From: "Mang Erica" <emang@keysys.ro>
To: <AESround2@nist.gov>
Subject: RC6
Date: Thu, 11 May 2000 15:26:13 +0300
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook IMO, Build 9.0.2416 (9.0.2910.0)
Importance: Normal
X-MimeOLE: Produced By Microsoft MimeOLE V5.00.2314.1300

My name is Erica Mang and I am assistant professor at the University of Oradea, Romania.
I write you now to inform you that I implemented the RC6 cipher in a circuit named CRIPTOR. The circuit has off line self-test facilities. In the two files I attached this message I made an analysis of suitability for pseudorandom BIST for the RC6 Cipher, and present than shortly the hardware implementation of the cipher. The circuit was implemented first time last year in my laboratory, in Verilog and now in VHDL. For this implementation I used Xilinx Foundation Series 1.5i Software and the VIRTEX XCV1000 board family.

Yours sincerely,

Erica Mang

Analysis of Suitability for Pseudorandom BIST of the RC6 Cipher

Erica Mang
Computers Department,
University of Oradea, 5 Armatei Romane Str., 3700, Oradea, Romania
E-mail: emang@keysys.ro,

Abstract

Well-known cipher, like DES, FEAL, IDEA or RC6 gain their security by iterating a cryptographically weak function more times. Data are transformed by reusing the same hardware a wished number of times. The first step of a VLSI cipher implementation is the mapping of the cipher data flow onto the hardware architecture. To prove that each combinational level has by entrance a pseudo-random data, the order of the involved operations are relevant. In addition, the operation type is important and not the number of instances of a certain operation. This observation can lead to a number of potential architectures, which realize the same ordering of operations but differ in silicon area. In this paper, I will show that the Cipher Data Flow Graph can also be used to study the randomization maintenance and the production of randomness during cipher operation.

1. Formal Description

I begin with recalling a number of basic definitions from graph theory.

Definition 1. *A directed graph G is a pair (V,E) such that*

1. *$V=\{v_1,\ldots,v_n\}$ is a finite non-empty set, whose elements are called vertices, and*
2. *E is a non-empty set of pairs of vertices, whose elements are called directed edges. The edge (a,b) has direction from a to b.*

Definition 2*. Let (V,E) be a directed graph.*

1. *For an edge (v_i,v_j) in E, v_i is called predecessor of v_j, and v_j is called successor of v_i.*
2. *For $v \hat{I} V$, the set $\cdot v := \{v_i \mid (v_i,v) \hat{I} E\}$ is called the predecessor set of v.*
3. *For $v \hat{I} V$, the set $v\cdot := \{v_i \mid (v,v_j) \hat{I} E\}$ is called the successor set of v.*
4. *The vertex v is called isolated if $\cdot v = v\cdot = \{\}$, i.e. if no edge ends in v and no edge starts from v.*

Definition 3. *An m-tuple $(v_1,\ldots,v_m)$ of vertices is called a path from v_1 to v_m in the directed graph (V,E) if $\{(v_1, v_2),\ldots,(v_{m-1},v_m)\}$ is a subset of E.*

I define now a special graph well suited to describe the data flow of a cipher.

Definition 4. *A Cipher Data Flow Graph, CDFG, is a directed graph (V,E) with no isolated vertices such that is at least one vertex with no predecessor, at least one vertex with no successor, and at least one vertex having both predecessors and successors.*

2. CDFG and ADFG for the RC6 Cipher

We call the vertices with no predecessors the input vertices, those with no successors the output vertices, and the remaining vertices the inner vertices.

Having now the basic concepts, I will construct a CDFG from the data flow graph of RC6 [RRSY-98]. The inputs, both key and plain text, specify the input vertices of the CDFG, and the cipher text specifies an inner

vertex. Each of the inner vertices has at least one predecessor and one successor, but this predecessor and successor vertices need to be inner vertices. An edge (v,w) between two vertices v and w indicates that the output of the operation denoted by vertex v is an input for the operation denoted by w.

