSA m times (without reset) using random keys. Since P_n is non-zero and doubly stochastic, this process is known to converge to the uniform distribution and we have $\Pr(i \to j) = 1/n + \epsilon_{ij}$ where ϵ_{ij} converges geometrically to zero with m.

Quoting from [3], Arcfour is 'believed to be fully interoperable with the RC4 algorithm', where RC4 [1] is a trademark of RSA Data Security. If this is the case then our results can also be applied to RC4, which is under-analysed relative to its widespread use.

7 Appendix - P_8

Using Theorem 5.1 and corollaries 5.1 and 5.2, the row medians of P_8 are $\sigma_8 = [3, 3, 3, 3, 3, 4, 4, 4]$, while P_8 itself is

$$P_8 = \begin{bmatrix} 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 \\ 0.158 & 0.116 & 0.117 & 0.119 & 0.120 & 0.121 & 0.123 & 0.125 \\ 0.145 & 0.152 & 0.111 & 0.113 & 0.115 & 0.118 & 0.121 & 0.125 \\ 0.133 & 0.140 & 0.148 & 0.108 & 0.111 & 0.115 & 0.120 & 0.125 \\ 0.122 & 0.129 & 0.137 & 0.147 & 0.108 & 0.113 & 0.119 & 0.125 \\ 0.113 & 0.120 & 0.128 & 0.137 & 0.147 & 0.111 & 0.117 & 0.125 \\ 0.105 & 0.112 & 0.120 & 0.129 & 0.140 & 0.152 & 0.116 & 0.125 \\ 0.098 & 0.105 & 0.113 & 0.122 & 0.132 & 0.145 & 0.158 & 0.125 \end{bmatrix}.$$

The values of P_8 are graphed in Figure 5, and notice that element 0 is the only uniformly distributed element. From σ_8 we see that $0, 1, 2, 3$ and 4 are almost equally likely to appear in the first or second halves of S. In fact, for $i \in \{0, 1, 2, 3, 4\}$, $0.50 \leq \Pr(i \to j \mid 0 < j \leq 3) \leq 0.5356$.

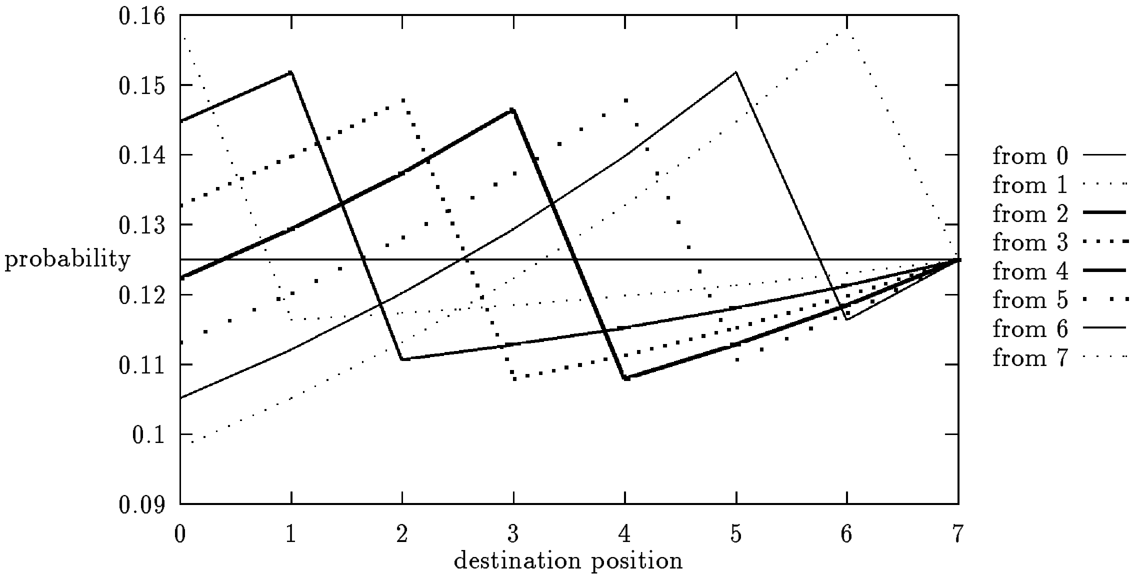

Figure 5: A graph of P_8. The i-th row of P_8 is plotted on the graph labeled 'from i' for $0 \leq i < n$. Apart from the horizontal line representing the uniform distribution of $\Pr(0 \to j)$, the general pattern for $P_{n,i}$ seems to be for $P_{n,i,j}$ to increase until $j = i - 1$, then decrease at $P_{n,i,i}$ and steadily increase again until $P_{n,i,7} = \frac{1}{8}$. Thus each graph has a 'sawtooth' appearance.

References

[1] Proprietary algorithms. See http://rsa.com/rsalabs/faq/faq_misc.html#misc.6.

[2] L. Guibas and A. M. Odlyzko. Periods in strings. *Journal of Combinatorial Theory, Series A*, 30:19–42, 1981.

[3] K. Kaukonen and Thayer. R. A stream cipher encryption algorithm arcfour. Internet Draft, draft-kaukonen-cipher-arcfour-01.txt, July 1997, available at ftp://ftp.ietf.org/internet-drafts/draft-kaukonen-cipher-arcfour-01.txt.

[4] Thayer. R. The ESP ARCFOUR algorithm. Internet Draft, draft-ietf-ipsec-ciph-arcfour-00.txt, June 1997, available at ftp://ftp.ietf.org/internet-drafts/draft-ietf-ipsec-ciph-arcfour-00.txt.

[5] E. M. Reingold, J. Nievergeld, and N. Deo. *Combinatorial Algorithms: Theory and Practice.* Prentice-Hall, 1976.