On the Resynchronization Attack *

Jovan Dj. Golić[1] and Guglielmo Morgari[2]

[1] System on Chip, Telecom Italia Lab
Via Guglielmo Reiss Romoli 274, I-10148 Turin, Italy
golic@inwind.it
[2] Telsy Elettronica e Telecomunicazioni
Corso Svizzera 185, I-10149 Turin, Italy
guglielmo.morgari@telsy.it

Abstract. The resynchronization attack on stream ciphers with a linear next-state function and a nonlinear output function is further investigated. The number of initialization vectors required for the secret key reconstruction when the output function is known is studied in more detail and a connection with the so-called 0-order linear structures of the output function is established. A more difficult problem when the output function is unknown is also considered. An efficient branching algorithm for reconstructing this function along with the secret key is proposed and analyzed. The number of initialization vectors required is larger in this case than when the output function is known, and the larger the number, the lower the complexity.

Keywords: Stream ciphers, Boolean functions, Resynchronization, Reconstruction algorithms

1 Introduction

A typical stream cipher is based on a keystream generator as an autonomous finite-state automaton whose output sequence is reversibly combined with a plaintext sequence to yield a ciphertext sequence. A practical stream cipher also uses a reinitialization algorithm which combines a secret key and a known parameter called initialization vector (IV) into an initial state of the keystream generator. Reinitialization enables reusage of the same secret key with different IV's for encrypting relatively short messages by different keystreams. This is important for resynchronization purposes as well as for late entry in (multiparty) communication links.

Reinitialization can increase the security due to shorter keystreams available for cryptanalysis, but can also decrease the security due to multiple keystreams derived from the same secret key. It is known that reasonably secure keystream generators can be constructed from a linear next-state function and a nonlinear output function, e.g., nonlinear filter generators and memoryless combiners, both based on linear feedback shift registers. However, it is shown in [2] that such a

* Most of this work was done while the authors were with Rome CryptoDesign Center, Gemplus, Italy.

keystream generator when used together with a linear reinitialization algorithm is totally insecure if the output function depends on a relatively small number of input variables. More precisely, for an n-bit output Boolean function and a k-bit secret key, the complexity of the attack is about $k2^n$ evaluations of the output function to obtain a system of linear equations for the secret key, which is then reconstructed by solving the system. A common and more secure technique for reinitialization is to produce the initial state of the keystream generator from the output of the keystream generator itself when loaded with a linear combination of the secret key and IV (e.g., see [1]). The resynchronization attack [2] may then be useful for recovering the secret key from a set of previously reconstructed initial states of the keystream generator.

A keystream generator can be rendered more secure by letting the secret key control the structure, that is, the next-state and/or output functions. In particular, only the output function can be chosen by the secret key. As in this case the resynchronization attack [2] is no longer applicable, it is interesting to investigate if other, more sophisticated attacks can then be developed.

Sections 2, 3, and 4 are devoted to the first objective of this paper which is to conduct a more in-depth analysis of the resynchronization attack and thus obtain more precise estimates of the number of IV's required for the secret key reconstruction given a general output Boolean function. The main properties of the so-called 0-order linear structures of Boolean functions are pointed out and their impact on the attack is determined. A characterization of Boolean functions in terms of 0-order and 1-order linear structures is also established. The second objective, which is to investigate the more difficult case when the output function is not known, is treated in Sections 5 and 6. An efficient algorithm for reconstructing this function along with the secret key is developed and its complexity is analyzed in terms of the number of IV's available. The main results and open problems are summarized in Section 7.

2 Problem Statement

According to [2], consider a general binary keystream generator with a linear next-state function $S_{t+1} = L_{\mathrm{state}}(S_t)$, where S_t is the internal state at time t, with a linear initialization function $S_0 = L_{\mathrm{init}}(K, IV)$, where S_0 is the initial state, K is the secret key, and IV is the initialization vector, and with an output function $z_t = f(L_{\mathrm{out}}(S_t))$, where z_t is the output (keystream) bit at time t, f is a nonlinear Boolean function, and L_{out} is a linear function.

