Generalized Inversion Attack on Nonlinear Filter Generators

Jovan Dj. Golic, Andrew Clark, and Ed Dawson

Abstract—A nonlinear filter generator is a basic keystream generator for stream cipher applications consisting of a single linear feedback shift register whose output is filtered by a nonlinear combining function. A binary nonlinear filter generator is viewed as a finite input memory automaton with one binary input and one binary output. The generalized inversion attack on a binary nonlinear filter generator is developed and analyzed by the theory of critical branching processes. Its objective is to recover the unknown input sequence from a given segment of the output sequence, provided that the filter function is known. Unlike the inversion attack, which requires that the filter function be linear in the first or the last input variable, this attack can be applied for any filter function. Both theory and systematic experiments show that its time complexity remains close to 2^M, which is the time complexity of the inversion attack, where M denotes the input memory size in bits.

Index Terms—Binary trees, Boolean functions, critical branching processes, inversion of finite automata, keystream generators.

1 Introduction

Nonlinear filter generators are popular building blocks in shift register based keystream generators for stream cipher applications as they enable one to achieve the cryptographic security with a relatively small number of shift registers (e.g., see [7] and [6]). A binary nonlinear filter generator consists of a single binary linear feedback shift register (LFSR), with a typically primitive feedback polynomial, and a nonlinear Boolean function whose inputs are taken from some shift register stages to produce the output. A nonlinear filter generator should be designed so as to resist all known cryptanalytic attacks applicable. The objective of the cryptanalytic attacks considered is to determine the unknown, secret key controlled LFSR initial state from a sufficiently long segment of the known keystream sequence. A set of design criteria to achieve a long period, a high linear complexity, and good statistical properties of the keystream sequence, as well as the resistance to the fast correlation attack [5], to the conditional correlation attack [1], and to the inversion attack [3] is recommended in [3].

Let r be the LFSR length, let n denote the number of nondegenerate input variables of the filter function f, let $\gamma = (\gamma_i)_{i=1}^n$ denote the tapping sequence specifying the inputs to f, and let $M = \gamma_n - \gamma_1$ denote the input memory size of the nonlinear filter generator regarded as a finite input memory combiner with one input and one output [3].

- *J.Dj. Golic is with the School of Electrical Engineering, University of Belgrade, Bulevar Revolucije 73, 11001 Belgrade, Yugoslavia. E-mail: golic@galeb.etf.bg.ac.yu.*
- *A. Clark and E. Dawson are with the Information Security Research Centre, Queensland University of Technology, GPO Box 2434, Brisbane, Queensland 4001, Australia. E-mail: {aclark, dawson}@fit.qut.edu.au.*

Manuscript received 19 Dec. 1996; revised 18 Apr. 2000; accepted 12 June 2000.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 102374.

The inversion attack [3] applies as such to the case when the filter function is linear in the first or the last input variable and runs forward or backward accordingly. This case is important as the only known case when the output sequence of a nonlinear filter generator as a combiner with one input and one output is purely random for every possible choice of the tapping sequence γ given that the input sequence is purely random. The attack consists of guessing the unknown M bits of the initial memory state and of the (unique) inversion of the first $r - M$ bits of the known keystream sequence into the remaining $r - M$ bits of the LFSR intitial state. Finally, the output sequence produced from the LFSR initial state is tested for consistency with the known keystream sequence. At worst, its computational complexity is 2^M appropriate steps, and 2^{M-1} on average.

To render the inversion attack infeasible, γ should be such that M is large and preferably close to its maximum possible value $r - 1$. In addition, to prevent reducing the effective input memory size by a uniform decimation technique [3], the greatest common divisor of $(\gamma_i - \gamma_1)_{i=1}^n$ should be equal to one.

Another way of preventing the inversion attack is to choose f that is linear in neither the first nor the last input variable. In this case, if M is large, then it is not possible to check by direct computation whether the output is purely random given that the input is such (see [3]). If the design criteria related to positive difference sets and correlation immunity are respected, then it is not practically possible to find a statistical weakness in the output even if it exists. However, a concept of a more general, so-called generalized inversion attack is also suggested in [3] which may work for any filter function. It goes along similar lines as the inversion attack with the only difference that the first $r - M$ keystream bits are not necessarily uniquely inverted into the corresponding $r - M$ input bits of the LFSR sequence.

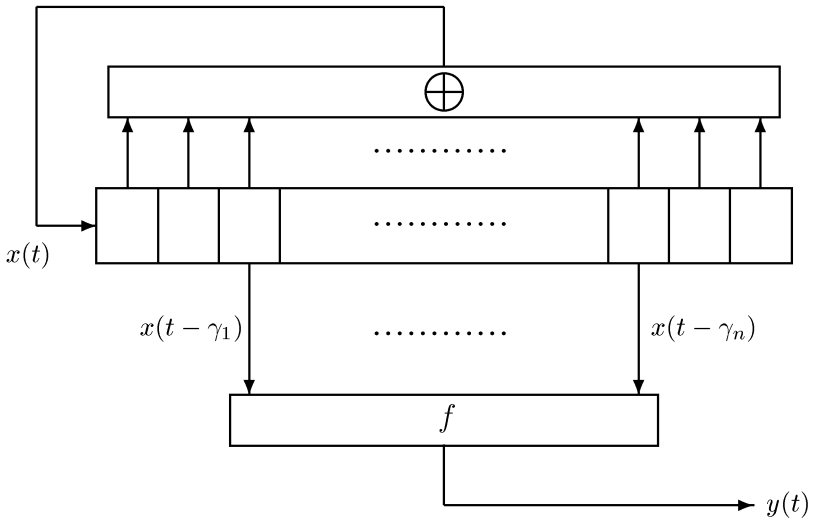


Fig. 1. Nonlinear filter generator.

