REFERENCES

[1] J. H. Tucker, "A transition calculus for Boolean functions," Ph.D. dissertation, Dep. Elec. Eng., Virginia Polytech. Inst. and State Univ., Blacksburgh, VA, May 1974.

[2] J. H. Tucker, M. A. Tapia, and A. W. Bennett, "Boolean differentiation and integration using Karnaugh maps," in *Proc. IEEE Southeast Conf.*, Williamsburg, VA, Apr. 1977.

[3] A. Thayse and M. Davio, "Boolean differential calculus and its application to switching theory," *IEEE Trans. Comput.*, vol. C-22, pp. 409–420, Apr. 1973.

[4] A. D. Talantsev, "On the analysis and synthesis of certain electrical circuits by means of special logical operators," *Avt. i Telem.*, vol. 20, no. 7, pp. 898–907, 1959.

[5] V. G. Lazarev and E. I. Piil, "The simplification of pulse-potential forms," *Avt. i Telem.*, vol. 24, no. 2, pp. 271–276, Feb. 1963.

[6] A. Brown and H. Young, "Toward an algebraic theory of the analysis and testing of digital networks," *AAS & ORS Annu. Meet.*, Denver, CO, June 17–20, 1969, AAS Paper 69-236.

[7] J. R. Smith, Jr. and C. H. Roth, Jr., "Analysis and synthesis of asynchronous sequential networks using edge-sensitive flip-flops," *IEEE Trans. Comput.*, vol. C-20, pp. 847–855, Aug. 1971.

[8] M. A. Tapia and J. H. Tucker, "Complete solution of Boolean equations," *IEEE Trans. Comput.*, vol. C-29, pp. 662–665, July 1980.

[9] M. A. Tapia, "Application of Boolean calculus to digital system design," in *Proc. IEEE Southeast Conf.*, Nashville, TN, Apr. 14–16, 1980.

Decrypting a Class of Stream Ciphers Using Ciphertext Only

T. SIEGENTHALER

Abstract **— Pseudonoise sequences generated by linear feedback shift registers [1] with some nonlinear combining function have been proposed [2]–[5] for cryptographic applications as running key generators in stream ciphers. In this correspondence it will be shown that the number of trials to break these ciphers can be significantly reduced by using correlation methods. By comparison of computer simulations and theoretical results based on a statistical model, the validity of this analysis is demonstrated. Rubin [6] has shown that it is computationally feasible to solve a cipher proposed by Pless [2] in a known plaintext attack, using as few as 15 characters. Here, the number of ciphertext symbols is determined to perform a ciphertext-only attack on the Pless cipher using the correlation attack. Our conclusion from the analysis is that the pseudonoise generator's output sequence and the sequences generated by the linear feedback shift registers should be uncorrelated. This leads to constraints for the nonlinear combining function to be used.**

Index Terms **— Correlation, cryptanalysis, exhaustive trials, pseudonoise generator.**

I. INTRODUCTION

In conventional cryptography pseudonoise (pn) generators consisting of s linear feedback shift registers (LFSR's) of length r_i ($i = 1, 2, \cdots, s$) are used. The combining function f is arbitrary but known. However, to avoid a cryptanalytic attack by the Berlekamp–Massey shift register synthesis algorithm [7], [8], only nonlinear functions can be used (see Fig. 1). These pn-generators have been proposed as running key generators in stream ciphers (see Fig. 2). The symbol $\oplus$ denotes bit-by-bit modulo-2 addition throughout the whole correspondence. We assume that the key of the cryptographic system specifies the initial states and the feedback coefficients of the different (binary) linear feedback shift registers. The initial condition and feedback connection of the LFSRi are referred to as the LFSRi part of the key. Further, it is assumed that the feedback connections of all LFSR's of length r_i ($i = 1, \cdots, s$) are primitive [1] or in other words that all LFSR's generate a maximal length sequence of period $2^{r_i} - 1$. The number R_i of different primitive feedback connections for an LFSR can be determined from its length r_i [1]. A length r_i (binary) LFSR has 2^{r_i} different initial states, however, the all zero state which generates the all zero sequence is not allowed. Therefore, a total of $R_i(2^{r_i} - 1)$ choices for the LFSRi part of the key exist and the total number K of keys for the pn-generator given in Fig. 1 is