Let IV_i, $1 \le i \le Q$, be given IV's and let the corresponding output bits be known at times $t \in T$, in the known keystream scenario. They define a system of nonlinear equations in K of the form

$$z_t^i \;=\; f(L_t(K) \oplus L_t^0(IV_i)), \quad 1 \le i \le Q, \quad t \in T, \tag{1}$$

where L_t and L_t^0 are linear functions derived from

$$L_{\mathrm{out}}(L_{\mathrm{state}}^t(L_{\mathrm{init}}(K, IV_i))) \;=\; L_t(K) \oplus L_t^0(IV_i) \tag{2}$$

and $\oplus$ denotes the bitwise addition. One problem, considered in [2], is to find a solution for K when f is known. Another, more difficult problem is to find a solution for K and f when f is unknown. Note that for a k-bit secret key K and an n-bit function f, the exhaustive search would require 2^k steps for the first problem and 2^{2^n+k} steps for the second problem.

3 Zero-Order and First-Order Linear Structures

Solving the system (1) depends on whether the output function has linear structures or not. Recall that an n-bit vector γ is called a linear structure of an n-bit Boolean function f if $f(X) \oplus f(X \oplus \gamma) \equiv$ const. It is known that the set of all linear structures of f is a vector space. It is shown in [4] that f has nonzero linear structures iff it can be expressed as $g(A(X))$ where A is a linear function and g is a function that is partially linear or that depends on less than n variables. According to [3], we can divide the linear structures into the so-called 0-order and 1-order linear structures. A vector γ is said to be a 0-order linear structure of f if $f(X) \oplus f(X \oplus \gamma) \equiv 0$. The all-zero vector is called the trivial (0-order) linear structure. Similarly, a vector γ is said to be a 1-order linear structure of f if $f(X) \oplus f(X \oplus \gamma) \equiv 1$. The linear structures of f are directly related to the autocorrelation function of f and can be determined with the complexity $O(n2^n)$ by using the Walsh-Hadamard transform of f (e.g., see [3]).

Here we give without proof a number of novel properties of Boolean functions related to 0-order linear structures which are interesting for the resynchronization attack. For the sake of completeness, we also give some properties of Boolean functions related to 1-order linear structures. Note that the distinction between 0-order and 1-order linear structures enables us to obtain novel characterizations of Boolean functions, by Propositions 4 and 8, which are more precise than the characterizations in terms of linear structures given in [4] and [3]. In particular, 0-order linear structures account for the degeneracy, whereas 1-order linear structures account for the partial linearity. Let $\mathcal{L}$, $\mathcal{L}_0$, and $\mathcal{L}_1$ denote the sets of all linear structures, all 0-order linear structures, and all 1-order linear structures of a given Boolean function f, respectively.

Proposition 1. *The set $\mathcal{L}_0$ is a vector space.*

Proposition 2. *The cardinality $|\mathcal{L}_0| = 2^m$ divides* $\gcd(|f^{-1}(0)|, |f^{-1}(1)|)$, *where $f^{-1}(i) = \{X | f(X) = i\}$, $i = 0, 1$, and for nonconstant f, m attains its maximum $n-1$ iff f is affine.*

Proposition 3. *The binary relation $X_1 \cong X_2$ iff $X_1 \oplus X_2 \in \mathcal{L}_0$ (i.e., iff $f(X \oplus X_1) \equiv f(X \oplus X_2)$) is an equivalence relation. The corresponding equivalence classes $\{X \oplus \gamma | \gamma \in \mathcal{L}_0\}$ all have cardinality $|\mathcal{L}_0|$.*

Proposition 4. *Let Λ_0 be an m-dimensional subspace of $\{0,1\}^n$ and let $\Lambda_0^\perp$ be the dual (orthogonal) space of Λ_0. Then $\mathcal{L}_0 = \Lambda_0$ iff $f(X) \equiv g(A(X))$ where g is an $(n-m)$-bit Boolean function without nontrivial 0-order linear structures and A is a linear function represented by a matrix, acting on one-column vectors, whose rows generate $\Lambda_0^\perp$.*

