AENSI Journals

Australian Journal of Basic and Applied Sciences

ISSN:1991-8178

Journal home page: www.ajbasweb.com

Design of SubBytes and InvSubBytes Transformations of AES Algorithm Using Power Analysis Attack Resistant Reversible Logic Gates

[1]P Saravanan and [2]P Kalpana

[1]Assistant Professor, PSG College of Technology, Department of Electronics and Communication Engineering, Coimbatore, India
[2]Professor, PSG College of Technology, Department of Electronics and Communication Engineering, Coimbatore, India

ARTICLE INFO

Article history:
Received 19 September 2014
Received in revised form
19 November 2014
Accepted 22 December 2014
Available online 2 January 2015

Keywords:
Advanced Encryption Standard (AES), Substitution Box (S-Box), Reversible Gates, Cryptography, Power Analysis Attacks, Quantum Cost.

ABSTRACT

Background: The SubBytes and InvSubBytes transformations of Advanced Encryption Standard (AES) algorithm are conventionally implemented by using either look-up tables or combinational logic circuits. Both implementations are susceptible to power analysis attacks as they consume substantial amount of power during their normal operation. **Objective:** To overcome the power analysis attacks in SubBytes and InvSubBytes transformations by using reversible logic gates. Since reversible logic gates ideally consume zero power, they found to be the right candidates for implementing the security algorithms against power analysis attacks. **Results:** The proposed reversible SubBytes and InvSubBytes transformations utilizes Toffoli family of reversible gates for their logic synthesis. Our proposed design shows 35% reduction in Gate count and 97% reduction in Quantum cost compared to the existing design of reversible SubBytes and InvSubBytes transformation module. This is mainly achieved by reusing the existing reversible gates in the structure. **Conclusion:** A Novel reversible gate design of SubBytes and InvSubBytes transformations (S-Box) of the AES algorithm is presented. Since the reversible gates ideally consume zero power, they are exploited here to construct the S-Box which makes the proposed design secure against power analysis attacks. The reversible gate design can further be extended to other round functions in AES algorithm to make it resistant against power analysis attacks.

To Cite This Article: P Saravanan and P Kalpana, Design of SubBytes and InvSubBytes Transformations of AES Algorithm Using Power Analysis Attack Resistant Reversible Logic Gates. ***Aust. J. Basic & Appl. Sci.**, 9(1): 8-18, 2015*

INTRODUCTION

A cryptographic algorithm is an essential part in network security. A well-known cryptographic algorithm is the Data Encryption Standard (DES) (Schneier, 1996) which has been widely adopted in many security products. However, serious considerations arise for long-term security because of the relatively short key word length of only 56 bits and from the highly successful cryptanalysis attacks. In November 2001, the National Institute of Standards and Technology (NIST) of the United States chose the Rijndael algorithm as the suitable Advanced Encryption Standard (AES) (NIST, 2001) to replace the DES algorithm. Since then, many hardware implementations have been proposed in literature. Some of them use Field Programmable Gate Arrays (FPGAs) (Chodowiec and Gaj, 2003) and some use Application Specific Integrated Circuits (ASICs) (Su *et al*., 2003). But both FPGA and ASIC implementations consume power from its source which leads to power analysis attacks.

The first concerns about the reversibility of computation were raised in the 1960s (Landauer, 1961). There were two related issues, logical reversibility and physical reversibility, which were intimately connected. Logical reversibility refers to the ability to reconstruct the input from the output of a computation, or gate function. The heat dissipated during a process is usually taken to be a sign of physical irreversibility, that the microscopic physical state of the system cannot be restored exactly as it was before the process took place. That classical computation can be done reversibly with no energy dissipated per computational step was discovered by Bennett (1973).

Reversible computing is a new paradigm for implementing cryptographic algorithms using reversible gates. The primary reason is due to the increasing demands for low power and more secured devices. As our computing demands become more complex, the power requirements tend to increase. These increased power traces will lead to side channel attacks on cryptographic systems such as Simple Power Analysis (SPA),

Corresponding Author: P Saravanan, Assistant Professor, Department of Electronics and Communication Engineering,
PSG College of Technology, Coimbatore -641004. India.
E-mail: dpsaravanan@yahoo.com

Differential Power Analysis (DPA) and High Order Power Analysis (HO-PA) attacks. Cryptographic systems implemented with reversible gates ideally consume zero power and hence thwart all side channel attacks related to power analysis. Saravanan and kalpana (2014) proposed energy efficient implementations of reversible building blocks to thwart power analysis attacks. By using proper charge sharing mechanism, resistance to power analysis attacks has been achieved in the proposed implementations.

Thapliyal and Zwolinski (2006) made an attempt to develop secure crypto system by using reversible logic gates. In this work, they presented a prototype of a reversible ALU for a crypto-processor but did not target any particular crypto algorithm. Quantum realization of a ternary full-adder was proposed using macro-level ternary Feynman and Toffoli gates built on the top of ion-trap realizable ternary 1-qutrit and Muthukrishnan–Stroud gates (Khan and Perkowski, 2007). Reversible gate design of single precision floating point multiplier was proposed (Nachtigal *et al*., 2010) based on operand decomposition approach. Datta *et al.* (2013) proposed reversible logic synthesis of 128-bit AES algorithm using Toffoli gate family. Since Exclusive-or-Sum-Of-Products (ESOP) based reversible logic synthesis method has been used in this work, the optimization is very poor in terms of number of reversible gates used and Quantum cost.

