An Optimized Real Time Generation of S-Box Inverses Using Arithmetic Modulo Powers of Two

Eltayeb Salih Abuelyman, and Mohamed Ahmed El-Affendi
College of Computer and Information Systems
Prince Sultan University, Riyadh 11586, Saudi Arabia

Summary

This paper proposes generation of entries of the S-Box using arithmetic modulo powers of two. The approach saves storage space and makes real time computations of the entries feasible. The inverse function used in the process is based on arithmetic modulo a power of two. The platform proposed for this project is modulo "2^n" arithmetic which builds upon the hypothesis that the set of odd residues of "2^n" forms a mathematical field. A first round optimization cuts the amount of required storage space by half. A second demonstrates that all entries of the S-Box could be derived from any of the remaining rows after the first round is completed; hence only a single row needs to be stored. For real time regeneration of the S-Box, each of the single row entries is unpacked into two digits. One array stores the most significant digits and other stores the least. The goal of this paper is therefore twofold: to enable real time computation of entries of the S-Box and to reduce the amount of stored information. The latter is small enough to reduce vulnerability and large enough to form a basis that enables real time generation of the complete S-Box.

Keywords: *Modulo Arithmetic, Mathematical Field, Rijndael, and S-Box.*

1. Introduction

In general, most of the low-level components of block ciphers emphasize the functional aspect of the likes of the S-Box. According to Vincent Rijmen, the co-designer of the Rijndael algorithm, the Advanced Encryption Standard (AES), which is based on the S-Box is still superior in security [1]. In fact, their algorithm enjoys extremely high level of confusion and diffusion. Furthermore, the resistance of the inverse function in GF(2^n) to linear, differential and higher-order differential attacks is exceptional according to Rijmen. However, one of the disadvantages of AES is the simplicity of description in GF (2^n), which is also the field in which the diffusion layer is linear. The AES designers believe that this may create uneasy feelings, but they are not aware of any vulnerability thereof. Should such a vulnerability exist, they suggest the replacement of the GF(2^n) by another field that has similar properties, but is not algebraic over GF(2^n).

Manuscript received December 5, 2007
Manuscript revised December 20, 2007

A potential vulnerability that will be dealt with here is not directly related to the aforementioned discussion. The vulnerability discussed here stems from storing the S-Box in an attempt to avoid the overhead of tedious real time computations. The authors of this paper have shown in a previous paper that arithmetic modulo prime numbers provides a valid, less complex alternative to real time computation of S-Box inverses [18]. One may argue that reducing the complexity relative to the Galois Field may increase vulnerability. Nevertheless, such tradeoff is acceptable considering the gain in the amount of information stored and the consequences thereto. However, more work is needed to confirm that the vulnerability of the resulting platform is less relative to that of the original implementation via Galois Fields.

This research is based on the finding that unique inverses of residues of, for example, 2^n may be obtained by a bijective mapping of these residues to the odd residues of 2^{n+1}. In a nutshell, the process of finding inverses starts with an injective mapping of residues of 2^n to the odd residues of 2^{n+1}. The inverses of the latter are computed using the field of odd numbers modulo 2^{n+1}. The resulting inverses are then mapped back by a bijective function to pseudo inverses of residues of 2^n. The rest of the paper is organized as follows. A brief introduction to the AES is presented in section 2. In section 3, a literature survey is presented. The arithmetic modulo a power of two platform is analyzed in section 4. Section 5 discusses performance optimizations and the conclusion is given in section 6.

2. The Advanced Encryption Standard

In Reference [2] the authors stated that "The Advanced Encryption Standard (AES) committee solicited proposals for an encryption algorithm that would become the first choice for most situations requiring a block cipher". Consequently, several algorithms were submitted and Rijndael was chosen by the American National Institute of Standards and Technology (NIST) [3]. More information about the AES is in the next subsection.

2.1 The Rijndael Algorithm

For Rijndael, the length of both the block to be encrypted and the encryption key are not fixed. They can be independently specified to 128, 192 or 256 bits. The number of rounds, however, varies according to the key length. It can be equal to 10, 12 and 14 when the key length is 128 bits, 192 bits and 256 bits, respectively[4]. The basic components of Rijndael are simple mathematical, logical, and table lookup operations. The latter is actually a composite function of an inversion over Galois Field (GF) with an affine mapping. Such structure makes Rijndael suitable for hardware implementation [2] Nevertheless, both hardware and software implementations have their own drawbacks. Hardware implementation is rigid as a block and key sizes must be held at fixed values. However, the running time is better compared to its software counterpart. All in all, Rijndael is considered to be the fastest algorithm in terms of the critical path between plaintext and cipher-text[2]. This paper proposes the design of a modulo prime-number based AES algorithm. The design will be simulated in a VHDL environment to confirm its superiority. The VHDL modules will not be included in this paper.

