Iterative Probabilistic Reconstruction of RC4 Internal States

Jovan Dj. Golić* and Guglielmo Morgari†

Abstract

It is shown that an improved version of a previously proposed iterative probabilistic algorithm, based on forward and backward probability recursions along a short keystream segment, is capable of reconstructing the RC4 internal states from a relatively small number of known initial permutation entries. Given a modulus N, it is argued that about $N/3$ and $N/10$ known entries are sufficient for success, for consecutive and specially generated entries, respectively. The complexities of the corresponding guess-and-determine attacks are analyzed and, e.g., for $N = 256$, the data and time complexities are (conservatively) estimated to be around $D \approx 2^{41}$, $C \approx 2^{689}$ and $D \approx 2^{211}$, $C \approx 2^{262}$, for the two types of guessed entries considered, respectively.

Keywords Guess-and-determine attacks, internal state reconstruction, iterative algorithms, probabilistic cryptanalysis, RC4

1 Introduction

RC4 [10] is a well known software-oriented stream cipher, which is widely used for data encryption on the Internet within the standardized SSL/TLS protocol, in wireless networks within the standardized WEP and WPA protocols, as well as in many commercial products. Given a parameter n (nominally, $n = 8$), the internal state of RC4 consists of a permutation S of 2^n different n-bit words and two pointer n-bit words i and j, which, at each time, define the positions of two words in the permutation to be swapped to produce the permutation at the next time. The i pointer is generated by a counter and both pointers are set to to zero initially. At each time, the output of RC4 is an n-bit word which is taken from an appropriate position in the permutation table S. For experimental investigation, it is convenient to define a more general RC4 family [8], where, given an integer N, the internal state consists of a permutation of all elements of $\mathbb{Z}_N = \{0, 1, \ldots, N-1\}$ and two pointers taking values in $\mathbb{Z}_N$, while all the operations remain the same, with the modulus N instead of 2^n.

*J. Dj. Golić is with Security Innovation, Telecom Italia, Via Reiss Romoli 274, 10148 Turin, Italy (Email: jovan.golic@telecomitalia.it).

†G. Morgari is with Telsy Elettronica e Telecomunicazioni, Corso Svizzera 185, 10149 Turin, Italy (Email: guglielmo.morgari@telsy.it).

In practical applications, RC4 keystream generator is used together with a key scheduling or initialization algorithm, which combines a preshared secret key with an initial value (IV), transmitted in the clear, into a secret session key defining the initial permutation for keystream generation. In [2] and some follow-up papers, it is shown that the key scheduling algorithm [10] is vulnerable to secret key reconstruction attacks from a number of known IV's and the corresponding initial keystream bytes if the IV is concatenated to the preshared secret key. In [9] and [1], it is shown that relatively short secret keys can be reconstructed from the initial permutation of RC4 by a probabilistic analysis.

The huge internal state size of $\log_2(N!N^2)$ bits very likely results in high periods and renders the complexity of time-data-memory tradeoff attacks far beyond the practical limits. Due to the simplicity of the next-state function, RC4 keystream generator is shown to be vulnerable to a number of statistical distinguishing attacks, starting from a linear statistical distinguisher [3] up to a digraph repetition distinguisher [7]. However, this is not the case when it comes to published attacks aiming at reconstructing an internal state from known keystream. Note that the initial state (permutation) of RC4 can be recovered from any reconstructed internal state by reversing the next-state function. The internal state reconstruction problem for RC4 is thus very difficult and any progress in this direction is practically very important.

Two algorithms for reconstructing the initial state or any internal state of RC4 from a short keystream segment are proposed in [5]. One is a search algorithm consisting in the sequential search through the values of the internal state components that are consistent with a given keystream segment, with backtracking in case of found contradictions. Its approximate time complexity can be computed recursively and turns out to be slightly smaller than the time of searching through the square root of all possible initial states. For example, for $N = 256$, the complexity is about 2^{779}, to be compared with the internal state size of about 1700 bits. If a number of consecutive entries of the initial permutation table are assumed to be known, then the complexity reduces and tables of the resulting systematically computed complexities are given in [6].

The other algorithm from [5] is a probabilistic algorithm consisting in a recursive forward computation of the approximate *a posteriori* probabilities of the internal state components given a keystream segment. Its time complexity is about 2^{6n} (N^6) steps each consisting of computing a small number of real number operations. However, it can recover the initial permutation only if a sufficient number of its consecutive entries are assumed to be known, which is estimated to be about 155 for $N = 256$. So, the resulting guess-and-determine attack appears to be less effective than the one corresponding to the search algorithm.

In a recent unrefereed publication [8], the starting point is an observation that the complexity of the search algorithm [5] significantly reduces if an internal state is such that both the pointers are known during the search. To this end, it is proposed to search for special internal states that contain a pattern consisting of two pointer values and a number, d, of known permutation entries that result in w known consecutive values of the j pointer, where the ratio w/d is as large as possible. Such patterns, which are here called (d, w) patterns, can be obtained in precomputation time with the complexity much smaller than the attack complexity and it is argued that constructing such patterns may be feasible for $N = 256$. To find such a pattern, a long keystream, i.e., a high data complexity is needed. However, in order to detect a chosen pattern along a keystream without running the search algorithm,

some additional properties of the pattern are required and, because of that, the conjectured existence of such patterns for larger N (e.g., for $N = 256$) may be questionable. Accordingly, the conjectured time and data complexities around 2^{241} for $N = 256$ are not very reliable. Besides, it is overlooked that the additional properties are not shift invariant, which implies that the i pointer value also needs to be matched apart from the j pointer value. Hence, the data complexity is in fact N times higher, i.e., around 2^{248}.