The SubBytes and the InvSubBytes transformations in the AES algorithm are susceptible to power analysis attacks hence many researchers have proposed different mechanisms to make the implementation more robust to power analysis attacks (Mazumdar *et al*., 2012). In this work, gate level designs of the SubBytes and InvSubBytes transformations are first deduced by using composite field arithmetic operations (Zhang and Parhi, 2004; Mui, 2007). And then, conventional logic gates are replaced by reversible gates and they are reused wherever possible in order to reduce the Gate count and Quantum cost. The proposed reversible gate designs are optimized in terms of reduced number of reversible gates and quantum cost. The proposed reversible gate designs of SubBytes and InvSubBytes transformations are resistant to power analysis attacks due to the zero power consumption of reversible gates.

Reversible logic:

In conventional CMOS logic design, the logic 1's and 0's needed for the internal operation of the computers are created by exchanging charge from one of the DC power supply rails. During this process, the entire switching energy is converted into heat and results in a loss of energy. Bennet in 1973 discovered that classical computation can be done reversibly with no energy dissipated per computational step. The energy dissipation per state can be expressed as $k_B Tlnm$, where m is the mean number of immediate predecessors 1) averaged over states near the intended path, or 2) averaged over all accessible states, whichever is greater. For a typical irreversible system, which throws away about one bit per logical operation, m is approximately two, and thus $k_B Tln2$ is the approximate lower bound on the energy dissipation of such systems (Landauer, 1961; Landauer, 1991). For a logically reversible system, however, *m* is exactly one by construction and hence, the energy dissipation theoretically approaches zero under ideal physical circumstances (Bennett, 1973; Bennett, 1985).

Reversible gates are bijective transformations on the inputs, i.e. number of inputs and outputs are equal and every distinct input gives a distinct output (Wille, 2011). Reversible gates do not allow any fan-out or feedback. The commonly used reversible gates are NOT (Toffoli gate with zero control line), Controlled-NOT alias CNOT (Toffoli gate with one control line), Controlled-Controlled-NOT alias CCNOT (Toffoli gate with two control lines), SWAP and Fredkin gates as shown in Fig. 1. The behavior of some reversible gates is defined as follows:

NOT: $a'=1\oplus a$

CNOT: $a'=a, \quad b'=a\oplus b$ (a is the control line)

CCNOT: $a'=a, \quad b'=b, \quad c'=c\oplus ab$ (a,b are the control lines)

SWAP: $b'=a, \quad a'=b$

FREDKIN: $a'=a, \quad b'=b\oplus ab\oplus ac, \quad c'=c\oplus ab\oplus ac$ (a is the control line)

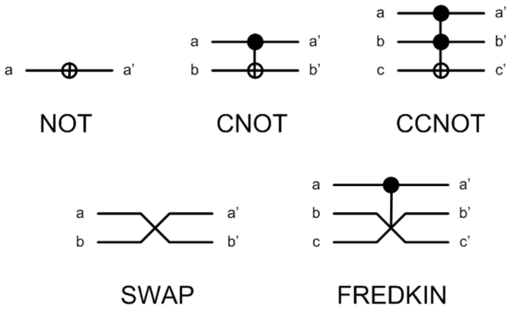

Fig. 1: Symbol of reversible gates.

The reversible logic synthesis can be done either with Toffoli gate family or Fredkin gate family since both are universal gates. In our proposed reversible design, Toffoli family of reversible gates has been used for reversible logic synthesis. The Quantum cost of Toffoli gate with 0, 1 and 2 control lines is 1, 1, and 5 respectively. The performance metrics considered for reversible gate design are number of ancilla inputs, number of garbage outputs, number of reversible gates used, its Quantum cost and delay in terms of number of stages. Ancilla inputs (Constants with either 0 or 1) and Garbage outputs are information that are not needed for the actual computation. They are required since the reversibility necessitates an equal number of outputs and inputs. Quantum cost denotes the effort needed to transform a reversible circuit to a quantum circuit (Wille, 2011). Since Gate count and Quantum cost are the important performance metrics directly related to the hardware resources, they are analyzed in this work and the improvements are tabulated.

Aes algorithm:

The AES algorithm (NIST, 2001) is a symmetric block cipher that processes data blocks of 128 bits using a cipher key of length 128, 192, or 256 bits. Each data block consists of a $4x4$ array of bytes called the state S, on which the basic operations of the AES algorithm are performed. In the encryption procedure, after an initial round key addition, a round function consisting of four different transformations—SubBytes, ShiftRows, MixColumns, and AddRoundKey—is applied to the data block in the encryption procedure.

The SubBytes transformation is a nonlinear byte substitution that operates independently on each byte of the state S using a substitution table (S-Box). The ShiftRows operation is a circular shifting on the rows of the state with different numbers of bytes (offsets). The MixColumns transformation mixes the bytes in each column of the state by the multiplication with a fixed polynomial modulo x^4+1. AddRoundKey is an XOR operation that adds a round key to the state S in each iteration, where the round keys are generated during the key expansion phase. The round function is performed iteratively 10, 12, or 14 times (Nr), depending on the key length of 128, 192 or 256 bits respectively. The MixColumns transformation is not applied to the final round.

Subbytes and invsubbytes transformations:

In the encryption module, the SubBytes transformation is a non-linear transformation, which computes the multiplicative inverse of each byte of the state S in *GF(2^8^)* with irreducible polynomial $P(x) = x^8 + x^4 + x^3 + x + 1$ followed by an affine transformation (Good, 2006). The transformation in the decryption module performs the inverse of the corresponding transformation in the encryption module. The SubBytes and InvSubBytes transformations can be implemented by two different approaches. We can either construct a single circuit directly whose input-output relation is equivalent to the SubBytes transformation known as Look-up table approach or construct a multiplicative inversion circuit and an affine transformation circuit independently, and then cascade these two circuits to design the SubBytes transformation known as Composite field approach. Applying composite field arithmetic, the elements of large-order fields are mapped to those of small-order fields in which the field operations can be carried out in a simpler way with reduced hardware cost.