Proposition 5. *The dimension m of $\mathcal{L}_0$ is the maximal nonnegative integer j such that $f(X) \equiv g(A(X))$ where g is an $(n-j)$-bit Boolean function and A is linear.*

Proposition 6. *Let $\mathcal{S}(f)$ denote the set of all n-bit Boolean functions h such that for some C, $h(X) \equiv f(X \oplus C)$. Then $|\mathcal{S}(f)| = 2^{n-m}$ if $|\mathcal{L}_0| = 2^m$.*

Proposition 7. *Either $|\mathcal{L}_1| = 0$ or $|\mathcal{L}_1| = |\mathcal{L}_0|$. If $|\mathcal{L}_1| > 0$, then $|f^{-1}(0)| = |f^{-1}(1)|$.*

Proposition 8. *Let Λ be an $(m+1)$-dimensional subspace of $\{0,1\}^n$, let Λ_0 be an m-dimensional subspace of Λ, and let $\Lambda_1 = \Lambda \setminus \Lambda_0$. Then $\mathcal{L}_0 = \Lambda_0$ and $\mathcal{L}_1 = \Lambda_1$ iff $f(X) \equiv g(A(X))$ where g is an $(n-m+1)$-bit Boolean function without nontrivial 0-order linear structures that is linear in the first variable (g has exactly one 1-order linear structure, that is, the vector $(1,0,\cdots,0)$) and A is a linear function represented by a matrix whose rows generate $\Lambda_0^\perp$ and whose rows without the first row generate $\Lambda^\perp$. Also, $\mathcal{L}_0 = \Lambda_0$ and $\mathcal{L}_1$ is empty iff $f(X) \equiv g(A(X))$ where g is an $(n-m)$-bit Boolean function without nontrivial linear structures and A is a linear function represented by a matrix whose rows generate $\Lambda_0^\perp$.*

4 Known Output Function

In the system (1), let q_t denote the number of different $L_t^0(IV_i)$ for a given t. If we assume that the rank of L_t^0 is maximal, n, then for moderately large 2^n, q_t can be approximated by the classical occupancy probabilistic model as

$$q_t \;\approx\; q \;=\; 2^n(1 - e^{-Q/2^n}). \tag{3}$$

Consequently, the system (1) can be put in the form

$$z_t^i \;=\; f(X_t \oplus C_t^i), \quad 1 \le i \le q_t, \quad t \in T, \tag{4}$$

where $X_t = L_t(K)$ and C_t^i are all different for each $t \in T$. For simplicity, in view of (3), we can assume that $q_t = q$ for every $t \in T$. Let k be the bit length of K and let $\tau = |T|$.

When f is a known n-bit function and n is relatively small, the system (4) can be solved by the method proposed in [2]. Namely, for each chosen t, find X_t by exhaustive search, where the required number of different C_t^i, q, is estimated to be about n, because the required number of different equations in X_t should roughly be equal to the number of binary variables in X_t. This then takes about $n2^n$ evaluations of f. As each found X_t defines n linear equations in K, T has to be sufficiently large so as to obtain k linearly independent equations in K. In particular, if T is such that the equations determined by X_t corresponding to different $t \in T$ are all linearly independent, then the required cardinality of T is $\tau = \lceil k/n \rceil$. Altogether, this takes about $k2^n$ evaluations of f. At the final stage,

K is obtained by solving the resulting system of linear equations. Our objective here is to study in more detail the required q and τ for a general f.

The first note is about the number of solutions for X_t. If f has nontrivial 0-order linear structures, i.e., if the number of 0-order linear structures is 2^m, $m > 0$, then Proposition 3 implies that the number of solutions is a multiple of 2^m and for large enough q it is exactly 2^m. More precisely, since f can then be put in the form specified by Propositions 4 and 5, where the function g has no nontrivial 0-order linear structures, it follows that for large enough q there is a unique solution for the linear function of X_t, $A(X_t)$. As $A(X_t)$ defines $n - m$ linear equations, the required cardinality of T then increases to $\tau = \lceil k/(n-m) \rceil$. However, as the number of variables is then effectively reduced from n to $n - m$, the total complexity of obtaining the system of linear equations in K reduces to about $k2^{n-m}$ evaluations of g.