$$K = \prod_{i=1}^{s} R_i(2^{r_i} - 1).$$

In a brute force attack and a worst case situation all of the K keys have to be applied which is by definition not feasible for a computationally secure pn-generator. However, a weakness of the generators which belong to the class of Fig. 1 may be the correlation between some of the inputs x_i and the output z. Based on this correlation [11] it is demonstrated in Section II that the LFSRi part of the key can be found independent of the LFSRj parts ($j = 1, \cdots, s; j \neq i$) with approximately $R_i 2^{r_i}$ tests. Making use of that for finding the key of the pn-generator, the number of trials can be significantly reduced from K to approximately

$$\sum_{i=1}^{s} R_i 2^{r_i}.$$

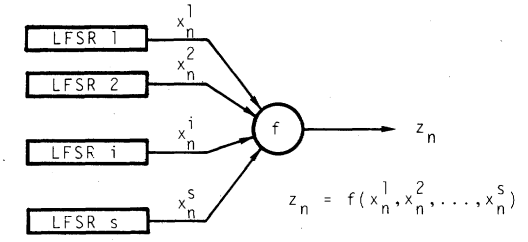

Fig. 1. A class of pn-generators with a nonlinear combining function f.

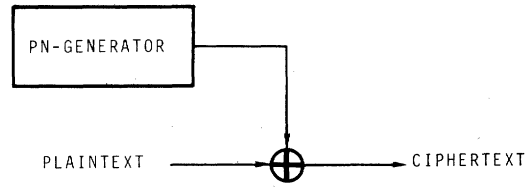

Fig. 2. Running key generator in a stream cipher.

II. STATISTICAL MODEL FOR A CIPHERTEXT-ONLY ATTACK

In this section, a statistical model is used to find the LFSRi part of the key, i.e., the initial state and feedback connection of the LFSRi $i \in \{1, \cdots, s\}$. Further, the number of tests to find the LFSRi-part of the key is determined as a function of the number of ciphertext digits used in the correlation attack. Let the inputs $x_n^1, x_n^2, \cdots, x_n^s$ of the function f in Fig. 3 be generated by independent and identically distributed (i.i.d) random variables (r.v.) X_n^i with probability distribution P_X such that $P(X_n^i = 0) = P(X_n^i = 1)$ for all i and n. The function f generates i.i.d. r.v. $Z_n = f(X_n^1, X_n^2, \cdots, X_n^s)$ with probability distribution P_Z where $P(Z_n = 0) = P(Z_n = 1)$ and

$$P(Z_n = X_n^i) = q_i. \tag{1}$$

Manuscript received July 11, 1983; revised November 18, 1983.

The author is with the Institute for Communication Technology, Federal Institute of Technology, 8092 Zurich, Switzerland.

0018-9340/85/0100-0081$01.00

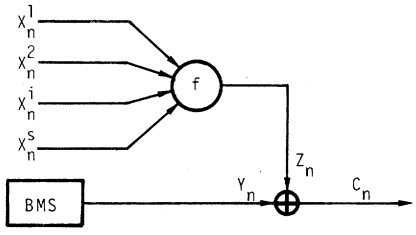

Fig. 3. Statistical model.

The plaintext is assumed to be the output of a binary memoryless source (BMS) with

$$P(Y_n = 0) = p_0 . \tag{2}$$

The r.v. α as a measure for the correlation between C_n and X_n^i is defined as

$$\alpha = \sum_{n=1}^{N}(1 - 2(C_n \oplus X_n^j)) = N - 2\sum_{n=1}^{N}(C_n \oplus X_n^j) \qquad j \in \{0, 1, \cdots, s\} \tag{3}$$

where X_n^0 shall be defined as virtual r.v.'s which are independent of X_n^i $(i = 1, 2, \cdots, s)$ and i.i.d. with probability distribution P_X and $P(X_n^0 = 0) = P(X_n^0 = 1) = 0.5$. Thus, the probability $P(C_n \oplus X_n^j = 0) = p_e$ can be determined