In order to maintain additive and multiplicative homomorphism, an isomorphic mapping function δ need to be applied to map the representation of an element in $GF(2^8)$ to its composite field (Mui, 2007). After completing the operations in composite field, it is necessary to remap the elements in composite field back to $GF(2^8)$. This can be done by the inverse isomorphic function δ^{-1} (Mui, 2007).

Subbytes and invsubbytes transformations using composite field arithmetic:

The SubBytes transformation can be performed by taking the multiplicative inverse in $GF(2^8)$ followed by affine transformation and the InvSubBytes transformation can be performed by inverse affine transformation followed by multiplicative inverse in $GF(2^8)$. The sequence of steps to carryout both transformations are shown below.

SubBytes transformation : Multiplicative inversion in $GF(2^8)$ → Affine transformation
InvSubBytes transformation : Inverse affine transformation → Multiplicative inversion in $GF(2^8)$.

a. Multiplicative Inverse in GF(2^8) using Composite Field Arithmetic:

The $GF(2^8)$ multiplicative inversion involved in the SubBytes and InvSubBytes transformations is a hardware demanding operation when it is directly implemented in $GF(2^8)$ (Jing *et al.*, 2001). As mentioned earlier, the hardware complexity can be reduced to a greater extent when the field operations are carried out in composite field rather than $GF(2^8)$. The basic building blocks required to implement $GF(2^8)$ multiplicative inversion module is shown in Fig. 2 (Zhang and Parhi, 2004).

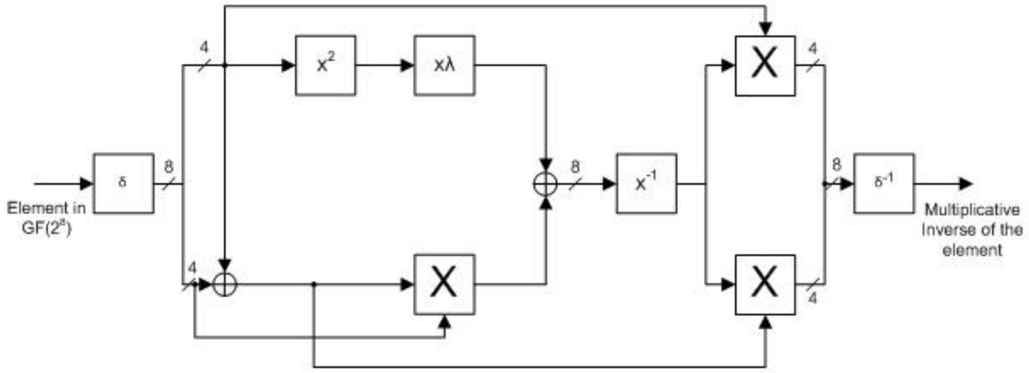

Fig. 2: Basic building blocks to implement *GF(2^8)* multiplicative inversion module.

b. Implementation of Affine Transformation and Inverse Affine Transformation:

The Affine transformation should be done at the final stage after taking the multiplicative inverse in *GF(2^8)* to perform SubBytes transformation. Similarly, Inverse affine transformation should be done at the initial stage before taking the multiplicative inverse in *GF(2^8)* to perform InvSubBytes transformation (Mui, 2007).

Proposed Reversible GF(2^8) Multiplicative Inverse Module:

As mentioned in the previous section, the SubBytes transformation requires the computation of multiplicative inverse followed by the affine transformation and the InvSubBytes transformation requires the computation of inverse affine transformation followed by finding the multiplicative inverse. Hence both SubBytes and InvSubBytes transformations require the calculation of multiplicative inverse in *GF(2^8)*.

a. Reversible Isomorphic and Inverse Isomorphic Mapping Block $(\delta$ ***and*** $\delta^{-1})$***:***

Reversible gate designs are functionally reversible, hence, it is enough if either forward isomorphic mapping also called as δ matrix (Mui, 2007) or inverse isomorphic mapping called as δ^{-1} matrix (Mui, 2007) can be designed. The number of XOR operations required in the forward isomorphic mapping is 24 whereas inverse isomorphic mapping takes only 23 XOR operations. Hence, inverse isomorphic mapping function is designed in this work by using reversible gates and the same design can be used for forward isomorphic mapping also.

The reversible gate design of inverse isomorphic mapping requires 23 CNOT gates which results in a Quantum cost of 23. This calculation is based on the assumption that each XOR operation requires one CNOT gate when the reversible logic synthesis is performed by conventional design using one-to-one mapping from logic operations to equivalent reversible gates. But, in our proposed design, the Gate count and Quantum cost is optimized by properly reusing the existing reversible gates. Our proposed reversible inverse isomorphic mapping block takes only 15 CNOT gates and has a Quantum cost of 15 which gives 35% savings in both Gate count and Quantum cost compared to the conventional design. The proposed design has zero ancilla inputs, zero garbage outputs and has delay of 13 as shown in Table 1. The reversible gate design of forward / inverse isomorphic mapping block is shown in Fig. 3.