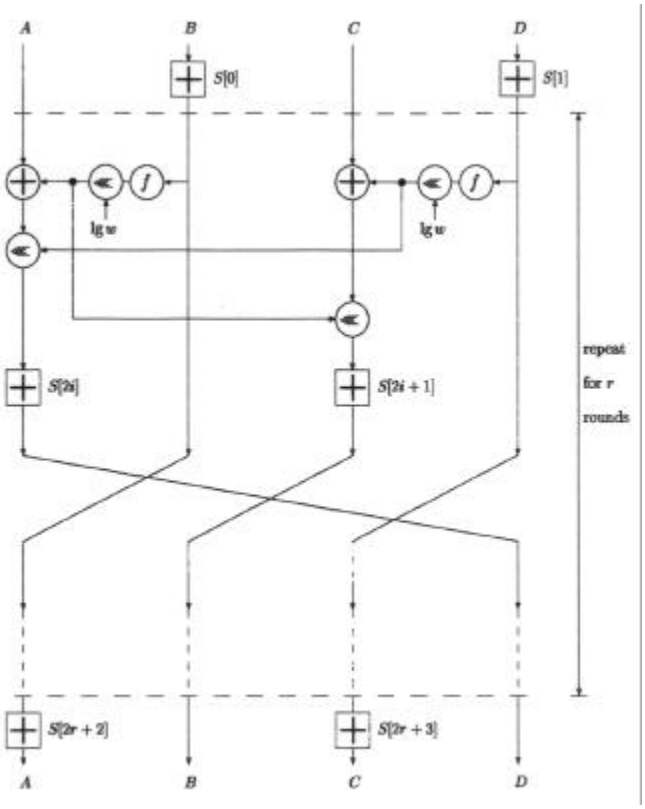

Figure 1. Data flow graph for the RC6 cipher.

In principle, a hardware implementation of a cipher could be done by implementing hardware sub blocks for each operation required, putting the required number of instances into silicon and connecting theses blocks according to the edges in the cipher's CDFG. Obviously, each type of operation requires a fixed number of inputs. Constraints with regard to silicon area preclude a direct mapping of most ciphers data flow graphs into hardware architecture. To find a convenient hardware architecture well suited to perform the data transformation as prescribed by the cipher algorithm, but with less hardware, is the task of the designer. We can describe a potential hardware architecture by a slightly modified CDFG. This special CDFG is called Architecture Data Flow Graph, ADFG, and can be mapped into silicon basically in the same way as the CDFG of the cipher's complete data flow graph. The smaller hardware architecture is able to perform the data transformation as prescribed by the cipher algorithm if:

1. the n types of inner vertices are the same in the CDFG and the ADFG and
2. every path in the CDFG from an input to an output is also a path in the ADFG from the same input to the same output with intermediate inner vertices of the same type.

Reusing of the same hardware block for computing several rounds requires additional feedback path from the outputs of a number of computational sub blocks to some inputs to computational sub blocks [CuBo93].

Definition 5. *An Architecture Data Flow Graph, ADFG, is a CDFG with two disjoint sets S and W of inner vertices such that*

1. *each s in S has at least one input and/or operation vertex as predecessor and exactly one operation vertex as successor;*
2. *each w in W has selection vertices as predecessors and selection and output vertices as successors; all operation vertices of the same operation type W_i have the same number of predecessor selection vertices.*

Note that no selection vertex is followed by another operation vertex. Then, a CDFG can be made an ADFG by introducing a selection vertex between all edges (v,w) with v being an input or inner vertex and w being an inner vertex.

Figure 2 shows the ADFG of an architecture implementing one round of the RC6. Selection vertices are drawn as black bullets.

The set of input vertices in the CDFG of RC6 is {A,B,C,D,S[0],S[1], …,S[2r+3]}and the set of output vertices is {A,B,C,D}. Important proprieties of the cipher are that all operations are group operations and that no operation is succeeded by an operation of the same type. This manifests in the predecessor list of RC6: the entry for the currently

considered operation type never appears as its own predecessor.

In general many inner vertices in a CDFG may be of the same computational type. If there are n different types of operations, we will let W_i denote the set of inner vertices of type i. An architecture is able to perform the data transformation as prescribed by the cipher algorithm if each sequence of operations in the CDFG is a sequence of operation types in the ADFG. Therefore, we consider the sequence of operation types in CDFGs. This sequence is collected in a list of predecessors of operation types. For each vertex in the graph, an entry is made in a predecessor list containing the operation type of the current vertex and type of its predecessors list containing the operation type of the current vertex and the type of its predecessors from left to right, either being operation vertices or input vertices. Identical entries in the list are omitted.

