Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/090133

A Novel Way to Implement WDDL Logic to Resist Power Analysis Attack in Algorithm Level

*Nianhao Zhu**, *Yujie Zhou and Hongming Liu*

School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China

Received: 2 Apr. 2014, Revised: 3 Jul. 2014, Accepted: 4 Jul. 2014
Published online: 1 Jan. 2015

Abstract: Power analysis attacks focus on recovering the secret key of a cryptographic core from measurements of its consumed power when the cryptographic core is in encryption or decryption process. This paper designs a complementary AES decryption algorithm which is implemented in hardware as a complementary counterpart of AES decryption engine to resist power analysis attack. The algorithm which is complementary to AES is denoted as CAES. CAES decryption engine can provide complementary power to AES decryption engine by emulating the ideal of wave dynamic differential logic (WDDL), a power balanced hardware gate style. CAES decryption algorithm is an algorithm level countermeasure which can be easily implemented by hardware description language. This enables designers to design a security IC in a traditional design flow, while WDDL logic circuits employ a customer design flow. This paper specifies the detailed description of the CAES decryption algorithm and its hardware implementation. Correspondingly, we carried out power analysis attacks to AES decryption engines without CAES counterpart and with CAES counterpart. We use very accurate power traces through simulation and FPGA experiment to exhaustively examine our proposed countermeasure. The results show that CAES counterpart can thwart power analysis attacks and it is a promising approach to implement resistant crypto core.

Keywords: AES, correlation power analysis, power analysis attack, WDDL

1 Introduction

With the massive spreading out of inexpensive integrated circuits which are able to store and process confidential data, the phenomena that more and more research on information security issues has been sprung up [1]. Side-channel attacks (SCA) exploit the leaked physical information from chips to analyze the cryptographic devices and recovery the secret key stored in cryptographic devices [2]. Recently, SCA, especially power analysis attacks, have been extensively shown to be a major threat to the security of data that processed and stored in cryptographic devices, such as smart card. Simple power analysis (SPA), differential power analysis (DPA) [2] and correlation power analysis (CPA) [3] are three types of power analysis attacks. Correspondingly, a lot of countermeasures have been proposed in the last few years [4,5,6,7,8,9].

Masking is a very prevalent countermeasure, which randomizes intermediate values that are processed by the cryptographic device [10,11]. The goal of masking is to make the power consumption of a cryptographic device independent of the intermediate values of the cryptographic algorithm. An advantage of this approach is that it can be implemented at the algorithm level without changing the traditional integrated circuit (IC) design flow. Masking, as a countermeasure to power analysis attack, has been extensively discussed in the scientific community [9,12,13].

On the other hand, more generic countermeasures are also under discussion. These countermeasures are all on circuit level. We call them more generic in that they are not constrained to a certain cryptographic algorithm. Once a practical method is found, designers need not to care about the security of implementations for a specific algorithm. This makes possible the automatic design. These countermeasures fall into two categories: complementary circuits and gate level mask circuits.

Kris Tiri and Ingrid Verbauwhede [14] proposed a complementary logic style called sense amplifier based logic (SABL), in which dual-rail and pre-charge technology are employed. Considering SABL requires a new customer design cell library, simple dynamic differential logic (SDDL) and its improvement wave dynamic differential (WDDL) came into being afterward

* Corresponding author e-mail: zhunianhao2005@163.com

also under efforts of Kris Tiri [15]. Compared with SABL, WDDL only makes use of common cells. The complementary cell of WDDL is opposite to its original cell. In contrast, Kazuyuki et al. proposed homogeneous dual-rail logic (HDRL) in [16], which has the same complementary cell as original cell.

Besides complementary circuits, masking on gate level is analyzed in [17]. It is a masked and dual-rail pre-charge logic style and can be implemented using common CMOS standard cell libraries. The implementation of masked gate circuits in logic level has been presented by Thomas Popp and Stefan Mangard in [18].

Though the above methods, in both algorithm level and circuit level, aim at preventing DPA completely, they still leak side channel information. For masking methods, outputs? transitions of logic gates are dependent on the input signal when glitches exist [19]. What?s more, in [20], Stefan Mangard et al. did a successful attack on masked AES hardware implementations with glitches. For complementary circuits, loading capacitance is hard to control for deep submicron. process technologies where the transistor sizes and wiring widths continuously shrink [18].

Place and route of WDDL logic is different from traditional IC design flow, and no EDA tool is suitable for it. In this paper, we proposed a complementary AES (CAES) decryption algorithm, as an opposite counterpart of AES decryption algorithm to balance consumed power in algorithm level. The concept of CAES is the same as WDDL. WDDL aims to balance consumed power in gate level, while our proposed CAES decryption algorithm is to balance power in algorithm level. The functions of CAES decryption algorithm are always opposite to the function of AES decryption in algorithm, which leads to the balanced consumed power.

For example, original logic operation of AES decryption algorithm is bitwise AND, then complementary logic operation of CAES decryption algorithm is bitwise OR. That is the basic ideal which we used to design CAES decryption algorithm.

The level of achieving resistance to DPA of CAES is higher than WDDL logic, so our proposed countermeasure can be easily implemented in hardware description language (HDL). The traditional IC design flow can be adopted in security circuit with the help of our proposed countermeasure.

This paper serves for the purpose of summarizing all the work on CAES decryption algorithm done by authors. The structure is as follows. In Section 2, power analysis attack and WDDL technology are introduced. In Section 3, we give the detail decryption algorithm of CAES, and simulate power analysis attacks to transformations of AES decryption engine without counterpart CAES and with counterpart CAES. The effects of our proposed countermeasure to power analysis attack are also shown in this section. In Section 4, we use PFGA experiment to validate our proposed countermeasure. And Section 5 concludes all of our work.

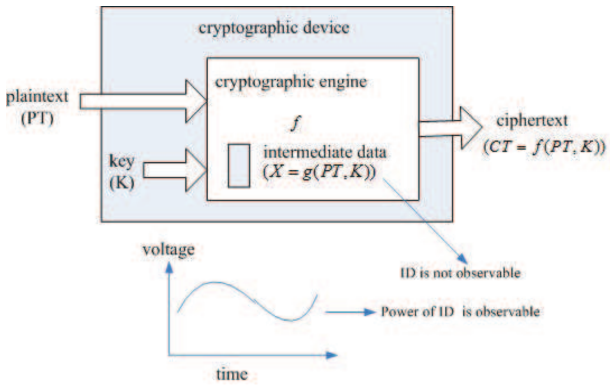

Fig. 1: The basic idea of power analysis attacks:to reveal secret information from a cryptographic device.

2 Preliminaries

This section introduces some preliminary knowledge on power analysis attacks and a typical hiding technique, WDDL logic. The ideal of our proposed countermeasure is just from the WDDL logic.

2.1 Power Analysis Attacks

Fig.1 shows the basic ideal of power analysis attacks. The cryptographic device implements a cryptographic algorithm, represented by f. Algorithm f takes the plaintext (PT) and the key (K) as inputs, and generates the encryption result called cipher text (CT) ($CT = f(PT, K)$). The internal secret key K is not directly observable through the ports of the device. The objective of power analysis attack is to reveal the value of K.

Power analysis attacks recover the whole key part by part in the following way. There are always intermediate values that are only related to a small part of K. Assume that an intermediate data X depends on a single key byte $K[7:0]$ and the plaintext PT. Then, $K[7:0]$ can be discovered with only 2^8 guesses.