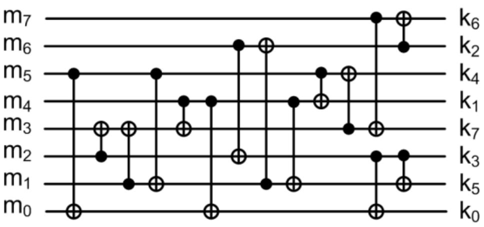

Fig. 3: Reversible IsoMap / InvIsoMap block.

b. Reversible Squarer and Multiply with Constant λ ***Block:***

Squaring Operation in GF(2^4):

Let $k = m^2$, where k and m are elements of *GF(2^4)*, which can be represented in binary as $\{k_3k_2k_1k_0\}_2$ and $\{m_3m_2m_1m_0\}_2$ respectively. The squaring operation in *GF(2^4)* is shown in equation (1) (Mui, 2007).

$$\begin{aligned} k_3 &= m_3 \\ k_2 &= m_3 \oplus m_2 \\ k_1 &= m_2 \oplus m_1 \\ k_0 &= m_3 \oplus m_1 \oplus m_0 \end{aligned} \tag{1}$$

Multiplication with Constant λ :

Let $k = m\lambda$, where k, m and λ are elements of *GF(2⁴)*, which can be represented in binary as $\{k_3k_2k_1k_0\}_2$, $\{m_3m_2m_1m_0\}_2$ and $\lambda = \{1100\}_2$ respectively. The multiplication with constant λ in *GF(2⁴)* is shown in equation (2) (Mui, 2007).

$$\begin{aligned} k_3 &= m_2 \oplus m_0 \\ k_2 &= m_3 \oplus m_2 \oplus m_1 \oplus m_0 \\ k_1 &= m_3 \\ k_0 &= m_2 \end{aligned} \tag{2}$$

Table 1: Performance metrics of reversible building blocks of *GF(2⁸)* multiplicative inversion module

S.No	Name of the Block	No. of Ancilla Inputs	No. of Garbage Outputs	No. of Reversible Gates	Quantum Cost	Delay
1.	IsoMap / InvIsoMap	0	0	CNOT - 15	15	13
2.	Squarer and Multiplication by Constant λ	0	0	CNOT - 4	4	3
3.	Adder (XOR Block)	0	4	CNOT - 4	4	1
4.	Multiplication in *GF(2⁴)*	13	17	CNOT - 25 CCNOT - 9	70	18
5.	Multiplicative Inverse in *GF(2⁴)*	8	8	CNOT - 14 CCNOT - 8	54	19

Proposed Merged Reversible Block:

In our proposed design, the squarer block and multiplication with constant λ block are merged together in order to reduce the operations. The output of the squarer block k_3, k_2, k_1, k_0 in equation (1) is substituted as input to m_3, m_2, m_1, m_0 of multiplication with constant λ block in equation (2). After simplifying the terms, the output of the merged squarer and multiplication with constant λ block is given in equation (3).

$$\begin{aligned} k_3 &= m_2 \oplus m_1 \oplus m_0 \\ k_2 &= m_3 \oplus m_0 \\ k_1 &= m_3 \\ k_0 &= m_3 \oplus m_2 \end{aligned} \tag{3}$$

The squarer block takes 4 CNOT gates and multiplication with constant λ block takes 4 CNOT gates when they are synthesized separately by using reversible gates. But the proposed reversible gate design of the merged squarer and multiplication with constant λ block takes only 4 CNOT gates and has a Quantum cost of 4 which gives 50% savings in Gate count and Quantum cost compared to the individual designs. Also it takes zero ancilla input, zero garbage output and has a delay of 3 as shown in Table 1. The reversible squarer and multiplication with constant λ block in *GF(2⁴)* is shown in Fig. 4.

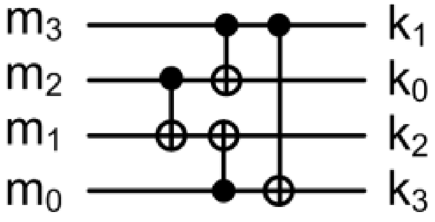

Fig. 4: Reversible squarer and multiplication with constant λ block.

c. Reversible GF(2^4) Adder:

The addition operation in *GF(2^4)* can be performed by simple logical XOR operation. The reversible gate design of the *GF(2^4)* adder block is shown in Fig. 5, which takes 4 CNOT gates and has a Quantum cost of 4. Also it takes zero ancilla inputs, 4 garbage outputs and has a delay of 1 as shown in Table 1.

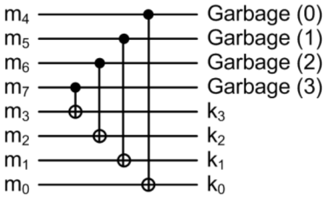

Fig. 5: Reversible *GF(2^4)* adder.

d. Reversible GF(2^4) Multiplier:

Let $k = mw$, where k, m and w are elements of *GF(2^4)* which can be represented in binary as $\{k_3k_2k_1k_0\}_2$, $\{m_3m_2m_1m_0\}_2$ and $w = \{w_3w_2w_1w_0\}_2$ respectively. The gate level implementation of multiplication in *GF(2^4)* is shown in Fig. 6 (Zhang and Parhi, 2004).

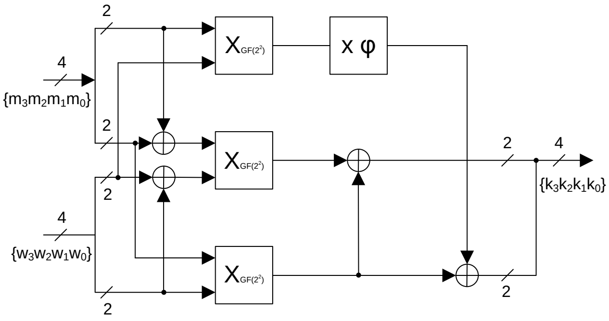

Fig. 6: Block level implementation of *GF(2^4)* multiplication.