The second note is about how large q has to be in order to reach the minimal number of solutions for X_t, under the assumption that the vectors C_t^i and X_t are all chosen at random. The minimal value of q needed, $q_{\min}$, depends on the choice of these vectors and on f. As nontrivial 0-order linear structures effectively reduce the number of variables, it is appropriate to investigate randomly chosen f without nontrivial 0-order linear structures. In the experiments, for a chosen f, X_t is chosen at random, and the q output bits z_t^i are then produced by (4) from randomly chosen different vectors C_t^i. It turns out that an important parameter affecting $q_{\min}$ is the relative number, p, of 1's in the truth table of f, $p = |f^{-1}(1)|/2^n$. Let $\bar{q}_{\min}$ be the average of $q_{\min}$ over random f and over random choices of the vectors C_t^i and X_t. A simple information-theoretic argument then yields a necessary condition $\bar{q}_{\min} \geq n/H(p)$, where the value of the binary entropy function $H(p)$ is the average number of bits of information about X_t provided by each equation in the system. In particular, for $p = 1/2$ we get $\bar{q}_{\min} \geq n$, whereas for $p = 0$ or $p = 1$ we naturally get that $\bar{q}_{\min} = \infty$, which means that the secret key cannot be recovered at all.

However, the experiments by computer simulations show that $\bar{q}_{\min}$ is larger than $n/H(p)$ as well as that the probability that $q_{\min}$ is considerably larger than this lower bound is not small. As a consequence, both the number of IV's required and the attack complexity increase. An explanation for this is that the information contents of individual (nonlinear) equations are generally not mutually independent. Fig. 1 displays the estimates of the probability distribution of $q_{\min}$ obtained from 10,000 randomly chosen 8-bit f for $p = 1/2$ and $p = 1/4$. Fig. 2 displays the average values of $q_{\min}$ as a function of p obtained from 100,000 randomly chosen 8-bit f along with the lower bound $n/H(p)$. Similar curves were also obtained for $n = 6, 10$. Accordingly, more IV's are required for nonbalanced than balanced f for the attack to be successful. In other words, the cryptographically most interesting case $p = 1/2$ requires the minimal number of IV's on average.

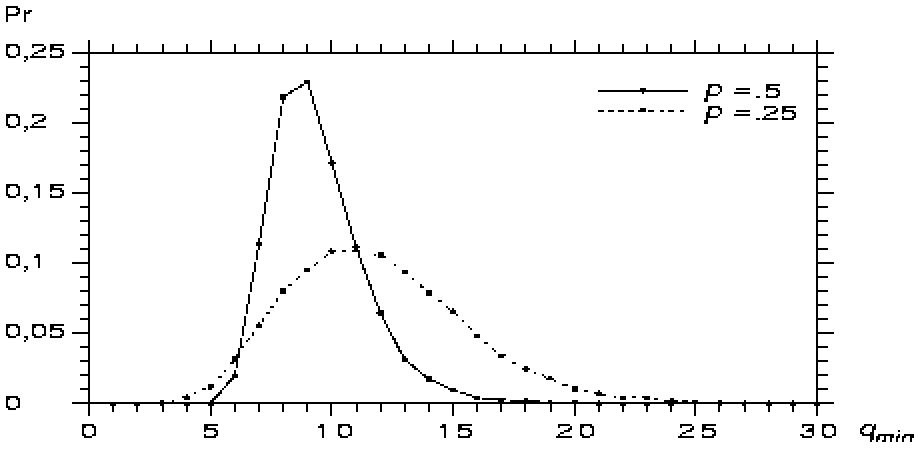

Fig. 1. Probability distributions of $q_{\min}$, for $n = 8$.

