Construction of High Quality Key-dependent S-boxes

Tianyong Ao, Jinli Rao, Kui Dai, and Xuecheng Zou

***Abstract*—High quality key-dependent S-boxes can break the preconditions of many cryptanalysis technologies, but it is difficult to construct them efficiently. Here, we proposed a method for fast constructing good key-dependent S-boxes, which are generated by means of a key-dependent affine transformation on a good base S-box. We proved their security by classifying the evaluative criteria of S-boxes into three categories: affine static, conditional affine static and affine dynamic criteria. We found that algebraic degree of an S-box may be decreased through an affine transformation and proved the condition for keeping it invariant under an affine transformation. In order to efficiently get affine key-dependent S-boxes, we presented three fast constructing algorithms for obtaining key-dependent nonsingular Boolean matrix, matrix multiplication of an S-box, and eliminating fixed points, respectively. The theoretical analysis and statistical results show that the key-dependent S-boxes generated by this method have good cryptographic properties.**

***Index Terms*—key-dependent S-Box; affine transformation; fast matrix multiplication; Block cipher;**

I. Introduction

BLOCK ciphers are widely used in the field of information security. With the development of cryptanalysis technology and computing power, improving block ciphers is always required. Substitution boxes (S-boxes) play a core role in block ciphers. The precondition of many cryptanalysis methods, such as differential cryptanalysis [1], linear cryptanalysis [2], and algebraic attacks [3], is that the characteristics of S-boxes can be known by attackers. An S-box is called a key-dependent S-box if its content depends on a secret key. If key-dependent S-boxes are used in ciphers, it is hard to take advantage of special properties of S-boxes for cryptanalysis, since attackers cannot know the content of a key-dependent S-box. Many studies have shown that good key-dependent S-boxes can enhance the security of block ciphers [4], [5], [6], and key-dependent S-boxes are used in many ciphers such as Khufu [7], Blowfish [4] and Twofish [8].

The previous methods of constructing key-dependent S-boxes can be summarized as the following three categories.

(1) Key-dependent S-boxes are created by a randomized permutation. For example, Khufu uses a pseudo-random function to generate an S-box from a user key. Blowfish uses iterations of its encryption function. K. Kazlauskas and J. Kazlauskas present a method that key-dependent S-boxes are generated by key pseudo-expansion words adding on a pseudo-permutation [9].

(2) Key-dependent S-boxes are produced by chaos system, e.g., Masuda et al. propose chaos block cipher with key-dependent S-boxes generated from modified skew tent map [10].

(3) Several good S-boxes are given in a cipher in advance, but which one will be used depending on the user key, e.g., Stoianov proposes one of four S-boxes being selected by an user key instead of only a fixed S-box in AES [11].

However, the previous methods have disadvantages. For example, weak S-boxes or weak keys maybe appear in the designs of the first two categories [12]. There will be a huge risk if weak key-dependent S-boxes are used in ciphers. Although weak S-boxes can be filtrated by the technologies of evaluating S-boxes, it will require too much time to evaluate them completely. The number of key-dependent S-boxes is limited in the last category. In addition, practical ciphers should be both high security and high performance. In a word, how to fleetly construct good key-dependent S-boxes is still a problem. In order to address this problem, a method of fast constructing good key-dependent S-boxes is proposed in this paper.

This paper makes the following major contributions.

(1) We gave the notion of affine key-dependent S-boxes and proposed a method for constructing high quality affine key-dependent S-boxes.

(2) We comprehensively investigated the effect of affine transformation on the evaluative criteria of an S-box, and classified the evaluative criteria of an S-box into three categories: affine static, conditional affine static and affine dynamic. We found that the algebraic degree of an S-box may be decreased under affine transformation, and proved the condition of keeping it invariant under affine transformations.

(3) Three fast algorithms were proposed for obtaining a key-dependent nonsingular Boolean matrix, doing matrix multiplication of an S-box, and eliminating fixed points, respectively. The fast algorithm for matrix multiplication of an S-box, which is based on Gray code, only required 2^m XOR operations rather than $m(m+2)*2^m$ operations in straightforward way. These algorithms not only can be used in fast constructing affine key-dependent S-boxes, but also can be used in other applications such as searching golden S-boxes.

The remainder of this paper is structured as follows. The following section introduces preliminaries of affine key-dependent S-boxes. The security of affine key-dependent S-boxes will be analyzed in section 3. In section 4, fast algorithms for constructing affine key-dependent S-boxes are

Manuscript received February 8, 2017; revised June 29, 2017. This work is supported by Science and Technology Project of Henan Province(152102210055).

Tianyong Ao is with the School of Physics and Electronics, Henan University, Kaifeng, China (e-mail: tyaohust@gmail.com).

Jinli Rao is with the School of Optical and Electronic Information, Huazhong University of Science and Technology, Wuhan, China (e-mail: ary.xsnow@gmail.com).

Kui Dai is with the School of Optical and Electronic Information, Huazhong University of Science and Technology, Wuhan, China (e-mail:daikui@mail.hust.edu.cn).

Xuecheng Zou is with the School of Optical and Electronic Information, Huazhong University of Science and Technology, Wuhan, China (e-mail:estxczou@gmail.com).

presented. In section 5, an instantiation for generating affine key-dependent S-boxes is given and discussed. Finally, the conclusions are drawn in section 6.

II. Preliminaries of Affine Key-dependent S-boxes

An S-box is the mapping $S : F_2^n \to F_2^m$. S can be represented by the vector $\{f_1, f_2, \cdots, f_m\}$, where $f_i : F_2^n \to F_2$ $(1 \le i \le m)$ represents a component function of the S.