To check which guessed key is correct, ideal power consumed is calculated according to intermediate value X. The variable, X, is indirectly observed through its power dissipation, which is a part of the power dissipated by the entire device. Through proper correlation techniques, the overall power dissipation from the device can be used in place of the power dissipation from intermediate data X. Power dissipated by unrelated components can be treated as noise. In such way, power analysis attack successfully obtains the information of the internal states and finally attacks the device.

The remainder of this part, we will give procedures of power analysis attacks in Fig.1. In the first step, the attacker chooses an intermediate m bit data X that is physically generated within the cryptographic circuit under attack. In most attack scenarios, signal X depends on both the input PT and the secret key K of the cryptographic algorithm according to a well-defined function g in cryptographic algorithm

$$X = g(PT, K) \tag{1}$$

where g is set by the algorithm and, hence is known by the adversary.

In the second step, the attacker inputs M different input values pt_i (with $i = 1, 2 \ldots M$) and measures the corresponding power p_i of the cryptographic device while it encrypts or decrypts different data blocks. We write input known data values as vector $PT = (pt_1, \ldots, pt_M)^T$. Measured power p_i has T sample points, so p_i is also a vector denoted as $p_i = (p_{i,1}, \ldots, p_{i,T})$. The attacker measures a trace for each of input data, and hence, the traces can be written as matrix P of size M x T.

In the third step, hypothetical intermediated data is calculated according to equation (1). Since the generic input pt_i is applied by the adversary, the only unknown variable in equation(1) is the secret key K. We write these possible choices as vector $K = (k_1, \ldots, k_N)$, where N denotes the total number of possible choices. This calculation equation (2) results in a matrix X of size M x N, which stands for intermediate value.

$$X_{i,j} = f(pt_i, k_j) \; i = 1, \ldots, M \; j = 1, \ldots, N \tag{2}$$

In the fourth step, intermediate values are mapped to power consumption values. In this step, attacker map the hypothetical intermediate values X to a matrix H of hypothetical power consumption values. H is a matrix with size of M x N.

In the fifth step, the measured power P and the estimated power H are compared. Each column h_i of the matrix H is compared with each column p_j of the matrix P. This means that the attacker compares the hypothetical power consumption values of each key hypothesis with the recorded traces at every position. The result of this comparison is a matrix R of size N x T, which each element $r_{i,j}$ contains the result of the comparison between the columns p_j and h_i. The comparison is done base on correlation efficient calculated according to equation (3).

$$r_{i,j} = \frac{\sum_{i=1}^{M}(h_{d,j} - \overline{h_i}).(p_{d,j} - \overline{p_j})}{\sqrt{\sum_{i=1}^{M}(h_{d,j} - \overline{h^i})^2.(p_{d,j} - \overline{p_j})^2}} \tag{3}$$

The highest correlation coefficient of the matrix R reveal the positions at which the chosen intermediate data has been processed and the key that is used by the device. In equation (3) $\overline{h_i}$ and $\overline{p_j}$ is calculated as following equations (4) and (5).

$$\overline{h_i} = \frac{1}{M-1}\sum_{i=1}^{M} h_{d,i} \tag{4}$$

$$\overline{p_j} = \frac{1}{M-1}\sum_{i=1}^{M} p_{d,j} \tag{5}$$

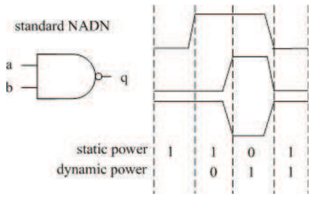

Fig. 2: CMOS standard NAND has data-dependent power dissipation.

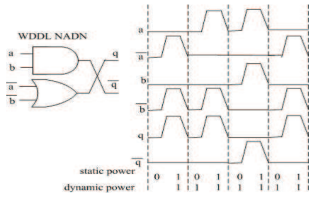

Fig. 3: WDDL NAND gate has data-independent power dissipation.

2.2 The WDDL Technique

WDDL logic is a typical logic of differential and pre-charge technique which is used for cryptographic devices to make the power consumption of the logic cells in the device constant in each clock cycle. It is an effective countermeasure against SCA. Its basic idea is to reduce the dependency of power consumption and intermediate value. Hardware circuits implemented as WDDL have a constant power dissipation and electromagnetic emanation.

Fig. 2 illustrates the operations of a standard NAND gate and WDDL NAND gate. This example approximates static and dynamic power dissipation of a logic gate through the hamming weight and hamming distance of its output, respectively. In the case of a single NAND gate in Fig. 2, the static and dynamic power dissipation depend on the input values of the gate. For example, if the static power is 0, both inputs must be 1. This side-channel leakage is critical to SCA.

Fig. 3 shows the same test case on a WDDL NAND gate. In this case, the circuit encodes each logic value with a complementary pair. Furthermore, each pair is pre-charged to (0,0) in each clock cycle before evaluation. As a result, each clock cycle, every WDDL signal pair shows exactly one transition from 0 to 1 and another one from 1 to 0. The resulting static and dynamic power dissipation are independent of the input values of the WDDL gate.

So far, WDDL technique has been broadly used to protect hardware circuits. But WDDL logic is implemented in transistor level. No commercial EDA tool is available for integrated circuit design. So we proposed CAES decryption algorithm as a complementary counterpart circuit to AES decryption circuit to resist

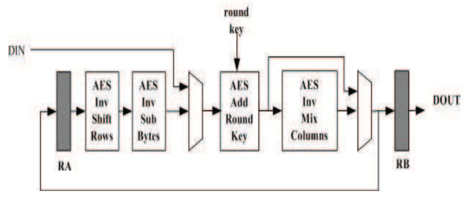

Fig. 4: Architecture of the AES decryption engine.

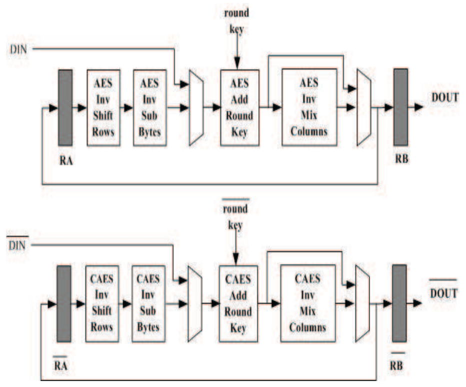

Fig. 5: Our proposed countermeasure based on commentary CAES decryption engine.

power analysis attack by emulating the ideal of WDDL. CASE decryption engine always execute opposite operations to AES decryption engine. This characteristic can make power consumption constant in every clock cycle.

3 Using CAES decryption algorithm to resist power analysis attack

This section discusses the concept of our proposed countermeasure CAES decryption algorithm, and then we build a model to evaluate how resistant to power analysis attack the hardware circuit is. And then, we give the detail CAES decryption algorithm and analyze its resistance to power analysis attack.

3.1 Concept

While CAES decryption engine protects AES decryption engine, its ultimate objective is to reduce the side-channel leakage originating from AES decryption engine.

The implementation of AES decryption engine in hardware has different parts which are potential sources of side-channel leakage. Fig. 4 shows the architecture of AES decryption engine implemented in hardware. There are four transformations: InvSubBytes, InvShiftRows, InvMixColumns and AddRoundKey. These transformations lead to data-dependent power dissipation which must be avoided. InvSubBytes transformation in AES is most vulnerable to power analysis attack, because power consumed by InvSubByes transformation attributes to the most part of the whole power consumed by AES decrypiton engine. So InvSubBytes is a main transformation that should be carefully considered.