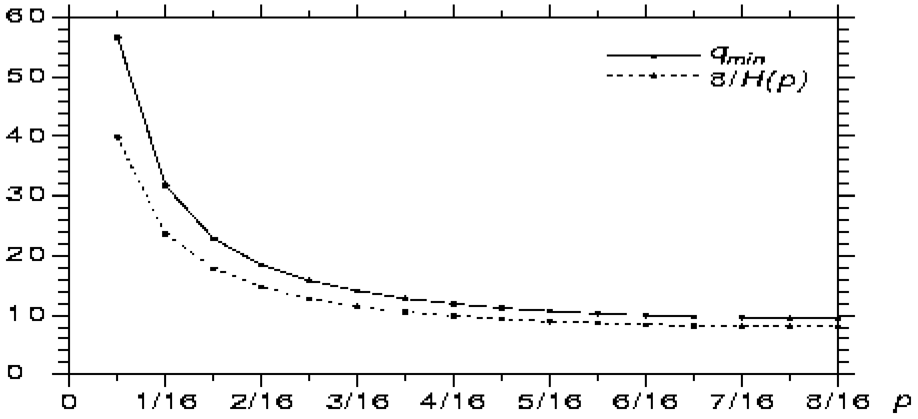

Fig. 2. Average values of $q_{\min}$ and $n/H(p)$, for $n = 8$.

5 Unknown Output Function - Complete IV Set

When f is not known, e.g., when it is defined by a secret key, the attack from the preceding section is not applicable. We first consider the case when there exists at least one value of t such that $q_t = 2^n$. In accordance with (3), this on average requires that $Q = 2^n \ln 2^n$, provided that $\tau = 1$. For $\tau > 1$, this average value is somewhat reduced, depending on τ. For any such t, the set of all n-bit functions h consistent with (4) is exactly the set $\mathcal{S}(f)$ of cardinality 2^{n-m} from Proposition 6. So, the most one can get from the system (4) is the set $\mathcal{S}(f)$, to which f belongs. Once the set $\mathcal{S}(f)$ is recovered, the method then consists of running the attack from the preceding section for each candidate function $h \in \mathcal{S}(f)$ and of testing the obtained secret key K on additional output bits, $n - m$ on average. Both f and K are thus recovered with the complexity only 2^{n-m} times larger than when f is known.

6 Unknown Output Function - Incomplete IV Set

In this section, we consider a more difficult case when $q_t < 2^n$ for every $t \in T$, so that the set $\mathcal{S}(f)$ cannot be recovered from any single value of t. In order to obtain this set, we have to combine the information from different observation times t, and this can be achieved by the following algorithm. The times t are first arranged in order of descending values of q_t and denoted as $1, 2, \cdots, \tau$.

Then initially, for each $1 \leq t \leq \tau$, compute a partially defined function h_t^0 representing $\mathcal{S}(h_t^0)$ that is consistent with (4) at time t. More precisely, q_t binary values of h_t^0 are defined by using (4) with the all-zero vector instead of X_t, that is,

$$h_t^0(C_t^i) \;=\; z_t^i, \quad 1 \leq i \leq q_t, \quad 1 \leq t \leq \tau, \tag{5}$$

while the remaining values of h_t^0 remain undefined. If we denote the undefined values by the symbol b, then each h_t^0 effectively takes three output values: 0, 1, and b.

The algorithm essentially consists of searching through a tree of candidate functions h representing all $\mathcal{S}(h)$ that are consistent with the current and previous observations combined. More precisely, a three-valued function, with a generic notation h, is assigned to each node in the tree in the following way. Initially, at level 1, start from a single node with the associated candidate function $h = h_1^0$, and then proceed iteratively. Consider a node at level j with the associated function h. The successor nodes to this node are derived from all different $Y \in \{0,1\}^n$ such that

$$h(X) \;=\; h_{j+1}^0(X \oplus Y) \tag{6}$$

for every $X \in \{0,1\}^n$ such that neither $h(X)$ nor $h_{j+1}^0(X \oplus Y)$ is equal to b. For each such Y, then use (6) to modify h for every X such that $h(X) = b \neq h_{j+1}^0(X \oplus Y)$ by setting $h(X) = h_{j+1}^0(X \oplus Y)$. The modified h is the candidate function associated with the successor node corresponding to Y.