The S-boxes can be called affine key-dependent S-boxes if they are produced by an S-box under a key-dependent affine transformations. An affine transformation consists of multiplication by a matrix followed by addition of a vector. This paper focuses on the key-dependent affine transformation on the output of a bijection S-box, i.e.

$$S_A(X) = \boldsymbol{A} * S_B(X) \oplus \boldsymbol{C}, \tag{1}$$

where S_A and S_B is called an affine key-dependent S-box and a base S-box, respectively, and the values of both $\boldsymbol{A}$ and $\boldsymbol{C}$ are dependent on a secret key. In order to make all key-dependent S-boxes injective and consistent with base S-box in size, $\boldsymbol{A}$ should be a nonsingular Boolean square matrix.

Let $\boldsymbol{A} = \{a_{ij}\}_{m*m}$ $(1 \le i, j \le m)$, $\boldsymbol{C} = \{c_m \cdots c_2 c_1\}$, $S_B(X) = \{b_m \cdots b_2 b_1\}$, $S_A(X) = \{b_m^{'} \cdots b_2^{'} b_1^{'}\}$, where $a_{ij}, c_i, b_i, b_i^{'} \in F_2$, $X \in F_2^m$. Then each element of $S_A(X)$, $X \in F_2^m$ can be computed as:

$$\begin{bmatrix} b_1^{'} \\ b_2^{'} \\ \vdots \\ b_m^{'} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \cdots a_{1m} \\ a_{21} & a_{22} \cdots a_{2m} \\ \vdots & \vdots \quad \vdots \quad \vdots \\ a_{m1} & a_{m2} \cdots a_{mm} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \oplus \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix}. \tag{2}$$

In the following, we will analyze the security of affine key-dependent S-boxes.

III. Security of Affine Key-dependent S-box

The mainly evaluative criteria of S-boxes are the following [13], [14], [15]:

(1) Nonlinearity (N_s),
(2) Differential uniformity (δ_s),
(3) Algebraic degree (def),
(4) Resistance of S-box against Algebraic Attack (RAA),
(5) Strict Avalanche Criterion (SAC),
(6) Number of monomials ($Nmon$),
(7) Number of fixed points ($Nfpt$).

In addition, C.Boura and A. Canteaut given a criterion of an S-box, named (v, w)-linearity, to evaluate the propagation of linear relations [16].

A high quality S-box should have the following characteristics: high nonlinearity, low differential uniformity, high resistance against algebraic attack, high algebraic degree, a great number of monomials, good strict avalanche effect and without fixed points.

A good S-box also should be a balanced function. For $S : F_2^n \to F_2^m$, S is said to be balanced if every element $y \in F_2^m$ has the same number of pre-images by S. The affine key-dependent S-boxes will be balanced if the base S-box is balanced, since different inputs will generate different outputs under affine transformation.

Many studies have investigated the impact on the security of S-boxes under affine transformations. It has been proved that the nonlinearity and differential uniformity of S-boxes are invariant by means of affine transformations [17]. The effect of affine transformations of an S-box on the maximal expected differential probability and linear potential is investigated in [18]. The effects of affine transformations on other evaluative criteria of an S-box are discussed in the following.

A. Algebraic Degree

Any Boolean function $f : F_2^n \to F_2$ can be uniquely represented as a multivariate polynomial over GF(2), called Algebraic Normal Form (ANF), i.e.

$$f(X) = f(x_1, \cdots, x_n) = \sum_{\mathbf{u} \in F_2^n} \alpha_{\mathbf{u}} x_1^{u_1} \cdots x_n^{u_n}, \tag{3}$$

where the coefficient $\alpha_{\mathbf{u}} \in F_2$, $u_i \in F_2$,$\mathbf{u} = \{u_1 \cdots u_n\}$, and $x_i^{u_i} = x_i$, if $u_i = 1$ and $x_i^{u_i} = 1$, if $u_i = 0$. The $x_1^{u_1} \cdots x_n^{u_n}$ in the ANF is called a monomial of f.

The algebraic degree of f is the maximum degree of those monomials with nonzero coefficients in the ANF. The algebraic degree of an S-box is the minimum algebraic degree of all the component functions of the S-box.

The algebraic degree is considered as invariant under affine transformations in [18]. However, we found that algebraic degree of S-boxes may be decreased by means of affine transformation.

For example, Let $S_{A1} = M1 * S_{B1}$, $M1 = [1,0,0,0;0,1,0,0;0,0,0,1;0,1,1,1]$, seeing S_{B1} and S_{A1} in table 1. Obviously, the algebraic degree of each component function of the S-box S_{B1} is 3, but the algebraic degree of the first component function of the S-box S_{A1} is 2.

TABLE I
The S-boxes S_{A1} and S_{B1}

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$S_{B1}(x)$	12	6	7	9	10	0	11	14	1	15	13	8	4	5	2	3
$S_{A1}(x)$	12	5	6	11	9	0	10	13	3	14	15	8	4	7	1	2

Here, we give a theorem about the algebraic degree of affine key-dependent S-boxes.

Theorem 1. Let d be the maximum algebraic degree of all component functions of the base S-box S_B, and M be the matrix whose i-th row vector consists of all the coefficients of d order monomials in the ANF of f_i, where f_i is the i-th component function of S_B. If the rank of M is greater than or equal to d, the algebraic degree of each component function of S_A is equal to d, where $S_A = \boldsymbol{A} * S_B(X) \oplus \boldsymbol{C}$.