The reversible gate design of *GF(2^4)* multiplier block requires 9 AND operations and 46 XOR operations. The reversible logic synthesis by one-to-one mapping takes 9 CCNOT gates and 46 CNOT gates which results in a Quantum cost of 91. The proposed design is optimized by reusing the reversible gates properly so that the reversible *GF(2^4)* multiplier block takes only 9 CCNOT gates and 25 CNOT gates and has a Quantum cost of 70 which results in 38% savings in Gate count and 23% savings in Quantum cost. The proposed design takes 13 ancilla inputs, 17 garbage outputs and has a delay of 18 as shown in Table 1. The reversible gate design of the multiplication in *GF(2^4)* is shown in Fig. 7.

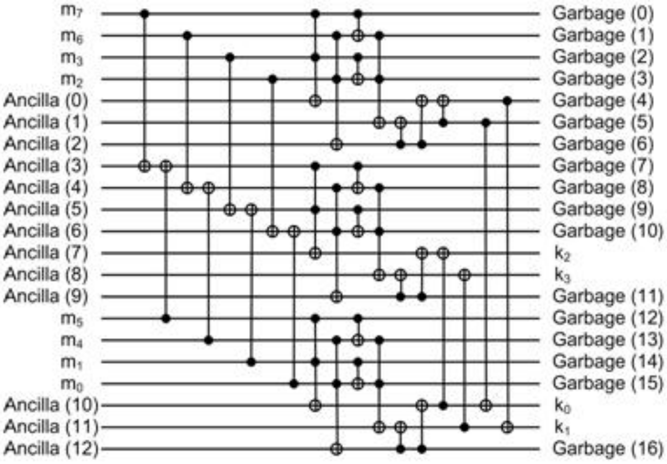

Fig. 7: Reversible *GF(2^4)* multiplier.

e. Reversible Multiplicative Inversion in $GF(2^4)$:

The Multiplicative inverse in $GF(2^4)$ can be calculated by three different approaches such as square-multiply approach, multiple decomposition approach and direct mapping approach (Zhang and Parhi, 2004). The reversible gate design of multiplicative inverse in $GF(2^4)$ using Square and Multiply approach takes 70 CNOT, 18 CCNOT gates and has a Quantum cost of 160. Also it takes 34 ancilla inputs, 42 garbage outputs and has a delay of 56 as shown in Table 2. The reversible gate design of multiple decomposition approach is shown in Fig. 8. It takes 22 CNOT, 9 CCNOT gates and has a Quantum cost of 67. Also it takes 14 ancilla inputs, 14 garbage outputs and has a delay of 19 as shown in Table 2.

Table 2: Performance analysis of different reversible multiplicative inverse approaches in $GF(2^4)$

S.No	Name of the Approach	No. of Ancilla Inputs	No. of Garbage Outputs	No. of Reversible Gates	Quantum Cost	Delay
1.	Square - Multiply	34	42	CNOT - 70 CCNOT - 18	160	56
2.	Multiple Decomposition	14	14	CNOT - 22 CCNOT - 9	67	19
3.	Direct Mapping	8	8	CNOT - 14 CCNOT - 8	54	19

Let $m = \{m_3m_2m_1m_0\}_2$ is an element in $GF(2^4)$ and its inverse is given by $m^{-1} = \{m_3^{-1}m_2^{-1}m_1^{-1}m_0^{-1}\}_2$ which can be obtained by direct mapping approach as given in equation (4) (Zhang and Parhi, 2004). By analyzing equation (4), it can be inferred that the direct mapping approach requires 25 AND operations and 21 XOR operations to calculate the multiplicative inverse. The reversible logic synthesis of equation (4) using one-to-one mapping takes 25 CCNOT gates, 21 CNOT gates and has a Quantum cost of 146. In our proposed reversible gate design of $GF(2^4)$ multiplicative inversion module, the reversible gates are properly reused so that it takes only 8 CCNOT and 14 CNOT gates with a Quantum cost of 54 as shown in Fig. 9. The reusability of reversible gates results in 52% savings in Gate count and 63% savings in Quantum cost. The proposed design takes 8 ancilla inputs and 8 garbage outputs and has a delay of 19 as shown in Table 2.

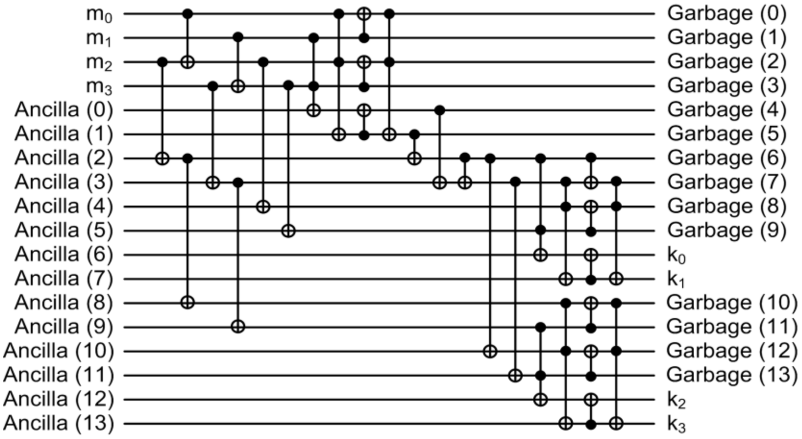

Fig. 8: Reversible gate design using multiple decomposition approach.