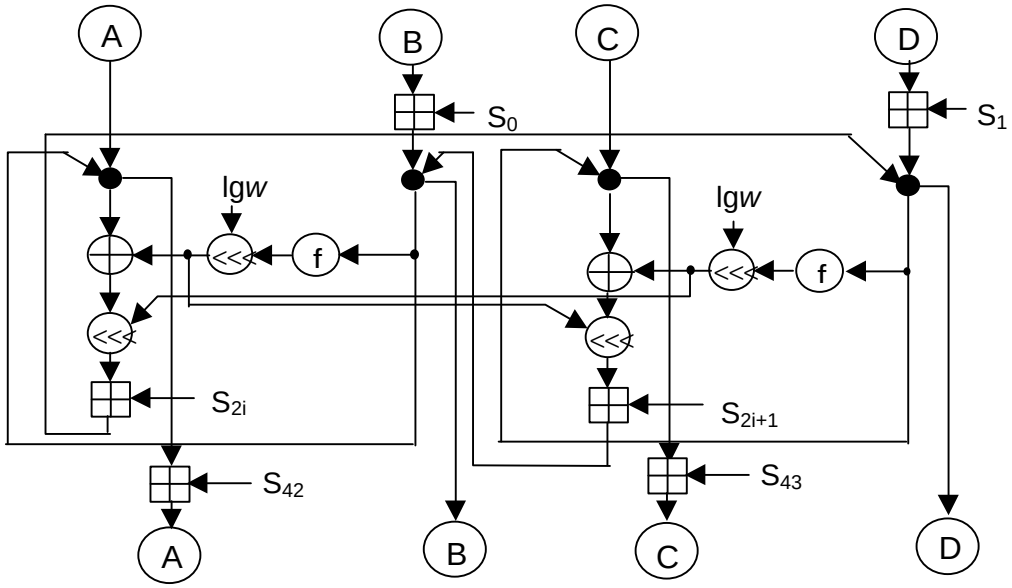

Figure 2. An ADFG of an architecture for the RC6 implementing one round in silicon

The extraction of an ADFG's predecessor list can be done following the next algorithm:

1. *Starting vertices are all ADFG output vertices; choose one of them that have not yet been visited and select its preceding operation vertices. This is the entry in the predecessor list's first column.*
2. *Select the leftmost, not yet visited predecessors of each preceding selection vertex. Make an entry of their operation types. If an identical entry is already present, omit the whole entry. Repeat this step until all predecessors of the preceding selection vertices select the rightmost predecessor operation type of those selection vertices with less predecessors. Mark the current vertex as visited. Select the leftmost predecessor of the preceding selection vertex that is not an input vertex and has not yet been visited. Choose it as the current vertex and continue with step 2. If all predecessors of all predecessor selection vertices have been visited, return to the successor of the current vertex and to step 2.*
3. *Upon returning to the initially chosen vertex, return to step 1 until all output vertices have been visited.*

Each vertex is visited once. The ADFG is finite and therefore the algorithm terminates after visiting all vertices because a visited operation vertex is marked, and all vertices are reachable from an output vertex.

I present the predecessors list for the RC6 algorithm in the figure 3.

3. Random Pattern Propagation Capabilities of RC6 Cipher

A cipher is defined to have a perfect secrecy when the cipher text is statistically independent of the plaintext.

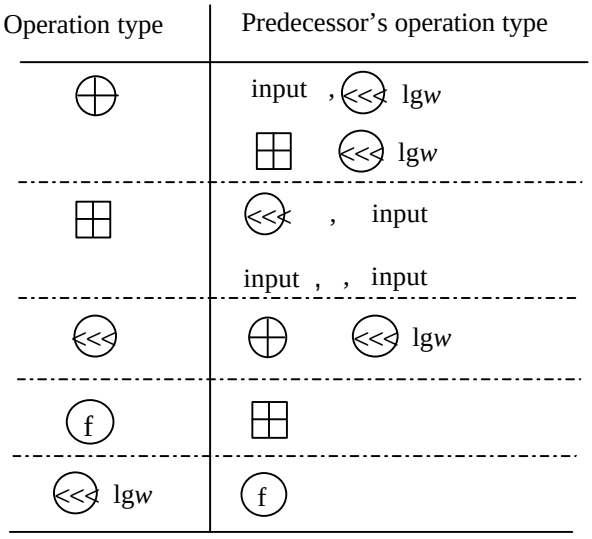

Figure 3. Predecessor list of the operation vertices of the RC6 cipher

Shannon demonstrated the perfect secrecy of the Vernam cipher given in the following theorem:

Theorem 1. *Suppose the three random variables X,Y, and Z take the values in a finite group, whose operation we denote by *****, and Y=X*Z. If X and Z are statistically independent and Z is equally likely to be any group element, then X and Y are statistically independent.*