$$\begin{aligned} p_e &= P(Z_n = X_n^j) \cdot P(Y_n = 0) + P(Z_n \neq X_n^j) \cdot P(Y_n = 1) \\ &= 1 - (p_0 + q_j) + 2p_0q_j = P(C_n \oplus X_n^j = 0), \end{aligned} \tag{4}$$

and therefore

$$P(C_n \oplus X_n^j = 1) = 1 - p_e .$$

Due to the fact that for all n and some fixed $j \in \{0, 1, \cdots, s\}$ all of the terms $(C_n \oplus X_n^j)$ in the sum given in (3) are i.i.d. r.v. $\in \{0, 1\}$ the random variable $\beta = \sum_{n=1}^{N}(C_n \oplus X_n^j)$ is binomially distributed with mean value m_β and variance σ_β^2

$$\begin{aligned} m_\beta &= N(1 - p_e) \\ \sigma_\beta^2 &= Np_e(1 - p_e) . \end{aligned} \tag{5}$$

The expected value and variance of α, m_α, and σ_α^2 will be

$$\begin{aligned} m_\alpha &= N - 2N(1 - p_e) = N(2p_e - 1) \\ \sigma_\alpha^2 &= \text{var}(N - 2\beta) = 2^2\sigma_\beta^2 \end{aligned} \tag{6}$$

and using (5)

$$\sigma_\alpha^2 = 4Np_e(1 - p_e) . \tag{7}$$

Because of the independence of Z_n and X_n^0 and because of the statistics of X_n^0 we have for $j = 0$: $q_0 = 0.5$ and $p_e = 0.5$ and with (6) and (7)

$$\left.\begin{aligned} m_\alpha &= 0 \\ \sigma_\alpha^2 &= \text{N} \end{aligned}\right\} \quad \text{for } j = 0 . \tag{8}$$

For large N, the r.v. α can be assumed to be normally distributed with parameters m_α and σ_α^2 due to the central limit theorem. In an attack an actual value α_0 for α is determined from N ciphertext digits and N digits generated by a LFSR of length r_i with an arbitrary initial state and an arbitrary of the R_i sets of feedback coefficients. There are two hypotheses to be considered.

H_1: The $N > r_i$ digits of the LFSR of length r_i coincide with N digits generated by the LFSRi of the pn-generator. This case corresponds to α being the correlation of C_n and X_n^i $i \in \{1, 2, \cdots, s\}$.

H_0: The $N > r_i$ digits of the LFSR of length r_i do not coincide with N digits generated by the LFSRi of the pn-generator. This case corresponds to α being the correlation of C_n and X_n^0.

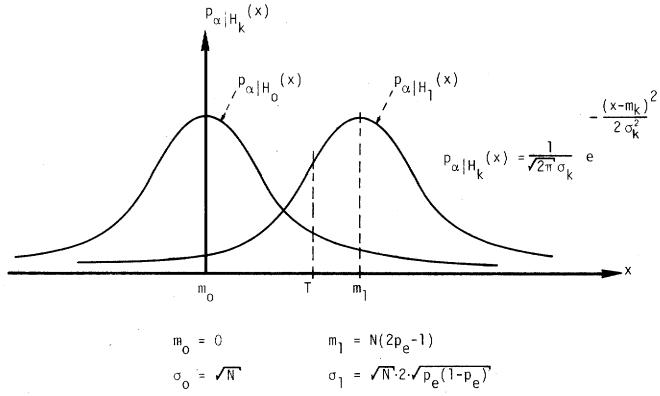

Fig. 4. Probability density function for random variable α.

Fig. 4 shows the normally distributed probability density function $p_{\alpha|H_K}(x)$ for $H_k, k = 0, 1$.

