Grey-Box Implementation of Block Ciphers Preserving the Confidentiality of their Design[*]

Vincent Carlier, Hervé Chabanne and Emmanuelle Dottax

SAGEM SA, avenue du Gros Chêne, 95610 Éragny-sur-Oise, France
{vincent.carlier, herve.chabanne, emmanuelle.dottax}@sagem.com

Abstract. In 1997, Patarin and Goubin introduce new asymmetric cryptosystems based on the difficulty of recovering two systems of multivariate polynomials from their composition. We make a different use of this difficult algorithmic problem to obtain a way of representing block ciphers concealing their design but leaving them executable. We show how to implement our solution giving a compact representation with Binary Decision Diagrams.

Keywords: design confidentiality, BDDs.

1 Introduction

Protection of the design of ciphering algorithms is not a new problem. The situation is a bit paradoxical, as usually confidentiality is addressed by encryption. At one hand, algorithms can be enclosed into a tamper resistant circuit. A famous example is given by the Skipjack algorithm history. Its specifications were once classified and it had to stay into devices such as the Capstone or Clipper chips, known to implement specific protections against reverse engineering [Roe95]. We talk in this case of *black-box* environment, where confidentiality of design relies on physical resistance. At the other hand, the new concept of *white-box* cryptography emerges [CEJvO03a, CEJvO03b, LN04]. Here, the whole source code is supposed to be known to the attacker, and the security is provided by logical ways. Note, however, that in these articles, the sole keys used by the algorithm are to be hidden, the design of the algorithm being known. We place ourselves in-between and call

[*]This work has been presented at the BFCA'05 workshop, Rouen, France, March 2005.

our solution *grey-box* cryptography. We propose a solution using well-known tamper response techniques where volatile memories are zeroized whenever an intrusion is detected, and we accept that some information is recovered by an intruder. This hypothesis is confirmed by experiments [Sko02] and seems quite reasonable to assume. We then find ourselves with an instance related to a well-known algorithmic problem, introduced for cryptographic purposes by Goubin and Patarin in the *two rounds schemes* with partial revelation of the polynomials, noted $2\mathbf{R}^-$ schemes [GP97]. In this, we follow works of Sander *et al.* [ST98, LS97] where the Quadratic Residue Hypothesis [GM84] is used to hide polynomials and subsequently programs, and more recently of Billet and Gilbert [BG03] who utilize the Isomorphism of Polynomials problem [Pat96, PGC98, CKPS00] to implement a concealed block cipher with a traceability property.

The remainder of this paper is organized as follows. Section 2 explains the setting of our solution. Section 3 goes further in details giving some concrete examples and explaining the method used. Section 4 concludes.

2 A New Way to Implement a Block Cipher Protecting the Confidentiality of its Design

2.1 $2\mathbf{R}^-$ Schemes

Goubin and Patarin introduce in [GP97] new asymmetric cryptosystems based on the idea of hiding one or two rounds of small S-box computations with secret functions of degree one or two. The public key is given by multivariate polynomials of small degree. In the following we recall the so-called *two-rounds schemes*, designed to be more secure than one-round schemes.

Let $\mathbf{K}$ be a finite field with $q = p^m$ elements. Plaintexts and ciphertexts are elements of $\mathbf{K}^n$. The secret key consists of three affine bijections $r, s, t : \mathbf{K}^n \to \mathbf{K}^n$, and two applications f, g, each given by n quadratic equations over $\mathbf{K}$. The public key consists of the polynomials $P_1, \ldots, P_n$ of degree 4 in n variables that describe the composed mapping $H = t \circ g \circ s \circ f \circ r$. When all these polynomials are given, the scheme is called a $2\mathbf{R}$ scheme. When only some of them are given, the scheme is called a $2\mathbf{R}^-$ scheme. The public-key side computation is just an application of the mapping H. For the secret-key computations, we need to invert the functions f and g. The authors propose to choose them among the following classes of functions:

- $C^\star$-functions: monomials over an extension of degree n over $\mathbf{K}$;

– triangular functions:

$$(a_1, \dots, a_n) \mapsto (a_1, a_2 + q_1(a_1), \dots, a_n + q_{n-1}(a_1, \dots, a_{n-1}))$$

where the q_i are quadratic;

– S-boxes functions, which map $(a_1, \dots, a_n) \in \mathbf{K}^n$ to:

$$(S_1(a_1, \dots, a_{n_1}), S_2(a_{n_1+1}, \dots, a_{n_1+n_2}), \dots, S_d(a_{n_1+n_2+\dots+n_{d-1}+1}, \dots, a_{n_1+\dots+n_d}))$$

where $n = \sum n_i$ and the $S_i : \mathbf{K}^{n_i} \to \mathbf{K}^{n_i}$ are quadratic;

– Combinations of S-boxes with triangular functions;

– $D^{\star\star}$-functions: squaring in an extension of $\mathbf{K}$ of degree n.

These schemes are based on the difficulty of decomposing compositions of multivariate polynomials, *i.e.* given $h = f \circ g$, recover f and g. Note that if we drop t and g in above description, we get the one-round schemes, and they have all been shown to be insecure [GP97]. The two-rounds schemes have also been shown to be insecure when g lies in the first two classes [GP97]. The variant that we are interested in is $2\mathbf{R}$ with S-boxes, where both f and g are S-boxes functions.

So far, there exist two different attacks against $2\mathbf{R}$ with S-boxes. In [Bih00], Biham succeeds in cryptanalysing the scheme. Note that this attack is not based on functionnal decomposition. Another attack has been published [DFKYZD99], based on the algebraic structure of the scheme and with the intention of decomposing the composition. However, the attack imposes restrictions on the scheme:

1. the field $\mathbf{K}$ should have more than 4 elements;

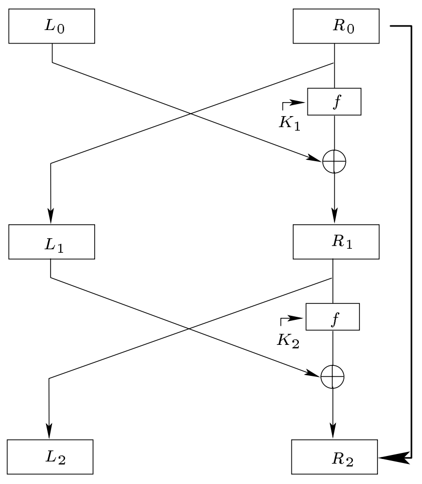
2. the attack would not work if the S-box functions are not quadratic.

Note as well that the $2\mathbf{R}^-$ schemes, *i.e.* when some of the polynomials describing the composition are kept secret, are not subject to these atttacks.

2.2 Our Idea and a Way to Implement It

Our idea is to use the same problem as in $2\mathbf{R}^-$ schemes for protecting the confidentiality of the design of block ciphers.

A block cipher is usually composed of several rounds, and a round itself is composed of different operations. The description of these operations constitutes the design of the algorithm: they have been chosen by the designer and they are an evidence of its know-how. Our method aims at keeping these design secret by composing rounds. For a given cipher acting on n-bit blocks, let $x_1, \ldots, x_n$ be the boolean input variables and $y_1, \ldots, y_n$ the output bits after the first round. Each y_i can be expressed as a boolean function $p_{1,i}$ in the variables $x_1, \ldots, x_n$ and obtained by combination of its component functions, including an S-box function. We compute as well the boolean functions $(p_{2,i})_{1 \le i \le n}$ corresponding to the second round of the cipher. Let $(q_i)_{1 \le i \le n}$ be the boolean functions that implement these two rounds (an example with DES is shown on Fig. 1). This system of boolean functions allows us to describe the two rounds of the cipher in an executable way, but without revealing information about the design of the algorithm.



$$\begin{cases} R_2(1) = q_1(L_0(1), \ldots, L_0(32), R_0(1), \ldots, R_0(32)) \\ R_2(2) = q_2(L_0(1), \ldots, L_0(32), R_0(1), \ldots, R_0(32)) \\ \vdots \\ R_2(32) = q_{32}(L_0(1), \ldots, L_0(32), R_0(1), \ldots, R_0(32)) \end{cases}$$

Figure 1: Two rounds of DES.