Proof: Z is uniformly chosen, so that its statistic $P_Z(z)=1/\#group\ elements$, with z being an element of the underlying group. Because of $*$ being a group operation, $Y=X*Z$ holds if $Z=X'*Y$ and vice-versa, where X' denotes the inverse of X. With the probability $P_{Y|X}(y|x)$ being the cipher text statistics of a cipher text y, regarding the plaintext statistics of a plaintext x, the following equations hold:

$$P_{Y|X}(y|x) \quad (1)$$

$$= P_{X'*Y|X}(x'*y|x) \quad (2)$$

$$= P_{Z|X}(x'*y|x) \quad (3)$$

$$= P_Z(x'*y) \quad (4)$$

$$= \frac{1}{\#\,group\ elements} \quad (5)$$

Equation 2 is obtained by "left-associating" X' to Y. Replacing $X'*Y$ by Z, we obtain equation 3. Then, we exploit the statistical independence of X and Z (equation 4). The uniform distribution of Z yields equation 5. Hence, we have proven that the cipher text statistics regarding the plaintext statistics is uniformly distributed in the group and thus that Y is statistically independent from X [Mas93][Kon81].

This theorem can be expanded for the sequence of several group operations:

Corollary 1. *Suppose that ***** and # are group operations on the same finite set and suppose that A, B, and C are statistically independent; the random variable A is equally likely to be any group element. Then, for E=(A*B)#D, B and E are statistically independent and D and E are statistically independent.*

Proof: Because A is equally likely to be any group element and is independent of B, $(A*B)$ is equally likely to be any group element. Because A, B and D are statistically independent, $(A*B)$ and D are statistically independent. Hence, by theorem 1 D and $E=(A*B)\#D$ are statistically independent.

Now let consider the cipher RC6. It is constructed from group operations. Let consider that the inputs of the cipher are randomly chosen and pair wise independent. Than it can be made the following statements about RC6's group operations:

Theorem 2*. If X and Y are two random and pair wise independent variables and Z=X+Y is the output of an adder in the RC6 cipher, than the inputs and outputs of each adder are statistically independent.*

Proof: Knowing that all operations are group operations and none of them is succeeded by an operation of the same type, it can be seen in the predecessors list, there are two cases to be considered:

a. If the operands are input vertices, independence follows directly from theorem 1 and the quality of the input vertices. Because the "+" operation is a group operation, than Z=X+Y supposes that Y=X'+Y and vice-versa, where X' denotes the inverse of X. The probability $P_{z|x}(z \mid x)$ being the cipher text statistics of a cipher text y regarding the plaintext statistics of a plaintext x is:

$$P_{z|x}(z \mid x) = P_{y|x}(x'+z \mid x) =$$

$$P_y(x'+z) = \frac{1}{\#\, group\ elements}$$

Therefore, it is proven that the cipher text statistics regarding the plaintext statistics is uniformly distributed in the group and thus the output Z is independent from the input X, respectively Y.

b. If the inputs are: one is the result of a rotation and the other one is an input node, following the same way it can be proven that the inputs and outputs of each adder are statistically independent.

Theorem 3*. If X is a random and independent variable and Z=X(2X+1) is the output of a circuitry that implements the quadratic function of the RC6 cipher, than the inputs and outputs of this circuitry are statistically independent.*

Proof: Knowing that all operations in the RC6 cipher are group operations, and none of these operations is succeeded by an operation of the same type, (see the predecessors list), it can be seen that the predecessor of the f function node is the adder. As demonstrated in theorem 2, it's inputs and outputs are statistically independent. On the other hand, the cipher text statistics regarding the plaintext statistics is:

$$P_{z|x}(z \mid x) = \frac{1}{\#\, group\ elements}$$

Using the statistically independence of X and Z and regarding the uniform distribution of Z, it can be said that the cipher text statistics regarding the plaintext statistics is uniformly distributed in the group and thus the output Z is independent from the input X.

Theorem 4*. If X and Y are two random and pair wise independent variables and Z=XÅY is the output of an XOR in the RC6 cipher, than the inputs and outputs of each XOR are statistically independent.*

Proof: Following the CDFG and predecessors list of RC6, it can be observed that the predecessors of the XOR operation are either inputs that are considered to be pair wise independent, or rotation registers. For all this operations it was proven the statistically independence propriety. If Z=X⊕Y=XY'+X'Y and knowing that the XOR operation is a group operation, and because X can be any of the group elements with the same probability and X is statistically independent from Y', than XY' can also be any of the group elements with the same probability. Because X, Y' and X' are independent, than XY' and X'Y respectively are independent. Applying the theorem 1, we can say that the output Z is independent from the input X, respectively Y.