The concept of CAES is to protect all of these four transformations with complementary transformation. Fig. 5 is our proposed countermeasure to power analysis attack. CAES decryption engine, as a counterpart of AES decryption engine, provides complementary power which makes the whole power consumed by AES and CAES constant at any time.

To implement the above concept, this work proposes CAES decryption algorithm which has four complementary transformations accordingly to AES decryption algorithm transformations. For example, when output of AES S-Box is 0x5a, the output of CAES S-box must be 0xa5 at the same time. It promises the same hamming weight at all the time. From Fig. 4 and Fig. 5, initial data of CAES is bitwise complementary to the initial data of AES. And even more, very intermediate value of CAES is bitwise complementary to the intermediate value of AES accordingly, because very transformations of CAES is complementary to the AES. This can guarantee the same hamming weight of all intermediate value. This is the basic ideal of our proposed countermeasure.

CAES decryption algorithm can guarantee that very transformation of AES decryption algorithm has a complementary transformation in CAES decryption algorithm. Equation (6) to equation (9) can easily be get.

$$AES\_InvSubBytes(x) = \overline{CAES\_InvSubBytes(\overline{x})} \tag{6}$$

$$AES\_InvShiftRows(x) = \overline{CAES\_InvShiftRows(\overline{x})} \tag{7}$$

$$AES\_InvMixColumns(x) = \overline{CAES\_InvMixColumns(\overline{x})} \tag{8}$$

$$AES\_AddRoundKey(x) = \overline{CAES\_AddRoundKey(\overline{x})} \tag{9}$$

Theorem 1. Let symbol $\wedge$ denote as bitwise exclusive OR, and $\sim\wedge$ as bitwise exclusive NOR. Symbol $\wedge$ and $\sim\wedge$ are complementary logic operation. The complementary operation is given by

$$a \wedge b = \overline{\overline{a} \sim\wedge \overline{b}} \tag{10}$$

This can easily be proved. Left part of equation (10) can be written

$$a \wedge b = (\overline{a}\&b)|(a\&\overline{b}) \tag{11}$$

where symbol & is represented as bitwise AND, and symbol | as bitwise OR. Right part of equation (10) can

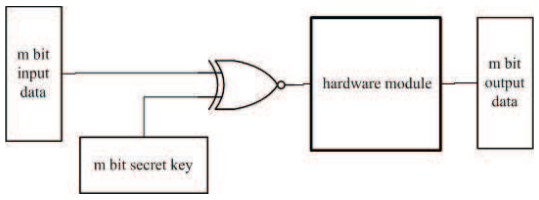

Fig. 6: Model to evaluate resistance to power analysis attack.

be rewritten in the following

$$\begin{aligned}\overline{\overline{a} \sim \wedge \overline{b}} &= \overline{(\overline{a}\&\overline{b})|(a\&b)} \\ &= \overline{\overline{a}\&\overline{b}}\ \&\ \overline{a\&b} \\ &= (a \mid b)\ \&\ (\overline{a} \mid \overline{b}) \\ &= (\overline{a}\ \&\ b) \mid (a\ \&\ \overline{b})\end{aligned} \quad (12)$$

From equation (11) and equation (12), we can easily get that equation (10) is true. Then, bitwise XOR operation and bitwise XNOR operation are complementary logic operations.

Theorem 2. Bitwise AND operation and bitwise OR operation are complementary logic operations. The equation (13) is given

$$a\ \&\ b = \overline{\overline{a} \mid \overline{b}} \quad (13)$$

Equation (13) is very easily to prove.

3.2 Model to Evaluate Resistance to Power Analysis Attack

Fig. (6) shows a way to evaluate resistance to power analysis attack. *M* bit input data, exclusive OR with *m* bit secret key, are input to hardware module which is just being evaluated. For the sake of illustration, we tie secret key to 0x2b. Attacker applies all possible input data patters to circuit in Fig. (6). If the attacker can recover the secret key, we can assert that the hardware module in Fig. (6) cannot thwart power analysis attack. And if the attacker cannot retrieve the secret key, it is apparent that the hardware module can resist power analysis attack. In this work, we just used this model to evaluate the resistance of our proposed countermeasure to power analysis attack.

3.3 CAES Decryption Algorithm

3.3.1 InvSubBytes

AES InvSubBytes transformation is a non-linear byte substitution that operates independently on each byte of the input data using AES inverse S-Box. AES inverse S-Box is constructed by composing two transformations:

1. Apply the following affine transformation which is expressed in matrix form:

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 0&0&1&0&0&1&0&1 \\ 1&0&0&1&0&0&1&0 \\ 0&1&0&0&1&0&0&1 \\ 1&0&1&0&0&1&0&0 \\ 0&1&0&1&0&0&1&0 \\ 0&0&1&0&1&0&0&1 \\ 1&0&0&1&0&1&0&0 \\ 0&1&0&0&1&0&1&0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} b_2 \wedge b_5 \wedge b_7 \wedge 1 \\ b_0 \wedge b_3 \wedge b_6 \wedge 0 \\ b_1 \wedge b_4 \wedge b_7 \wedge 1 \\ b_0 \wedge b_2 \wedge b_5 \wedge 0 \\ b_1 \wedge b_3 \wedge b_6 \wedge 0 \\ b_2 \wedge b_4 \wedge b_7 \wedge 0 \\ b_0 \wedge b_3 \wedge b_5 \wedge 0 \\ b_1 \wedge b_4 \wedge b_6 \wedge 0 \end{bmatrix} \quad (14)$$

where b_i is the i^{th} bit of the input byte, b'_i is the i^{th} bit of the output byte.

2. Take the multiplicative inverse in the finite field GF (2^8), the element 00 is mapped to itself.

Calculating the multiplicative inverse of a byte in the finite field GF (2^8) is much more difficult. Related work in [21] provided an efficient approach which is used to calculate the inverse of a byte in the finite field GF (2^8). The approach in work [21] is shown In Fig. 7, calculating the multiplicative inverse in the finite field (2^8) is converted to compute in a composite fields GF $((2^4)^2)$. There are three steps needed to calculate the multiplicative inverse.

Step 2.1. Convert a data from the finite field GF (2^8) to composite fields GF $((2^4)^2)$.

To implement this, an isomorphism function *T* is needed. According to the work in [21], the isomorphism

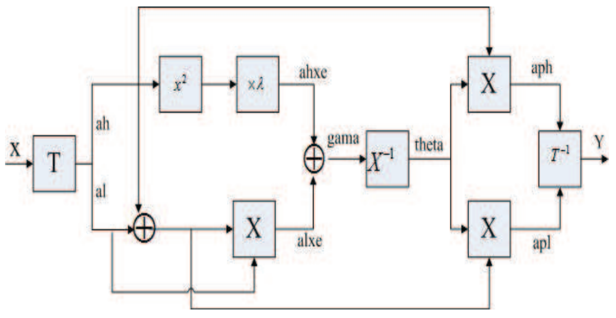

Fig. 7: Multiplicative inversion in GF $((2^4)^2)$

function T is given in the following:

$$\{ah,al\} = \begin{bmatrix} al_0 \\ al_1 \\ al_2 \\ al_3 \\ ah_0 \\ ah_1 \\ ah_2 \\ ah_3 \end{bmatrix} = TX = \begin{bmatrix} 1&0&0&0&1&1&1&0 \\ 0&1&1&0&0&0&0&0 \\ 0&1&0&0&0&0&0&1 \\ 0&0&1&0&1&0&0&0 \\ 0&0&0&0&1&1&1&0 \\ 0&1&0&0&1&0&1&1 \\ 0&0&1&1&0&1&0&1 \\ 0&0&0&0&0&1&0&1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix}$$
$$= \begin{bmatrix} x_0 \wedge x_4 \wedge x_5 \wedge x_6 \\ x_1 \wedge x_2 \\ x_1 \wedge x_7 \\ x_2 \wedge x_4 \\ x_4 \wedge x_5 \wedge x_6 \\ x_1 \wedge x_4 \wedge x_6 \wedge x_7 \\ x_2 \wedge x_3 \wedge x_5 \wedge x_7 \\ x_5 \wedge x_7 \end{bmatrix} \tag{15}$$

where ah and al are four bit element in GF $((2^4)^2)$, ah_i is the i^{th} bit of ah, al_i is i^{th} bit of al; X is an element in (2^8), x_i is the i^{th} bit of X.