We further subtract some equations from attackers analysis by a physical mean. The design of the algorithm is stored in a volatile memory which is zeroized when an intrusion is detected. Such techniques known as *tamper response* can be implemented following various ways [Wei00]. The simplest one is a quick drop of the power line of the memory (see Fig. 2). Due to data remanence phenomena [Gut01] and external conditions [Sko02], it is hard to exactly predict how many equations will be erased. Now the confidentiality of the design is based on the same problem as the $2\mathbf{R}^-$ scheme.

There is one point that we have not tackled yet but that is worthy of attention: the treatment of the secret key. As the key is usually diversified into several subkeys (one for each round), we have several possibilities to implement the block cipher, among them:

1. The boolean functions have additionnal variables k_i so that for each round, the corresponding subkey can be input, the key schedule being performed separately. At one hand, this allows flexibility in key injection, and has the advantage that each round can be represented by the same boolean functions, so we can implement the whole cipher with only one composition of two rounds. On the other hand, it adds a lot of variables to the boolean functions we have to compute.

2. The secret key is integrated into the boolean functions, *i.e.* we perform the key schedule before the implementation of the cipher and we compute the boolean functions with only the text variables as input. This has the drawback that every round will have a different expression, so we have to compute and implement every composition of 2 rounds separately.

3. If the cipher permits it, we can envisage an intermediate solution. When the block cipher has a very simple key-schedule, it is possible to integrate the main key and the key-schedule into the boolean functions. We can think for instance about the block cipher 3-Way [DGV94] of which key-schedule is reduced to bitwise XOR-ing a short round constant to the main key. This allows us to implement all the cipher with only one composition of two rounds, but without adding too many variables to the boolean functions.

These different solutions may lead to different levels of security.

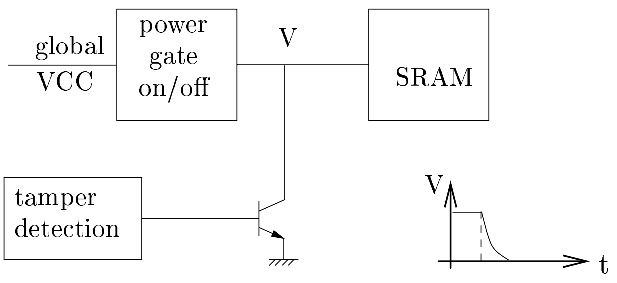


Figure 2: Physical implementation of the "$-$" of $2\mathbf{R}^-$.

3 An example

3.1 BDDs

We choose binary decision diagrams (BDDs) for representing boolean circuits. They were introduced in 1986 by Bryant [Bry86] and are known to give a compact representation of logical functions. Some operations are defined on BDDs. For instance, we can use their composition for step by step computing the BDDs standing for one or many rounds. As well, the algorithm for evaluating BDDs can be considered as a trivial way to implement our solution with a network of multiplexers.

BDDs are data structures used to represent boolean functions. Here we shortly present their properties, the interested reader is referred to [Bry86, Bry92, And97].

Let f be a boolean function of n variables. If $f|_{x_i=b}$ denotes the function resulting when the i-th variable is replaced by the constant b, the *Shannon expansion* of the function f around variable x_i is given by:

$$f = x_i \cdot f|_{x_i=1} + \overline{x_i} \cdot f|_{x_i=0}$$

This simple relation is used to represent boolean functions as particular graphs in an if-then-else notation.