Another probabilistic algorithm for the internal state reconstruction from a short keystream is proposed in [4]. It is pointed out that in the probabilistic model in which the internal state components are assumed to be independent, the expressions for the *a posteriori* probabilities given in [5] are in fact approximations since the two so-called 'change of state' and 'observation of output symbol' effects should be considered simultaneously rather than separately. Accordingly, the new expressions derived in [4] are expected to be more effective, especially in the case when the j pointer values are unknown. Another observation in [4] is that since the assumed probabilistic model is only approximate, it would make sense to recursively compute the *a posteriori* probabilities also in the backward direction, apart from the forward direction. It is hence suggested in [4] that the resulting iterative algorithm consisting of a number of rounds, where, in each round, the *a posteriori* probabilities are first computed forwards and then backwards, could be more effective in that a smaller number of permutation entries may be required to be known for success, while keeping essentially the same time complexity of about N^5 steps per iteration (i.e., about $2N^6$ steps per round). The conducted preliminary experiments for $N = 8, 16$ indicate that this indeed may be the case, but are clearly insufficient for extrapolation to larger N.

The main objective of this paper is to continue the study of iterative probabilistic algorithms, to propose a number of improvements, and to estimate more reliably the number of known permutation entries that are required for success in the guess-and-determine scenario. This is achieved for two types of patterns: the consecutive d patterns, where a number d of consecutive permutation entries are assumed to be known, and the special (d, w) patterns [8], where d known permutation entries result in w known consecutive values of the j pointer and w is maximal. Systematic experimental analysis, conducted for $N \leq 68$, leads us to conjecture that asymptotically, for larger N, about $d = N/3$ and $d = N/10$ permutation entries are sufficient for success in the case of consecutive and special patterns, respectively. For smaller N, less entries are required in both the cases. The data and time complexities of the corresponding guess-and-determine internal state reconstruction attacks are then estimated to be around $D = 2^9$, $C = 2^{717}$ and $D = 2^{215}$, $C = 2^{266}$, for the basic versions, and $D \approx 2^{41}$, $C \approx 2^{689}$ and $D \approx 2^{211}$, $C \approx 2^{262}$, for the optimized versions, respectively.

A description of RC4 keystream generator is briefly recalled in Section 2 and the forward and backward recursions for the iterative probabilistic analysis are outlined in Section 3. The corresponding improved iterative probabilistic algorithm is described in Section 4 and the obtained experimental results are presented in Section 5, whereas the supporting graphs of the success probabilities are given in the Appendix. The basic and optimized versions of the resulting guess-and-determine internal state reconstruction attacks are discussed in Section 6. A summary and conclusions are given in Section 7.

2 Description of RC4 Keystream Generator

Given a modulus N, the internal state of RC4 at time t consists of a permutation (table) $S_t = (S_t[l])_{l=0}^{N-1}$ of N elements of $\mathbb{Z}_N$ and two pointers i_t and j_t taking values in $\mathbb{Z}_N$. The output Z_t of RC4 at time t is also an integer from $\mathbb{Z}_N$. Initially, we have $i_0 = j_0 = 0$, whereas the initial permutation S_0 is defined by the secret key in a way which is irrelevant for this paper. For every $t \geq 1$, the next-state and output functions of RC4 are defined by

$$i_t \ = \ i_{t-1} + 1, \quad j_t \ = \ j_{t-1} + S_{t-1}[i_t] \tag{1}$$

$$S_t[i_t] \ = \ S_{t-1}[j_t], \quad S_t[j_t] \ = \ S_{t-1}[i_t] \tag{2}$$

$$Z_t \ = \ S_t[S_t[i_t] + S_t[j_t]] \tag{3}$$

respectively, where the additions are modulo N. The output sequence is thus $Z = (Z_t)_{t=1}^{\infty}$.

3 Iterative Probabilistic Analysis

In a probabilistic model where the initial permutation S_0 is chosen randomly according to the *a priori* uniform probability distribution, our ideal objective would be to compute the *a posteriori* probabilities $\Pr\{S_0[i]|Z_1^T\}$, $0 \leq i \leq N-1$, given a kestream segment Z_1^T, where $Z_{t_1}^{t_2} = Z_{t_1}, \ldots, Z_{t_2}$. For simplicity, unless specified otherwise, we keep the same notation for random variables and their values. If $T \approx N$, then these probabilities should be close to 0 or 1, which would imply a successful reconstruction of the initial permutation. Since the exact computation does not appear to be feasible, we proceed by computing recursively in time the approximate *a posteriori* probabilities for the internal state components, i.e., for the individual permutation entries $S_t[i]$, $0 \leq i \leq N-1$, and the pointer j_t, under *the independence assumption* that, at each time, the components are mutually independent, except for the constraint that the permutation entries have to be all different.

More precisely, according to [4], we use the forward recursions for $\Pr\{S_t[i]|Z_1^t\}$, $0 \leq i \leq N-1$, and $\Pr\{j_t|Z_1^t\}$, for $t = 1, \ldots, T$, and the backward recursions for $\Pr\{S_t[i]|Z_{t+1}^T\}$, $0 \leq i \leq N-1$, and $\Pr\{j_t|Z_{t+1}^T\}$, for $t = T-1, \ldots, 0$. The forward recursions can start from arbitrary *a priori* probability distributions $\Pr\{S_0[i]\}$, $0 \leq i \leq N-1$, and $j_0 = 0$ (or $j_0 = j$, for any given j), while the backward recursions can start from arbitrary *a priori* probability distributions $\Pr\{S_T[i]\}$, $0 \leq i \leq N-1$, and $\Pr\{j_T\}$. The essence of the iterative algorithm [4] is to compute initially the forward recursions starting from the uniform probability distributions and then to alternatively compute the backward and forward recursions along the keystream segment, each time using the last *a posteriori* probability distributions computed by the previous forward/backward recursions as the *a priori* probability distributions for the current backward/forward recursions. Computing in both directions is expected to be advantageous because the recursions and the resulting probabilities are approximate rather than exact, due to the underlying independence assumption.