Step 2.2. Calculate in composite fields GF $((2^4)^2)$. In Fig. 7, computations are expressed in the following equations.

$$ahxe = ah^2 \bullet \lambda \tag{16}$$
$$alxe = (ah \,\Lambda\, al) \bullet al \tag{17}$$
$$gama = ahxe \,\Lambda\, alxe \tag{18}$$
$$theta = gama^{-1} \tag{19}$$
$$aph = theta \bullet ah \tag{20}$$
$$apl = theta \bullet (ah \,\Lambda al) \tag{21}$$

where λ is a constant element in GF$((2^4)^2)$, the value of λ is 0xe. Symbol $\bullet$ denoted as multiplication in GF $((2^4)^2)$.

From equation (16) to equation (21), it is clear that there are four operations: x^2 (squaring), $\times\lambda$ (multiplication by a constant), $\times$ (multiplication) and X^{-1} (multiplicative inverse) in GF (2^4). We denote two polynomials, $a(x)$ and $b(x)$ in GF (2^4) as following.

$$\begin{aligned} a(x) &= a_3^3 x_3^3 + a_2^2 x_2^2 + a_1 x_1 + a_0 \\ b(x) &= b_3^3 x_3^3 + b_2^2 x_2^2 + b_1 x_1 + b_0 \end{aligned} \tag{22}$$

The multiplication of two-term polynomials involves multiplication of elements in GF (2^4) which requires an irreducible polynomial of degree 4 which is given by

$$m(x) = x^4 + x + 1 \tag{23}$$

Multiplication in GF (2^4) is given by

$$\begin{aligned} &c(x) = a(x) \times b(x) mod m(x) \\ &\quad a_A = a_0 \wedge a_3, a_B = a_2 \wedge a_3, a_C = a_1 \wedge a_2 \\ &\quad c_0 = (a_0 \,\&\, b_0) \wedge (a_3 \,\&\, b_1) \wedge (a_2 \,\&\, b_2) \wedge (a_1 \,\&\, b_3) \\ &\quad c_1 = (a_1 \,\&\, b_0) \wedge (a_A \,\&\, b_1) \wedge (a_B \,\&\, b_2) \wedge (a_C \,\&\, b_3) \\ &\quad c_2 = (a_2 \,\&\, b_0) \wedge (a_1 \,\&\, b_1) \wedge (a_A \,\&\, b_2) \wedge (a_B \,\&\, b_3) \\ &\quad c_3 = (a_3 \,\&\, b_0) \wedge (a_2 \,\&\, b_1) \wedge (a_1 \,\&\, b_2) \wedge (a_A \,\&\, b_3) \end{aligned} \tag{24}$$

Squaring in GF (2^4) is a special case of multiplication and is given by

$$\begin{aligned} &c(x) = a(x)^2 mod m(x) \\ &\quad c_0 = a_0 \wedge a_2 \\ &\quad c_1 = a_2 \\ &\quad c_2 = a_1 \wedge a_2 \\ &\quad c_3 = a_3 \end{aligned} \tag{25}$$

The inverse of an element $a(x)$ in GF (2^4) is given by

$$\begin{aligned} &c(x) = a(x)^{-1} mod m(x) \\ &\quad a_A = a_0 \wedge a_1, a_B = a_2 \wedge a_3 \\ &\quad c_0 = a_A \wedge a_B \wedge (a_0 \,\&\, a_2 \,\&\, a_3) \wedge a_2 \,\&\, (a_0 | a_1) \\ &\quad c_1 = (a_A \,\&\, a_2) \wedge a_3 \wedge a_1 \,\&\, (a_0 | a_3) \\ &\quad c_2 = a_0 \,\&\, a_1 \wedge a_B \wedge a_0 \,\&\, (a_2 | a_3) \\ &\quad c_3 = a_1 \wedge a_B \wedge (a_0 \,\&\, a_3) \wedge a_3 \,\&\, (a_1 | a_2) \end{aligned} \tag{26}$$

Multiplication by $\wedge$ in GF (2^4) is given by

$$\begin{aligned} &c(x) = \lambda a(x) mod m(x) \\ &\quad c_0 = a_1 \wedge a_2 \wedge a_3 \\ &\quad c_1 = a_0 \wedge a_1 \\ &\quad c_2 = a_0 \wedge a_1 \wedge a_2 \\ &\quad c_3 = a_0 \wedge a_1 \wedge a_2 \wedge a_3 \end{aligned} \tag{27}$$

Step 2.3. Apply isomorphism function T^{-1} to convert an element from GF $((2^4)^2)$ to GF (2^8).

To implement this, an isomorphism function T^{-1} which is the inverse isomorphism function T is given in the following:

$$Y = T^{-1}\{aph, apl\} = \begin{bmatrix} 1&0&0&0&1&0&0&0 \\ 0&0&0&0&1&1&0&1 \\ 0&1&0&0&1&1&0&1 \\ 0&1&0&0&1&1&1&0 \\ 0&1&0&0&1&1&0&1 \\ 0&0&1&0&1&1&0&0 \\ 0&1&1&1&1&0&0&1 \\ 0&0&1&0&1&1&0&1 \end{bmatrix} \begin{bmatrix} apl_0 \\ apl_1 \\ apl_2 \\ apl_3 \\ aph_0 \\ aph_1 \\ aph_2 \\ aph_3 \end{bmatrix}$$
$$= \begin{bmatrix} apl_0 \wedge aph0 \\ aph_0 \wedge aph1 \wedge aph_3 \\ apl_1 \wedge aph0 \wedge aph_1 \wedge aph_3 \\ apl_1 \wedge aph0 \wedge aph_1 \wedge aph_2 \\ apl_1 \wedge aph0 \wedge aph_1 \wedge aph_3 \\ apl_2 \wedge aph0 \wedge aph_1 \\ apl_1 \wedge apl2 \wedge apl_3 \wedge aph_0 \wedge aph_3 \\ apl_2 \wedge aph0 \wedge aph_1 \wedge aph_3 \end{bmatrix} \tag{28}$$

The transformation of CAES InvSubBytes is complementary to AES InvSubBytes absolutely. Input and output of CAES inverse S-Box must be bitwise complementary to AES inverse S-Box. To accomplish this goal, every basic logic unit of CAES and AES must be bitwise complementary to each other. CAES inverse

S-Box is constructed by composing two transformations in the following.