Proof: Let $S_B : F_2^n \to F_2^m, S_B = \{f_1, \cdots, f_m\}$;
$S_A = \boldsymbol{A} * S_B(X) \oplus \boldsymbol{C} = \{f_{A1}, \cdots, f_{Am}\}$;
$f_j(X) = \sum_{\mathbf{u}_j \in F_2^n} \alpha_{\mathbf{u}_j} x_1^{u_{j1}} \cdots x_n^{u_{jn}}$,
where $\boldsymbol{A} = \{a_{ij}\}_{m*m}$ $(1 \le i, j \le m)$, $\boldsymbol{C} = \{c_1 \cdots c_m\}$, and $\mathbf{u}_j = \{u_{j1} \cdots u_{jn}\}$, then,

$$f_{Ai}(X) = c_i \oplus \sum_{1 \le j \le m} a_{ij} \Big(\sum_{\mathbf{u}_j \in F_2^n} \alpha_{\mathbf{u}_j} x_1^{u_{j1}} \cdots x_n^{u_{jn}} \Big). \tag{4}$$

The algebraic degree of f_{Ai} is only dependent on the values of a_{ij} and the coefficients of d order monomials in the ANF of f_j. So we only need to consider them. For simplicity, let the $v_k(X)$ represent the k-th d order monomial in the ANF of each the component function, and β_{jk} represent

the coefficient of k-th d order monomial in the ANF of f_j, where $v_k(X) = x_1^{u_{k1}} \cdots x_n^{u_{kn}}$, $wt(u_{k1} \cdots u_{kn}) = d$, $k \leq C(m,d) = q$. Then, the sum of all d order monomials in the ANF of f_j can be represented as:

$$P_d(f_j) = \sum_{1 \leq k \leq q} \beta_{jk} v_k(X). \tag{5}$$

The sum of all d order monomials in the ANF of f_{Ai} can be represented as:

$$\begin{aligned} P_d(f_{Ai}) &= \sum_{1 \leq j \leq m} a_{ij} * P_d(f_j) \\ &= \sum_{1 \leq j \leq m} a_{ij} \sum_{1 \leq k \leq q} \beta_{jk} v_k(X) \\ &= \sum_{1 \leq j \leq m} \sum_{1 \leq k \leq q} a_{ij} \beta_{jk} v_k(X). \end{aligned} \tag{6}$$

Explicitly, if and only if all the coefficients of $v_k(X)$ in the $P_d(f_{Ai})$ are equal to zero, the algebraic degree of f_{Ai} is less than d.

For a given f_{Ai}, the problem whether all the coefficients of $v_k(X)$ in $P_d(f_{Ai})$ are simultaneously zero is equivalent to this problem: whether there are non-zero solutions for the system of homogeneous equations in which a_{ij} $(1 \leq j \leq m)$ are m unknown variables and its coefficients matrix is $B = \{\beta_{jk}\}_{m*q}$ $(1 \leq j \leq m, 1 \leq k \leq q)$. When the rank of the matrix B is greater than or equal to m, The system of homogeneous equations only have zero solutions, i.e. all the m variables a_{ij} $(1 \leq j \leq m)$ are equal to zero. However, for any $1 \leq i \leq m$, a_{ij} $(1 \leq j \leq m)$ can not be zero simultaneously since A is a nonsingular matrix.

Therefore, when the rank of the matrix B is greater than or equal to m, the algebraic degrees of all the component functions of S_A are equal to d.

B. Resistance of S-boxes against Algebraic Attack

To do algebraic attack on a cipher, attackers should find adequate independent equations from the cipher system and then solve them to get the key. Unlike using algebraic immunity [19] to evaluate the resistance of S-box against algebraic attack for stream ciphers, Courtois et al. [20] defined

$$\Gamma = ((t-r)/s)^{(t-r)/s} \tag{7}$$

to evaluate the actual contribution of S-boxes to the complexity of algebraic attacks for block ciphers, where t representing the number of monomials, r representing the dimension of the space of equations, and s representing the size of bijective S-boxes. Cheon and Lee [21] called the Γ as the resistance of S-box against algebraic attacks. Generally, the complexity of algebraic attack on block cipher is almost decided by the number of independent equations and monomials[3].

The number of independent equations from a given affine key-dependent S-box will not change since the affine transformation is reversible. Therefore, the resistance of S-boxes against algebraic attack is invariant under affine transformation when the S-boxes are known by attackers. In fact, more independent equations are required for algebraic attacks on a cipher with key-dependent S-boxes since those S-boxes are secret. Therefore, affine key-dependent S-boxes can tone up the resistance of S-boxes against algebraic attacks for the block cipher.

C. Strict Avalanche Criterion

An S-box satisfies the strict avalanche criterion if each output bit of the S-box changes with a probability of one half whenever single input bit is complemented [13]. Many S-boxes in practical block ciphers, for example AES, are not satisfies the strict avalanche criterion since there are restriction relationships among the evaluative criteria of S-box. Distance to Strict Avalanche Criterion ($DSAC$) can be used to evaluate strict avalanche effect for S-boxes [17], which can be defined as

$$\begin{aligned} DSAC(f) = \frac{1}{2} \max_{\substack{e \in F_2^n \\ wt(e)=1}} \Big| 2^{n-1} \\ - \sum_{x \in F_2^n} wt(f(x) \oplus f(x \oplus e)) \Big|. \end{aligned} \tag{8}$$

The value of $DSAC$ is smaller, the strict avalanche effect is better. The $DSAC$ will be affected under affine transformation on the output of base S-boxes.

D. (v,w)-linearity of an S-box

Let $S : F_2^n \to F_2^m$, and $S_\lambda : F2^n \to F_2$, $S_\lambda(x) = \lambda \cdot S(x)$, where $x \in F_2^n$, $\lambda \in F_2^m$. S is (v,w)-linear if there exists a v-dimensional linear subspaces $V \subset F_2^n$ and a w-dimensional subspace $W \subset F_2^m$, and for all $\lambda \in W$, the degree of S_λ is not more than 1 [16].