The value T shall be the decision threshold for the two hypotheses H_0 and H_1 (Fig. 4). For $\alpha_0 < T$, H_0 is accepted; for $\alpha_0 \geq T$, H_1 is accepted. For $p_e = 0.5$ (i.e., $q_i = 0.5$ or/and $p_0 = 0.5$) the two probability density functions $p_{\alpha|H_0}(x)$ and $p_{\alpha|H_1}(x)$ are identical and therefore no decision can be made. The computational effort depends on the number of wrong decisions, i.e., on the number of values α_0 exceeding the threshold T. Therefore, the probability P_f for a "false alarm" $P(\alpha \geq T \,|\, H_0)$ is of primary interest. To determine the decision threshold, however, the probability P_m for "missing the event" $P(\alpha < T \,|\, H_1)$ must also be taken into account

$$P_f = \int_T^\infty p_{\alpha|H_0}(x)\,dx, \qquad P_m = \int_{-\infty}^T p_{\alpha|H_1}(x)\,dx .$$

With

$$Q(x) = \frac{1}{\sqrt{2\pi}}\int_x^\infty e^{-y^2/2}\,dy ,$$

$$P_f = Q\left(\left|\frac{T}{\sqrt{N}}\right|\right)$$

and

$$P_m = Q\left[\left|\frac{N(2p_e - 1) - T}{2\sqrt{N}\sqrt{p_e(1 - p_e)}}\right|\right].$$

Instead of the threshold T we use for convenience

$$\gamma_0 = \frac{N(2p_e - 1) - T}{2\sqrt{N}\sqrt{p_e(1 - p_e)}}$$

and

$$\frac{T}{\sqrt{N}} = \sqrt{N}(2p_e - 1) - 2\gamma_0\sqrt{p_e(1 - p_e)}$$

which give the final expressions for P_m and P_f

$$P_m = Q(|\gamma_0|) \tag{9}$$

$$P_f = Q(|\sqrt{N}(2p_e - 1) - 2\gamma_0\sqrt{p_e(1 - p_e)}|) . \tag{10}$$

To attack a generator (Fig. 1) we proceed as follows. First, the probabilities q_i are determined from the function f. The probability p_0 is known from the code (e.g., ASCII) and the language of the plaintext. Using (4) we calculate the probability p_e. For a chosen value P_m the parameter γ_0 of (9) is a constant and from (10) the

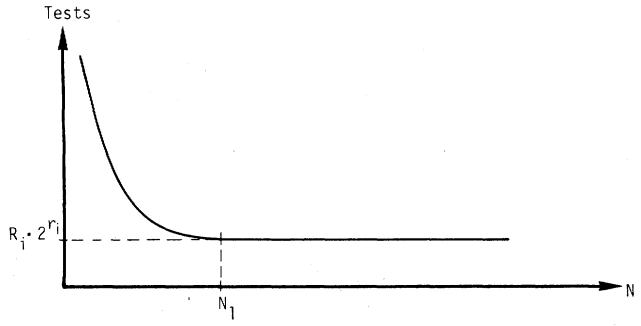

Fig. 5. Number of tests to be performed to find the LFSRi part of the key of a pn-generator (as given in Fig. 1) used in a stream cipher.

probability for "false alarm" $P(\alpha \geq T | H_0)$ can be determined as a function of the N ciphertext digits used in (3). To find the LFSRi part of the key, we choose one out of the R_i possible primitive feedback connections and generate the corresponding maximal length sequence $\{s_i\}$ of period $2^{r_i} - 1$. Then, for each of the $2^{r_i} - 1$ possible positions of $\{s_i\}$ and the N ciphertext digits, the correlation α is computed. For each event $\alpha \geq T$ it is assumed that the correct feedback connection and the correct position is used, hence the LFSRi part of the key is known. Because the event $\alpha \geq T | H_0$ occurs with probability P_f, our decision may be wrong. Therefore, additional tests with new ciphertext segments have to be performed at all positions with $(\alpha \geq T)$. If H_1 is rejected for all of the $2^{r_i} - 1$ positions, a new primitive feedback connection out of the R_i possibilities is chosen and tested. In the worst case, all of the $2^{r_i} - 1$ positions of all the possible R_i feedback connections have to be tested, i.e., approximately $R_i 2^{r_i}$ tests are necessary to find the LFSRi part of the key. The number of "false alarms" $(\alpha \geq T | H_0)$ and consequently the number of tests necessary depend on the number N of ciphertext digits used. If we choose N_1 such that

$$P_f = \frac{1}{R_i 2^{r_i}} \tag{11}$$

the expected number of "false alarms" over all $\sim R_i 2^{r_i}$ basic tests is one and the total number of tests to find the LFSRi part of the key is approximately $R_i 2^{r_i}$. The number of tests to be performed cannot be decreased by choosing $N > N_1$. From (10) and (11) we get

$$\frac{1}{R_i 2^{r_i}} = Q\left(\left|\sqrt{N_1}\,(2p_e - 1) - 2\gamma_0\sqrt{p_e(1 - p_e)}\right|\right).$$