$$\begin{aligned}
k_3 &= m_3 \oplus m_3m_2m_1 \oplus m_3m_0 \oplus m_2 \\
k_2 &= m_3m_2m_1 \oplus m_3m_2m_0 \oplus m_3m_0 \oplus m_2 \oplus m_2m_1 \\
k_1 &= m_3 \oplus m_3m_2m_1 \oplus m_3m_1m_0 \oplus m_2 \oplus m_2m_0 \oplus m_1 \\
k_0 &= m_3m_2m_1 \oplus m_3m_2m_0 \oplus m_3m_1 \oplus m_3m_1m_0 \oplus m_3m_0 \oplus m_2 \oplus m_2m_1 \oplus m_2m_1m_0 \oplus m_1 \oplus m_0
\end{aligned} \tag{4}$$

From Table 2, it can be inferred that the multiplicative inverse in $GF(2^4)$ can be efficiently computed with direct mapping approach since it takes less number of reversible gates and the Quantum cost involved is also less. This is because, the composite field approach will not give optimum results when the order of the field involved is small such as $GF(2^4)$. Hence in our proposed reversible SubBytes and InvSubBytes transformations, the multiplicative inverse in $GF(2^4)$ is calculated by direct mapping approach.

f. Proposed Reversible Multiplicative Inversion in GF(2^8):

The performance metrics of the reversible building blocks of *GF(2^8)* multiplicative inversion module are given in Table 1. The proposed reversible *GF(2^8)* multiplicative inversion module requires two reversible IsoMap/InvIsoMap block, two reversible *GF(2^4)* adder, one reversible Squarer and Multiplication by constant λ block, three reversible *GF(2^4)* multiplier, one reversible *GF(2^4)* multiplicative inversion module as shown in Fig. 2 (Zhang and Parhi, 2004). In order to verify the functionality of the proposed reversible design, Verilog code for each reversible gate has been written and then all the reversible gates are instantiated to construct the complete design. The simulation of the Verilog code has been carried out in Xilinx ISim simulator. The simulation output of the proposed reversible *GF(2^8)* multiplicative inversion module is shown in Fig. 10 where the input x is an element in *GF(2^8)* and the output y is its multiplicative inverse. The performance metrics of the proposed reversible *GF(2^8)* multiplicative inversion module are tabulated in Table 3.

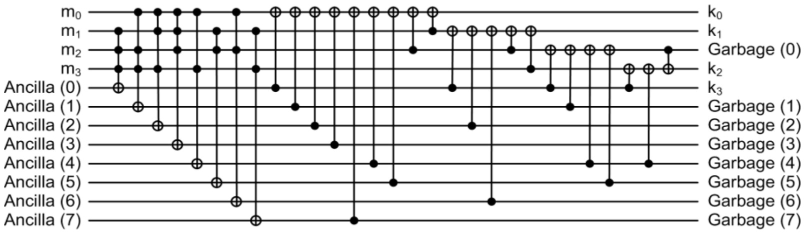

Fig. 9: Reversible gate design using direct mapping approach.

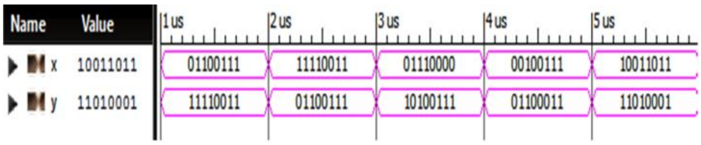
```
Name   Value      1 us       2 us       3 us       4 us       5 us
x      10011011   01100111   11110011   01110000   00100111   10011011
y      11010001   11110011   01100111   10100111   01100011   11010001
```

Fig. 10: Simulation output of the proposed reversible *GF(2^8)* multiplicative inversion module.

Table 3: Performance metrics of proposed reversible *GF(2^8)* multiplicative inversion module

S.No	Name of the Block	No. of Ancilla Inputs	No. of Garbage Outputs	No. of Reversible Gates	Quantum Cost	Delay
1.	IsoMap / InvIsoMap	0	0	CNOT - 30	30	26
2.	Squarer and Multiplication by Constant λ	0	0	CNOT - 4	4	3
3.	Adder (XOR Block)	0	8	CNOT - 8	8	2
4.	Multiplication in *GF(2^4)*	39	51	CNOT - 75 CCNOT - 27	210	36
5.	Multiplicative Inverse in *GF(2^4)*	8	8	CNOT - 14 CCNOT - 8	54	19
6.	Proposed Reversible *GF(2^8)* Multiplicative Inversion Module	47	67	CNOT - 131 CCNOT - 35	306	83

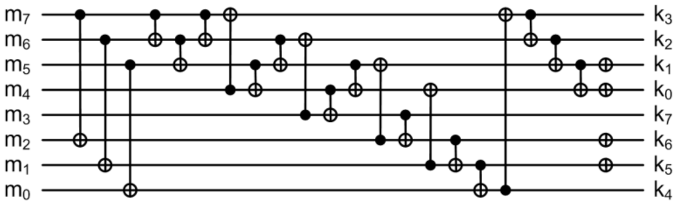

Fig. 11: Reversible affine transformation block.