3.1 Forward Recursions

Let $P_t[i](k) = \Pr\{S_t[i] = k|Z_1^t\}$, $p_t(k) = \Pr\{j_t = k|Z_1^t\}$, $P_t = (P_t[i](k))_{i,k=0}^{N-1}$, and $p_t = (p_t(k))_{k=0}^{N-1}$, for $0 \le t \le T$, where, initially, $P_0[i](k) = \Pr\{S_0[i] = k\}$ and $p_0(k) = \delta_{0,k}$, where $\delta_{i,k}$ is the Kronecker delta symbol (or $p_0(k) = \delta_{j,k}$). P_t and p_t are called the probability matrices, where P_t has rows indexed by i and columns indexed by k. Then the forward recursion for $P_t[i](k)$, in terms of P_{t-1} and p_{t-1}, can be put into the form:

$$P_t[i](k) \propto \sum_{a,b,c} \Pr\{S_t[i] = k, Z_t \mid j_{t-1} = a - b, S_{t-1}[i_t] = b, S_{t-1}[a] = c, S_{t-1}[i] = k, Z_1^{t-1}\} \cdot p_{t-1}(a-b) P_{t-1}[i_t](b) \frac{P_{t-1}[a](c)}{1 - P_{t-1}[a](b)} \frac{P_{t-1}[i](k)}{1 - P_{t-1}[i](b) - P_{t-1}[i](c)} \quad (4)$$

(where formally $0/0 = 0$). It is assumed that $P_{t-1}[a](c)$ and $P_{t-1}[i](c)$ are included only if $a \neq i_t$ and that $P_{t-1}[i](k)$ is included only if $i \neq i_t, a$ (where $j_t = a$). The proportionality constant is determined from the normalization requirement

$$\sum_k P_t[i](k) = 1. \quad (5)$$

Similarly, the forward recursion for $p_t(k)$, in terms of P_{t-1} and p_{t-1}, can be put into the following form:

$$p_t(k) \propto \sum_{b,c} \Pr\{j_t = k, Z_t \mid j_{t-1} = k - b, S_{t-1}[i_t] = b, S_{t-1}[k] = c, Z_1^{t-1}\} \cdot p_{t-1}(k-b) P_{t-1}[i_t](b) \frac{P_{t-1}[k](c)}{1 - P_{t-1}[k](b)}, \quad (6)$$

where $P_{t-1}[k](c)$ is included only if $k \neq i_t$, and the proportionality constant is determined from the normalization requirement

$$\sum_k p_t(k) = 1. \quad (7)$$

The conditional probabilities in the first lines of (4) and (6) are derived in [4], by distinguishing between 15 and 5 cases, respectively, depending on the values of i, i_t, j_t, and $\sigma = S_{t-1}[i_t] + S_{t-1}[j_t]$ being equal to or different from each other. Each iteration of (4) and (6) requires N^5 and N^3 computational steps, where each step consists of a small number of real number operations, respectively.

3.2 Backward Recursions

Let $P'_t[i](k) = \Pr\{S_t[i] = k|Z_{t+1}^T\}$, $p'_t(k) = \Pr\{j_t = k|Z_{t+1}^T\}$, $P'_t = (P'_t[i](k))_{i,k=0}^{N-1}$, and $p'_t = (p'_t(k))_{k=0}^{N-1}$, for $0 \le t \le T$, where, initially, $P'_T[i](k) = \Pr\{S_T[i] = k\}$ and $p'_T(k) = \Pr\{j_T = k\}$. Then the backward recursions for $P'_t[i](k)$ and $p'_t(k)$, in terms of P'_{t+1} and p'_{t+1},

can be put into the form, respectively:

$$P'_t[i](k) \propto \sum_{a,b,c} \Pr\{S_t[i] = k, Z_{t+1} \mid j_{t+1} = a, S_{t+1}[i_{t+1}] = b, S_{t+1}[a] = c, S_{t+1}[i] = k, Z_{t+2}^T\} \cdot p'_{t+1}(a) P'_{t+1}[i_{t+1}](b) \frac{P'_{t+1}[a](c)}{1 - P'_{t+1}[a](b)} \frac{P'_{t+1}[i](k)}{1 - P'_{t+1}[i](b) - P'_{t+1}[i](c)} \quad (8)$$

$$p'_t(k) \propto \sum_{b,c} \Pr\{j_t = k, Z_{t+1} \mid j_{t+1} = k + c, S_{t+1}[i_{t+1}] = b, S_{t+1}[k + c] = c, Z_{t+2}^T\} \cdot p'_{t+1}(k + c) P'_{t+1}[i_{t+1}](b) \frac{P'_{t+1}[k + c](c)}{1 - P'_{t+1}[k + c](b)}. \quad (9)$$

In (8), it is assumed that $P'_{t+1}[a](c)$ and $P'_{t+1}[i](c)$ are included only if $a \neq i_{t+1}$ and that $P'_{t+1}[i](k)$ is included only if $i \neq i_{t+1}, a$ (where $j_{t+1} = a$). In (9), $P'_{t+1}[k + c](c)$ is included only if $k + c \neq i_{t+1}$ (where $j_{t+1} = k + c$). The proportionality constants are determined from analogous normalization requirements. The conditional probabilities in the first lines of (8) and (9) are derived in [4], by distinguishing between 15 and 5 cases, respectively, depending on the values of i, i_{t+1}, j_{t+1}, and $\sigma = S_{t+1}[i_{t+1}] + S_{t+1}[j_{t+1}]$ being equal to or different from each other. Each iteration of (8) and (9) requires N^5 and N^3 computational steps, where each step consists of a small number of real number operations, respectively.