This expression can be evaluated using tables or tight upper and lower bounds for the Q-function given in [9]. N_1 can be upper bounded by using

$$Q(x) < \frac{1}{2} e^{-x^2/2}, \qquad x \geq 0$$

which leads to

$$\frac{1}{R_i 2^{r_i}} < \frac{1}{2} e^{-1/2(\sqrt{N_1}(2p_e - 1) - 2\gamma_0\sqrt{p_e(1-p_e)})^2}$$

and

$$N_1 < \left[\frac{\frac{1}{\sqrt{2}}\sqrt{\ln(R_i 2^{r_i - 1})} + \gamma_0\sqrt{p_e(1 - p_e)}}{p_e - \frac{1}{2}}\right]^2. \tag{12}$$

The upper bound (12) can be used to roughly estimate the number of ciphertext digits to perform an attack on a system as given in Fig. 2. From bound (12) it can be seen that for a given system N_1 increases due to the relation $0 \leq \sqrt{p_e(1 - p_e)} \leq 1/2$ and the fact

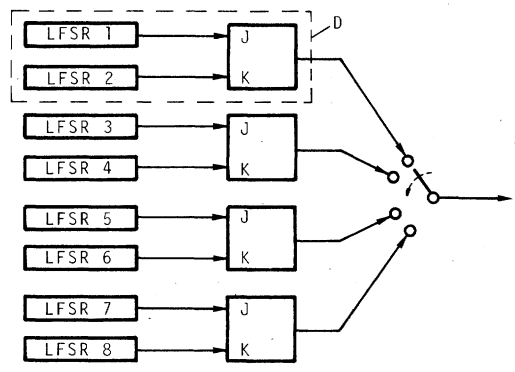

Fig. 6. pn-generator proposed by Pless, consisting of 8 LFSR's, 4 JK-flip-flops, and a sampler. Every fourth digit of part D (referred to as "Minipless") is connected to the output.

that $\sqrt{\ln(R_i 2^{r_i - 1})}$ is a very slowly increasing function of R_i and r_i approximately with $(1/2 - p_e)^{-2}$.

III. Results from the Simulations with pn-Generators

The pn-generators proposed by Pless [2], Geffe [3], and Bruer [4], [5] were investigated. The cross-correlation function (CCF) of N output digits generated by a pn-generator and one of the linear feedback shift registers LFSRi were computed for 1000 points. First, the values exceeding the threshold T were counted for different values of T. Then the number of values exceeding T was estimated using the statistical model described above. All simulations done with these pn-generators show an excellent agreement with theory. A sample of the simulation results is given for the generator proposed by Pless [2] in Fig. 8. The values obtained by the simulation are plotted by squares; the solid line represents the result of the statistical model.

This pn-generator can be split into four parts. The analysis has been done for one register pair and the JK-flip-flop (part D in Fig. 6). Fig. 7 shows a sample of 280 points of the cross-correlation function. Each point of the CCF has been computed using 100 digits generated by part D of the pn-generator. "True peak" indicates the in-phase position of one of the two incoming pn-sequences with the sequence generated by a LFSR with properly chosen initial state and feedback connection.

The value of the CCF at the in-phase position in this sample exceeds the absolute maximum of all other values by a factor of approximately 2.0.

IV. Examples

In this section the parameters q_i defined by (1) are given for the pn-generators proposed by Pless [2], Geffe [3], and Bruer [4], [5]. Furthermore, the exact values of N_1 (see Fig. 5) are given for the Pless pn-generator.

a) Geffe type (see Fig. 9). The parameters q_i $(i = 1, 2, 3)$ can be determined as $q_1 = q_3 = 0.75$; $q_2 = 0.5$. As mentioned in the section above in the case $q_i = 0.5$ a correlation attack cannot be performed. For LFSR2, however, a slightly modified correlation attack can be done after the LFSR1 and LFSR3 parts of the key have been found.

b) Bruer type (see Fig. 10). The output is equal to 1 if the number of ones at the input exceeds the threshold $n/2$; otherwise the output is zero. The parameter q_i $(i = 1, 2, \cdots, n)$ can be determined as

$$q_i = \frac{1}{2} + \binom{n - 1}{\frac{n - 1}{2}} \frac{1}{2^n} \qquad \text{for all } i = 1, 2, \cdots, n. \tag{13}$$

As an example, the probability of coincidence for input i and output of the pn-generator proposed by Bruer [4], [5] is given in Table I.