Reversible affine transformation:

The reversible gate design of affine transformation is shown in Fig. 11. Since the design is functionally reversible, it is enough if either affine transformation or inverse affine transformation is considered. In this work, reversible gate design of affine transformation has been carried out which actually requires 4 NOT operations and 32 XOR operations. The reversible logic synthesis using one-to-one mapping approach requires 4 NOT and 32 CNOT gates with a Quantum cost of 36. By properly reusing the existing reversible gates, the proposed reversible affine transformation block is optimized to 4 NOT and 21 CNOT gates with a Quantum cost of 25. This optimization results in 31% savings in both Gate count and Quantum cost. The proposed design takes zero ancilla inputs, zero garbage outputs and has a delay of 21 as shown in Table 4.

Table 4: Performance metrics of proposed reversible SubBytes / InvSubBytes transformation module

S.No	Name of the Block	No. of Ancilla Inputs	No. of Garbage Outputs	No. of Reversible Gates	Quantum Cost	Delay
1.	Multiplicative Inverse in *GF(2^8)*	47	67	CNOT - 131 CCNOT - 35	306	83
2.	Affine Transformation	0	0	NOT - 4 CNOT - 21	25	21
3.	Proposed Reversible SubBytes / InvSubBytes Transformation Module	47	67	NOT - 4 CNOT - 152 CCNOT - 35	331	104

Proposed designs:

a. Reversible SubBytes Transformation:

The reversible SubBytes transformation can be obtained by cascading reversible GF(2^8) multiplicative inversion module and reversible affine transformation block as shown in Fig. 12 (Zhang, 2004). The simulation output of the proposed reversible SubBytes transformation is shown in Fig. 13. The proposed reversible SubBytes transformation takes 4 NOT gates, 152 CNOT gates, 35 CCNOT gates and has a Quantum cost of 331. Also it takes 47 ancilla inputs, 67 garbage outputs and has a delay of 104.

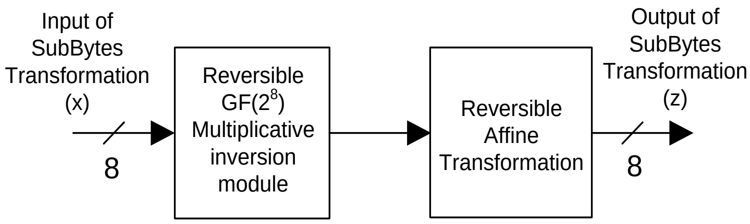
Input of SubBytes Transformation (x)
8
Reversible GF(2^8) Multiplicative inversion module
Reversible Affine Transformation
Output of SubBytes Transformation (z)
8

Fig. 12: Block diagram of the proposed reversible SubBytes transformation.

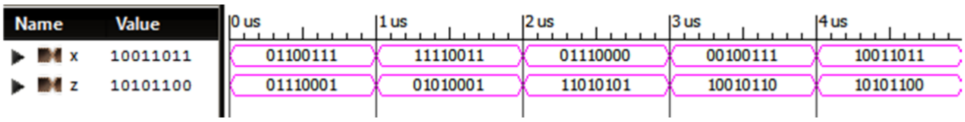

Fig. 13: Simulation output of the proposed reversible SubBytes transformation .

b. Reversible InvSubBytes Transformation:

The reversible InvSubBytes transformation can be obtained by cascading reversible affine transformation and reversible *GF(2^8)* multiplicative inversion module as shown in Fig. 14 (Zhang and Parhi, 2004). The performance metrics of the proposed reversible InvSubBytes transformation is similar to reversible SubBytes transformation as the same building blocks are used for both transformations. The simulation output of the proposed reversible InvSubBytes transformation is shown in Fig. 15. The proposed reversible InvSubBytes transformation takes 4 NOT gates, 152 CNOT gates, 35 CCNOT gates and has a Quantum cost of 331. Also it takes 47 ancilla inputs, 67 garbage outputs and has a delay of 104. The performance metrics of the proposed reversible SubBytes / InvSubBytes transformation are given in Table 4.

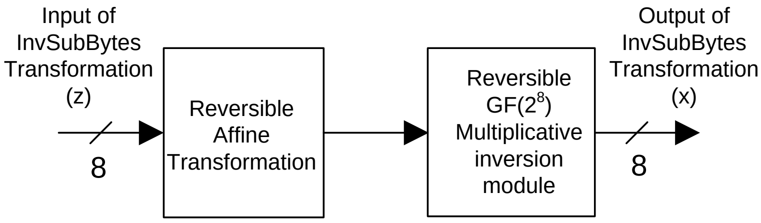
Input of InvSubBytes Transformation (z)
8
Reversible Affine Transformation
Reversible GF(2^8) Multiplicative inversion module
Output of InvSubBytes Transformation (x)
8

Fig. 14: Block diagram of the proposed reversible InvSubBytes transformation.

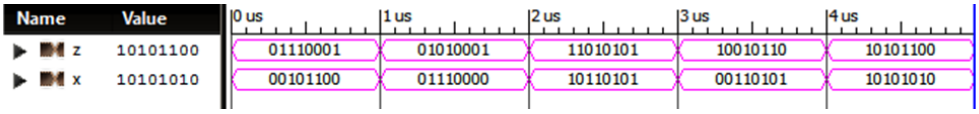

Fig. 15: Simulation output of the proposed reversible InvSubBytes transformation.

RESULTS AND DISCUSSIONS

Table 5 summarizes the performance improvement in the proposed reversible SubBytes / InvSubBytes transformation module compared to the conventional reversible designs. An important point to be noted here is that the conventional reversible designs are nothing but the reversible designs obtained by direct one-to-one mapping of logic operations to reversible gates. The two important performance metrics known as Gate count and Quantum cost are analysed in this work to show the improvement in the proposed reversible designs. Our proposed design of reversible SubBytes / InvSubBytes transformation module shows 36% reduction in Gate count and 35% reduction in Quantum cost compared to the conventional reversible designs. In addition, our proposed reversible gate design shows 35% reduction in Gate count and 97% reduction in Quantum cost compared to the existing reversible SubBytes and InvSubBytes transformation module (Datta *et al.*, 2013) as shown in Table 6.