1. Apply CAES affine transformation which is expressed in the following equation:

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 0&0&1&0&0&1&0&1 \\ 1&0&0&1&0&0&1&0 \\ 0&1&0&0&1&0&0&1 \\ 1&0&1&0&0&1&0&0 \\ 0&1&0&1&0&0&1&0 \\ 0&0&1&0&1&0&0&1 \\ 1&0&0&1&0&1&0&0 \\ 0&1&0&0&1&0&1&0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$
$$= \begin{bmatrix} b_2 \sim\wedge b_5 \sim\wedge b_7 \sim\wedge 1 \\ b_0 \sim\wedge b_3 \sim\wedge b_6 \sim\wedge 0 \\ b_1 \sim\wedge b_4 \sim\wedge b_7 \sim\wedge 1 \\ b_0 \sim\wedge b_2 \sim\wedge b_5 \sim\wedge 0 \\ b_1 \sim\wedge b_3 \sim\wedge b_6 \sim\wedge 0 \\ b_2 \sim\wedge b_4 \sim\wedge b_7 \sim\wedge 0 \\ b_0 \sim\wedge b_3 \sim\wedge b_5 \sim\wedge 0 \\ b_1 \sim\wedge b_4 \sim\wedge b_6 \sim\wedge 0 \end{bmatrix} \tag{29}$$

where b_i is the i^{th} bit of the input byte, b_i' is the i^{th} bit of the output byte. Basic operation unit of affine transformation of AES is bitwise XOR in equation (14). According to Lemma 1, it is apparent that outputs of equation (14) and equation 29) are bitwise complementary to each other when inputs of them are bitwise complementary to each other.

2. Take the multiplicative inverse in the finite field GF (2^8), the element FF is mapped to itself.

Calculating multiplicative inverse of CAES also has three steps which are complementary to multiplicative inverse of AES.

Step 2.1. Convert a data from the finite field GF (2^8) to composite fields GF $((2^4)^2)$. The transformation is given by:

$$\{ah, al\} = \begin{bmatrix} al_0 \\ al_1 \\ al_2 \\ al_3 \\ ah_0 \\ ah_1 \\ ah_2 \\ ah_3 \end{bmatrix} = TX = \begin{bmatrix} 1&0&0&0&1&1&1&0 \\ 0&1&1&0&0&0&0&0 \\ 0&1&0&0&0&0&0&1 \\ 0&0&1&0&1&0&0&0 \\ 0&0&0&0&1&1&1&0 \\ 0&1&0&0&1&0&1&1 \\ 0&0&1&1&0&1&0&1 \\ 0&0&0&0&0&1&0&1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix}$$
$$= \begin{bmatrix} x_0 \sim\wedge x_4 \sim\wedge x_5 \sim\wedge x_6 \\ x_1 \sim\wedge x_2 \\ x_1 \sim\wedge x_7 \\ x_2 \sim\wedge x_4 \\ x_4 \sim\wedge x_5 \sim\wedge x_6 \\ x_1 \sim\wedge x_4 \sim\wedge x_6 \sim\wedge x_7 \\ x_2 \sim\wedge x_3 \sim\wedge x_5 \sim\wedge x_7 \\ x_5 \sim\wedge x_7 \end{bmatrix} \tag{30}$$

According to Lemma 1, it is apparent that outputs of equation (15) and equation (30) are bitwise complementary to each other when inputs of them are bitwise complementary to each other.

Step 2.2. Calculate in composite fields GF $((2^4)^2)$. Computations of CAES in GF $((2^4)^2)$ have the same structure as AES. Computations of CAES in GF $((2^4)^2)$ is also according to equation (16) to equation (20). But main operation: squaring, multiplication by a constant, multiplication and multiplicative inverse, are complementary to AES.

Multiplication of CAES in GF (2^4) is given by

$$\begin{aligned} & c(x) = a(x) \times b(x) mod m(x) \\ & a_A = a_0 \sim\wedge a_3, a_B = a_2 \sim\wedge a_3, a_C = a_1 \sim\wedge a_2 \\ & c_0 = (a_0|b_0) \sim\wedge (a_3|b_1) \sim\wedge (a_2|b_2) \sim\wedge (a_1|b_3) \\ & c_1 = (a_1|b_0) \sim\wedge (a_A|b_1) \sim\wedge (a_B|b_2) \sim\wedge (a_C|b_3) \\ & c_2 = (a_2|b_0) \sim\wedge (a_1|b_1) \sim\wedge (a_A|b_2) \sim\wedge (a_B|b_3) \\ & c_3 = (a_3|b_0) \sim\wedge (a_2|b_1) \sim\wedge (a_1|b_2) \sim\wedge (a_A|b_3) \end{aligned} \tag{31}$$

Equation (31) is modified from equation (24), we modified XOR to XNOR, and AND to OR. According to Lemma 1 and Lemma 2, it is apparent that outputs of equation (24) and equation (31) are bitwise complementary to each other when inputs of them are bitwise complementary to each other.

Squaring of CAES in GF (2^4) is a special case of multiplication and is given by

$$\begin{aligned} & c(x) = a(x)^2 mod m(x) \\ & c_0 = a_0 \sim\wedge a_2 \\ & c_1 = a_2 \\ & c_2 = a_1 \sim\wedge a_2 \\ & c_3 = a_3 \end{aligned} \tag{32}$$

Equation (32) is modified from equation (25), we modified XOR to XNOR. According to Lemma 1, it is apparent that outputs of equation (25) and equation (32) are bitwise complementary to each other when inputs of them are bitwise complementary to each other.

The multiplicative inverse of CAES in GF (2^4) is given by

$$\begin{aligned} & c(x) = a(x)^{-1} mod m(x) \\ & a_A = a_0 \sim\wedge a_1, a_B = a_2 \sim\wedge a_3 \\ & c_0 = a_A \sim\wedge a_B \sim\wedge (a_0|a_2|a_3) \sim\wedge (a_2|(a_0 \& a_1)) \\ & c_1 = (a_A|a_2) \sim\wedge a_3 \sim\wedge (a_1|(a_0 \& a_3)) \\ & c_2 = (a_0|a_1) \sim\wedge a_B \sim\wedge (a_0|(a_2 \& a_3)) \\ & c_3 = a_1 \sim\wedge a_B \sim\wedge (a_0|a_3) \sim\wedge (a_3|(a_1 \& a_2)) \end{aligned} \tag{33}$$

Equation (33) is modified from equation (26), we modified XOR to XNOR, and AND to OR. According to Lemma 1 and Lemma 2, it is apparent that outputs of equation (26) and equation (33) are bitwise complementary to each other when inputs of them are bitwise complementary to each other.

Multiplication by λ of CAES in GF (2^4) is given by

$$\begin{aligned} c(x) &= \lambda a(x) mod m(x) \\ c_0 &= a_1 \sim \wedge a_2 \sim \wedge a_3 \\ c_1 &= a_0 \sim \wedge a_1 \\ c_2 &= a_0 \sim \wedge a_1 \sim \wedge a_2 \\ c_3 &= a_0 \sim \wedge a_1 \sim \wedge a_2 \sim \wedge a_3 \end{aligned} \tag{34}$$

Equation (34) is modified from equation (27), we modified XOR to XNOR. According to Lemma 1, it is apparent that outputs of equation (27) and equation (34) are bitwise complementary to each other when inputs of them are bitwise complementary to each other.