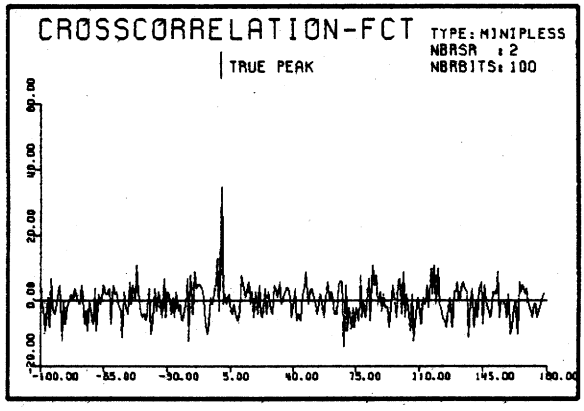

Fig. 7. Window of 280 points of the cross-correlation function.

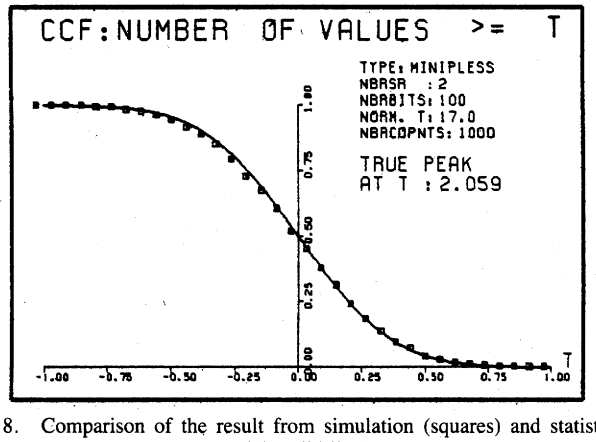

Fig. 8. Comparison of the result from simulation (squares) and statistical model (solid line).

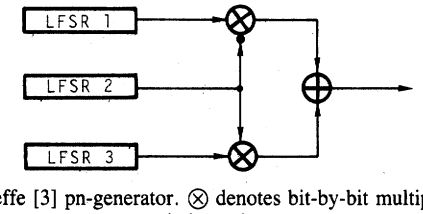

Fig. 9. Geffe [3] pn-generator. ⊗ denotes bit-by-bit multiplication and • is inversion.

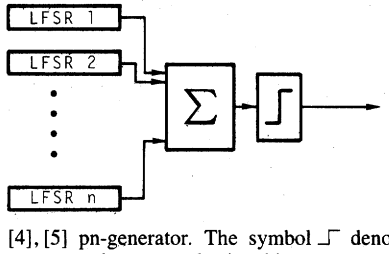

Fig. 10. Bruer [4], [5] pn-generator. The symbol ⌐ denotes a threshold detector and n is odd.

c) Pless type. Pless proposed the structure given in Fig. 6 for a pn-generator. The parameters q_i $(i = 1, 2, \cdots, 8)$ can be determined as $q_i = q = 0.75$ for all i and $p_e = 0.25 + 0.5p_0$ from (4). Only the maximal length LFSR with length 19 and 27 594 primitive feedback connections is considered. Table II lists the number of ciphertext bits N_1 to be used in a correlation attack with a minimum number of tests as discussed in Fig. 5. The rough estimate for N_1 according to the upper bound (12) is also given. The values computed by (12) using $\gamma_0 = 3$ exceed N_1 by only 7 percent.

In Table II, the number of ciphertext digits to be used for a correlation attack is shown. The Pless pn-generator is employed. The number is given as a function of the probability p_0 that the plaintext source emits a zero.