Definition 1 *A* binary decision diagram (BDD) *is a rooted, directed acyclic graph with two types of nodes. A* non-terminal node *N is labelled $i \in \{0, \dots, n\}$ and has two children noted $low(N)$ and $high(N)$. A* terminal node *is labelled 0 or 1 and has no child.*

A graph having root node labelled i denotes the function f_i

$$f_i(x_1, \dots, x_n) = x_i \cdot f_{high(N)}(x_1, \dots, x_n) + \overline{x_i} \cdot f_{low(N)}(x_1, \dots, x_n)$$

The set of values $\{x_1, \dots, x_n\}$ describes a path in the graph starting from the root : at each node labelled i, we follow the high child if $x_i = 1$ ("`THEN`") and the low child otherwise ("`ELSE`").

Definition 2 *A BDD is* ordered (OBDD) *if, on all paths through the graph, the labels respect a given order. An OBDD is* reduced (ROBDD) *if the following conditions are satified:*

uniqueness: *two nodes having the same label and children are equal;*

non-redondant tests: *there are no node with both children leading to the same node.*

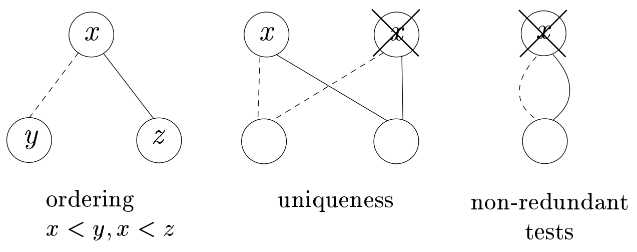


Figure 3: The conditions of ROBDDs.

These three conditions for constructing an ROBDD are illustrated on Fig. 3. Figure 4 shows an example on the function $f(x, y, z) = x \cdot y + z$.

Note that ROBBDs depend only on the order of the variables, so they are canonical representations of functions. In other words, for a given variables order, any way of computing an ROBBD leads to the same result. There

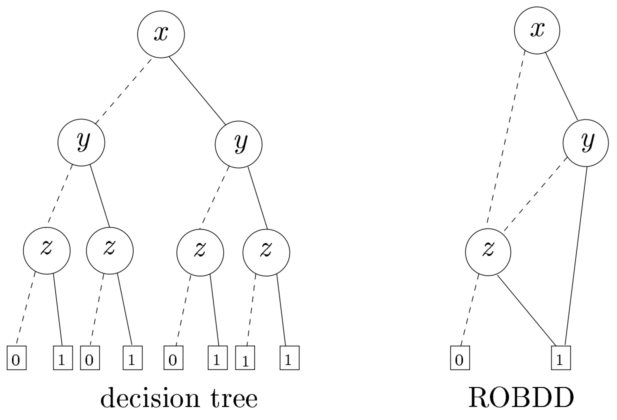


Figure 4: Representations of the function $f(x, y, z) = x \cdot y + z$.

exist various types of BDDs. Here we use *signed* BDDs, where a tag is added on each link for if we have to complement the result. This leads to more compact representations, as a function and its complementary can be represented by the same BDDs.

3.2 Grouping together two rounds of DES

To fix ideas, we here give some figures (see Tab. 1) on the number of nodes needed to represent the right part of the composition of the first two rounds of DES with signed ROBDDs. What we exactly compute is illustrated by Fig. 1. Each one of the 32 bits here stands as a logical function of 34 variables. Note that when represented by polynomials, each of these logical functions has degree 25 and more than 150000 terms. We used the BDD library [Yun] to compute the composition of two rounds. The following table gives the exact complexity of the resulting BDDs, which varies from 6667 to 34947 nodes (13750 on average), R(i, j) standing for the j-th bit of the right block after i rounds of DES. Note that in this experiment the key is fixed to a random value. As for the variable ordering, it has a great influence on the size of the BDDs, for instance the size of the BDD for one bit can be more than 1.5 millions of nodes with some orders. The figures given here were obtained with an order that gives acceptable size for all BDDs. However, we think that the complexity can be further reduced, for instance by using a specific order for each output bit. Further research is needed to explicit the relation between the input variable ordering and the size of the resulting BDD.