3. Literature Survey

Ichikawa, Kasuya, and Mastui published a paper they called "Hardware Evaluation of AES finalists"[5]. The paper evaluates hardware implementations of the AES finalists; Twofish [13], Serpent [14], RC6 [15], Mars [16], and Rijndael[17]. Commenting on Mars, the authors stated two problems: the keyed transformations take a long time and, the algorithm is very complex. They also concluded that RC6 gives a poor performance since the critical path is long. The RC6, according to them, did not satisfy the need for fast encryption. They believe Serpent has the best security but it requires the largest circuit. They also believed that Twofish has quite a long critical path. In their paper titled "Comparison of the Hardware Performance of AES candidates using reconfigurable hardware" Pawel Chodowiec and Kris Gai gave data supporting Rijndael [16]. The throughput of Rijndael came second. However, considering all the other criteria, Rijndael was found to be the best. Ian Harvey discussed the selection of encryption algorithm in practical situations in his paper titled "The Effects of Multiple Algorithms in the Advanced Encryption Standard" [2]. AES finalists are compared based on the factors considered for algorithm selection. Bryan Weeks et al presented an overview of the methods and architectures used for the AES hardware comparison in their paper titled "Hardware Performance Simulations of Round 2 Advanced Encryption Standard Algorithms" [4]. In general, throughput, area and latency are the characteristics considered for design tradeoffs in hardware engineering. The five finalists were examined from the standpoint of minimum area and maximum throughput. Interested readers may consult reference [4] for further details. A. Satoh, S. et al presented an AES hardware implementation they considered to be efficient in their paper "A Compact Rijndael Hardware architecture with S-Box Optimization"[8]. However, the main drawback of their architecture is the critical path time. The SubBytes, MixColumns and AddRoundKey transformations are done for one column within one clock cycle. This increases the critical path time. In the next subsection, a survey of some of the VHDL implementations is presented.

3.1 VHDL implementations

Algotronix AES Core [9] represents the second generation of their AES VHDL technology. It is a stable implementation of the entire algorithm. It offers competitive density and performance on all the main Field Programmable Gate Arrays (FPGA) families from Xilinx, Altera and Actel. It is supplied as synthesizable source code to allow for customer code review in security sensitive applications. The core is highly configurable with many implementation options but unlike most competitive products, this is achieved using VHDL generic parameters and does not require customizing the VHDL code.

In their paper "Configurable Design and Implementation of the Rijndael Algorithm-AES", Arda Yurdakul et al [10] discussed the design and implementation of three configurable and flexible cores of Rijndael. The three cores are; an encryptor, a decryptor and a combined encryptor-decryptor. These cores support not only the AES, but also the whole Rijndael algorithm. Another feature of the cores is that they are all designed using Electronic Code Book (ECB) mode, meaning that every single data block is encrypted and decrypted independently from each other. Since ECB is the basic element of all other main modes such as Cipher Block Chaining (CBC), Cipher Feedback (CFB) and Output Feedback (OFB), it is easy to extend their design and implement the other modes. All the modules in these flexible cores are realized using VHDL language. Some modules are designed by using behavioral style and some are designed using Register Transfer Language. In the next section the implementation of the modulo arithmetic based AES is presented.

4. AES Implementation using modulo arithmetic

Generally speaking for the hardware implementations of Rijndael, Ian Harvey [2] states that the average time for one lookup table is 3.2 nanoseconds for Rijndael (8x8). If one is able to optimize the S-Box lookup process, then the speed of Rijndael can be greatly increased. The S-Box computation is the most time-consuming operation in Rijndael. This is the case because it is required in every round. Current implementations pre-compute the S-Box and store it on a Read Only Memory (ROM). However, there is a chance that in a highly sensitive data environment, storing such information may pose a threat to its security. To circumvent such vulnerability, the S-Box values must be computed in a real-time basis. However, using the Galois Fields (GF) renders this option undesirable. To speed up real-time S-Box construction, an environment other than the GF must be used. The reason real time computation of the S-Box is advantageous is twofold: first, when the lookup table is stored for future reference, it is vulnerable to attacks; hence, it is a security concern. Second, if a device doesn't have enough resources, real-time computation of inverses of numbers for the S-Box in the Galois Field environment becomes a bottleneck. To overcome these limitations, real time computations of these inverses can be performed using modulo arithmetic. The proof that modulo arithmetic approach is efficient and takes significantly less time and space compared to the GF can be found in reference [11]. Henceforth, arithmetic modulo a power of two will be referred to as AM(2^n). In the next subsections we will present analysis of AM(2^n).