Step 2.3. Apply isomorphism function T^{-1} to convert an element from GF $((2^4)^2)$ to GF (2^8).The transformation is expressed by

$$\begin{aligned} Y = T^{-1}\{aph, apl\} &= \begin{bmatrix} 1&0&0&0&1&0&0&0 \\ 0&0&0&0&1&1&0&1 \\ 0&1&0&0&1&1&0&1 \\ 0&1&0&0&1&1&1&0 \\ 0&1&0&0&1&1&0&1 \\ 0&0&1&0&1&1&0&0 \\ 0&1&1&1&1&0&0&1 \\ 0&0&1&0&1&1&0&1 \end{bmatrix} \begin{bmatrix} apl_0 \\ apl_1 \\ apl_2 \\ apl_3 \\ aph_0 \\ aph_1 \\ aph_2 \\ aph_3 \end{bmatrix} \\ &= \begin{bmatrix} apl_0 \sim \wedge aph0 \\ aph_0 \sim \wedge aph1 \sim \wedge aph_3 \\ apl_1 \sim \wedge aph0 \sim \wedge aph_1 \sim \wedge aph_3 \\ apl_1 \sim \wedge aph0 \sim \wedge aph_1 \sim \wedge aph_2 \\ apl_1 \sim \wedge aph0 \sim \wedge aph_1 \sim \wedge aph_3 \\ apl_2 \sim \wedge aph0 \sim \wedge aph_1 \\ apl_1 \sim \wedge apl2 \sim \wedge apl_3 \sim \wedge aph_0 \sim \wedge aph_3 \\ apl_2 \sim \wedge aph0 \sim \wedge aph_1 \sim \wedge aph_3 \end{bmatrix} \end{aligned} \tag{35}$$

Equation (35) is modified from equation (28), we modified XOR to XNOR. According to Lemma 1, it is apparent that outputs of equation (28) and equation (35) are bitwise complementary to each other when inputs of them are bitwise complementary to each other.

Inverse S-Box of CAES is constructed in the same way with Inverse S-Box of AES according the equations above. Regarding every logic operation units are always the complementary between CAES and AES, hamming weight of every intermediate value are always in a constant value.

3.3.2 InvShiftRows

Intermediate cipher result can be pictured as a rectangular array of bytes having four rows and four columns called state. In the AES InvShiftRows transformation, the bytes in the last three rows of the state are cyclically shifted over different number of bytes. The AES InvShiftRows transformation is very simply hardwired as no logic involved.

CAES InvShiftRows has the same transformation with AES InvShiftRows. Inputs and outputs of CASE and AES InvShiftRows are always complementary with each other when inputs of them are bitwise complementary to each other, because the transformation is just cyclically shifted.

3.3.3 InvMixColumns

AES InvMixColumns operates on the state column by column, treating each column as a four-term polynomial. The transformation can described in the matrix which is given by

$$\begin{bmatrix} S^{`}_{0,c} \\ S^{`}_{1,c} \\ S^{`}_{2,c} \\ S^{`}_{3,c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix} \tag{36}$$

In order to simplify the calculation, equation (36) can be rewritten as

$$\begin{bmatrix} S^{`}_{0,c} \\ S^{`}_{1,c} \\ S^{`}_{2,c} \\ S^{`}_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} 05 & 00 & 04 & 00 \\ 00 & 05 & 00 & 04 \\ 04 & 00 & 05 & 00 \\ 00 & 04 & 00 & 05 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix} \tag{37}$$

We denote $\begin{bmatrix} T_{0,c} & T_{1,c} & T_{2,c} & T_{3,c} \end{bmatrix}^T$ as

$$\begin{bmatrix} T_{0,c} \\ T_{1,c} \\ T_{2,c} \\ T_{3,c} \end{bmatrix} = \begin{bmatrix} 05 & 00 & 04 & 00 \\ 00 & 05 & 00 & 04 \\ 04 & 00 & 05 & 00 \\ 00 & 04 & 00 & 05 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix} \tag{38}$$

Equation (37) can be rewritten in the following

$$\begin{bmatrix} S^{`}_{0,c} \\ S^{`}_{1,c} \\ S^{`}_{2,c} \\ S^{`}_{3,c} \end{bmatrix} = \begin{bmatrix} (\{02\} \bullet (T_{0,c} \wedge T_{1,c})) \wedge (T_{2,c} \wedge T_{3,c}) \wedge T_{0,c} \\ (\{02\} \bullet (T_{1,c} \wedge T_{2,c})) \wedge (T_{3,c} \wedge T_{0,c}) \wedge T_{2,c} \\ (\{02\} \bullet (T_{2,c} \wedge T_{3,c})) \wedge (T_{0,c} \wedge T_{1,c}) \wedge T_{3,c} \\ (\{02\} \bullet (T_{3,c} \wedge T_{1,c})) \wedge (T_{1,c} \wedge T_{2,c}) \wedge T_{0,c} \end{bmatrix} \tag{39}$$

Where $02 \bullet T_{i,c}$ is a multiplication $\{b^{`}_7 b^{`}_6 b^{`}_5 b^{`}_4 b^{`}_3 b^{`}_2 b^{`}_1 b^{`}_0\} = 02 \bullet \{b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0\}$ in GF (2^8). This multiplication which is denoted XTIME in GF (2^8), can be represent in the following equation

$$\begin{bmatrix} b^{`}_0 \\ b^{`}_1 \\ b^{`}_2 \\ b^{`}_3 \\ b^{`}_4 \\ b^{`}_5 \\ b^{`}_6 \\ b^{`}_7 \end{bmatrix} = \begin{bmatrix} b_7 \\ b_7 \wedge b_0 \\ b_1 \\ b_7 \wedge b_2 \\ b_7 \wedge b_3 \\ b_4 \\ b_5 \\ b_6 \end{bmatrix} \tag{40}$$

In order to get the result of equation (39), equation (38) must be simplified in the following

$$\begin{bmatrix} T_{0,c} \\ T_{1,c} \\ T_{2,c} \\ T_{3,c} \end{bmatrix} = \begin{bmatrix} 04 & 00 & 04 & 00 \\ 00 & 04 & 00 & 04 \\ 04 & 00 & 04 & 00 \\ 00 & 04 & 00 & 04 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix} + \begin{bmatrix} 01 & 00 & 01 & 00 \\ 00 & 01 & 00 & 01 \\ 01 & 00 & 01 & 00 \\ 00 & 01 & 00 & 01 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix} = \begin{bmatrix} (\{02\} \bullet (\{02\} \bullet (S_{0,c} \wedge S_{2,c}))) \wedge S_{0,c} \\ (\{02\} \bullet (\{02\} \bullet (S_{1,c} \wedge S_{3,c}))) \wedge S_{1,c} \\ (\{02\} \bullet (\{02\} \bullet (S_{0,c} \wedge S_{2,c}))) \wedge S_{3,c} \\ (\{02\} \bullet (\{02\} \bullet (S_{1,c} \wedge S_{3,c}))) \wedge S_{0,c} \end{bmatrix} \tag{41}$$

Through the equation (39), equation (40) and equation (41), we can easily get the result of equation (36) by the way mentioned above.

The transformation of CAES InvMixColumns is complementary to AES InvMixColumns absolutely. Input and output of CAES InvMixColumns must be bitwise complementary to AES InvMixColumns. To accomplish this goal, every basic logic unit of CAES and AES must be bitwise complementary to each other. CAES InvMixColumns transformation is designed by modified equation (39), equation (40) and equation (41).