Supposed that S_A is an affine S-box with S, i.e. $S_A(x) = A \cdot S(x) \oplus C$, where A is a nonsingular Boolean square matrix and C is a vector. The inverse matrix of A is denoted as A^{-1}. Then, $S(x) = A^{-1} \cdot (S_A(x) \oplus C)$.

If S is (v,w)-linear, then, for $x \in V$, $\lambda \in W$, $S_\lambda(x) = \lambda \cdot S(x) = \lambda \cdot (A^{-1} \cdot (S_A(x) \oplus C)) = (\lambda \cdot A^{-1} \cdot S_A(x)) \oplus (\lambda \cdot A^{-1} \cdot C)$. Obviously,

$$S_\lambda(x) \oplus (\lambda \cdot A^{-1} \cdot C) = \lambda \cdot A^{-1} \cdot S_A(x) \tag{9}$$

Let

$$S'_\lambda(x) = S_\lambda(x) \oplus (\lambda \cdot A^{-1} \cdot C) \tag{10}$$

and

$$\lambda' = \lambda \cdot A^{-1} \tag{11}$$

Then,

$$S'_\lambda(x) = \lambda' \cdot S_A(x) \tag{12}$$

For each $\lambda \in W$, there is a unique λ' since A^{-1} is a nonsingular Boolean square matrix. Let $W' = \{\lambda' | \lambda' = \lambda \cdot A^{-1}, \lambda \in W\}$. In addition, W is w-dimensional subspace. Therefore, W' is w-dimensional subspace.

If the degree of $S_\lambda(x)$ is not more than 1, then the degree of $S'_\lambda(x)$ is also not greater than 1, since $S'_\lambda(x)$ is affine with $S_\lambda(x)$. Therefore, there exist two subspaces V and W', and for all $\lambda' \in W'$, $S'_\lambda(x)$ has degree at most 1.

In conclusion, (v,w)-linearity property of an S-box will be not affected under affine transformation.

E. Classification of Evaluative Criteria of S-boxes

The number of monomials of f is the number of all monomials with nonzero coefficients in its ANF. Let $S : F_2^n \to F_2^m$, X is a fixed point of S if $S(X) = X$. The number of fixed points of S is the dimensions of the set $\{X \in F_2^n | S(X) = X\}$. Both the number of monomials of component functions and the number of fixed points in affine key-dependent S-boxes are affected under affine transformation.

According to the evaluative criteria of an S-box are whether affected under affine transformation, we classify them into three categories:

(1) Static affine criteria. They are invariant under affine transformations, such as nonlinearity, differential uniformity, resistance against algebraic attacks, balance and (v, w)-linearity.

(2) Conditional affine static criteria. They can be maintained under certain conditions, like algebraic degree.

(3) Dynamic affine criteria. They maybe change under affine transformation, such as number of monomials and $DSAC$.

According to above analysis, if the base S-box is a good one, the key-dependent affine S-boxes can keep the affine static criteria and conditional affine static criteria as well as that of the base S-box. A good S-box must have high nonlinearity. Generally, the number of monomials and avalanche effect are highly dependent on the nonlinearity of S-boxes, which can be proved by the following statistical results. Therefore, affine dynamic criteria of affine key-dependent S-boxes can also be accepted. In a word, if we choose a good S-box as the base S-box, we can get good key-dependent S-boxes by key-dependent affine transformation.

IV. Fast Algorithm for Constructing Affine Key-dependent S-box

A. Overview of the Fast Algorithm

Performance is an important indicator of block ciphers. In the following sections, we introduce how to fast construct affine key-dependent S-boxes. Supposed $Sbox_B$ is the base S-box, a fast algorithm for constructing affine key-dependent S-boxes is presented as Algorithm 1. It consists of three steps. Firstly, a key-dependent Boolean matrix $\boldsymbol{A}$ and a vector $\boldsymbol{C}$ are generated from a secret key. Secondly, a temporary S-box, denoted $Sbox_t$, is gotten by the matrix multiplication of an S-box, in which each element of $Sbox_B$ will be multiplied by matrix $\boldsymbol{A}$. In the third step, it is checked that whether fixed points will appear in the S-box generated by adding vector $\boldsymbol{C}$ to $Sbox_t$. If there are fixed pointed, the value of $\boldsymbol{C}$ will be changed, until it meets the requirement of the number of fixed pointed.

The key problems of fast constructing affine key-dependent S-boxes are how to get a nonsingular Boolean matrix A, how to do the matrix multiplication of an S-box, and how to make sure there are not fixed points in key-dependent S-boxes. We will present three fast algorithms to solve the three problems in the following section, respectively.

Algorithm 1 Generating Key-dependent S-boxes

Input :Key, base S-box
Output:Key-dependent S-box

1: Generate a key-dependent nonsingular Boolean matrix A and a vector C with a secret Key by the key-dependent nonsingular matrix generation algorithm;
2: Get a temporary S-box, denoted $Sbox_t$, by matrix multiplication of the S-box, where $Sbox_t[X] = A * Sbox_B[X]$, $X \in F_2^m$;
3: Check whether fixed points will appear in the S-box $Sbox_A$ for the given value of $\boldsymbol{C}$, where $Sbox_A[X] = Sbox_t[X] \oplus \boldsymbol{C}, X \in F_2^m$. If there are fixed points in $Sbox_A$, the value of $\boldsymbol{C}$ will be adjusted until it meets requirement of fixed points, and then output $Sbox_A$.