4.1 Computing inverses modulo (2^n)

A VHDL module for computing the S-Box values has been simulated successfully. Without loss of generality, the module was used to compute the inverses of residues modulo 2^4. For convenience, the output is rearranged in table 1 below. The table shows an AM(2^4) mapping of numbers less than 2^4.

Number	Inverse Modulo16
1	1
2	-
3	11
4	-
5	13
6	-
7	7
8	-
9	9
10	-
11	3
12	-
13	5
14	-
15	15

Table 1. Inverses Modulo 16 mapping the first 15 integers

Only odd numbers on the first row have inverses modulo 16. The inverses are shown in the second row, which also shows a '-' for numbers without inverses. In this case, all the even numbers do not have inverses as expected. The following is a generalization of the relationship between numbers and their multiplicative inverses modulo a power of 2.

Lemma 1

Given any integer n and any number 'a' that is less than 2^n, if 'a' has a multiplicative inverse modulo 2^n, then both 'a' and its multiplicative inverse must be odd numbers.

Proof

If 'a' and 'b' are multiplicative inverses of each other modulo 2^n, then for some integer z less than 2^n the following equation holds:

$$a * b = (z * 2^n) + 1 \qquad (1)$$

Since the right hand side is an odd number, it follows that 'a' must be an odd number and 'b' must also be an odd number. This proves that there are no multiplicative inverses for even numbers modulo 2^n. The Eueler's Totiet[12] will be used to prove that every odd number has a multiplicative inverse modulo 2^n. In general, mathematicians use Euler's Totiet function 'Φ' to compute the number of integers that are relatively prime (or multiplicative inverses) to a particular integer n. The function, denoted Φ(n), is given by the following product:

$$\Phi(n) = [(p_1 - 1) * p_1^{k1-1}] * [(p_2 - 1) * p_2^{k2-1}] * \ldots * [(p_m - 1) * p_m^{km-1}] \qquad (2)$$

Where n in this case is expressed in terms of its prime factors [$p_1, p_2, \ldots p_k$].

A special case of the Euler's function can be used to find the number of integers that are relatively prime to 2^n. Since a power of 2 has the number 2 as its only prime factor then replacing "n" with "2^n"; "p_1"with the number 2; "k1" with "n" we reach equation 3 .

$$\Phi(2^n) = (2-1)(2^{n-1}) = 2^{n-1} \qquad (3)$$

The formula shows that half of the numbers less than 2^n are relatively prime to it. Since no even number is relatively prime to 2^n and there are exactly 2^{n-1} odd number less that 2^n, it follows that all the odd numbers less than 2^n have multiplicative inverses. This completes the proof.

Lemma 2

For any integer n, if we divide the sequence of odd numbers from 1 to "2^n-1" into two disjoint subsets where the first contains the sequence of all the odd numbers that are less than 2^{n-1} and the second contains the rest in ascending order as well, then we can say that the first and last number in each subset is the multiplicative inverse of itself modulo 2^n.

Proof:

Assume a sequence of odd numbers divided into two subsets as shown:

$[1, 3, .., 2^{n-1} - 1], [2^{n-1} + 1, 2^{n-1} + 3, \dots, 2^n - 1]$

The proof will be divided into four parts for the four boundary conditions:

$\{1, (2^{n-1} - 1), (2^{n-1} + 1)$ and $(2^n - 1)\}$.

a) The proof for 1 as the multiplicative inverse of itself is trivial.

b) The proof for $(2^n - 1)$ as the multiplicative inverse of itself can be given as follows for some integer z that is less than 2^n:

$(2^n - 1) * (2^n - 1) = 2^{2n} - 2^{n+1} + 1 =$
$2^n (2^n - 2) + 1 = 2^n (z) + 1 = 1 \bmod (2^n)$

c) The proof for $(2^{n-1} - 1)$ is as follows:

$(2^{n-1} - 1)*(2^{n-1} - 1) = 2^{2n-2} - 2^n + 1 = 2^n * (2^{n-2} - 1) + 1 = 1 \bmod (2^n)$

d) The proof for $(2^{n-1} + 1)$ is also achievable in a similar way:

$(2^{n-1} + 1)*(2^{n-1} + 1) = 2^{2n-2} + 2^n + 1 = 2^n * (2^{n-2} + 1) + 1 = 1 \bmod (2^n)$

We will also show that for any power of 2, there are only four numbers that are inverses of themselves modulo the power of 2.

Corollary 1

There are exactly four numbers that are the inverses of themselves modulo a power of 2.

Proof

For any number 2^n we proved in Lemma 1 that each of the numbers in the set $S = \{1, (2^{n-1}-1), (2^{n-1}+1), (2^n-1)\}$ equals its inverse modulo (2^n). We need to show that if a number "a" equals its inverse then "a" must be a member of the set S. Let us assume “a” to be the inverse of itself and that “a” is not a member of S, we conclude the following four inequalities:

(i) $\mathbf{a < (2^n - 1)}$
(ii) $\mathbf{a > 1}$
(iii) **a is not equal to** $\mathbf{(2^{n-1} - 1)}$
(iv) **a is not equal to** $\mathbf{(2^{n-1} + 1)}$

We will prove that if “a” satisfies conditions (i) and (ii), then “a” must be equal to $(2^{n-1} - 1)$ or $(2^{n-1} + 1)$, thereby contradicting statements (iii) and (iv) and proving that the value of “a” can only be equal to one of the members of the set S.

Case (i) $\mathbf{a < (2^n - 1)}$

Therefore: $a = (2^n - 1) - 2*r$
for any $r > 0$ (4)
and
$a^2 = (2^n - 1)^2 - 4*r(2^n - 1) + 4*r^2$
$= 2^{2n} - 2^{n+1} + 1 - 4*r*2^n + 4*r + 4*r^2$
$= 2^n(2^n - 2 - 4*r) + 4*r^2 + 4*r + 1$ (5)

Since a^2 is congruent to 1 modulo 2^n, equation (5) implies that:

$(4*r^2 + 4*r) \bmod (2^n) = 0$

Therefore $(4*r^2 + 4*r) = t*2^n$
for $0 < t < 2^n$

Rearranging the terms results in the following equation

$r*\{(r+1)/t\} = 2^{n-2}$ (6)

Equation (6) implies that both "r" and "{(r+1)/t}" are powers of 2. Since the numerator (r+1) must be an odd number, it follows that {(r+1)/t} must be equal to 1. Therefore "r" must be equal to 2^{n-2}. Consequently, using equation (4), "a" will satisfy the following equation:

$a = (2^n - 1) - 2*r$
$= (2^n - 1) - 2*2^{n-2}$
$= (2^{n-1} - 1)$

Case (ii) $\mathbf{a > 1}$

Therefore $a = 1 + 2*r$
for any $r > 0$ (7)
and $a^2 = 1 + 4*r + 4*r^2$

since “a” is congruent to 1 modulo 2^n it follows that:

$(4*r^2 + 4*r) \bmod (2^n) = 0$

and eventually we find the value of “r” to be equal to 2^{n-2} following the same steps of *case (i)*,. Plugging this value in equation (7) will lead to:

$a = 1 + 2*2^{n-2} = (2^{n-1} + 1)$

which completes the proof.
This ends the proof for Corollary 1.

Since mapping of the boundary conditions results in numbers that are inverses of themselves, AM(2^n), and for that matter GF(2^n), may look vulnerable through the S-Box Values. We used corollary 1 to show that regardless of the value of n, the number of integers which are equal to their inverses modulo (2^n) will be exactly four. However, that is not the only concern we have with this approach. AM(2^n) also suffers from another disadvantage, that is, even numbers have no inverses modulo(2^n). The question that may be asked is whether or not modifying the modulo (2^n) arithmetic will lead to a better approach. In the next subsection the proposed “pseudo inverses modulo powers of two” will be introduced.