CASE InvMixColumns can be described in the matrix which is given by

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} (\{02\} \otimes (T_{0,c} \sim \wedge T_{1,c})) \sim \wedge (T_{2,c} \sim \wedge T_{3,c}) \sim \wedge T_{0,c} \\ (\{02\} \otimes (T_{1,c} \sim \wedge T_{2,c})) \sim \wedge (T_{3,c} \sim \wedge T_{0,c}) \sim \wedge T_{2,c} \\ (\{02\} \otimes (T_{2,c} \sim \wedge T_{3,c})) \sim \wedge (T_{0,c} \sim \wedge T_{1,c}) \sim \wedge T_{3,c} \\ (\{02\} \otimes (T_{3,c} \sim \wedge T_{1,c})) \sim \wedge (T_{1,c} \sim \wedge T_{2,c}) \sim \wedge T_{0,c} \end{bmatrix} \tag{42}$$

Where $\otimes$ denotes a transformation which is complementary to XTIME operation. This transformation is defined by

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} b_7 \\ b_7 \sim \wedge b_0 \\ b_1 \\ b_7 \sim \wedge b_2 \\ b_7 \sim \wedge b_3 \\ b_4 \\ b_5 \\ b_6 \end{bmatrix} \tag{43}$$

From equation (40) and equation (43), outputs AES XTIME and CAES XTIME are absolutely bitwise complementary to each other when inputs of them are bitwise complementary to each other. And in equation (42) and equation (39), XOR and XNOR are complementary logic operation according to lemma 2. If $T_{i,c}$ in equation (42) and $T_{i,c}$ in equation (39) are complementary to each other, outputs of equation (42) and equation (39) are absolutely bitwise complementary to each other.

To achieve this goal, we modified equation (41) in the following way

$$\begin{bmatrix} T_{0,c} \\ T_{1,c} \\ T_{2,c} \\ T_{3,c} \end{bmatrix} = \begin{bmatrix} (\{02\} \otimes (\{02\} \otimes (S_{0,c} \sim \wedge S_{2,c}))) \sim \wedge S_{0,c} \\ (\{02\} \otimes (\{02\} \otimes (S_{1,c} \sim \wedge S_{3,c}))) \sim \wedge S_{1,c} \\ (\{02\} \otimes (\{02\} \otimes (S_{0,c} \sim \wedge S_{2,c}))) \sim \wedge S_{3,c} \\ (\{02\} \otimes (\{02\} \otimes (S_{1,c} \sim \wedge S_{3,c}))) \sim \wedge S_{0,c} \end{bmatrix} \tag{44}$$

It is apparent that outputs of equation (41) and equation (44) are complementary to each other. So outputs of equation (42) and equation (39) are complementary to each other. That is to say, CAES InvMixColumns and AES InvMixColumns are complementary transformations.

3.3.4 AddRoundKey

In the AddRoundKey transformation of AES, a Round key is added to the state by a simple bitwise XOR operation. AES AddRounKey can be given by

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix} = \begin{bmatrix} b_0 \wedge k_0 \\ b_1 \wedge k_1 \\ b_2 \wedge k_2 \\ b_3 \wedge k_3 \\ b_4 \wedge k_4 \\ b_5 \wedge k_5 \\ b_6 \wedge k_6 \\ b_7 \wedge k_7 \end{bmatrix} \tag{45}$$

where c_i is the i^{th} bit of the output byte of AddRoundKey, b_i is the i^{th} bit of the input byte, k_i is the i^{th} bit of the round key byte. The only logic operation in AES AddRoundKey is bitwise XOR, the complementary operation to bitwise XOR is bitwise XNOR. So AddRoundkey transformation of CAES is just simple bitwise XNOR operation by a roudkey.

The transformation of CAES is as follows

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix} = \begin{bmatrix} b_0 \sim \wedge k_0 \\ b_1 \sim \wedge k_1 \\ b_2 \sim \wedge k_2 \\ b_3 \sim \wedge k_3 \\ b_4 \sim \wedge k_4 \\ b_5 \sim \wedge k_5 \\ b_6 \sim \wedge k_6 \\ b_7 \sim \wedge k_7 \end{bmatrix} \tag{46}$$

4 Experiment Results

To verify function of CAES decryption algorithm and its resistance to power analysis attack, this section presents the experiment results, including logic function verification, simulation of power analysis attack and real-world attack in FPGA.

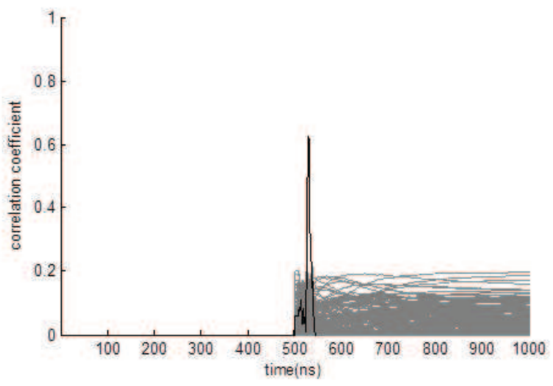

Fig. 8: Power analysis attack to AES InvSubBytes.

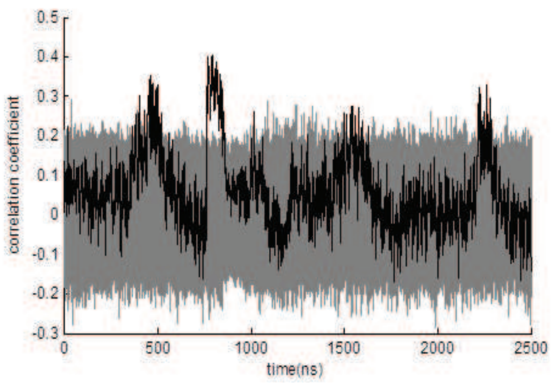

Fig. 10: Power analysis attack to AES decryption engine.

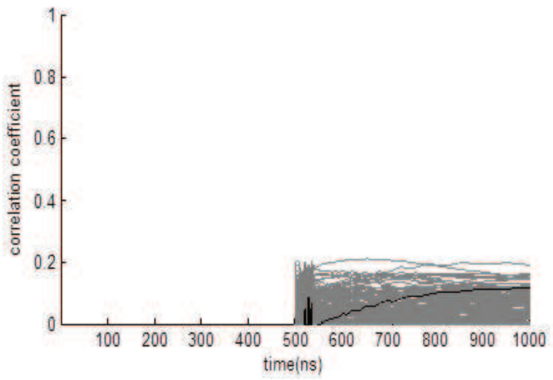

Fig. 9: Power analysis attack to AES InvSubBytes with CAES InvSubBytes

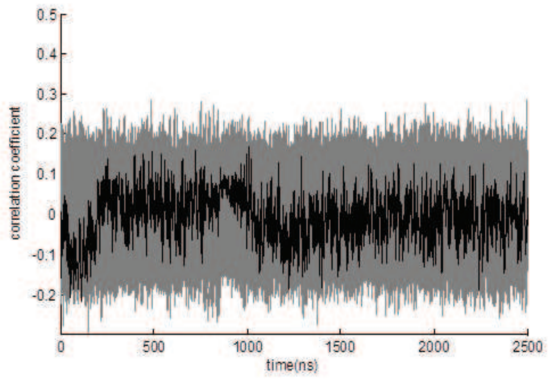

Fig. 11: Power analysis attack to AES decryption engine with counterpart CAES

4.1 Logic Function Verification

In this subpart, we verify the logic function of our proposed CASE decryption algorithm. To verify whether the outputs of very round of AES and CAES decryption are absolutely bitwise complementary is called logic function verification.