Instead, a binary tree structure of depth $r - M$ is formed to store all possible solutions for the $r - M$ input bits, for every guessed initial memory state. It is also suggested in [3] that the theory of branching processes may be useful for analyzing the size of the resulting trees. While it is certainly true that the correct (very likely unique) LFSR sequence must be found by this attack, it remains to analyze its complexity, especially if $r - M$ is large. The main question to be answered is whether the resulting trees are then so large that the complexity gets close to 2^r, which would render the attack ineffective. This is the main objective of this paper. We will show both by the theory of branching processes and experimentally that the complexity of the generalized inversion attack is, perhaps surprisingly, also very close to 2^M, regardless of the choice of the filter function. Consequently, the choice of f cannot prevent the inversion attack in its generalized form.

The inversion attack [3] is briefly reviewed in Section 2, the generalized inversion attack is described and further developed in Section 3 and analyzed by the theory of critical branching processes (outlined in the Appendix) in Section 4, experimental results are presented and discussed in Section 5, and the conclusions are given in Section 6.

2 Inversion Attack

Let $x = (x(t))_{t=-r}^{\infty}$ be a binary maximum-length sequence of period $2^r - 1$ ($(x(t))_{t=-r}^{-1}$ is the LFSR initial state), let $f(z_1, \ldots, z_n)$ be a Boolean function of n, $n \leq r$, nondegenerate input variables, and let $\gamma = (\gamma_i)_{i=1}^{n}$ be an increasing sequence of nonnegative integers such that $\gamma_1 = 0$ and $\gamma_n \leq r - 1$. Then, the output sequence $y = (y(t))_{t=0}^{\infty}$ of the nonlinear filter generator shown in Fig. 1 is defined by

$$y(t) = f(x(t-\gamma_1), \ldots, x(t-\gamma_n)), \quad t \geq 0. \tag{1}$$

If we assume that the input sequence is purely random, that is, a sequence of balanced (uniformly distributed) and independent bits (binary random variables) and that the filter function is balanced (has balanced output given a balanced input), then the output sequence is not necessarily such. It is shown in [3] that the output sequence is purely random for every tapping sequence if $f(z_1, \ldots, z_n) = z_1 + g(z_2, \ldots, z_n)$ or $f(z_1, \ldots, z_n) = g(z_1, \ldots, z_{n-1}) + z_n$.

The objective of the inversion attack is to reconstruct the LFSR initial state from a segment of the keystream sequence, given the LFSR feedback polynomial of degree r, the filter function f, and the tapping sequence γ. The attack runs forward or backward depending on whether f is linear in the first or the last input variable, respectively. In the former case, put (1) into the form

$$x(t) = y(t) + g(x(t-\gamma_2), \ldots, x(t-\gamma_n)), \quad t \geq 0, \tag{2}$$

which means that the nonlinear filter generator as a combiner with one input and one output is invertible if the initial memory state is known. The forward inversion attack then goes as follows:

1. Assume (not previously checked) M bits $(x(t))_{t=-M}^{-1}$ of the unknown initial memory state.
2. By using (2), generate a segment $(x(t))_{t=0}^{r-M-1}$ of the input sequence from a known segment $(y(t))_{t=0}^{r-M-1}$ of the keystream sequence.
3. By using the LFSR linear recursion, generate a sequence $(x(t))_{t=r-M}^{N-1}$ from the first r bits $(x(t))_{t=-M}^{r-M-1}$.
4. By using (1), compute $(\hat{y}(t))_{t=r-M}^{N-1}$ from $(x(t))_{t=r-2M}^{N-1}$ and compare with the observed $(y(t))_{t=r-M}^{N-1}$. If they are the same, then accept the assumed initial memory state and stop. Otherwise, go to Step 1.

It takes 2^{M-1} trials on average to find a correct initial memory state. One may as well examine all 2^M initial memory states. In that case, the algorithm yields all the LFSR sequences that produce the given keystream sequence of length N. The found candidate initial states could then be examined on a longer sequence as well, which may reduce their number. More precisely, under a reasonable assumption that different LFSR initial states give rise to bitwise uncorrelated periodic keystream sequences, the expected number of false alarms for candidate initial states does not

exceed 2^{-c} if the length of the keystream sequence is only $N = r + c$. If the determined LFSR sequence is not unique, then any such sequence is a satisfactory solution (equivalent LFSR initial states yielding the same keystream sequence), but, for most filter functions, this situation is very unlikely.

3 Generalized Inversion Attack

The generalized inversion attack as suggested in [3] applies to an arbitrary filter function f which need not be linear in the first or the last input variable. Without essential loss of generality, f is assumed to be balanced. For such a function, there exists a nonzero fraction p_+ of values of the input variables $(z_2, \ldots, z_n)$, where f is equal to zero or one (equally likely) regardless of z_1, and, similarly, a nonzero fraction p_- of values of the input variables $(z_1, \ldots, z_{n-1})$, where f is equal to zero or one (equally likely) regardless of z_n. In this case, one should find the minimum of p_+ and p_- and then accordingly apply the generalized inversion attack in the forward or backward direction. In the generalized inversion attack, the objective is to find all possible, not necessarily unique, input sequences of length $r - M$ consistent with a given segment of the keystream sequence of the same length, for each assumed initial memory state, whereas the rest is the same as in the inversion attack. The (generalized) inversion attack thus exploits the dependence between the input and the output sequence to the maximum possible extent.