4 Iterative Probabilistic Algorithms

The basic iterative algorithm [4] consists of rounds, where in each round the last computed probability matrices P_T and p_T by the forward recursions are used as the initial probability matrices P'_T and p'_T for the backward recursions and the last computed probability matrix P'_0 by the backward recursions is used as the initial probability matrix P_0 for the forward recursions, while p_0 is kept unchanged in each round (as j_0 is fixed), respectively. One pass of the forward and backward recursions is called Forward and Backward, respectively. The time complexity of each round is thus around $2N^6$ steps, provided that $T \approx N$.

In the first round, if no element of the initial permutation S_0 is known, then the initial probability matrix P_0 of Forward would be defined as uniform, i.e., $P_0 = U$, where all elements of the (uniform) matrix U are equal to $1/N$. However, experiments show that such an algorithm cannot be successful. Accordingly, in the guess-and-determine scenario, a number of elements of S_0 need to be known or guessed correctly. In [5], [4], the known elements are $S_0[i]$, $1 \leq i \leq d$. In general, if d elements of S_0 are known, then the initial probability matrix P_0 is defined as $P_0 = G_d$, where the (guessed) matrix G_d contains d 1's corresponding to the known elements, $N - 1$ 0's in each row and column corresponding to a known element, and the remaining entries equal to $1/(N - d)$. After a (small) number of rounds, the last obtained probability matrix P'_0 represents a soft estimate of the initial permutation S_0 that produced the given keystream segment Z_1^T. A hard estimate of S_0 is then obtained by rounding off the probabilities to the binary values.

The main problem addressed in this paper is to determine the minimal d that is sufficient for a successful reconstruction of S_0. To this end, on the basis of some theoretical considerations and conducted numeorous experiments, we first propose a number of improvements to the basic iterative algorithm described above. The iterative probabilistic algorithm incorporating the improvements is denoted as IPA.

Consecutive and Special Patterns: An internal state pattern is defined as a partially specified internal state. More precisely, for a pattern $\mathcal{P} = (i, j, (p_k, v_k)_{k=1}^d)$, an internal state at time t, (i_t, j_t, S_t), is said to be compliant with $\mathcal{P}$ (or vice versa) if $(i_t, j_t) = (i, j)$ and $S_t[p_k] = v_k, 1 \leq k \leq d$. An initial consecutive pattern [5], [4] is then defined as $\mathcal{P} = (0, 0, (k, v_k)_{k=1}^d)$, and a consecutive pattern is defined as $\mathcal{P} = (i, j, (i + k, v_k)_{k=1}^d)$, where the addition is modulo N. Such patterns are here also called d patterns. It follows that if an internal state is compliant with a consecutive pattern, then the subsequent d values of the j pointer are uniquely determined, which means that one can then update the (partially known) internal state at least d times, before the value of the j pointer is lost. As already noted in [5], if a value of the j pointer is known, then the forward iterative update of the *a posteriori* probabilities is more effective, and the same is true for the backward iterative update. This is why consecutive patterns are important, and initial consecutive patterns are more effective as the value of the j pointer then does not have to be guessed. Since consecutive patterns are generic, the required keystream length is very short, i.e., $T \approx N$.

In addition, one may use the special (d, w) patterns proposed in [8] which ensure that the subsequent w values of the j pointer are uniquely determined. If w is maximal given d, then such a pattern is called in [8] a maximum (w-generative d-order) pattern. A reliable conjecture regarding maximum patterns justified in [8] is that $w/d \approx 6$, for moderately large d, while, for smaller d, this ratio is somewhat smaller than 6. Accordingly, maximum patterns are expected to considerably increase the effectivenes of iterative probabilistic algorithms, in comparison with consecutive patterns. However, since a special pattern is fixed, then a very long keystream segment is required for such a pattern to be found along the keystream.

Hard Preprocessing: Given a keystream segment Z_1^T, we determine the extended pattern $\mathcal{P}_{\bar{d},\bar{w}}$, for both consecutive and special patterns, $\bar{w} \geq \bar{d} \geq d$, where the extended parameters $\bar{w}$ and $\bar{d}$ are defined as the maximal numbers of the pointer j values and the permutation entries at any time $0 \leq t \leq \bar{w}$, respectively, that are uniquely determined by Z_1^T if the guessed pattern $\mathcal{P}$ is compliant with S_0. Consequently, $\bar{d}$, $\bar{w}$, and $\mathcal{P}_{\bar{d},\bar{w}}$ are computed in the preprocessing stage, by running the next-state function forwards and backwards for a (small) number of rounds as long as the j pointer is known and by updating the internal state at each time t according to Z_t and (3) if $S_t[i_t]$ is known (where $S_t[j_t] = S_{t-1}[i_t]$ is known since j_t and j_{t-1} are known). This way, $\bar{d}$ values of S_t are determined and stored for each $0 \leq t \leq \bar{w}$. The preprocessing algorithm is stopped if the internal state is not updated any more in a full forward or backward pass. A pattern $\mathcal{P}$ that is not compliant with S_0 may yield inconsistencies already at this stage.

Hard Reset of Backward: Known j pointer values are included automatically in Forward. It is advantageous to include them also in Backward, and this can be achieved by hard resetting Backward only at time $t = \bar{w}$, by replacing p'_t and the corresponding rows and columns of P'_t by the binary vectors corresponding to known j_t and known entries of S_t determined by hard preprocessing, respectively, and by renormalizing the rows.