TABLE I
PROBABILITY OF COINCIDENCE

n	$q_i = q$
3	0.75
5	0.6875
7	0.6562
9	0.6367

TABLE II
NUMBER OF CIPHERTEXT DIGITS

p_0	N_1	upper bound(12)
1.0 [a]	325	348
0.9	525	562
0.8	957	1024
0.7	2190	2342
0.6	8845	9457
0.55	35466	37917

[a] corresponds to a known plaintext attack

V. SUMMARY AND CONCLUSIONS

For a class of pn-generators it has been shown that the number of trials to find the key can be reduced significantly in those cases where a correlation exists between input and output of the combining function f of the pn-generator. The number of trials can be minimized by using $N \geq N_1$ ciphertext digits for the correlation attack. An upper bound for N_1 is given. This bound is valid for arbitrary pn-generators of the considered class and can be used to determine the number of digits for an attack with a minimal number of tests. Numerical values are given for the probability q_i for a coincidence between an input and corresponding output digit for some recently proposed pn-generators. For the proposal made by Pless numerical values are given for the number of ciphertext digits N_1 to be used in the attack. If a hardware correlator is used, this pn-generator can easily be broken. Our conclusion from the above analysis is that pn-generators should be constructed using nonlinear functions f (see Fig. 1) which have the property that the output z is uncorrelated to all inputs x_i. It is pointed out that the analysis is only valid for stream cipher systems which use pn-generators as given in Fig. 1. For an example of a completely different system where this analysis cannot be applied, refer to the well-known DES [10].

ACKNOWLEDGMENT

The author wishes to acknowledge the valuable discussions with P. Schoebi of the Institute for Communication Technology, Swiss Federal Institute of Technology, Zurich, who initialized this work. Also, thanks go to Prof. Dr. P. Leuthold for the support of this work.

REFERENCES

[1] S. W. Golomb, *Shift Register Sequences*. San Francisco, CA: Holden-Day, 1967.
[2] V. S. Pless, "Encryption schemes for computer confidentiality," *IEEE Trans. Comput.*, vol. C-26, pp. 1133–1136, Nov. 1977.
[3] P. R. Geffe, "How to protect data with ciphers that are really hard to break," *Electronics*, pp. 99–101, Jan. 4, 1973.
[4] J. O. Bruer, "On nonlinear combinations of linear shift register sequences," in *Proc. IEEE Int. Symp. Inform. Theory*, les Arcs, France, June 21–25, 1982.

[5] ——, "On nonlinear combinations of linear shift register sequences," Linkoeping Univ., Sweden, Intern. Rep., Mar. 1983.
[6] F. Rubin, "Decrypting a stream cipher based on JK-flip-flops," *IEEE Trans. Comput.*, vol. C-28, pp. 483–487, July 1979.
[7] J. L. Massey, "Shift register synthesis and BCH decoding," *IEEE Trans. Inform. Theory*, vol. IT-15, pp. 122–127, Jan. 1969.
[8] E. R. Berlekamp, *Algebraic Coding Theory*. New York: McGraw-Hill, 1968, ch. 7, 10.
[9] J. M. Wozencraft and I. M. Jacobs, *Principles of Communication Engineering*. New York: Wiley, Jan. 1967, 2nd printing, p. 83.
[10] Nat. Bureau of Standards, Pub. 46, Jan. 1977.
[11] W. Blaser and P. Heinzmann, "New cryptographic device with high security using public key distribution," *IEEE Student Papers*, p. 150, 1979–1980.

Testing Strategy and Technique for Macro-Based Circuits

FABIO SOMENZI, SILVANO GAI, MARCO MEZZALAMA, AND PAOLO PRINETTO

***Abstract* — The increasing complexity of VLSI systems demands structured approaches to reduce both design time and test generation effort. PLA's and scan paths have both been widely reported to be efficient in this sense. This correspondence presents an easily testable structure and its related testing strategies. The circuits are assumed to be based on the interconnection of combinatorial macros, mostly implemented by PLA's; tests are generated locally, considering the involved macro as an isolated item, and then are expressed in terms of primary inputs and outputs using a topological approach as general strategy and algebraic techniques for the propagation of signals through macros. Propagation is dealt with by new algorithms. Since the problem of test generation is NP-hard, a set of heuristics is introduced to keep the amount of computation reasonable; several implementation issues are finally investigated.**

***Index Terms* — Automated test pattern generation (ATPG), Boolean differences, controlability, *D*-algorithm, design for testability, LSSD, observability, programmable logic array (PLA), structured design.**

I. Introduction

The challenge for VLSI designers is the management of increasing complexity. One natural approach to this problem is to pursue the maximum order in each phase of design and to partition the task on a hierarchical basis, thus providing the designer with a complete understanding of the problem at the level he is considering. Several studies have been published on these general guidelines [1]–[4].