3.1 Forward and Backward Attacks

In the forward generalized inversion attack, given the current output bit $y(t)$ and a guessed current memory state $(x(i))_{i=t-M}^{t-1}$ (the preceding M input bits), the basic equation (1) may have a unique solution for $x(t)$, may have no solution for $x(t)$, or may have two solutions for $x(t)$ (both zero and one). Given a segment of $r - M$ successive output bits, proceeding forward one bit at a time, one can thus obtain and store all possible solutions for an input sequence in a binary tree structure of maximum depth $r - M$. Each node in the tree represents an internal memory state of M successive input bits. Similarly, in the backward inversion attack, one proceeds backward one bit at a time, each time finding from (1) all possible solutions for $x(t - M)$ given the current output bit $y(t)$ and a guessed current memory state $(x(i))_{i=t-M+1}^{t}$ (the next M input bits). Without loss of generality, we will now deal only with the forward generalized inversion attack.

Let $p = p_+$ for simplicity. In the probabilistic model where the LFSR initial state is chosen uniformly at random, any $M + 1$, $M \leq r - 1$, successive input bits (defining the inputs to f) are balanced and independent. Without essential difference, the given keystream sequence can be considered either as fixed or as purely random and independent of the LFSR sequence. In this model, for any $t \geq 0$, the number of possible solutions for the current input bit $x(t)$ is a nonnegative integer random variable Z with the probability distribution, independent of t,

$$\Pr\{Z = 0\} = \frac{p}{2}, \quad \Pr\{Z = 1\} = 1 - p, \quad \Pr\{Z = 2\} = \frac{p}{2}. \tag{3}$$

Its expected value and variance are given by

$$\mu = 1, \quad \sigma^2 = p. \tag{4}$$

3.2 Basic Attack

It is interesting to examine the case $p = 1$, when f does not effectively depend on the first input variable z_1, in more detail. Then, M is bigger than the effective memory size and guessing M successive input bits is the same as guessing all the input bits to f as well as some additional input bits if $\gamma_2 - \gamma_1 > 1$. Accordingly, the attack can then be reduced to the so-called *basic generalized inversion attack*, in which one guesses $M + 1$ successive input bits $(x(i))_{i=t-M}^{t}$ and then checks whether the corresponding output bit determined by f is the same as the observed $y(t)$ or not. If not, then there is no solution for the next input bit $x(t + 1)$ and the guess is discarded as incorrect. If yes, then there are two possible solutions for $x(t + 1)$ and the search is continued in the same manner for both of them. In the probabilistic model as above, the number of solutions for $x(t + 1)$ is a random variable Z defined by (3) for $p = 1$. It takes only two values, 0 and 2, each with probability 1/2, and has the expected value and variance both equal to 1 (see (4)).

Initially, exactly one half of the guesses are discarded so that the total effective number of initial guesses is in fact 2^M, which is the same as before. Of course, the corresponding 2^{M+1} trees, half of which are empty, store all the solutions for the input sequences of length $r - M$ given the known output sequence of the same length. The solutions are the same as above, but the trees are different. Each node contains $M + 1$ rather than M successive input bits and the trees have maximum depth $r - M - 1$ rather than $r - M$, but the nodes at the first and the last level have to be checked if they are consistent with the first and the last output bit, respectively. The main difference from the generalized inversion attack described above is that the nodes at each level have to be generated before they are tested for consistency with the corresponding output bit. The trees can be grouped in 2^M pairs, each corresponding to the same initial memory state and each pair can be aggregated into a single tree, the same as above, in an obvious way by discarding all the nodes without branches leaving out. So, the basic generalized inversion attack is less efficient, as should be expected, since it does not make use of the fact that p_+ is smaller than 1. The basic attack can also run in the backward direction as well.

3.3 Binary Trees

In the forward generalized inversion attack, for each initial memory state $(x(t))_{t=-M}^{-1}$, the obtained binary tree, representing all the solutions for the next $r - M$ bits $(x(t))_{t=0}^{r-M-1}$ consistent with the known $r - M$ output bits $(y(t))_{t=0}^{r-M-1}$ is unique given $(y(t))_{t=0}^{r-M-1}$. For each $1 \leq n \leq r - M$, let Z_n denote the number of nodes at level n, that is, the number input segments $(x(t))_{t=0}^{n-1}$ of length n that are consistent with the output segment $(y(t))_{t=0}^{n-1}$. The initial level $n = 0$ contains only one node representing an initial memory state $(x(t))_{t=-M}^{-1}$ and each node in the tree represents an internal memory state of M successive input bits. As well, one can

also store only one level at a time, but then each node at level n should represent an input segment of variable size n, $1 \le n \le r - M$, rather than of constant size M. In practice, instead of storing the tree, one may conduct a depth-first search, in which case the required space complexity is negligible.

Let $Y_n = \sum_{l=1}^{n} Z_l$ denote the total number of nodes in the tree up to level n, without counting the initial node. Then, the (normalized) time complexity of the tree search process is $\sum Y_{r-M}/(r-M)$ steps, the sum being over all 2^M initial memory states, where the step complexity is approximately the same as in the inversion attack. The total number of the obtained solutions for input segments $(x(t))_{t=-M}^{r-M-1}$ of length r that are consistent with the given output segment is given as $\sum Z_{r-M}$, where the sum is over all 2^M initial memory states. Note that for the basic generalized inversion attack the figures are slightly different. Namely, the time complexity is $\sum(1 + Y_{r-M-1})/(r-M)$ and the total number of solutions is $\sum Z_{r-M}/2$ (the nodes at level $r-M$ are not effectively produced), where the sums are over all 2^{M+1} initial guesses. Consequently, the main problem to be addressed is how large these values can grow as $r-M$ increases.