B. Fast Key-dependent Nonsingular Matrix Generation Algorithm

Each row vector of a $m * m$ nonsingular Boolean matrix A can be seen as a m-bit vector, denoted $v_r, v_r \in F_2^m \backslash 0$. To get key-dependent nonsingular Boolean matrix A, we can use m sub-word of a secret key as m indexes to select m linear independent m-bit vector from the set $\{i | i \in F_2^m \backslash 0\}$ as the m row vectors of A, respectively. The value of the r-th index to select the r-th row vector is denoted as r_{ind}. Supposed S_c is a candidate set consisting of the elements which can be selected as the r-th row vector of A. After the r-th row vector being selected, S_c is updated by removing the elements which are linear dependent on the former r row vectors from S_c.

To make different keys generate different nonsingular Boolean matrixes, the elements in S_c are always ranged in ascending order and labeled with 0, 1, 2, $\cdots$, and so on, respectively. There is a little trouble to find the r_{ind} row vector based on r_{ind} since the label of the elements in S_c will be changed when S_c is updated. there are two intuitive approaches to get the element labeled with r_{ind} in S_c. One is that all candidate elements are copied into an array and ranged in ascending order when S_c is updated each time, and then the $r_{ind} - th$ element in the array is selected as the row vector. The other is that a linked list is used to represent S_c. However, those approaches are low efficient.

To fast update the candidate S_c and find the r-th row vector, we present a fast key-dependent nonsingular matrix generation algorithm, seeing Algorithm 2. In the algorithm, three arrays, denoted S_c, S_l, S_v, are used to represent three sets: candidate set, linear dependent set and selected set, respectively. Their sizes are 2^m, 2^{m-1} and m, respectively. The index value of S_c (i.e. subscript of array) represents the value of a candidate element. The value of elements in S_c is either 1 or 0. i is a candidate element if $S_c[i] = 1$. Otherwise, it cannot be used as a row vector of the nonsingular Boolean matrix. In initial state, all the element of S_c except the first one are set to 1. When i is linear dependent on the selected row vectors, $S_c[i]$ will be set to 0. Let $c_{nt} = \sum_{i=1}^{n} S_c[i]$. The value of c_{nt} indicates how many elements ranged from 1 to n can be selected as row vectors. When the value of c_{nt} is equal to r_{ind}, indicating the $r_{ind} - th$ elements being found. As 0 is always not the candidate element, the value of $i - 1$ is the expected row vector, denoted as v_r. When

Algorithm 2 Generating Key-dependent Nonsingular Matrix

Input :Key
Output:Nonsingular matrix S_v

Initialization: $S_l[0] = 0, S_c[i] = \begin{cases} 0, i = 0 \\ 1, i \neq 0 \end{cases}$
for r=0 to m-1 **do**
    Generate the r-th index (denoted r_{ind}) from Key;

    //Get the r-th row vector (denoted v_r) with r_{ind}
    $c_{nt} = 0;\ i = 1;$
    while $c_{nt} \leq r_{ind}$ **do**
        $c_{nt} = c_{nt} + S_c[i];$
        $i = i + 1;$
    end while
    $v_r = i - 1;$

    //Update S_l and S_c.
    $S_v[r] = v_r$
    if $r < m - 1$ **then**
        for k=0 to 2^r **do**
            $temp = v_r \oplus S_l[k];$
            $S_l[k + 2^r] = temp;$
            $S_c[temp] = 0;$
        end for
    end if
end for

the r-th row vector (v_r) is chosen, make v_r be added to all the elements in the linear dependent set, and the results are appendant into the linear dependent set S_l.

The first two row vectors in S_v can be directly gotten from key with the result of r_{ind} subtracted 1 or 2 instead of using c_{nt} to further optimize this algorithm. The main characteristics of the fast algorithm include: (1) Using index value of an array to represent the candidate elements and the value of array element indicating whether its index is a candidate element. (2) To find the r_{ind}-th element in the updated candidate set S_c, we use a counter rather than intuitive approaches such as rearranging the element of array S_c or using a linked list.

C. Fast Algorithm for Matrix Multiplication of S-boxes

In matrix multiplication of an S-box, each element of the base S-box should be multiplied with the Boolean matrix A, where each bit of an element in the target S-box should be computed as equation (13):

$$b_i^t = \bigoplus_{i=0}^{m-1} a_{ij} b_j. \tag{13}$$

To finish the matrix multiplication of an S-box, $m * 2^m$ times computation like equation (13) are required in straightforward way. For this equation, one AND, m times Shift and $m-1$ times XOR operations (or one look up table operation instead of $m-1$ XOR operations) are required in general CPU. Therefore, the computation complexity of the matrix multiplication of an S-box is $O(m(m+2) * 2^m)$ at least. It is a very low effective algorithm.

Here, we proposed a fast algorithm for the matrix multiplication of an S-box by taking the advantage of Gray code. In

Algorithm 3 Fast Matrix Multiplication of S-boxes

Input :A
Output:$Sbox_t$

Initialization: $Sbox_t[0] = 0;\ rst = 0;\ G_1 = 0;$
for $r = 0\ to\ 2^m - 1$ **do**
    $G = i \oplus (i >> 1);$
    $diff = Gi \oplus G_1;$
    Switch the column vector $A[:, j]$ from matrix A according to the value of $diff$
    $rst = rst \oplus A[:, j];$
    $Sbox_t[invSbox_B[G]] = rst;$
    $G_1 = G;$
end for

Gray code, the representation of two successive values only differs in one bit. It is very easy to get the Gray number of an original number in a binary representation. Let G is the resulting Gray number, i is the original number, then $G = i \oplus (i >> 1)$,where $>>$ represents the binary right shift.