Soft Preprocessing: Run one round of Forward from $P_0 = G_{\bar{d}}$ and one round of Back-

ward from $P'_T = U$ and uniform p'_T, and then perform termwise multiplication of the computed probability matrices ($P_t \cdot P'_t$ and $p_t \cdot p'_t$) and row normalization to obtain $\tilde{P}_t$ and $\tilde{p}_t$, respectively, for each $0 \leq t \leq T$. It is assumed that hard reset of Backward is applied. The rationale for this operation can be found in the approximate expressions

$$\Pr\{S_t[i]|Z_1^T\} \propto \Pr\{S_t[i]|Z_1^t\} \cdot \Pr\{S_t[i]|Z_{t+1}^T\} \tag{10}$$

$$\Pr\{j_t|Z_1^T\} \propto \Pr\{j_t|Z_1^t\} \cdot \Pr\{j_t|Z_{t+1}^T\}. \tag{11}$$

The expressions are only approximate, because the Markov chain property does not hold for the internal state components, but only for the internal state as a whole. The obtained soft reset probability matrices are used as described below.

Modifying Initial Probability Matrix of Forward: In the first round, start Forward from $\tilde{P}_0$, and in subsequent rounds, start Forward from the last permutation probability matrix of Backward modified by $\tilde{P}_0$ by using termwise multiplication ($P'_0 \cdot \tilde{P}_0$) and row normalization.

Soft Zero Row Reset: During the iterations of the iterative algorithm, even if the guessed pattern is compliant with S_0, at times it may happen that the all-zero rows occur in the probability matrices of Forward or Backward. This, wrong convergence to partially inconsistent values, which is due to repeated iterations of approximate recursions for the probabilities, may be dealt with by resetting each all-zero row occuring at time t with the corresponding row of the soft reset probability matrices $\tilde{P}_t$ and $\tilde{p}_t$, except when all the rows of P_t or P'_t are zero, in which case the iterative algorithm is stopped.

Soft Inconsistent Column Reset: The permutation probability matrices P_t and P'_t of Forward and Backward are approximations of the corresponding matrices of exact probabilities for individual permutation entries at time t, respectively. These exact probability matrices, which are infeasible to compute directly, are doubly stochastic matrices, satisfying the condition that the sum of entries in each row and column is equal to 1. Experiments confirm that it is more effective to perform row than column normalization in each iteration of the iterative algorithm, as described above. However, an improvement is obtained if at each time t, inconsistent columns are replaced by the corresponding columns of the soft reset probability matrix $\tilde{P}_t$, and all the rows are then renormalized. A column is considered inconsistent with a doubly stochastic matrix if the sum of entries is too small or too large (e.g., if the sum is out of the range $[0.1, 1.9]$).

5 Experimental Results

The main objective of the experiments was to estimate the minimal d that is needed for the IPA to successfully recover the whole initial permutation S_0 from a short keystream segment of length $T = N$, provided that S_0 is compliant with a known initial consecutive d pattern or a known special maximum (d, w) pattern [8]. Implications for guess-and-determine cryptanalytic attacks, where the patterns need to be guessed, are discussed in Section 6.

Accordingly, we conducted two sets of experiments, related to initial consecutive and special maximum patterns, respectively. In the former case, for each assumed value of N

and each assumed value of d, we ran the IPA for a number of randomly generated S_0 and then estimated the corresponding probabilities of success as a function of d given N, where a d pattern is derived from S_0. Similarly, in the latter case, for each assumed (d, w) pattern and each assumed value of N, we ran the IPA for a number of randomly generated S_0 and (i_0, j_0) that are compliant with the pattern, and then estimated the corresponding probabilities of success as a function of N given a (d, w) pattern. An experiment was considered to be successful if the IPA was able to reconstruct S_0. The values of the parameters are chosen according to the time complexity N^6, given the PC computational power available. For simplicity, the number of rounds was 5 in all the experiments, but it was noted that increasing the number of rounds tended to improve the success probabilities to some extent.

5.1 Initial Consecutive Patterns

The values of N, d tested for initial consecutive patterns are given in Table 1. For each N, the values of d are chosen so as to encompass the success probabilities close to 0.5 or smaller. For each N, the average number of conducted experiments per each value of d is also shown.

Table 1: Experiments with initial consecutive patterns

N	16	24	32	40	48	56	64
d	$[2, 5]$	$[5, 8]$	$[8, 11]$	$[10, 15]$	$[12, 18]$	$[14, 20]$	$[16, 22]$
# Exp	1000	100	100	38	21	19	24
d_0	4.10	6.86	9.72	12.74	15.89	18.93	21.59
a	0.26	0.34	0.42	0.63	0.59	0.82	0.71

Graphs of the obtained success probabilities are displayed in Fig. 1 in the Appendix. Each graph is expected to be an increasing function of d that reaches the value 0.5 for a threshold $d = d_0$, to be numerically estimated. To this end, for d around d_0, such a graph can be approximated as the exponential function $p(d) = (1 + \hat{a}^{d-d_0})^{-1}, \hat{a} < 1$, where the parameters d_0 and $\hat{a}$, which depend on N, can be estimated by applying the standard linear regression method to the function $(d - d_0) \log \hat{a} = \log (p(d)^{-1} - 1)$. However, to find an approximation for large $|d - d_0|$, which will be used in Section 6, it is more realistic to use the exponential function $p(d) = (1 + a^{(d-\hat{d}_0)^2 \operatorname{sgn}(d-\hat{d}_0)})^{-1}, a < 1$, which corresponds to the cumulative distribution of a normal distribution. The linear regression method is then applied to the function $\sqrt{\log a^{-1}}(d - \hat{d}_0) = \operatorname{sgn}(p(d) - 0.5) \sqrt{|\log (p(d)^{-1} - 1)|}$.

The computed values of d_0 and a, as functions of N, are shown in Table 1. It thus appears that d_0/N is a slowly increasing function of N that stabilizes around $1/3$ already for $N \geq 48$. On the other hand, it is natural to expect that for large N, the minimal d_0 required for success is proportional to N.