4 Probabilistic Analysis via Branching Processes

4.1 Probabilistic Models

The basic probabilistic model to be considered is one in which $(x(t))_{t=-M}^{-1}$ is uniformly distributed and $(y(t))_{t=0}^{r-M-1}$ is a random variable independently generated from a uniformly distributed LFSR initial state. Note that $(y(t))_{t=0}^{r-M-1}$ need not be uniformly distributed and, in fact, is not likely to be such if $p > 0$. In particular, some output segments may not be possible at all. In the related, simplified model where the output segment is uniformly distributed, the expected values of both Z_n and Y_n/n are equal to 1 for each $1 \le n \le r - M$. So, in this model, the expected total number of consistent solutions for $(x(t))_{t=-M}^{r-M-1}$ (to be checked in the final stage of the generalized inversion attack) as well as the expected time complexity of the tree construction are both exactly 2^M, which is the same as in the inversion attack.

Not only can the expected values be different in the basic model, but also it is conceivable that Z_{r-M} and/or $Y_{r-M}/(r-M)$ can be big depending on a particular output segment. In the inversion attack, where $p = 0$, this is not possible because the variance of Z_n is zero for every guessed initial memory state. More generally, if the output segment is uniformly distributed and $p > 0$, then the number of solutions is exactly 2^M for each output segment, but the variance of Z_n need not be equal to zero for every guessed initial memory state. Consequently, the problem here is to estimate the expected values and the variances, as well as the probability distributions of both Z_n and Y_n/n in the basic probabilistic model.

The variances and the probability distributions of Z_n and Y_n/n in general depend on a particular filter function and on a chosen tapping sequence as well. They could be estimated empirically in various cases of interest, as is demonstrated in the next section. However, a reasonably good approximation providing insight into the size of the random tree spanned can be obtained by the theory of critical branching processes outlined in the Appendix. One may consider the random tree produced by the random initial memory state and the random or a fixed output segment. In both cases, the associated branching process is one with the branching probability distribution defined by (3). It is a *critical Galton-Watson process* with the expected value 1 and the variance p of the branching random variable Z_1.

The random tree produced by the associated branching process is not the same as the random tree obtained by the tree construction process. The reason for this is that, in the branching process, the branching probability distribution for a given node is independent of the nodes at the same or the preceding levels (the history), whereas, in the tree construction process, there is a dependence between the nodes as a result of successive inputs to the filter function having some bits in common. Note that the dependence is not influenced by the LFSR recursion since only r successive bits of the LFSR sequence are examined. This dependence is relatively weak if the tapping sequence defines a positive difference set and is stronger if it is equidistant, that is, if $\gamma = (\delta i)_{i=0}^{n-1}$, where δ is a positive integer. As a consequence, the probability distributions of both the variables Z_n and Y_n/n are somewhat different. However, the difference is expected to be relatively small for both their expected values and variances, as they are only affected by relatively weak pairwise and triplewise dependences between different levels in the random tree generated by the tree construction process.

4.2 Expected Values and Variances

In view of Theorem 1 from the Appendix, we get that, for the associated branching process, $E(Z_n) = 1$, $\mathrm{Var}(Z_n) = p\,n$, and $\Pr\{Z_n > 0\} = 1 - f^{(n)}(0)$, where $f^{(n)}(s)$ is the self-composition (7) of the generating function,

$$f(s) = p/2 + (1-p)s + ps^2/2,$$

of the branching probability distribution (3). This probability can be evaluated numerically. For any n, $\Pr\{Z_n > 0\} \le 1 - p/2$ and, for large n, $\Pr\{Z_n > 0\} \approx 2/(p\,n)$, provided $p > 0$. If p is very small, then this probability is close to 1 unless n is very large. Accordingly, the expected fraction of the guessed initial memory states giving rise to at least one input segment of length n that is consistent with the given output segment of length n is $1 - f^{(n)}(0)$. On the other hand, Theorem 2 from the Appendix gives that $E(Y_n/n) = 1$ and $\mathrm{Var}(Y_n/n) = p\,n/3$. In view of the Chebyshev inequality $\Pr\{|Y_n/n - E(Y_n/n)| \ge \varepsilon\} \le \mathrm{Var}(Y_n/n)/\varepsilon^2$, we then get that Y_n/n is, with high probability, $O(\sqrt{n})$ and the multiplicative constant is not big. Note that, in the case of interest, $n = r - M$.

It is interesting to see how large Z_n and Y_n/n can grow when conditioned on the event that there exists at least one input segment of length n that is consistent with the output segment. At least one such initial memory state exists, corresponding to the original LFSR sequence producing the

given output sequence. Theorem 3 from the Appendix shows that, for $p > 0$ and large n, $E(Z_n|Z_n > 0) \approx p\,n/2$ and $\mathrm{Var}(Z_n|Z_n > 0) \approx p^2n^2/4$. This means that the number of solutions is, with high probability, linear in n, provided at least one such solution exists. As for Y_n/n, the note from the Appendix shows that, for $p > 0$ and large n, $E(Y_n/n|Z_n > 0) = O(p\,n)$ and $\mathrm{Var}(Y_n/n|Z_n > 0) = O(p^2n^2)$ so that Y_n/n is then, with high probability, $O(p\,n)$. Consequently, the resulting tree is then bigger than on average, but still relatively small, even if $n = r - M$ is big.