4.2 Pseudo inverses modulo powers of two

So far the following points have been emphasized:

1. All the odd numbers are relatively prime to powers of two

2. No even number has an inverse modulo a power of two

It is clear from the first point that every odd number has an inverse in modulo a power of two arithmetic, which is a plus. However, the second point limits the use of powers of two because the same conclusion could not be reached for the even numbers. The next move is to show that a mapping of all the integers less than 2^n to the odd numbers less than 2^{n+1} will enable the use of modulo a power of two arithmetic as a valid alternative to computing inverses via the GF approach. The resulting inverses are considered pseudo inverses because they are computed in a different mathematical field. Four functions are needed to compute the pseudo inverses as follows, where "OdRs" stands for Odd Residues and "IOdRs" stands for Inverse of Odd Residues:

F1: $(2^n) \rightarrow$ OdRs(2^{n+1});
F2: OdRs $(2^{n+1}) \rightarrow$ IOdRs (2^{n+1})
F3: IOdRs $(2^{n+1}) \rightarrow (2^n)$
F4: $(2^n) \rightarrow (2^n)$

The algorithm for computing the pseudo inverses is a direct implementation of these functions and is called "pseudo inverses modulo 2^n algorithm". Figure 1 shows the implementation of algorithm in steps: the first function (F1) is implemented on the first two steps; the second function (F2) is implemented at the third step; the third function (F3) is given by the fourth step; the fourth function (F4) is shown in step 5.

```
1. given a remainder "r" in modulo "$2^n$"
2. compute "s" = "2r – 1"
3. compute "t" the inverse of "s" in modulo "$2^{n+1}$"
4. compute "u" = "(s+1)/2"
5. "u" is the pseudo inverse of "r" modulo $2^n$
```

Figure 1. Algorithm for finding pseudo inverses modulo 2^n

To demonstrate the algorithm, we use the value "4" for the variable "n" and display the results on table 2 for all the remainders in ascending order. The left most column is considered column number 1, consequently, the results of each of the steps of the algorithm are displayed in the column sharing its number.

Except for step number "3", the rest are self explanatory. It will be sufficient to state that if "x" and "y" are inverses of each other in modulo 2^{n+1}, then there is an integer "m" that relates them as follows:

$$x = (2^n * m + 1)/y \qquad (8)$$

The next step is to optimize the process which is covered in section 4.

r	s	t	u
1	1	1	1
2	3	11	6
3	5	13	7
4	7	23	12
5	9	25	13
6	11	3	2
7	13	5	3
8	15	15	8
9	17	17	9
10	19	27	14
11	21	29	15
12	23	7	4
13	25	9	5
14	27	19	10
15	29	21	11

Table 2. Demonstration of the pseudo inverses of 2^4

5. Optimizing the pseudo modulo approach

The proposed approach will be optimized by reducing both the computation complexity and storage requirement. For this purpose, a practically larger value for "n" will be needed. Without loss of generality, the value chosen for "n" is "7". This gives an S-Box with 128 entries computed as pseudo inverses using arithmetic modulo 2^8. Table 3 below shows inverses computed using the field of odd numbers modulo 2^8. Even numbered columns are eliminated because their entries are not members of this field.

	1	3	5	7	9	B	D	F
0	01	AB	CD	B7	39	A3	C5	EF
1	F1	1B	3D	A7	29	13	35	DF
2	E1	8B	AD	97	19	83	A5	CF
3	D1	FB	1D	87	09	F3	15	BF
4	C1	6B	8D	77	F9	63	85	AF
5	B1	DB	FD	67	E9	D3	F5	9F
6	A1	4B	6D	57	D9	43	65	8F
7	91	BB	DD	47	C9	B3	D5	7F
8	81	2B	4D	37	B9	23	45	6F
9	71	9B	BD	27	A9	93	B5	5F
A	61	0B	2D	17	99	03	25	4F
B	51	7B	9D	07	89	73	95	3F

C	41	EB	0D	F7	79	E3	05	2F
D	31	5B	7D	E7	69	53	75	1F
E	21	CB	ED	D7	59	C3	E5	0F
F	11	3B	5D	C7	49	33	55	FF

Table 3. Inverses of off numbers modulo 2^8

The table gives 128 inverses for an S-Box with 128 entries. For a 256 entries S-Box, one has to take the value of "n" to be equal to 8.