Conjecture 1 *For consecutive d patterns, the minimal value of d that is sufficient for the IPA success probability to be about* 0.5 *is around $N/3$ for $N \geq 48$.*

In each experiment, we also computed the parameters $\bar{w}$ and $\bar{d}$ in the hard preprocessing stage and found that in most cases, $\bar{w}$ was equal to or just slightly bigger than d and that the success probability was not significantly correlated to $\bar{d}$.

5.2 Special Maximum Patterns

The maximum (d, w) patterns and N tested are given in Table 2. Given a pattern, the values of N are chosen so as to encompass the success probabilities close to 0.5 or smaller. The description of the patterns is given in [8] and is irrelevant for this paper. For each pattern, the average number of conducted experiments per each value of N is also shown.

Table 2: Experiments with special maximum patterns

(d, w)	$(4, 15)$	$(5, 21)$	$(6, 27)$	$(7, 31)$	$(8, 37)$
N	$[24, 30]$	$[34, 40]$	$[43, 48]$	$[50, 56]$	$[60, 68]$
# Exp	100	59	21	23	11
N_0	27.67	35.86	44.24	51.08	59.07
b	0.82	0.82	0.84	0.82	0.98

Graphs of the obtained success probabilities are displayed in Fig. 2 in the Appendix. Each graph is expected to be a decreasing function of N that reaches the value 0.5 for a threshold $N = N_0$, to be numerically estimated. To this end, for N around N_0, such a graph can be approximated as the exponential function $p(N) = (1 + \hat{b}^{N_0 - N})^{-1}, \hat{b} < 1$, where the parameters N_0 and $\hat{b}$, which depend on the special pattern, can be estimated by applying the linear regression method to the function $(N_0 - N) \log \hat{b} = \log (p(N)^{-1} - 1)$. Similarly, an approximation for large $|N - N_0|$, which will be used in Section 6, is obtained by using the exponential function $p(N) = (1 + b^{(N - \hat{N}_0)^2 \operatorname{sgn}(\hat{N}_0 - N)})^{-1}, b < 1$, and by applying the linear regression method to $\sqrt{\log b^{-1}}(\hat{N}_0 - N) = \operatorname{sgn}(p(N) - 0.5) \sqrt{|\log (p(N)^{-1} - 1)|}$.

The computed values of N_0 and b, for the tested special patterns, are shown in Table 2. The obtained values of N_0 are consistent with a conclusion that N_0/w is a slowly increasing function of w that stabilizes around $5/3$ for large w. (We also performed experiments for other special (d, w) patterns, which confirm that the success probability predominantly depends on w and weakly on d.) If we utilize a plausible conjecture [8] for maximum patterns that $w/d \approx 6$ for large d, then we obtain that $N_0/d \approx 10$ is sufficient for success for large d. On the other hand, it is natural to expect, also for maximum patterns, that the minimal d_0 required for success is proportional to N, for large N.

Conjecture 2 *For maximum (d, w) patterns and large N, the minimal value of w that is sufficient for the IPA success probability to be about 0.5 is around $0.6N$ and, accordingly, the corresponding minimal value of d is around $N/10$ (e.g., for $N \geq 256$).*

In each experiment, we also computed the parameters $\bar{w}$ and $\bar{d}$ in the hard preprocessing stage and found that in many cases, $\bar{w}$ was considerably bigger than w and that the success probability, while not significantly correlated to $\bar{d}$, was somewhat correlated to $\bar{w}$.

6 Guess-and-Determine Attacks

6.1 Basic Versions

In the guess-and-determine scenario, no element of S_0 is known and, hence, the initial consecutive d patterns need to be tested exhaustively in order to find the one compliant with S_0, which will then result in the reconstruction of S_0 with a success probability p, depending on N and the adopted value of d. As there are $N^{(d)} = N(N-1)\cdots(N-d+1)$ such patterns, the IPA algorithm, with a maximum number r of rounds, is then repeated $N^{(d)}$ times on the same keystream segment of length $T = N$. Note that each guessed pattern is easily tested for correctness by comparing the given keystream segment with the one produced from the reconstructed S_0. The required data and time complexities are then $D = N$ and $C = 2rN^6N^{(d)}$, respectively. For simplicity, we here neglect the reduction in C due to the fact that the patterns not compliant with S_0 can possibly be detected already in the hard preprocessing stage or before the completion of all r rounds, when the all-zero probability matrix is obtained. To achieve the success probability close to 1, the algorithm can be repeated $O(1/p)$ times on distinct keystream segments. The average time and data complexities are thus increased $1/p$ times, which is very small if p is around 0.5.

For special maximum (d, w) patterns, the situation is essentially different, because the IPA needs to be repeated for every position in the keystream sequence until an internal state compliant with the assumed pattern is found. The guesses to be tested thus relate to the times. In fact, since the (d, w) property is preserved under the shift operation $(i+\tau, j+\tau, (p_k+\tau, v_k)_{k=1}^{d})$, where the addition is modulo N, at time t, a shifted pattern such that $i + \tau = i_t$ is tested. Under the assumption that the cycle of internal states generated from S_0 is sufficiently long, the required keystream length is on average $D = NN^{(d)}$, because the value of the j pointer needs to be matched too. The average data and time complexities are then $D = NN^{(d)}/p$ and $C = 2rN^7N^{(d)}/p$, respectively, where p is the success probability.

Under the Conjectures 1 and 2, for the consecutive and maximum patterns, the required values of d are $d \approx N/3$ and $d \approx N/10$, for moderately large N, respectively. In this case, we can use the numerical approximation $N^{(d)} \approx N^d e^{-d^2/(2N)}$. The average data and time complexities then become $D = 2N$, $C \approx 4rN^{N/3+6}e^{-N/18}$ and $D \approx 2N^{N/10+1}e^{-N/200}$, $C \approx 4rN^{N/10+7}e^{-N/200}$, respectively.