4.3 Correction Factor

One may take the estimates given above as good approximations for the random tree generated by the tree construction process. Recall that, in the basic probabilistic model, one first chooses a random uniformly distributed LFSR initial state, then generates the corresponding output segment of length $r - M$, and, finally, independently chooses a uniformly distributed initial memory state and constructs the corresponding tree. So, for each achievable output segment of length $r - M$, one in fact constructs 2^M trees corresponding to all possible initial memory states. The above estimates would have been good approximations if all 2^{r-M} output segments were achievable. Since this is not the case, a correction has to be made. Namely, the random variables Z_n and Y_n/n have to be conditioned on the *achievability* event that there exists at least one initial memory state, among 2^M of them, with at least one input segment of length n consistent with the output segment. The conditioning event is the same as the one that the output segment is achievable or, in terms of the theory of branching processes, that among 2^M independently generated trees there exists at least one of depth n. It is easily seen that the expected fraction of achievable output segments of length n is then

$$q_n \;=\; 1 - f^{(n)}(0)^{2^M} \;\sim\; 1 - \left(1 - \frac{2}{p\,n}\right)^{2^M}. \qquad (5)$$

Thus, the theory of branching processes helps one analyze how many output segments of a given length are expected to occur at the output of a nonlinear filter generator. This reflects its statistical properties. Consequently, for any $n \geq 1$, the random variable Z_n in the original branching process is a mixture of the zero random variable, with probability $1 - q_n$, and the random variable Z_n conditioned on the achievability event, with probability q_n. Both $E(Z_n)$ and $\Pr\{Z_n > 0\}$ then increase by the multiplicative factor q_n^{-1}, whereas $\mathrm{Var}(Z_n)$ approximately increases by the same factor. The random variable Y_n/n is more difficult to analyze, but it is clear that one may expect that the trees produced from achievable output segments by the basic probabilistic model are bigger in size about q_n^{-1} times up to level n than the ones produced by arbitrary output segments. For $n = r - M$, the correction factor q_{r-M}^{-1} becomes significant if $2^M(1 - f^{(r-M)}(0)) \leq 1$.

4.4 Time Complexity and Number of Solutions

As noted before, the time complexity of the tree construction process is given as $T = \sum Y_{r-M}/(r - M)$, where the sum and is over all 2^M initial memory states. The analysis conducted above, based on the theory of critical branching processes, shows that the expected time complexity is about $q_{r-M}^{-1}2^M$ and that, with high probability, under the reasonable independence assumption,

$$T \;\leq\; q_{r-M}^{-1}2^M + 2^{M/2}(r - M)^{1/2}\sqrt{3pq_{r-M}^{-1}}. \qquad (6)$$

Note that the correction factor q_{r-M}^{-1} depends on $r - M$, M, and p. For $p > 0$ and large $r - M$, we have $q_{r-M}^{-1} \leq p(r - M)/2$. The total number of obtained input segments of length r consistent with the given output segment of length $r - M$ is $K = \sum Z_{r-M}$ and has about the same expected value as T and the variance three times bigger. As a result, it satisfies a relation analogous to (6).

It is clear that, unlike the time complexity, the fraction of achievable output segments of any given length, as well as the total number of input segmens consistent with a given output segment, are both independent of whether the attack is applied in its forward ($p = p_+$), backward ($p = p_-$), or basic ($p = 1$) form. So, (5) is only an approximation. It is reasonable to expect that the approximation is better if p is taken to be the minimum of p_+ and p_-, especially if this minimum is relatively small.

5 Experimental Results

In this section, we present results obtained by systematic experimental analysis of various nonlinear filter generators. The shift register length chosen is $r = 100$ which is sufficiently big to study the effect of a large tree depth $r - M$. The primitive feedback polynomial chosen is $1 + x^2 + x^7 + x^8 + x^{100}$. We study the filter Boolean functions, f, with $n = 5$ and $n = 10$ input variables. For each n, two tap settings, γ, are considered, one adjacent and the other corresponding to a full positive difference set (for $n = 5$), and to a random set (for $n = 10$). The experimental results for each of the four cases are shown in Tables 1, 2, 3, and 4. In each case, we have randomly chosen five filter functions f with different probabilities $(p_+, p_-) = (0, 0.5)$, $(0.125, 0.125)$, $(0.125, 0.875)$, $(0.5, 0.5)$, and $(0.875, 0.875)$. For each of them we have run the forward and backward generalized inversion attacks, as well as the forward and backward basic generalized inversion attacks for 50 randomly chosen LFSR initial states.

The results shown are the average normalized number of solutions for consistent input segments of length r, per each initial memory state guessed ("#Solutions" being equal to $K/2^M$ for the forward and backward attacks and to $K/2^{M+1}$ for the basic ones), the average normalized time complexity of the tree construction process, per each initial memory state guessed ("Time" being equal to $T/2^M$ for the forward and backward attacks and to $T/2^{M+1}$ for the basic ones), and the fraction of trees reaching the full depth $r - M$ ("Prob") for all the attacks. Note that, in the basic attacks, the level $r - M - 1$ is not empty after checking for consistency if and only if the level $r - M$ is not empty before checking for consistency. All the results are averaged over 50 randomly chosen LFSR initial states.

Table 5 contains the probability for a tree to reach the full depth $(1 - f^{(r-M)}(0))$, to be compared with the "Prob" column of Tables 1, 2, 3, and 4, the fraction of achievable

TABLE 1
$(r, n, M) = (100, 5, 4)$, $\gamma = (0, 1, 2, 3, 4)$

p_+	p_-	#Solutions	Time	Prob	Attack
0.000	0.500	1.000	1.010	1.000	Forward
		1.000	1.010	0.062	Backward
		0.500	1.010	0.500	Basic Fwd
		0.500	1.010	0.031	Basic Bwd
0.125	0.125	6.832	2.669	0.313	Forward
		6.832	2.489	0.244	Backward
		3.416	2.626	0.183	Basic Fwd
		3.416	2.443	0.124	Basic Bwd
0.125	0.875	52.185	12.152	0.611	Forward
		52.185	10.727	0.133	Backward
		26.092	11.731	0.325	Basic Fwd
		26.092	10.291	0.069	Basic Bwd
0.500	0.500	101.983	22.641	0.160	Forward
		101.983	21.382	0.371	Backward
		50.992	21.806	0.110	Basic Fwd
		50.992	20.534	0.193	Basic Bwd
0.875	0.875	42.626	10.093	0.209	Forward
		42.626	11.199	0.300	Backward
		21.313	9.751	0.134	Basic Fwd
		21.313	10.868	0.159	Basic Bwd

output segments of required length (q_{r-M}), and the corresponding correction factor for the number of solutions and for the time complexity (q_{r-M}^{-1}). They are computed according to the theory of critical branching processes. Each found consistent input segment of length r was then tested on an additional segment of the keystream sequence (the final stage of the inversion attack). For each examined nonlinear filter generator and every