Although there are 128 computed inverses on table 3, one needs to determine the inverses of only half of them and the other half will follow by symmetry. That is, if "x" is the inverse of "y", then "y" is the inverse of "x". For example, if the inverse of "45" can easily be determined from the 4th row and the 5th column of the table as "8D", then it will be redundant to have the number "45" on the table in the eighth row and the D' column. It is interesting to observe that further optimization can be achieved through the following lemma:

Lemma 3
If "x" is the inverse of "y" in modulo 2^n arithmetic, then the 2^n's complement of x is the inverse of the 2^n's complement of y. That is, $(2^n - x)$ is the inverse of $(2^n - y)$ modulo 2^n arithmetic.

Proof:
Given "x" and "y" to be inverses in modulo 2^n arithmetic, then for $(2^n - x)$ to be the inverse of $(2^n - y)$ they must satisfy for some integer "m" the following:

$$(2^n - x) * (2^n - y) \;=\; 2^n * m + 1$$

Rewriting the left hand side we get:

$$(2^{2n} - x*2^n - y*2^n + x*y) = 2^n[2^n - (x+y)] + x*y$$

Since $x*y$ can be rewritten as $(2^n *k +1)$, it follows that

$$(2^{2n} - x*2^n - y*2^n + x*y) = 2^n [2^n +k- (x+y)] + 1$$

Taking "m" to be equal to $[2^n +k- (x+y)]$ completes the proof.

Based on Lemma 3, table 3 will therefore reduce to table 4 below:

	1	3	5	7	9	B	D	F
0	01	AB	CD	B7	39	A3	C5	EF
1	F1	1B	3D	A7	29	13	35	DF
2	E1	8B	AD	97	19	83	A5	CF
3	D1	FB	1D	87	09	F3	15	BF
4	C1	6B	8D	77	F9	63	85	AF
5	B1	DB	FD	67	E9	D3	F5	9F
6	A1	4B	6D	57	D9	43	65	8F
7	91	BB	DD	47	C9	B3	D5	7F

Table 4. A Reduced version of table 3

Table 4 enables the lookup of inverses for the odd hexadecimal numbers from 01 to 7F, where the column headers are the Least Significant Digits (LSD) and the row headers are the Most Significant Digits (MSD). If one needs to find the inverse of a double hexadecimal digits number where the MSD is less than "8", then the answer can be looked up from table 4. However, if the MSD is more than "7", then lemma 3 has to be used. Here is an example for a case where the MSD is greater than "7".

To find the inverse of "9D" one must use lemma 3 because the MSD is greater than 7. The lemma will be demonstrated in this case in three steps:

1. obtain the 2^8's complement of "9D" which is equal to "63".
2. find the inverse of "63" using table 4, which is equal to "4B".
3. find the 2^8's complement of "4B" which is equal to "B5".

The result from step 3 in this case "B5" is the inverse of "9D" obtained indirectly from table 4. A quick look at table 3 confirms that the inverse of "9D" is indeed "B5". To even further optimize the process, table 4 can be reduced to a single row, namely, row number 1. The following definition is necessary for the said hypothesis.

Definition 1: The generator G_m (G_l) is a one dimensional array that contains the MSD (LSD) digit of the inverses in the first row of table 4 as follows:

$$G_m = [0, A, C, B, 3, A, C, E]$$
$$G_l = [1, B, D, 7, 9, 3, 5, F]$$

Based on this definition the following lemma will summarize the results of this section, hence no proof is presented.

Lemma 4

The single step for computing the inverse of a two hexadecimal digits number "H_mH_l" is as follows:

$$\text{Inverse of } (H_mH_l) = \begin{cases} (G_m[H_l] - H_m) \,\|\, G_l[H_l] & \text{for } H_m = 1,7,9 \text{ or } F \\ (G_m[H_l] - 9*H_m) \,\|\, G_l[H_l] & \text{for } H_m = 3,5,B \text{ or } D \end{cases}$$

6. Conclusion

An implementation of the Advanced Encryption Standard that uses the field of odd residues of a power of two as a platform has been investigated. The paper suggested an alternative to the Galois Field that may very well be desirable for situations where storing of the S-Box is not desirable or feasible. The proposed technique is based on arithmetic modulo powers of two and enabled the reduction of storage space to only ($1/2^n$) of the amount required by the complete S-Box. The eliminated entries of the S-Box can be regenerated from the stored part. The stored part was carefully chosen to be small enough to reduce the vulnerability but also large enough to form a basis that can be used to generate all the entries. Although this paper introduced a new mathematical field and significant savings in the storage requirement, the ultimate measure for its importance comes from one fact: that is, whether or not further work will confirm that the resulting AES implementation is less vulnerable compared to the original implementation via Galois Field.