Let Z, Z_j, Y and Y_j are m-bit vector, $A[:, j]$ represents the j-th column vector of matrix A, $Z = A * Y$, and $Z_j = A * Y_j$. If the vectors Y and Y_j only differs at the j-th bit of them, then

$$Z_j = Z \oplus A[:, j] \tag{14}$$

Therefore, to do matrix multiplication of an S-box, we can transform $Sbox_B(X)$, $X \in F_2^n$, into Gray codes in turn, and then multiplication of $Sbox_B(X+1)$ by matrix A can be computed as equation (14), and the results are saved into the appropriate location in the goal S-box ($Sbox_t$). The appropriate location can be found based on the inverse S-box of base S-box ($invSbox_B$). The fast algorithm of matrix multiplication of an S-box is described as Algorithm 3.

In this fast algorithm, only three XOR, one shift, one switch, one look-up-table and two store operations are required for generating one element of $Sbox_t$. The computation complexity of generating $Sbox_t$ is $O(8 * 2^m)$. The proposed method can save much time to do matrix multiplication of an S-box compared with straightforward methods.

D. Fast Checking Fixed Points and Adjusting the Value of C

To avoid the S-boxes with fixed points being used, it is necessary to check whether fixed points will be appear in the affine key-dependent S-boxes and maybe adjust the value of vector C. To address this problem, a straightforward method maybe judge whether the result of $Sbox_t[i] \oplus C$ is equal to i for all $i \in F_2^n$. If there are fixed points, the value of C should be changed. The key pseudocode for the straightforward method can be seen in Algorithm 4. This method is low performance, especially when the value of C should be adjusted many times to eliminate fixed points.

Here, we proposed a fast algorithm for judging and eliminating fixed points. In this algorithm, whether fixed points will appear in $Sbox_A$ for a given value of C can be directly judged by accessing an array rather than checking fixed points after adding C to elements of $Sbox_t$ one by one. We find the rule that there will be a fixed point at $Sbox_A[i]$ if C is equal to the result of $Sbox_t[i] \oplus i$. Because if $C = Sbox_t[i] \oplus$

Algorithm 4 Straightforward Algorithm for Checking Fixed Points and Adjusting the Value of C

Input :C, $Sbox_t$
Output:$Sbox_A$

```
for j = 0 to 2^m − 1 do
    for i = 0 to 2^m − 1 do
        temp = Sbox_t[i] ⊕ C;
        if temp == i then
            C = C + 1 mod 2^m; break;
        end if
        pst_s box[i] = temp;
    end for
end for
```

Algorithm 5 Fast algorithm for Checking Fixed Points and Adjusting the Value of C

Input :C, $Sbox_t$
Output:$Sbox_A$

```
for i = 0 to 2^m − 1 do
    fixpt[Sbox_t[i] ⊕ i] = fixpt[Sbox_t[i] ⊕ i] + 1
end for

for i = 0 to 2^m − 1 do
    if fixpt[C] == 0 then
        break;
    else
        C = C + 1 mod 2^m;
    end if
end for
```

i, then $Sbox_A[i] = Sbox_t[i] \oplus C = i$. According to this rule, we can use an array (denoted $fixpt[2^m]$) to indicate that how many fixed points will appear in $Sbox_A$ for the given value of C. The elements in this array are initialized to zero, and then let $fixpt[Sbox_t[i] \oplus i] = fixpt[Sbox_t[i] \oplus i] + 1$, $i \in F_2^n$. The value of $fixpt[i]$ denotes how many fixed points will appear in $Sbox_A$ if the value of C is equal to i. The fast algorithm for checking fixed points and adjusting the value of C is shown in Algorithm 5.

If $Sbox_t$ is such an S-box that there is always fixed points in the $Sbox_A$ for any $C \in F_2^m$, we called $Sbox_t$ as one-fixed-point S-box. If $Sbox_t$ is not a one-fixed-point S-box, we can find a value of C which make no fixed points in $Sbox_A$. It is a hard problem to prove whether one-fixed-point S-boxes will be generated by affine transformation on a base S-box, but we can prove that there is only one fixed point in $Sbox_A$ for any C if one-fixed-point S-boxes are generated.
Prove: If $Sbox_A$ always has fixed points, then each element in the array $fixpt$ is greater than zero. However, there are only 2^m elements and the sum of all the elements in the array $fixpt$ is not more than 2^m. Therefore, each element in the array $fixpt$ is 1. In other words, there is only one fixed point will appear in $Sbox_A$ in this case. A few fixed points can be accepted in block cipher with key-dependent S-boxes, because they are secret and diffusion units can drop their weak by round function iterations. For example, fixed points are accepted in Twofish cipher.

TABLE II
STATISTIC CHARACTERISTICS OF $DSAC$ AND NUMBER OF MONOMIALS

	Minimum	**Maximum**	**Average**	**Standard Deviation**
$DSAC$	4	8	7.96	0.29
$Nmon$	110	151	127	8.2

TABLE III
DISTRIBUTION OF NUMBER OF MONOMIALS WITH DIFFERENT DEGREE

	Degree of Monomials								
	0	1	2	3	4	5	6	7	8
Ideal	0.5	4	14	28	35	28	14	4	0.5
Statistical	0	4	14	28	35	28	14	4	0
Standard Deviation	0	1	2	3	4	4	3	1	0

V. INSTANTIATION AND DISCUSSIONS

In order to further validate the effectiveness of the proposed method for key-dependent S-boxes, we presented an instantiation of the key-dependent S-boxes, and analyzed their evaluative criteria. In the instantiation, the base S-box (denoted S_B) is inverse function in $GF(2^8)$ with 0 mapped on itself, where the irreducible polynomial is $m(x) = x^8+x^4+x^3+x+1$. The nonlinearity and differential uniformity of this base S-box are 112 and 4/256, respectively. The resistance of an S-box against algebraic attacks for all the affine key-dependent S-boxes are same with that of the S-box of AES cipher. Those affine static criteria are excellent.