If there does not exist such an Y, then there are no successors to the considered node. This means that the candidate function h assigned to this node is inconsistent with the observations and as such is incorrect, although it may be fully defined. Such an end node is called a failure end node. On the other hand, if there exists such an Y, then the candidate function assigned to any successor node has at most as many undefined values as the candidate function assigned to the considered node. Any node with a binary, fully defined candidate function is called a success end node if it has a (unique) successor and every subsequent node, if generated, would have a (unique) successor. In practice, if the tree is examined by the depth-first search with backtracking, with a negligible space complexity, this can be checked on a small number of additional nodes. Alternatively, one can apply the width-first search by storing and examining one level of the tree at a time. In this case, at each level, one does not have to examine different nodes with the same candidate function more than once, which means that only the nodes with different candidate functions assigned are further processed. The algorithm then stops when a level with a single node with a fully defined candidate function is reached.

In theory, even if $\tau = \infty$, it is possible, but extremely unlikely, that a success end node is never reached. In this case, after a certain point, the new observation times contain no additional information about f and the obtained candidate functions are not being updated at all.

After a success end node is found, the set $S(f)$ is recovered and the rest is the same as when the IV set is complete. The effectiveness of the algorithm can be measured by the number of different observation times required to reach a solution and by the total number of nodes examined. Both depend on the number of IV's available. The worst-case time complexity is reflected by the total number of nodes in the whole tree. The complexity per node is at most 2^{2^n} elementary operations with the values 0, 1, and b.

Example 1. Let $f(x_1, x_2, x_3) = 1 \oplus x_1 \oplus x_2 \oplus x_1x_2 \oplus x_1x_3$, $\tau = 6$, and $q_t = q = 4, 1 \leq t \leq 6$. Further, let the output bits z_t^i be produced by (4) from the vectors X_t and C_t^i as displayed in Fig. 3. Then all the different candidate functions obtained are shown in Fig. 4, where h_t stands for a generic candidate function at time t. For each $2 \leq t \leq 6$, the numbers of the form $l_1 \rightarrow l_2$ are also shown, where l_1 stands for the total number of candidate functions obtained and l_2 stands for the total number of (displayed) different functions among them, because different candidate functions at time $t-1$ can give rise to the same candidate function at time t.

At time $t = 3$ there appear 3 fully defined candidate functions, whereas at time $t = 4$ there are also 3 such functions, but only 2 of them are from time $t = 3$. So, at time $t = 4$, one fully defined candidate function disappears as inconsistent, while a new one appears. At time $t = 5$ the set of candidate functions is not updated and at time $t = 6$ a unique consistent solution, h_6, is found, which is different from all the previously obtained fully defined candidate functions. Note that $f \neq h_6$, but $f \in S(h_6)$. More precisely, $f(X) = h_6(X \oplus X_1)$, where $X_1 = 011$ (see Fig. 3).

X_1	X_2	X_3	X_4	X_5	X_6
011	100	000	001	100	100

	C^i_1	C^i_2	C^i_3	C^i_4	C^i_5	C^i_6
i=1	110	110	101	111	101	000
i=2	010	001	010	010	110	111
i=3	000	010	000	000	100	101
i=4	101	111	001	110	111	100

	z^i_1	z^i_2	z^i_3	z^i_4	z^i_5	z^i_6
i=1	1	0	1	0	0	1
i=2	0	1	0	0	0	0
i=3	0	0	1	0	1	0
i=4	0	0	0	1	0	1

Fig. 3. The vectors X_t and C^i_t and the output bits z^i_t, in Example 1.

h_1	h_2				h_3						h_4						h_5						h_6
	4 → 4				6 → 6						13 → 6						9 → 6						1 → 1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
b	b	b	0	0	0	b	1	0	0	0	0	b	1	1	0	0	0	b	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
b	1	0	b	b	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1
b	0	1	0	b	0	0	1	0	b	0	0	0	0	1	b	1	0	0	0	1	b	1	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
b	b	0	1	b	b	0	0	1	b	0	0	0	0	0	0	b	0	0	0	0	0	b	1

Fig. 4. Candidate functions h_t, in Example 1.