For example, for $N = 256$, we can assume that $d = 86$ and $d = 26$ for the two types of patterns, respectively. For $r = 5$, we then get $D = 2^9$, $C = 2^{716.95}$ and $D = 2^{215.1}$, $C = 2^{266.43}$, respectively.

6.2 Optimized Versions

For initial consecutive patterns, instead of choosing $d \approx d_0$ and the success probability p around 0.5, we can choose a smaller d and the success probability p much smaller than 0.5. The average data complexity then increases $1/p$ times, i.e., $D = N/p$, whereas the average time complexity $C = 2rN^6N^{(d)}/p$ may decrease or increase, depending on how fast p decreases as d decreases. Experimental results from Section 5.1 indicate that $1/p$ can be approximated as $1/p \approx a^{-(d-d_0)^2}$, for $d < d_0$, where, in view of Table 1 and also theoretically, the parameter $a < 1$ is expected to slowly increase with N. Therefore, C can

be approximated as:

$$C \approx 2re^{-d^2/(2N)}N^{d+6}a^{-(d-d_0)^2}. \tag{12}$$

Given the values from Table 1, it follows that C decreases as d decreases from d_0, reaches a minimum point for $d = d_{\text{opt}}$, and then increases as d further decreases. This means that by choosing $d = d_{\text{opt}}$, we obtain the minimal time complexity $C_{\min}$ at the cost of increasing the data complexity. If we neglect the slowly varying factor $e^{-d^2/(2N)}$, then we get that $d_0 - d_{\text{opt}} \approx \log_2 N/(2\log_2 a^{-1})$ and, hence,

$$C_{\min} \approx 2rN^6N^{(d_0)}N^{-\log_2 N/(4\log_2 a^{-1})}, \tag{13}$$

whereas $D_{\text{opt}} \approx N^{1+\log_2 N/(4\log_2 a^{-1})}$.

For example, for $N = 256$, in light of Table 1, we can (conservatively) assume that $a \approx 0.7$, so that $d_0 - d_{\text{opt}} \approx 8$. Then, for $d = 78$, we get optimized complexities $D_{\text{opt}} \approx 2^{40.93}$ and $C_{\min} \approx 2^{689.30}$. However, very likely, the parameter a is closer to 1, so that the achievable complexities are lower.

For special maximum patterns, we can similarly decide to choose N bigger than N_0 and get the success probability p much smaller than 0.5. In fact, in practice, where N is given, we can choose d smaller than $d_0 \approx N/10$ hoping that the data complexity $D = NN^{(d)}/p$ and the time complexity $C = 2rN^7N^{(d)}/p$ may then both become smaller. Whether this is at all possible depends on how fast $1/p$ increases as d decreases. Note that $1/D$ is the probability of finding an internal state compliant with the pattern that results in a successful reconstruction of the whole internal state by the IPA. Experimental results from Section 5.2 indicate that $1/p$ can be approximated as $1/p \approx b^{-(N-N_0)^2}$, for $N > N_0$, or, alternatively, as a function of d given N, as $1/p \approx b^{-100(d-d_0)^2}$, for $d < d_0 = N/10$. In view of Table 2 and also theoretically, the parameter $b < 1$ is expected to slowly increase with N. Therefore, the common factor of D and C can be approximated as:

$$N^{(d)}/p \approx e^{-d^2/(2N)}N^d b^{-100(d-d_0)^2} \tag{14}$$

which, similarly as in (12), is minimized for $d_0 - d_{\text{opt}} \approx \log_2 N/(200\log_2 b^{-1})$. Now, if b is close to 1 so that $d_0 - d_{\text{opt}}$ is approximately at least 1, then we can get a reduction of both the complexities. Note that, unlike the consecutive patterns, here there is no tradeoff between data and time complexities, but the expected reduction may be less significant.

For example, for $N = 256$, in light of Table 2, we can (conservatively) assume that $b \approx 0.97$, so that $d_0 - d_{\text{opt}} \approx 1$. Then, for $d = 25$, we get optimized complexities $D_{\text{opt}} \approx 2^{210.65}$ and $C_{\min} \approx 2^{261.97}$. However, very likely, the parameter b is closer to 1, so that the complexities are in fact lower. The reduction of the time complexity is much lower than in the case of initial consecutive patterns, because the values of d are much smaller, for a given N.

7 Conclusions

It is shown that an iterative probabilistic algorithm (IPA), based on forward and backward recursive computations of *a posteriori* probabilities of the internal state components given a short keystream segment, has a high potential for recovering the whole RC4 internal states

from a relatively small number of known initial permutation entries, in about N^6 computational steps, for a modulus N. This is experimentally verified for two types of patterns of known entries, that is, consecutive patterns and special maximum patterns introduced in [8]. Systematic experiments conducted support a conjecture that only around $N/3$ and $N/10$ known entries are sufficient for a successful reconstruction, for the two types of patterns, respectively. Apart from the data given in the current version of the paper, the conducted experiments also include data for larger N, that is, for $N = 72, 80$ in the case of initial consecutive patterns and for the special maximum (9,42) pattern. They will be presented in the final version of the paper.

In guess-and-determine cryptanalytic attacks, the entries required to be known need to be guessed and the corresponding data and time complexities are then shown to be around $D = 2N$, $C \approx 4rN^{N/3+6}e^{-N/18}$ and $D \approx 2N^{N/10+1}e^{-N/200}$, $C \approx 4rN^{N/10+7}e^{-N/200}$, respectively, where r is the number of IPA rounds. Moreover, in the optimized versions, the complexities can be further improved by choosing a smaller number of guessed entries, so that the corresponding IPA success probability p is reduced, and by repeating the IPA on about $1/p$ distinct keystream segments. For $N = 256$, the complexities are thus (conservatively) estimated to be around $D \approx 2^{41}$, $C \approx 2^{689}$ and $D \approx 2^{211}$, $C \approx 2^{262}$, for the two types of patterns, respectively.