TABLE 2
$(r, n, M) = (100, 5, 15)$, $\gamma = (0, 1, 3, 7, 15)$

p_+	p_-	#Solutions	Time	Prob	Attack
0.000	0.500	1.000	1.012	1.000	Forward
		1.000	1.012	0.002	Backward
		0.500	1.012	0.500	Basic Fwd
		0.500	1.012	0.001	Basic Bwd
0.125	0.125	1.076	1.042	0.136	Forward
		1.076	1.037	0.061	Backward
		0.538	1.041	0.069	Basic Fwd
		0.538	1.037	0.031	Basic Bwd
0.125	0.875	4.696	2.240	0.218	Forward
		4.696	2.080	0.000	Backward
		2.348	2.211	0.112	Basic Fwd
		2.348	2.049	0.000	Basic Bwd
0.500	0.500	5.376	2.150	0.017	Forward
		5.376	2.128	0.003	Backward
		2.688	2.112	0.009	Basic Fwd
		2.688	2.089	0.001	Basic Bwd
0.875	0.875	158.010	29.403	0.119	Forward
		158.010	26.335	0.001	Backward
		79.005	27.872	0.063	Basic Fwd
		79.005	24.768	0.000	Basic Bwd

TABLE 3
$(r, n, M) = (100, 10, 9)$, $\gamma = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)$

p_+	p_-	#Solutions	Time	Prob	Attack
0.000	0.500	1.000	1.011	1.000	Forward
		1.000	1.011	0.040	Backward
		0.500	1.011	0.500	Basic Fwd
		0.500	1.011	0.020	Basic Bwd
0.125	0.125	1.122	1.052	0.175	Forward
		1.122	1.040	0.178	Backward
		0.561	1.051	0.089	Basic Fwd
		0.561	1.039	0.090	Basic Bwd
0.125	0.875	2.056	1.256	0.198	Forward
		2.056	1.387	0.033	Backward
		1.028	1.247	0.101	Basic Fwd
		1.028	1.380	0.016	Basic Bwd
0.500	0.500	3.302	1.714	0.067	Forward
		3.302	1.815	0.063	Backward
		1.651	1.696	0.035	Basic Fwd
		1.651	1.799	0.032	Basic Bwd
0.875	0.875	26.262	6.761	0.066	Forward
		26.262	7.284	0.069	Backward
		13.131	6.545	0.036	Basic Fwd
		13.131	7.073	0.036	Basic Bwd

TABLE 4
$(r, n, M) = (100, 10, 15)$, $\gamma = (0, 2, 3, 6, 7, 9, 10, 11, 14, 15)$

p_+	p_-	#Solutions	Time	Prob	Attack
0.000	0.500	1.000	1.012	1.000	Forward
		1.000	1.012	0.027	Backward
		0.500	1.012	0.500	Basic Fwd
		0.500	1.012	0.013	Basic Bwd
0.125	0.125	1.002	1.015	0.142	Forward
		1.002	1.014	0.151	Backward
		0.501	1.015	0.072	Basic Fwd
		0.501	1.014	0.076	Basic Bwd
0.125	0.875	1.040	1.033	0.144	Forward
		1.040	1.017	0.012	Backward
		0.520	1.033	0.073	Basic Fwd
		0.520	1.017	0.006	Basic Bwd
0.500	0.500	1.100	1.049	0.023	Forward
		1.100	1.084	0.028	Backward
		0.550	1.048	0.012	Basic Fwd
		0.550	1.083	0.014	Basic Bwd
0.875	0.875	2.230	1.422	0.012	Forward
		2.230	1.476	0.014	Backward
		1.115	1.413	0.006	Basic Fwd
		1.115	1.467	0.007	Basic Bwd

chosen LFSR initial state, it turns out that exactly one input sequence is consistent with the given keystream sequence (no equivalent LFSR initial states), as should be expected since the number of input variables n is relatively small compared to r.

The experimental results shown generally agree very well with the theory of critical branching processes. In fact, by comparing Tables 2 and 4, where the memory sizes are the same, but the numbers of input variables are different, one may conclude that the dependence induced

TABLE 5
Full Depth Probabilities

(r,n,M)	p				
	0.000	0.125	0.500	0.875	1.000
	$1-f^{(r-M)}(0)$				
(100,5,4)	1.000	0.140	0.039	0.022	0.020
(100,5,15)	1.000	0.156	0.043	0.025	0.022
(100,10,9)	1.000	0.147	0.041	0.023	0.021
(100,10,15)	1.000	0.156	0.043	0.025	0.022
	q_{r-M}				
(100,5,4)	1.000	0.911	0.468	0.303	0.271
(100,5,15)	1.000	1.000	1.000	1.000	1.000
(100,10,9)	1.000	1.000	1.000	1.000	1.000
(100,10,15)	1.000	1.000	1.000	1.000	1.000
	q_{r-M}^{-1}				
(100,5,4)	1.000	1.098	2.136	3.300	3.695
(100,5,15)	1.000	1.000	1.000	1.000	1.000
(100,10,9)	1.000	1.000	1.000	1.000	1.000
(100,10,15)	1.000	1.000	1.000	1.000	1.000

by overlapping successive inputs to the filter function (Table 4) tends to reduce the size of the constructed trees. Interestingly, the tables show that the normalized time complexities are almost independent of the direction in which the attack is run. The normalized time complexities are exactly the same if p_+ or p_- is equal to zero. As the total number of solutions is the same in each case for all the attacks, the normalized number of solutions is halved for the basic attacks. Despite some irregularities in Tables 1 and 2 ($n = 5$), one can draw a general conclusion that the normalized number of solutions and the normalized time complexity are likely to increase as the minimum of p_+ and p_- increases. The trees produced by the basic generalized inversion attack are roughly twice as big as those produced by the generalized inversion attack in the same direction, as should be expected.