References

[1] Rijmen, V. "Security and Implementation of the AES" 2nd International Workshop on the state of the art in cryptology and new challenges ahead, Warsaw, Poland, Thursday, May 13th, 2004

[2] Harvey, I.,"The Effects of Multiple Algorithms in the Advanced Encryption Standard", nCipher Corporation Ltd., 4'Th January 2000 Retrieved on November 6, 2005

[3] Daemen,J. and Rijmen, V. , "AES Proposal: Rijndael," Document vers on 2, Date: 03/09/99. Retrieved on October 20, 2005

[4] Weeks, B., Mark, Bean, B., Rozylowicz, T., Ficke, C. "Hardware Performance Simulations of Round 2 Advanced Encryption Standard Algorithms", National Security Agency. Retrieved on November 8, 2005

[5] Ichikawa, T., Kasuya, T., Matsui, M. "Hardware Evaluation of AES Finalists", Kamakura Office, Mitsubishi Electric Engineering Company Limited. Retrieved on October 30, 2005.

[6] Chodowiec, P. and Gaj, K., "Comparison of the Hardware Performance of AES candidates using reconfigurable hardware" , spring 2002

[7] http://www.nist.gov/aes "Advanced Encryption Standard Development Effort"..

[8] Satoh, A., Morioka, S., Takano, K. and Munetoh, S. "A Compact Rijndael Hardware Architecture with S-Box Optimization," Proc.Advances in Cryptology—ASIACRYPT 2001, pp. 239-254, 2001.

[9] Algotronix http://www.algotronix.com/engineering/aes1.html

[10] http://www.cmpe.boun.edu.tr/~yurdakul/papers/Ozpinar DSD03.pdf

[11] Abuelyaman, E. "Alternative S-Box Computation Method for AES Environments." Technical Report, School of Information Technology, Illinois State University, Normal, IL. December 2005

[12] Guy, R. K. "Euler's Totient Function," "Does $\phi(n)$ Properly Divide $n-1$," "Solutions of $\phi(m)=\sigma(n)$," "Carmichael's Conjecture," "Gaps Between Totatives," "Iterations of ϕ and σ," "Behavior of $\phi(\sigma(n))$ and $\sigma(\phi(n))$." §B36-B42 in Unsolved Problems in Number Theory, 2nd ed. New York: Springer-Verlag, pp. 90-99, 1994.

[13] Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C., Ferguson, N. "Twofish: A 128-Bit Block Cipher", 15'Th June, 1998

[14] Anderson, R., Biham, E. and Knudsen, L. ," The Case for Serpent" , 24th March 2000

[15] Ronald L. Rivest1, M.J.B. Robshaw2, R. Sidney2, and Y.L. Yin2, "The RC6 Block Cipher", August 20, 1998

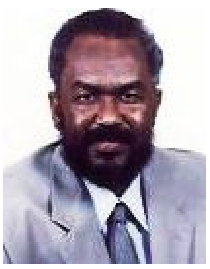
[16] IBM MARS Team, "MARS and the AES Selection Criteria", May 15, 2000

[17] Daemen, J. and Rijmen, V. "AES Proposal: Rijndael " Document version 2 1999 – May

[18] Abuelyaman, E. and El-Affendi, M. " S-Box Construction Using Arithmetic Modulo Prime Numbers", The first conference on digital communications and computer applications, Jordon University of Science & Technology, March 19-22, 2007.

[19] Swankoski,, E.J., Brooks, R.R., Narayanan, V., Kandemir, M., Irwin, M.J "A Parallel architecture for Secure FPGA Symmetric Encryption" , 2004

Eltayeb Salih Abuelyaman received a PhD degree in Computer Engineering from the University of Arizona in the US in 1988. He served as faculty member at various universities in the US for 18 years before moving to Prince Sultan University in Saudi Arabia where he served as a Faculty Member, a Director of the Information Technology and Computing Services and currently serves as the Dean of the College of Computer Science and Information Systems. His current research Interest is in the area of Computer Networks and Information Security.



Dr. El-Affendi obtained a PhD in computer science from Bradford University in 1983. He is currently an associate professor of computer science in the department of computer science, Prince Sultan University, KSA and the Director of the University Research Center. Current research interests of Dr. El-Affendi include computer networks, distributed systems and applied computational linguistics.