It thus turns out that, for special maximum patterns, the data complexity is considerably lower than that from [8]. The time complexity may possibly be reduced by introducing additional properties of the special patterns as in [8], which might allow a faster detection of inconsistent patterns, but the attacks would then become complicated and based on additional conjectures.

Appendix: Graphs of Success Probabilities

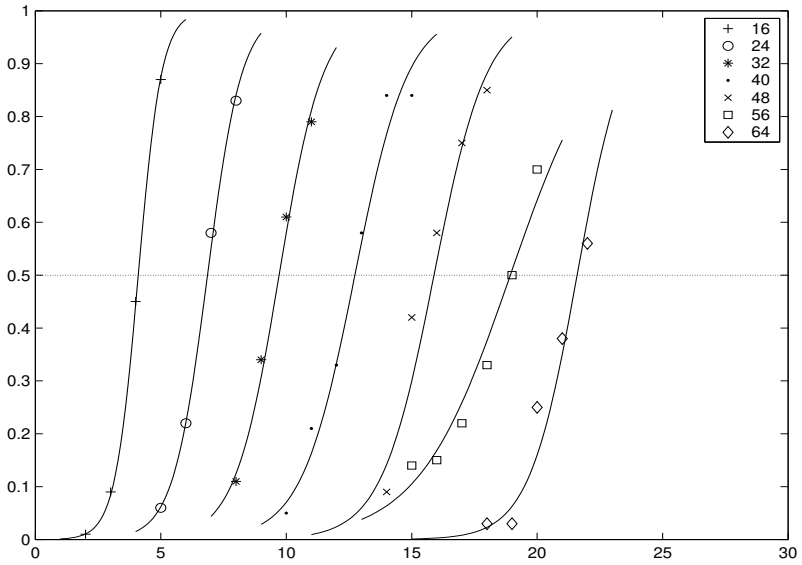

Figure 1: IPA success probabilities for initial consecutive patterns.

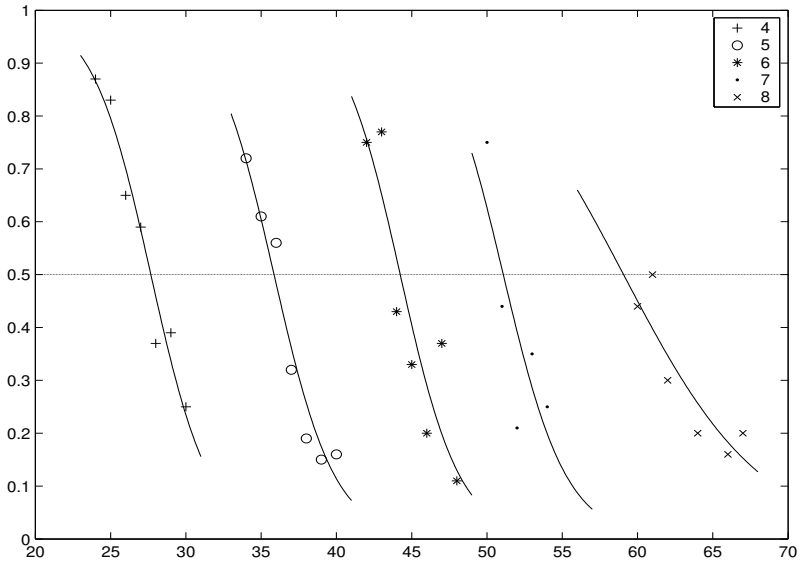

Figure 2: IPA success probabilities for special maximum patterns.

References

[1] E. Biham and Y. Carmeli, "Efficient reconstruction of RC4 secret keys from internal states," Technical Report CS-2008-06, Technion, Haifa, Israel, 2008.

[2] S. Fluhrer, I. Mantin, and A. Shamir, "Weakness in the key scheduling algorithm of RC4," Selected Areas in Cryptography - SAC '01, *Lecture Notes in Computer Science*, vol. 2259, pp. 1-24, 2001.

[3] J. Dj. Golić, "Linear statistical weakness of alleged RC4 keystream generator," Advances in Cryptology - EUROCRYPT '97, *Lecture Notes in Computer Science*, vol. 1233, pp. 226-238, 1997.

[4] J. Dj. Golić, "Iterative probabilistic cryptanalysis of RC4 keystream generator," Information Security and Privacy - ACISP 2000, *Lecture Notes in Computer Science*, vol. 1841, pp. 220-233, 2000.

[5] L. Knudsen, W. Meier, B. Preneel, V. Rijmen, and S. Verdoolaege, "Analysis methods for (alleged) RC4," Advances in Cryptology - ASIACRYPT '98, *Lecture Notes in Computer Science*, vol. 1514, pp. 327-341, 1998.

[6] I. Mantin, "Analysis of the stream cipher RC4," Master Thesis, The Weizmann Institute of Science, Rehovot, Israel, 2001.

[7] I. Mantin, "Predicting and distinguishing attacks on RC4 keystream generator," Advances in Cryptology - EUROCRYPT '05, *Lecture Notes in Computer Science*, vol. 3494, pp. 491-506, 2005.

[8] A. Maximov and D. Khovratovich, "New state recovering attack on RC4," Cryptology ePrint Archive, Report 2008/017, 2008.

[9] G. Paul and S. Maitra, "Permutation after RC4 key scheduling reveals the secret key," Selected Areas in Cryptography - SAC '07, *Lecture Notes in Computer Science*, vol. 4876, pp. 360-377, 2007.

[10] R. L. Rivest, "The RC4 encryption algorithm," RSA Data Security, Inc., Mar. 1992.