Table 5: Performance improvement in the proposed design compared to conventional design.

S.No	Name of the Block	Gate Count		Quantum Cost		% Reduction	
		Conventional Design	Proposed Design	Conventional Design	Proposed Design	Gate Count	Quantum Cost
1.	IsoMap / InvIsoMap	46	30	46	30	35	35
2.	Squarer and Multiplication by Constant λ	8	4	8	4	50	50
3.	Adder (XOR Block)	8	8	8	8	-	-
4.	Multiplication in $GF(2^4)$	153	102	261	210	33	20
5.	Multiplicative Inverse in $GF(2^4)$	46	22	146	54	52	63
6.	Affine Transformation	36	25	36	25	31	31
7.	Proposed Reversible SubBytes / InvSubBytes Transformation	297	191	505	331	36	35

Conclusion:

A Novel reversible gate design of SubBytes and InvSubBytes transformations (S-Box) of the AES algorithm is presented. Since the reversible gates theoretically consume zero power, they are exploited here to construct the S-Box which makes the proposed design secure against power analysis attacks. Our proposed reversible SubBytes / InvSubBytes transformation module shows 36% reduction in Gate count and 35% reduction in Quantum cost compared to the conventional reversible designs. And our proposed design shows 35% reduction in Gate count and 97% reduction in Quantum cost compared to the existing design of reversible SubBytes and InvSubBytes transformation module. This is mainly achieved by reusing the existing reversible gates in the structure. The reversible gate design can further be extended to other round functions in AES algorithm to make it resistant against power analysis attacks.

Table 6: Performance improvement in the proposed design compared to existing design.

S.No	Functional Block	Gate Count		Quantum Cost		% Reduction	
		Existing Design (Datta *et al.*, 2013)	Proposed Design	Existing Design (Datta *et al.*, 2013)	Proposed Design	Gate Count	Quantum Cost
1.	Reversible SubBytes / InvSubBytes Transformation	294	191	11602	331	35	97

REFERENCES

Bennett, C.H., 1973. Logical reversibility of computation. IBM journal of Research and Development, 17(6): 525-532.

Bennett, C.H. and R. Landauer, 1985. The fundamental physical limits of computation. Scientific American, 253(1): 48-56.

Chodowiec, P. and K. Gaj, 2003. Very compact FPGA implementation of the AES algorithm. In the Proceedings of the 5th International Workshop on Cryptographic Hardware and Embedded Systems-CHES 2003, pp: 319-333.

Datta, K., V. Shrivastav, I. Sengupta and H. Rahaman, 2013. Reversible logic implementation of AES algorithm. In the Proceedings of the 8th International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS), pp: 140-144.

Good T. and M. Benaissa, 2006. Very small FPGA application-specific instruction processor for AES. IEEE Transactions on Circuits and Systems, 53(7): 1477-1486.

Jing, M.H., Y.H. Chen, Y.T. Chang and C.H. Hsu, 2001. The design of a fast inverse module in AES. In the Proceedings of the International Conferences on Info-tech and Info-net, pp: 298-303.

Khan, M.H.A. and M.A. Perkowski, 2007. Quantum ternary parallel adder/subtractor with partially-look-ahead carry. Journal of Systems Architecture, 53(7): 453-464.

Landauer, R., 1961. Irreversibility and heat generation in the computing process. IBM journal of research and development, 5(3): 183-191.

Landauer, R., 1991. Information is physical. Physics Today, 44(23): 25.

Mazumdar, B., D. Mukhopadhyay and I. Sengupta, 2012. Design for Security of Block Cipher S-Boxes to Resist Differential Power Attacks. In the Proceedings of the 25th International Conference on VLSI Design, pp: 113-118.

Mui, E.N.C., 2007. Practical implementation of Rijndael S-Box using Combinational logic, Online Available:http://www.xess.com/ projects/ Rijndael_SBox.pdf.

Nachtigal, M., H. Thapliyal and N. Ranganathan, 2010. Design of a reversible single precision floating point multiplier based on operand decomposition. In the Proceedings of the 10th IEEE Conference on Nanotechnology, pp: 233-237.

NIST, 2001. Advanced Encryption Standard (AES), FIPS-197.

Saravanan, P. and P. Kalpana, 2014. Energy Efficient Reversible Building Blocks Resistant To Power Analysis Attacks. Journal of Circuits, Systems and Computers, 23(9): 14501271-145012740.

Schneier, B., 1996. Applied Cryptography. Wiley, New York.

Su, C., T. Lin, C. Huang and C. Wu, 2003. A high-throughput low-cost AES processor. IEEE Communications Magazine, 41(12): 86-91.

Thapliyal, H. and M. Zwolinski, 2006. Reversible logic to cryptographic hardware: a new paradigm. In the Proceedings of the 49th IEEE International Midwest Symposium on Circuits and Systems-MWSCAS'06, pp: 342-346.

Wille, R., 2011. An introduction to reversible circuit design. In the Proceedings of the Saudi International Electronics, Communications and Photonics Conference-SIECPC, pp: 1-4.

Zhang, X. and K.K. Parhi, 2004. High-speed VLSI architectures for the AES algorithm. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 12(9): 957-967.