bit	#nodes	bit	#nodes	bit	#nodes	bit	#nodes
R(2,1)	13448	R(2, 9)	16536	R(2,17)	17256	R(2,25)	9402
R(2,2)	30741	R(2,10)	13564	R(2,18)	34947	R(2,26)	13449
R(2,3)	9322	R(2,11)	6667	R(2,19)	7240	R(2,27)	6944
R(2,4)	7095	R(2,12)	7067	R(2,20)	13436	R(2,28)	25947
R(2,5)	6938	R(2,13)	32393	R(2,21)	7002	R(2,29)	6813
R(2,6)	19294	R(2,14)	9285	R(2,22)	7057	R(2,30)	20057
R(2,7)	7057	R(2,15)	6914	R(2,23)	16337	R(2,31)	17070
R(2,8)	9592	R(2,16)	16076	R(2,24)	18064	R(2,32)	7070

Table 1: Complexity of BDDs for the right block of DES after 2 rounds.

3.3 An implementation

In a straightforward implementation of the complete DES, 8 sets of 64 BDDs (one set for each pair of rounds) are placed in a memory and we run through them, according to the value of the 64 input bits. Each node (except the last one) is coded on 40 bits and consists of its variable number, 2 tags for the signs of the "`THEN`" and "`ELSE`" links (we use signed ROBBDs), and the addresses of its "`THEN`" and "`ELSE`" nodes. All the BDDs necessary to represent the 16 DES rounds need approximately 18 Mo of memory to be stored using this representation. We used a RAM memory, which is accessed by an FPGA (Field Programmable Gate Array). The FPGA is programmed to take as input the 64 plaintext bits, and to run through the BDDs in memory according to these values. When the FPGA reaches the leaves of the last set of BDDs, it gets the 64 output bits. The throughput of this implementation is 152 Kbits/s.

As a comparison, the white-box DES implementation of [LN04], which is a software implementation, occupies 4.5 MB and encrypts one block in 30ms.

4 Conclusion

We introduce a new way of implementing cryptographic algorithms preserving their confidentiality. Our technique demands some tamper response in case of intrusion to obtain an instance of a hard algorithmic problem. There is still avenues to improve this. For instance, and as usual with BDDs, variable ordering should have a great importance for size optimization of manipulated graphs [Bry92]. A very simple implementation of this scheme consists in storing BDDs in an external but tamper responsive SRAM, and to add some logic to run through this memory. From another point of view, note that some cryptographic algorithms are more difficult to represent this way as they rely on primitives for which BDDs are not so efficient such as multiplicators or rotators. Finally, we try to place ourselves outside known attacks but we are dealing with special instances where a sparse polynomial is composed with approximately itself. We invite readers to carefully analyze our solution before using it.

Acknowledgements. This work has been supported by the MEDEA+ programme under the CryptoSoC project.

We thank Jean-Francis Michon, Pierre Valarcher and Jean-Baptiste Yunès for profitable discussions and permitting us to use their BDD library.

References

[And97] Henrik Reif Andersen. An Introduction to Binary Decision Diagram. Lecture notes, October 1997.

[BG03] Olivier Billet and Henri Gilbert. A Traceable Block Cipher. In Chi-Sung Lailh, editor, *Proceedings of ASIACRYPT'03*, volume 2894 of *Lecture Notes in Computer Science*, pages 331–346. Springer-Verlag, 2003.

[Bih00] Eli Biham. Cryptanalysis of Patarin's 2-Round Public Key System with S-Boxes (2R). In *Proceedings of EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 408–416. Springer-Verlag, 2000.