Theorem 5*. If X is a random and independent variable and Z is the result of a variable rotation or a rotation with (lgw) positions, than the inputs and outputs of the rotation register are statistically independent.*

Proof: Observing the CDFG and predecessors list of the RC6, cipher it can be observed that the predecessors of the RC6 rotations – with (lgw) positions or variable rotations – are either outputs of an adder and a rotation or a quadratic function. As it has been proved in theorem 1 and Corollary 1, we can again say that the cipher text statistics regarding the plaintext statistics is uniformly distributed in the group and thus the output Z is independent from the input X.

Note that the random data property of RC6 has been proven based upon the predecessor list. Note also that if we make a distinction between plaintext and key inputs in RC6's CDFG and ADFGs, random selection remains necessary for the keys whereas the plaintext needs only be chosen independently from the keys, but not necessarily random.

It was shown that on each node of RC6's data flow graph random data appear if random sub keys are used for encryption. This random data propriety permits the implementation of an efficient built-in self-test scheme to test the complete data path.

The security standards recommend cryptographic equipment to be concurrently testable. Usually, concurrent checking capabilities are achieved by introduction of some kind of redundancy. There are several types of redundancy to achieve self-checking proprieties: hardware redundancy, that usually means duplication of hardware and code redundancy, that means application of error detecting codes [Bon92].

It is a basic design goal of cipher algorithms to make them well suited for both hardware and software implementations. One of the most important issues is the similarity of encryption and decryption that allows utilization of the same hardware for the both data transformation directions. The basic underlying concept is the use of involution ciphers.

Definition 6. *A function* $In(\chi,\chi)$ *from* $\{0,1\}^m x \{0,1\}^k \mapsto \{0,1\}^m$ *is called an involution cipher if for every* $z \hat{I} \{0,1\}^k$, $In(In(x,z),z)=x$ *for all* $x \hat{I} \{0,1\}^m$.

Thus, an arbitrary input block, say x, applied to an involution cipher In, yields an output block y of the same length for a certain key z. If y is applied to In for identical z, then x will come out: In(x,z)=y and In(y,z)=x. Figure 4 shows the structure that makes the according substructure of RC6 an involution cipher. The function f(..) can be chosen arbitrary and does not at all influence the involution cipher property.

The structure of involution ciphers can be utilized for a concurrent self-test in a number of situations. The involution property creates invariants: identical values are produced regardless of the data and key applied. If not only one, but at least two, rounds have been hard-wired on a single chip than two identical involution functions are present. Whenever one of them is idle during the data transformation process, it can be provided with the same key and output of the other involution structure. Its result must be identical to the working structure's input.

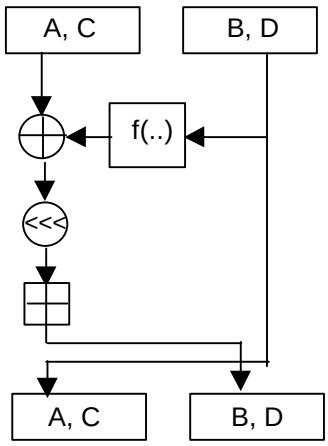

Figure 4. Involution cipher of RC6

Suppose now that there is only one involution structure. Whenever it is idle during the data transformation process, one of its results can be fed back to the inputs. The result must match the original data input. Note that this test does not check for static faults of the function f(..) inside the XOR structure. Remember that f(..) can be chosen arbitrarily. Nevertheless, the check is well suited for the test of the XOR structure itself. Whenever f(..) is implemented with a number stages, the intermediate stages needed for implementation of the involution cipher are tested as well.

References

[Bon92] H. Bonnenberg. Secure and self-testing implementation of the IPES cipher. Presentation at

the Eighth European Workshop on Design for Testability, Bruges, Belgium, June 2-4, 1992.

[CuBo93] A. Curiger and H. Bonnenberg. VINCI: VLSI-based high-speed encryption using the new cipher IDEA. Technical Report 05/93, Integrated System Laboratory, ETH Zurich, April 1993.

[Kon81] Alain G. Konheim. Cryptography: A Primer. Wiley Interscience Publication. John Wiley&Sons, Inc. New York, Chichester, Brisbane, Toronto, 1981.

[Mas93] J.L. Massey. A proposal for a new block encryption standard. In Proc. EuroCrypt'90, Aarhus, Denmark, May 1990.

[RRSY98] R. Rivest, M.J.B. Robshaw, R. Sidney, Y.L..Yin, The RC6 Block Cipher, M.I.T. Laboratory for Computer Science, RSA Laboratories, 1998.