The algebraic degrees of eight component functions of S_B are 7. It can be verified that the rank of the matrix which consists of all the coefficients of monomials with degree 7 in the eight component functions is equal to 8. According to theorem 1, the algebraic degrees of the component functions of all key-dependent S-boxes are 7, which is the best number for 8*8 S-boxes. So all affine key-dependent S-boxes with inverse function in $GF(2^8)$ as base S-box have good affine static and conditional affine static criteria.

We implemented an affine key-dependent S-boxes generator in C language according to our proposed fast algorithms. We used 64-bit key to generate the 8*8 nonsingular Boolean matrix $\boldsymbol{A}$ and 8 bit vector $\boldsymbol{C}$. The most left 8-bit of the key is used to generate $\boldsymbol{C}$. The remaining 56-bit is divided into eight 7-bit independent sub-key used as the indexes to select eight row vectors of $\boldsymbol{A}$. As the key space is too large to test all affine key-dependent S-boxes, we adopted a random sampling method for statistical analysis of the affine dynamic criteria from a statistical standpoint. We used a random generator to produce 2^{20} 64-bit keys and identical number affine key-dependent S-boxes were generated. We had tested $DSAC$ and the number of monomials for each component functions of those affine key-dependent S-boxes. The results are shown in table 2. In addition, the distributions of the number of monomials with different degrees are shown in table 3. The comparisons of evaluative criteria between these affine key-dependent S-boxes and the S-box of AES are shown in table 4. The one-fixed-point S-box is not found in our statistical experiment. As can be seen from Table 5, the key-dependent S-boxes obtained by our method have the same security level with the S-box of AES.

We implemented the affine key-dependent S-boxes gener-

TABLE IV
COMPARISONS OF EVALUATIVE CRITERIA BETWEEN THE AFFINE KEY-DEPENDENT S-BOXES AND THE S-BOX OF AES

Evaluative Criteria	Affine Key-dependent S-boxes	S-box of AES
N_s	112	112
δ_s	$4/256$	$4/256$
$DSAC$	4 - 8	8
def	7	7
$Nmon$	110 - 151	110 - 145
RAA	$2^{22.9}$	$2^{22.9}$

TABLE V
COMPARISONS OF SPEEDUP BETWEEN THE PROPOSED FAST ALGORITHMS AND STRAIGHTFORWARD ALGORITHM

Algorithms	Fast	Straightforward	Speedup
Generate affine key-dependent S-box	11070	36599	3.3
Get nonsingular Boolean matrixes	3025	14012	3.8
Matrix multiplication of S-boxes	4846	18183	3.9
Checking fixed points and adjusting C	2770	3975	1.4

ation algorithm in C language, and simulated it with GEM5 based on ARM instruction set. We also implemented the C functions for generating 8*8 nonsingular Boolean matrix, the matrix multiplication of an S-box and checking fixed points and adjusting the value of $\boldsymbol{C}$ in a straightforward way. The simulation results show that approximately 11070 dynamic instructions are required to get an 8*8 affine key-dependent S-box by the fast algorithms on average. If the affine key-dependent S-boxes are generated in a straightforward way, 36599 instructions will be required on average. The number of executed instructions for fixed points checking and adjusting the value of C is from 2758 to 3031 (2770 on average) by the fast algorithm, while it is from 2152 to 42980 (3975 on average) in a straightforward way. The speedup of the fast constructing affine key-dependent S-boxes can be up to 3.3 compared with the implementation in a straightforward way.

To enhance the security of encrypted data, more rounds and longer key may be used for a block cipher. For example, AES-192 which has 12 rounds may be used instead of AES-128 which has 10 rounds. However, the former is twenty percent slower than the later. This is a large performance overhead for encrypting bulk data such as a database and video data. If we used the enhanced AES-128 in which the fixed S-boxes are replaced by key-dependent S-boxes, the performance overhead only spent on key-dependent S-boxes establishment stage. According to results in [22], we can find that each round of AES encryption needs about 164 instructions by ARM7. It means 328 instructions cycles will be increased if using AES-192 instead of AES-128 to encrypt one data block (128 bits). When the size of the data to be encrypted is larger than 540 bytes, which is very small compared to bulk data generally, the enhanced AES-128 with key-dependent S-boxes will have an advantage of performance compared with AES-192.

In addition,the proposed method of affine key-dependent S-boxes may be useful in hash functions design since affine key-dependent S-boxes can enhance the security of block ciphers and it is becoming an interesting approach to implement a hash function using block ciphers [23], [24].

Therefore, the affine key-dependent S-boxes can be used to replace the fixed S-boxes in the AES cipher. It not only improve the security of the cipher, but also has an advantage in performance for encrypting bulk data compared with increasing the number of rounds to enhance the security. These fast algorithms can also be used for other applications such as fast searching golden S-boxes.