[Bry86] Randal E. Bryant. Graph-Based Algorithms for Boolean Functions Manipulation. *IEEE Transactions on Computer*, 8(C-35):677–691, 1986.

[Bry92] Randal E. Bryant. Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams. *ACM Computing Surveys*, 24(3):293–318, 1992.

[CEJvO03a] S. Chow, P. Eisen, H. Johnson, and P.C. van Oorschot. A White-Box DES Implementation for DRM Applications. In *Proceedings of ACM CCS-9 Workshop DRM 2002*, volume 2595 of *Lecture Notes in Computer Science*, pages 1–15. Springer-Verlag, 2003.

[CEJvO03b] S. Chow, P. Eisen, H. Johnson, and P.C. van Oorschot. White-Box Cryptography and an AES Implementation. In *Proccedings of SAC'02*, Lecture Notes in Computer Science, pages 250–270. Springer-Verlag, 2003.

[CKPS00] Nicolas T. Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In Bart Preneel, editor, *Proceedings of Eurocrypt'00*, volume 1807 of *Lecture Notes in Computer Science*, pages 392–407. Springer-Verlag, 2000.

[DFKYZD99] Ye Ding-Feng, Lam Kwok-Yan, and Dai Zong-Duo. Cryptanalysis of "2R" Schemes. In *Proceedings of CRYPTO'99*. Springer-Verlag, 1999.

[DGV94] Joan Daemen, René Govaerts, and Joos Vandewalle. A New Approach Towards Block Cipher Design. In R. Anderson, editor, *Proceedings of FSE'93*, volume 809 of *Lecture Notes in Computer Science*, pages 18–32. Springer-Verlag, 1994.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[GP97] Louis Goubin and Jacques Patarin. Asymmetric Cryptography with S-Boxes. In *Proceedings of 1st International Information and Communications Security Conference*, pages 369–380, 1997.

[Gut01] Peter Gutmann. Data Remanance in Semiconductor Devices. In *10th USENIX Security Symposium*, pages 39–54, 2001.

[LN04] Hamilton E. Link and William D. Neumann. Clarifying Obfuscation: Improving the Security of White-Box Encoding. Cryptology ePrint Archive, Report 2004/025, 2004. `http://eprint.iacr.org/2004/025`.

[LS97] Richard Lipton and Tomas Sander. An additively homomorphic encryption scheme or how to introduce a partial trapdoor in the discrete log, 1997.

[Pat96] Jacques Patarin. Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms. In Ueli M. Maurer, editor, *Proceedings of EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 33–48. Springer-Verlag, 1996.

[PGC98] Jacques Patarin, Louis Goubin, and Nicolas T. Courtois. Improved Algorithms for Isomorphisms of Polynomials. In Kaisa Nyberg, editor, *Proceedings of Eurocrypt'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 184–200. Springer-Verlag, 1998.

[Roe95] Michael Roe. How to Reverse Engineer an EES Device. In Bart Preneel, editor, *Proceedings of HFE'94*, volume 1008 of *Lecture Notes in Computer Science*, pages 305–328. Springer-Verlag, 1995.

[Sko02] Sergei Skorobogatov. Low temperature data remenence in static RAM. Technical report, University of Cambridge Computer Laboratory, 2002.

[ST98] Tomas Sander and Christin F. Tschudin. On Software Protection Via Function Hiding. In *Proceeding of Information Hiding, Second International Workshop*, volume 1525 of *Lecture Notes in Computer Science*, pages 111–123. Springer-Verlag, 1998.

[Wei00] Steve H. Weingart. Physical Security Devices for Computer Subsystems: A Survey of Attacks and Defense. In Çetin Kaya Koç and Christof Paar, editors, *Proceedings of CHES'00*, volume 1965 of *Lecture Notes in Computer Science*, pages 302–317. Springer-Verlag, 2000.

[Yun] Jean-Baptiste Yunès. BDD Library. Available from `http://www.liafa.jussieu.fr/web9/outils/outils_scientifiques_fr.php`.