To demonstrate the accordance with the theory, consider Tables 1 and 2, for example. The normalized number of solutions is, in most cases, bigger in Table 1 than in Table 2 because the number of possible initial memory states is much smaller, so that the variance becomes significant (see (6)), and because the correction factor is bigger than 1 for Table 1, unlike the other tables (see Table 5). The same holds for the normalized time complexity except that the figures are smaller since the variance is smaller (see (6)).

6 Conclusions

The theory of critical branching processes is applied to analyze the complexity of the generalized inversion attack on nonlinear filter generators. Both theory and systematic experimental results obtained show that, almost regardless of the choice of the filter function, the attack has time complexity close to 2^M, M being the input memory size. Consequently, the choice of the filter function that is linear in neither the first nor the last input variable is likely to spoil the output statistics, but does not prevent the inversion attack in its generalized form. The inversion attack is infeasible if M is sufficiently large, provided that the tapping sequence is such that M cannot be reduced by the uniform decimation technique.

The attack can be extended to deal with more than one bit at a time, in which case the resulting trees are no longer binary, but the associated branching process remains critical. This may reduce the required complexity for some filter functions, especially if the branching probabilities p_- and p_+ are both close to 1. The attack can also be applied to nonlinear filter generators with multibit outputs, where the theory of subcritical branching processes will be useful. More generally, the attack is also applicable to combinations of nonlinear filter generators with single or multiple binary outputs.

Appendix

Critical Branching Processes

Only the basic type of branching processes, called the Galton-Watson processes, will be considered (see [4] and [2]). Such a branching process is a Markov chain $\{Z_n\}_{n=0}^{\infty}$ on the nonnegative integers whose transition function is defined in terms of a given probability distribution $\{p_k\}_{k=0}^{\infty}$. The initial random variable Z_0 takes value 1 with probability 1 and, for any $n \geq 1$, the random variable Z_n conditioned on $Z_{n-1} = i$ is the sum of i independent identically distributed random variables with the probability distribution $\{p_k\}_{k=0}^{\infty}$ (if $i = 0$, then $Z_n = 0$). The process can be regarded as a random (finite or infinite) tree with Z_n being the number of nodes at level $n \geq 0$, where the number of branches leaving any node in the tree is equal to k with probability p_k, independently of other nodes at the same or previous levels. The generating function characterizing the probability distribution of Z_n can be expressed as the self-composition of the generating

function $f(s) = \sum_{k=0}^{\infty} p_k s^k$ of $\{p_k\}_{k=0}^{\infty}$, which is the probability distribution of Z_1. Precisely, if $f^{(n)}(s)$, $0 \le s \le 1$, denotes the generating function of the probability distribution of Z_n and if $f^{(0)} = s$, then, for every $n \ge 1$,

$$f^{(n)}(s) \;=\; f(f^{(n-1)}(s)). \tag{7}$$

The basic characteristic of a branching process is the expected number of branches leaving any node, that is,

$$\mu \;=\; E(Z_1) \;=\; \sum_{k=0}^{\infty} k\, p_k. \tag{8}$$

A branching process is called subcritical, critical, or supercritical if $\mu < 1$, $\mu = 1$, or $\mu > 1$, respectively. The extinction probability, defined as the probability of a tree being finite, is 1 for subcritical and critical (provided $p_0 > 0$) processes and smaller than 1 for supercritical processes. Here, we are only interested in critical processes, whose main properties are given by the following theorem (see [2] and [4]). Let $\sigma^2 = \mathrm{Var}(Z_1)$ be the variance of Z_1.

Theorem 1. *In the critical case, $\mu = 1$, if $\sigma^2 > 0$ ($p_1 < 1$) and $\sigma^2 < \infty$, then, for any $n \ge 1$,*

$$E(Z_n) \;=\; 1 \tag{9}$$

$$\mathrm{Var}(Z_n) \;=\; \sigma^2 n \tag{10}$$

$$\Pr\{Z_n > 0\} \;=\; 1 - f^{(n)}(0) \;\sim\; \frac{2}{\sigma^2 n}. \tag{11}$$

Equation (11) implies that the extinction probability, $\lim_{n\to\infty} \Pr\{Z_n > 0\}$, is equal to 1, while the rate of convergence is relatively slow. The variance grows linearly with n, although the expected value remains equal to 1.

It is also interesting to study the total number of nodes in a random tree up to level n, not counting the initial node, that is, the random variable $Y_n = \sum_{l=1}^{n} Z_l$, for any $n \ge 1$. Its generating function satisfies a recursion which reduces to a functional equation with a unique solution if $n \to \infty$ (see [4]). Its expected value follows trivially, and its variance can be determined after a certain manipulation.

Theorem 2. *In the critical case, $\mu = 1$, if $\sigma^2 > 0$, then, for any $n \ge 1$,*

$$E(Y_n) = n \tag{12}$$

$$\mathrm{Var}(Y_n) = \frac{\sigma^2}{6}\, n(n+1)(2n+1) \sim \frac{\sigma^2}{3}\, n^3. \tag{13}$$

Note that, although the extinction probability is 1, the expected value grows linearly with n and the variance increases as n^3, which is faster by a multiplicative factor n than what would hold if the random variables Z_l were independent.