We implemented AES decryption engine and CAES decryption engine in register transfer level (RTL) respectively. The simulation results of two hardware validate that outputs of two engines are always bitwise complementary to each other, when the inputs of them are bitwise complementary to each other.

4.2 Simulation Power Analysis Attack

We use the model in Fig. (6) to evaluate power leakage of InvSubBytes transformation. In Fig. (8) and Fig. (8), black line means correlation coefficient of correct key, gray line means that of wrong keys. From Fig. (8), it is apparent that power analysis attack can easily recover the correct key when the power analysis attack applied to AES InvSubBytes operation without any countermeasure. While Fig. (9) illustrates that power analysis attack cannot retrieve the correct key when InvSubBytes transformation are implemented with our proposed CAES InvSubBytes.

4.3 Real-world Attack in FPGA

To verify our proposed countermeasure for power analysis attack, this subpart presents the results of real-word attacks in FPGA. We implemented the AES decryption engine without counterpart CAES circuit and AES decryption engine with our proposed CAES circuit in FPGA. The power analysis attack attacks are carried out in these two circuits to evaluate our scheme.

We choose inverse S-Box as the attacking target, because attacker often chooses it as attacking target. We applied all possible values of input data to inverse S-Box, the power is recorded and analyzed. In Fig. (10) and Fig. (11), black line represent correct 0x2b, gray lines represents other possible keys. It is apparent that power analysis attack to FPGA with our countermeasure cannot retrieve the correct key.

From the above analysis results, our proposed countermeasure circuit can resist power analysis attacks by emulating the ideal of WDDL. Thus, the correlation between power traces and hamming weights can be effectively broken to hide the correct key.

5 Conclusion

This paper proposes CAES decryption algorithm as a solution to protect AES decryption engine from power analysis with the ideal of WDDL technique. This concept can be easily implemented by hardware description language.

The analysis results of our proposed countermeasure circuit showed that countermeasure using the CAES can resist power analysis attack. Our proposed countermeasure is a promising way to thwart power analysis attack.

References

[1] M. Alioto, M. Poli, S. Rocchi, A General Power Model of Differential Power Analysis Attacks to Static Logic Circuits, Very Large Scale Integration (VLSI) Systems, IEEE Transactions **18**, 711-724, 2010.
[2] P. Kocher, J. Jaffe, B. Jun, Differential power analysis, in Advances in Cryptology **99**, 789-789, 1999.
[3] E. Brier, C. Clavier, F. Olivier, Correlation Power Analysis with a Leakage Model, in Cryptographic Hardware and Embedded Systems - CHES **2004**, 135-152, 2004.
[4] N. Avirneni, A. Somani, Countering Power Analysis Attacks using Reliable and Aggressive Designs, IEEE Transactions on Computers, 1-10, 2013.
[5] M. Bucci, L. Giancane, R. Luzzi, A. Trifiletti, A Flip-Flop for the DPA Resistant Three-Phase Dual-Rail Pre-Charge Logic Family, Very Large Scale Integration (VLSI) Systems, IEEE Transactions **20**, 2128-2132, 2012.
[6] T. Popp, S. Mangard, Implementation aspects of the DPA-resistant logic style MDPL, in Circuits and Systems, IEEE International Symposium on, 2910-2916, 2006.
[7] Z. Chen and Y. Zhou, Dual-rail random switching logic: a countermeasure to reduce side channel leakage, Cryptographic Hardware and Embedded Systems-CHES **2006**, 242-254, 2006.
[8] M. Bucci, L. Giancane, R. Luzzi, A. Trifiletti, Three-Phase Dual-Rail Pre-charge Logic, Cryptographic Hardware and Embedded Systems - CHES **2006**, 232-241, 2006.
[9] E. Trichina, L. Korkishko, Secure and efficient aes software implementation for smart cards, in Information Security Applications, 425-439, 2005.
[10] H. Saputra, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, R. Brooks, Masking the energy behaviour of encryption algorithms, Computers and Digital Techniques, IEE Proceedings **150**, 274-84, 2003.
[11] H. Saputra, N. Vijaykrishnan, et al., Masking the energy behavior of DES encryption [smart cards], Automation and Test in Europe Conference and Exhibition, 84-89, 2003.
[12] M. Yoshikawa, Y. Kojima, Efficient Random Number for the Masking Method against DPA Attacks, in Systems Engineering, 21st International Conference on, 321-324, 2001.
[13] E. Trichina, T. Korkishko, Secure AES hardware module for resource constrained devices, in Security in Ad-hoc and Sensor Networks, 215-229, 2005.
[14] K. Tiri, M. Akmal, I. Verbauwhede, A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards, in Solid-State Circuits Conference, ESSCIRC 2002, Proceedings of the 28th European, 403-406, 2002.
[15] K. Tiri, I. Verbauwhede, A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation, in Design, Automation and Test in Europe Conference and Exhibition **1**, 246-251, 2004.
[16] K. Tanimura, N. D. Dutt, HDRL: Homogeneous Dual-Rail Logic for DPA Attack Resistive Secure Circuit Design, Embedded Systems Letters **4**, 57-60, 2012.
[17] Y. Ishai, A. Sahai, D. Wagner, Private circuits: Securing hardware against probing attacks, Advances in Cryptology-CRYPTO **2003**, 463-481, 2003.
[18] T. Popp, S. Mangard, Masked Dual-Rail Pre-charge Logic: DPA-Resistance Without Routing Constraints, Cryptographic Hardware and Embedded Systems ? CHES 2005, **3659**, 172-186, 2005.
[19] S. Mangard, T. Popp, B. M. Gammel, Side-channel leakage of masked CMOS gates, Topics in Cryptology?CT-RSA, 351-365, 2005.
[20] S. Mangard, N. Pramstaller, E. Oswald, Successfully Attacking Masked AES Hardware Implementations, Cryptographic Hardware and Embedded Systems-CHES 2005, **3659**, 157-171, 2005.
[21] J. Wolkerstorfer, E. Oswald, M. Lamberger, An ASIC implementation of the AES SBoxes, Topics in Cryptology, 29-52, 2002.

Nianhao Zhu received the BS degree in electrical and information engineering from NanJing University of Information Science and Technology, China, the MS degree in electrical engineering form Shanghai Jiao Tong University, china.He is now a PhD student in Shanghai Jiao Tong University, China.His research interests are in the areas of computer architecture, applied cryptography and information security.

Yujie Zhou received her Ph.D in 1997. She engaged in postdoctoral research work from 1997 to 1999 in State Key Laboratory Of Information Security(SKLOIS). In 2000,she began to participate in cryptographic algorithm chips design, and as Shenzhen ZTE IC design company technical director, leading to design the first commercial cryptographic algorithm chip.In 2001, she was selected national”ten-five””863” plan expert of information field information security technology theme China’s e-government overall group member. Since 2003,she served as professor and doctoral in electrical engineering department,Shanghai JiaoTong university.She led the design of high-end information security SoC chip”SSX17 high-performance security processor chip”.

Hongmin Liu received the the MS degree in electrical engineering form Shanghai Jiao Tong University, china.He is now a PhD student in Shanghai Jiao Tong University, China.His research interests are in the areas of computer architecture, applied cryptography and information security.