VI. CONCLUSION

A method for constructing good key-dependent S-boxes by means of key-dependent affine transformation on a good base S-box is proposed in this paper. We investigated the effect of affine transformation on the security of S-boxes and classified the evaluative criteria of S-boxes into affine static, conditional affine static and affine dynamic three categories. It is found that the algebraic degree of an S-box is conditional invariant under affine transformation and the condition for keeping algebraic degree invariant is given. Theoretical analysis shows that if the base S-box is a good one, all key-dependent S-boxes generated by the proposed method will be as good as the base S-box in nonlinearity, differential uniformity, algebraic degree, balance property, (v, w)-linearity and resistance of S-box against algebraic attack for block cipher. Experimental results show that the affine dynamic criteria such as $DSAC$ and number of monomials also have good cryptographic properties. In addition, to fast generate affine key-dependent S-boxes, three fast constructing algorithms for obtaining key-dependent nonsingular Boolean matrix, matrix multiplication of an S-box, and eliminating fixed points are proposed, respectively. The speedup of the proposed fast algorithms is more than 3.3 compared with the implementations in straightforward way. The highlight of the fast algorithm for matrix multiplication of an S-box is taking advantage of Gray code. It provides a new view of implementation of matrix multiplication of S-boxes and may be helpful for other aspects like fast searching golden S-boxes in intelligent algorithm. The proposed method can ensure that all key-dependent S-boxes have good cryptographic properties. It not only can enhance the security of a cipher, but also has an advantage in performance of encrypting bulk data compared with increasing the number of rounds to enhance the security level.

REFERENCES

[1] E. Biham and A. Shamir, "Differential cryptanalysis of des-like cryptosystems," *Journal of CRYPTOLOGY*, vol. 4, no. 1, pp. 3–72, 1991.

[2] M. Matsui, "Linear cryptanalysis method for des cipher," in *Advances in CryptologyEUROCRYPT93*. Springer, 1994, pp. 386–397.

[3] N. T. Courtois and J. Pieprzyk, "Cryptanalysis of block ciphers with overdefined systems of equations," in *Advances in CryptologyASIACRYPT 2002*. Springer, 2002, pp. 267–287.

[4] B. Schneier, "Description of a new variable-length key, 64-bit block cipher (blowfish)," in *Fast Software Encryption*. Springer, 1994, pp. 191–204.

[5] A. Biryukov and A. Shamir, "Structural cryptanalysis of sasas," in *Advances in CryptologyEUROCRYPT 2001*. Springer, 2001, pp. 395–405.

[6] L. Keliher, "Linear cryptanalysis of substitution-permutation networks," Ph.D. dissertation, Queens University, 2003.

[7] R. C. Merkle, "Fast software encryption functions," in *Advances in Cryptology-CRYPT090*. Springer, 1991, pp. 477–501.

[8] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson, "Twofish: A 128-bit block cipher," *NIST AES Proposal*, vol. 15, 1998.

[9] K. Kazlauskas and J. Kazlauskas, "Key-dependent s-box generation in aes block cipher system," *Informatica*, vol. 20, no. 1, pp. 23–34, 2009.

[10] N. Masuda, G. Jakimoski, K. Aihara, and L. Kocarev, "Chaotic block ciphers: from theory to practical algorithms," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 53, no. 6, pp. 1341–1352, 2006.

[11] N. Stoianov, "One approach of using key-dependent s-boxes in aes," in *Multimedia Communications, Services and Security*. Springer, 2011, pp. 317–323.

[12] S. Vaudenay, "On the weak keys of blowfish," in *Fast Software Encryption*. Springer, 1996, pp. 27–32.

[13] A. Webster and S. E. Tavares, "On the design of s-boxes," in *Advances in CryptologyCRYPTO85 Proceedings*. Springer, 1986, pp. 523–534.

[14] S. Fischer and W. Meier, "Algebraic immunity of s-boxes and augmented functions," in *Fast Software Encryption*. Springer, 2007, pp. 366–381.

[15] M.-J. O. Saarinen, "Cryptographic analysis of all $4\times$ 4-bit s-boxes," in *Selected Areas in Cryptography*. Springer, 2012, pp. 118–133.

[16] C. Boura and A. Canteaut, "A new criterion for avoiding the propagation of linear relations through an sbox," in *Fast Software Encryption*. Springer Berlin Heidelberg, 2013, pp. 763–764.

[17] S. Mister and C. Adams, "Practical s-box design," in *Workshop on Selected Areas in Cryptography, SAC*, vol. 96. Citeseer, 1996, pp. 61–76.

[18] A. Canteaut and J. Rou, "On the behaviors of affine equivalent sboxes regarding differential and linear attacks," in *Advances in Cryptology – EUROCRYPT 2015*. Springer, 2015, pp. 45–74.

[19] W. Meier, E. Pasalic, and C. Carlet, "Algebraic attacks and decomposition of boolean functions," in *Advances in Cryptology-EUROCRYPT 2004*. Springer, 2004, pp. 474–491.

[20] N. T. Courtois, B. Debraize, and E. Garrido, "On exact algebraic [non-] immunity of s-boxes based on power functions," in *Information Security and Privacy*. Springer, 2006, pp. 76–86.

[21] J. H. Cheon and D. H. Lee, "Resistance of s-boxes against algebraic attacks," in *Fast Software Encryption*. Springer, 2004, pp. 83–93.

[22] G. Bertoni, L. Breveglieri, P. Fragneto, M. Macchetti, and S. Marchesin, "Efficient software implementation of aes on 32-bit platforms," in *Cryptographic Hardware and Embedded Systems-CHES 2002*. Springer, 2003, pp. 159–171.

[23] N. Kishore and B. Kapoor, "Attacks on and advances in secure hash algorithms,," *Iaeng International Journal of Computer Science*, vol. 43, no. 3, pp. 326–335, 2016.

[24] S. Hirose and H. Kuwakado, "A block-cipher-based hash function using an mmo-type double-block compression function," *Ieice Technical Report Information Theory*, vol. 111, pp. 45–51, 2012.