Other interesting random variables to be considered are Z_n and Y_n conditioned on the event $\{Z_n > 0\}$. They are the number of nodes at level n and the total number of nodes up to level n, not counting the initial one, in a random tree reaching level n. The probability distribution of Z_n conditioned on $\{Z_n > 0\}$ is simply obtained by dividing the probability distribution of Z_n by $\Pr\{Z_n > 0\}$ (see Theorem 1). The limit distribution of Z_n/n conditioned on $\{Z_n > 0\}$ in the critical case has been characterized by Yaglom (see [4] and [2]). By computing the expected value [2] and variance, we can then formulate the following theorem.

Theorem 3. *In the critical case, $\mu = 1$, if $0 < \sigma^2 < \infty$, then*

$$\lim_{n\to\infty} \Pr\left\{ \frac{Z_n}{n} > z \,|\, Z_n > 0 \right\} \;=\; e^{-2z/\sigma^2}, \quad z \ge 0, \tag{14}$$

$$E(Z_n | Z_n > 0) = \frac{1}{1 - f^{(n)}(0)} \sim \frac{\sigma^2}{2}\, n \tag{15}$$

$$\mathrm{Var}(Z_n | Z_n > 0) = \frac{1}{1 - f^{(n)}(0)} \left(\sigma^2 n - \frac{f^{(n)}(0)}{1 - f^{(n)}(0)} \right) \sim \frac{\sigma^4}{4}\, n^2. \tag{16}$$

The probability distribution of the random variable Y_n conditioned on $\{Z_n > 0\}$ is not treated in the standard books on branching processes like [4] and [2]. Nevertheless, the previous theorems and the results regarding the random variable Z_n conditioned on $\{Z_{n+k} > 0\}$ presented in [2] lead us to conclude that, in the critical case,

$$E(Y_n | Z_n > 0) = O(n(1 - f^{(n)}(0))^{-1}) = O(\sigma^2 n^2)$$

and

$$\mathrm{Var}(Y_n | Z_n > 0) = O(\sigma^2 n^3 (1 - f^{(n)}(0))^{-1}) = O(\sigma^2 n^4).$$

REFERENCES

[1] R.J. Anderson, "Searching for the Optimum Correlation Attack," *Proc. Fast Software Encryption—Leuven '94,* B. Preneel, ed., pp. 137-143, 1995.

[2] K.B. Athreya and P.E. Ney, *Branching Processes.* Berlin: Springer-Verlag, 1972.

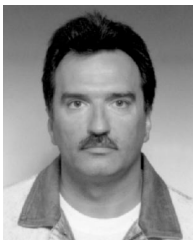
[3] J.Dj. Golic, "On the Security of Nonlinear Filter Generators," *Proc. Fast Software Encryption—Cambridge '96,* D. Gollmann, ed., pp. 173-188, 1996.

[4] T.H. Harris, *The Theory of Branching Processes.* Berlin: Springer-Verlag, 1963.

[5] W. Meier and O. Staffelbach, "Fast Correlation Attacks on Certain Stream Ciphers," *J. Cryptology,* vol. 1, no. 3, pp. 159-176, 1989.

[6] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography.* Boca Raton, Fla.: CRC Press, 1997.

[7] R.A. Rueppel, *Analysis and Design of Stream Ciphers.* Berlin: Springer-Verlag, 1986.



Jovan Dj. Golic received the BSc, MSc, and PhD degrees in electrical engineering from the School of Electrical Engineering, University of Belgrade, Yugoslavia, in 1979, 1981, and 1985, respectively. From 1979 to 1993, he worked with the Institute of Applied Mathematics and Electronics, Belgrade, where he was appointed a department head in 1986 and a senior research fellow in 1990. In 1987 and 1988, he was a Fulbright visiting scientist at the School of Electrical Engineering, Cornell University, Ithaca, New York. He was a part-time docent with the School of Electrical Engineering, University of Belgrade, from 1988 to 1997, as well as with the Faculty of Technical Sciences, University of Novi Sad, from 1986 to 1997. Since 1985, he has been a part-time research associate at the Institute of Mathematics, Serbian Academy of Science and Arts, Belgrade. From 1993 to 1997, he worked as a research scientist at the Information Security Research Centre, Queensland University of Technology, Australia. Since 1997, he has been an associate professor with the School of Electrical Engineering, University of Belgrade.

Professor Golic is a member of the International Association for Cryptologic Research. He has taught and developed undergraduate and graduate courses in cryptology, information theory, data compression and error control coding, algebra, numerical analysis, and discrete mathematics. His research interests include design and cryptanalysis of stream and block ciphers, sequences, coding and information theory, discrete mathematics, pattern recognition, and optimization techniques.

Andrew Clark received the BSc degree in mathematics from the University of Queensland in 1991, the BAppSc (Hons) degree in mathematics from the Queensland University of Technology (QUT) in 1992, and the PhD degree in information technology from QUT in 1998. Currently, he is a research scientist at the QUT within the Information Security Research Centre, where he is employed as a full-time senior research assistant. His research interest areas include automated cryptanalysis, use of optimization techniques in cryptology, and network security.

Ed Dawson received the BSc degree in mathematics in 1969 from the University of Washington, the MA degree in 1974 from the University of Sydney, the MLitStu degree in 1981 and the MSc degree in 1985, both from the University of Queensland, and the PhD degree in information technology from the Queensland University of Technology (QUT) in 1991. He is a member of the International Association for Cryptologic Research and a fellow of the Institute of Combinatorics and Its Applications.

From 1974 to 1991, he was a lecturer at the School of Mathematics, QUT. Since 1992, he has been with the Information Security Research Centre, QUT, first as an associate professor and, since 2000, as a professor. Currently he is currently the director of the centre. Professor Dawson has taught and developed undergraduate and graduate courses in algebra and cryptology. His research interests lie in the area of cryptology, particularly in the design and analysis of encryption algorithms.