As complexity has a dramatic impact on test generation, testability has become a common word when speaking about VLSI. Several techniques have been adopted to solve the problem; the most popular are partitioning, making internal points accessible, and the addition of special hardware for on-chip testing [5]–[9]. Most practical testability methodologies exploit some combination of the techniques just mentioned.

Manuscript received March 8, 1983; revised November 23, 1983.

F. Somenzi is with SGS ATES Componenti Elettronici S.P.A., 20041 Agrate Brianza, Italy.
S. Gai is with the Centro Elaborazione Numerale dei Segnali, 10129 Torino, Italy.
M. Mezzalama and P. Prinetto are with the Dipartimento di Automatica e Informatica, Politecnico di Torino, 10129 Torino, Italy.

Once again, partitioning plays a key role since it allows one to both split the problem into subtasks which are much easier to manage and to tailor the test generation approach to the different modules. Among these, the best suited to special algorithms are array modules (e.g., ROM's, RAM's, and PLA's) [10], [11].

The best-known methods to partition a network are those based on degating logic and/or tristate buses and the one proposed in [12] and based on LSSD [6]. These techniques can be applied to macro based designs, but both have some drawbacks:

- the use of degating logic and/or tristate buses tends to increase the number of accessible points and the placement of the extra logic has to be done manually by the designer;
- the technique discussed in [12] fails to deal with cascaded combinatorial modules (the network can only be cut in one direction).

This correspondence discusses a circuit structure and its related testing strategies; these are intended to overcome the drawbacks but at the same time meet the basic requirements of hierarchical organization and high testability.

II. Preliminaries

A. Circuit Model

We assume we are dealing with synchronous systems designed in the well-known top-down fashion, whose building blocks for the combinational modules (hereinafter referred to as macros) are acyclic networks whose Boolean equations are known. The use of the word macro instead of building block is to stress that each block is intended to perform a well-defined logical function (e.g., adding, multiplexing) as a natural result of functional partitioning.

We also suppose a scan path [6] is provided for the state register.

Taking these assumptions into account, we can represent the circuit according to Fig. 1 where, for the sake of clarity, state registers have been split into input and output parts since they may be considered as primary inputs and outputs with respect to the test.

Besides the adoption of a scan path, we assume that testability is pursued by means of an additional constraint to make the application of locally[1] generated tests easier.

This consists of forbidding "pseudocyclic" structures like the one in Fig. 2 where some of the outputs are fed back to the inputs (note that this does not prevent the network inside the dashed line from being combinational). Such a structure must be avoided since tests for each macro are generated locally, assuming both complete observability of local outputs and complete controllability of local inputs. This does not hold if pseudofeedback is allowed since inputs are no longer independent. A test is applicable only if the values assumed by fed back outputs are equal to those required on corresponding inputs. The designer can rearrange the description of his circuit by considering the structure inside the dashed line as a macro, provided that algorithms are available to accomplish the test generation task. This is not seen to be a problem for random logic macros, whereas standard ATPG techniques for PLA's [11] are no longer applicable.

B. Test Generation

As far as fault model is concerned, the following is assumed:

- only one macro at a time is faulty;
- the fault model inside each macro is the one supported by the specific test generation algorithm (e.g., for PLA's, the crosspoint defect), possibly augmented so as to include stuck-at faults on local inputs and outputs.

As a consequence, stuck-at faults on interconnections between

[1] In the following we shall use the adjective "local" when speaking about a single macro, regardless of the circuit it is embedded in, whereas "global" will refer to objects and activities at the topmost level.

0018-9340/85/0100-0085$01.00