In the experiments, we used (4) with randomly generated distinct vectors C_t^i, for variable values of q. It turns out that a typical tree first grows and then gradually shrinks to a single node with a partially defined candidate function. Finally, it takes a number of additional levels for this function to be updated into a fully defined function, at which point the algorithms stops. Fig. 5 shows the dependence of the logarithm to the base 2 of the number of nodes with different candidate functions upon the tree level, for the trees obtained from the same randomly chosen balanced 8-bit function f and a number of different q. It is not shown that in this particular experiment the success end nodes were reached after $t = 68, 45, 47$, and 57 levels for $q = 24, 28, 32$, and 36, respectively.

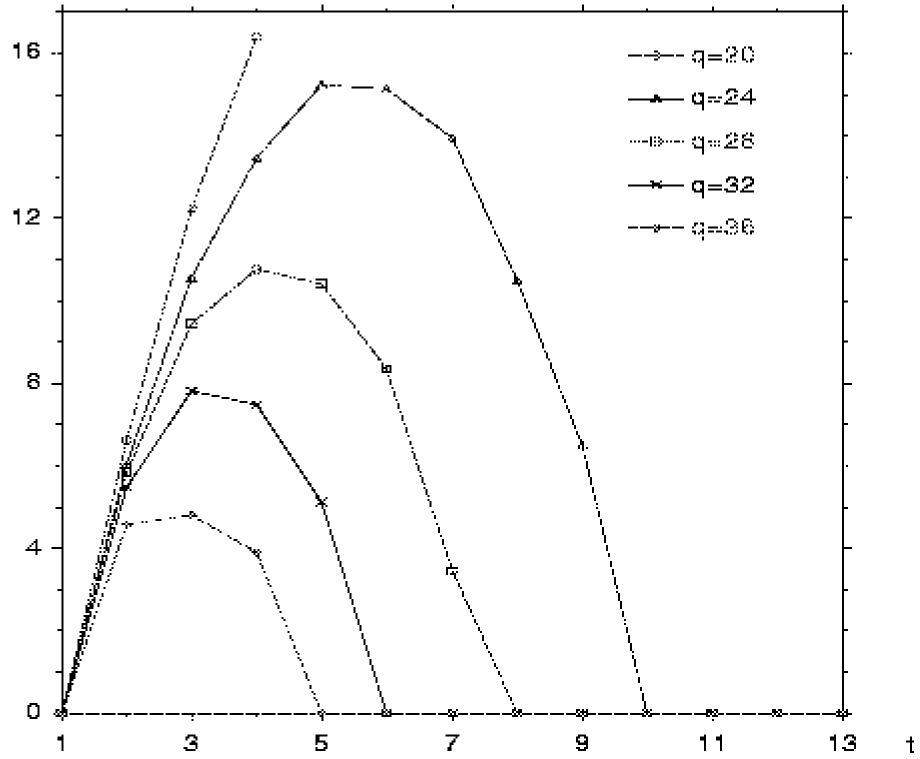

Fig. 5. Log$_2$ of the number of nodes as a function of the tree level.

Let $\tau_{\rm min}$ denote the minimal number of levels in a tree until a level with a single success end node is reached and let N denote the total number of nodes with different candidate functions at each level. Further, let $\hat{\tau}_{\rm min} = 2^n \ln 2^n/q$ denote an approximation for $\tau_{\rm min}$ according to the classical occupancy model.

Fig. 6 displays the logarithms to the base 2 of the average values of N, $\tau_{\min}$, and $\hat{\tau}_{\min}$, for variable values of q, where the averages were obtained over 1000 randomly generated 8-bit functions f.

It is interesting that for the values of q larger than a critical point q_2 ($q_2 \approx 55$, for $n = 8$), $\tau_{\min} \approx N$, meaning that for almost all f each node in the tree has a unique successor. Branching occurs for $q < q_2$, and for the values of q smaller than another critical point, q_1 ($q_1 \approx 35$, for $n = 8$), N grows rapidly as q decreases. In particular, if $q \approx n$, then the complexity appears to be prohibitively high. Recall that on average, q has to be larger than n for the rest of the attack to be applicable. For $q < q_2$, $\hat{\tau}_{\min}$ appears to be a very good approximation. Similar behavior is expected for any value of n.

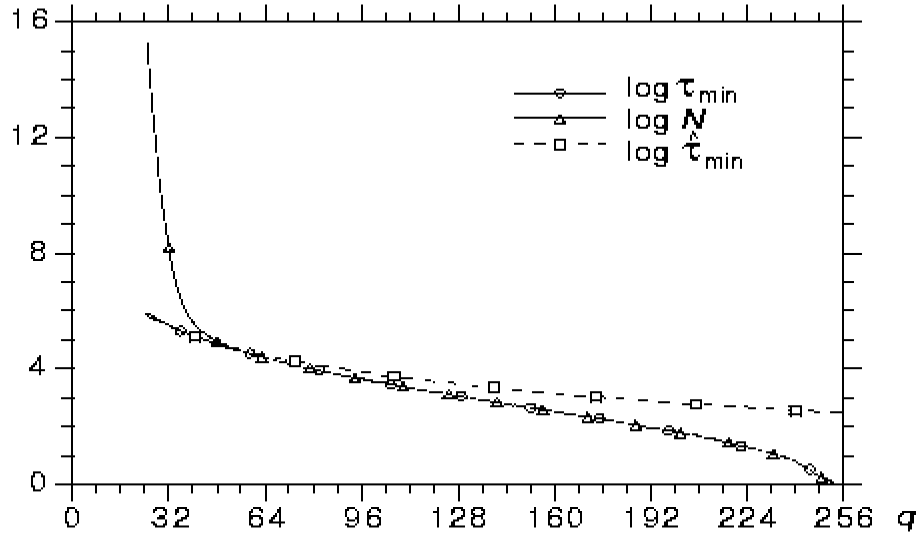

Fig. 6. Average values of $\tau_{\min}$, $\hat{\tau}_{\min}$, and N, for $n = 8$.

7 Conclusions

It is shown that the number of initialization vectors required for a successful resynchronization attack can be larger than the number of binary inputs to the output function. The main properties of the so-called 0-order and 1-order linear structures of Boolean functions are established and it is pointed out that the nonzero 0-order linear structures of the output function can simplify the resynchronization attack.

More importantly, a new algorithm is proposed which shows that the attack can also work when the output function is not known provided that the number of initialization vectors is sufficiently large. This algorithm is able of reconstructing both the output function and the secret key, and the larger the number of initialization vectors, the lower the complexity.

If the number of initialization vectors is relatively small, then the complexity becomes prohibitively high. In this case, analyzing the complexity of this algorithm theoretically as well as finding other, possibly more effective algorithms are problems interesting for future investigations.

References

1. A. Clark, E. Dawson, J. Fuller, J. Dj. Golić, H.-J. Lee, W. Millan, S.-J. Moon, and L. Simpson, "The LILI-II keystream generator," Information Security and Privacy - ACISP 2002, *Lecture Notes in Computer Science*, vol. 2384, pp. 25-39, 2002.
2. J. Daemen, R. Govaerts, and J. Vandewalle, "Resynchronization weakness in synchronous stream ciphers," Advances in Cryptology - EUROCRYPT '93, *Lecture Notes in Computer Science*, vol. 765, pp. 159-167, 1994.
3. S. Dubuc, "Characterization of linear structures," *Designs, Codes and Cryptography*, vol. 22, pp. 33-45, 2001.
4. X. Lai, "Additive and linear structures of cryptographic functions," Fast Software Encryption - FSE '94, *Lecture Notes in Computer Science*, vol. 1008, pp. 75-85, 1995.