Lightweight design of SM4 algorithm and realization of threshold scheme

B W Chen¹, X Xia¹, Q M Liang², W D Zhong³*

¹ Academy of cryptographic engineering, Engineering University of PAP, Xi’an, Shanxi, 710086, China
² Academy of information engineering, Engineering University of PAP, Xi’an, Shanxi, 710086, China
³ Key Laboratory of PAP of Network and Information Security of Engineering University of PAP, Xi’an, Shanxi, 710086, China
*Corresponding author’s e-mail: gxdxmmxy@sdsch.cn

Abstract. Aiming at the lightweight design of SM4 cryptographic algorithm, a lightweight cryptographic algorithm scheme is proposed and implemented in hardware. Based on the work of Li et al., a 4-bit S-box is selected. The original 8-bit look-up table S-box in the SM4 algorithm is replaced with a 4-bit cyclic shift S-box, and the same circuit structure is used to save resources in hardware implementation. A threshold scheme based on the principle of secret sharing is proposed to improve the overall security of the scheme against side-channel attacks. The input information is divided into three groups, and all operations are converted from $GF(2^4)$ to $GF(2^2)$ through a multiplier and a squarer, which further reduces the circuit consumption and improves the security of the algorithm. In the realization of the threshold scheme, virtual values are introduced to ensure that the scheme meets the uniformity, and the decomposition method is used in the inverter to reduce the number of operations and the proportion of space. After theoretical deduction, the threshold scheme as a whole has the ability to resist first-order and second-order DPA attacks. The overall hardware implementation of the program uses a total of 2803 logic elements, which proves that it achieves the goal of lightweight while ensuring safety.

1. Introduction

With the use of radio frequency identification devices (RFID) and wireless sensor networks (WSNs) becoming more and more normalized, the application range of lightweight cryptographic algorithms has gradually expanded. Under the conditions of low power consumption, limited computing resources, and small hardware footprint, lightweight passwords have extremely high practicality. Many lightweight algorithms such as PRESENT, LED, LBlock, Piccolo, Simon, Speck, have also been designed under the research of scholars[1]. SM4 algorithm is a commercial cryptographic algorithm independently developed by our country[2], which has extremely high practical value for lightweight research. However, because lightweight cryptographic algorithms are used in environments where hardware and software resources are very limited, their security is at a disadvantage compared with traditional cryptographic algorithms[3]. The security improvement work for lightweight cryptographic algorithms has been ongoing. There are two main research ideas: Improve the iterative structure of the algorithm itself to improve the mathematical security of the algorithm or improve the S-box to
Improve the side channel security. Among them, Liu proposed a 4-bit reflexive S-box that can be applied to lightweight cryptographic algorithms for GPU\cite{1}. By improving the iterative structure of the PRESENT algorithm, Zhang has improved the algorithm’s anti-differential analysis and other mathematical analysis performance, and improved the software and hardware efficiency of the algorithm\cite{5}. Based on the SM4 cryptographic algorithm, this paper proposes a lightweight cryptographic algorithm based on the work of literature [5] and literature [6]. The main innovations are as follows:

1) The 8-bit S-box of the original SM4 cryptographic algorithm is replaced with the 4-bit S-box proposed in literature [5]. Because the implementation circuit adopts the same form, this can greatly reduce the resources consumed by the hardware implementation of the circuit and meet the initial requirements of lightweight. And the 4-bit S-box also achieves the best cryptographic properties.

2) The realization of the secret sharing scheme is completed on the basis of the composite domain. The use of the composite domain can ensure the reduction of hardware resource consumption while ensuring resistance to basic power consumption attacks. In the secret sharing threshold scheme, information grouping with a share of 3 is used to ensure that the threshold scheme can resist first-order and second-order DPA (differential power analysis) attacks.

3) Use FPGA (Field Programmable Gate Array) to implement the algorithm in hardware to verify that it has achieved the goal of lightweight.

2. Algorithm introduction

The algorithm improvement scheme proposed in this paper is based on the national secret SM4 algorithm. The Feistel\cite{7} structure is still used for iterative structure. In order to adapt to the 4 bit S-box used, the plaintext block length and key of the algorithm are set by 64 bits, and in order to ensure the confusion and diffusion effect of each bit, the number of iterations of the algorithm is still 32. The system parameters of the original SM4 algorithm remain half.

2.1. Encryption and decryption algorithm

Define the relevant variables. $Z_2^e$ means that the element is a binary e-bit element. Circulating $i$ bits to the left is represented by $<<i$, $\oplus$ means 16-bit exclusive OR, and sbox(*) means 4-bit non-linear transformation S-box.

Suppose the input of this lightweight algorithm is $(X_0, X_1, X_2, X_3) \in (Z_2^{16})^4$, and its output is $(Y_0, Y_1, Y_2, Y_3) \in (Z_2^{16})^4$, which is used in 16 iterations. The key of is $r_k \in Z_2^{16}, i=0,1,\ldots,15$. According to the definition of the algorithm, after 16 iterations, another reverse order transformation is performed to obtain the final output. The process can be summarized by the formula as follows:

$$X_{i+4} = X_i \oplus T(X_{i+1} \oplus X_{i+2} \oplus X_{i+3} \oplus r_k)$$

(1)

$$Y_0, Y_1, Y_2, Y_3 = (X_{19}, X_{18}, X_{17}, X_{16})$$

(2)

In the above formula, the function $T$ is composed of a nonlinear function $\tau$ and a linear function $L$, where $\tau$ is four parallel S-box operations, and the input is $A=(a_0, a_1, a_2, a_3) \in (Z_2^4)^4$. The output is $B=(b_0, b_1, b_2, b_3) \in (Z_2^4)^4$, and its operation relation expression can be expressed as the following formula: $B=\tau(A) = (\text{sbox}(a_0), \text{sbox}(a_1), \text{sbox}(a_2), \text{sbox}(a_3))$.

Then through linear transformation, the output of function $T$ can be obtained by $L(B)=B \oplus (B<<1) \oplus (B<<5) \oplus (B<<9) \oplus (B<<12)$. The decryption and encryption algorithms use the same structure, it's just that the key is used in the reverse order. The parameters set in the above steps are all set in accordance with the design requirements of the lightweight algorithm and the parameters of the SM4 algorithm. Among them, the basic flow of the algorithm is shown in Figure 1.
2.2. The generation algorithm of round key

The round key is generated by the initially input key. In this lightweight algorithm, it involves the selection of fixed parameters of the cipher algorithm. This algorithm will select parameters according to the ratio. The formula for generating round key $r_k$ is as follows:

$$MK = (MK_0, MK_1, MK_2, MK_3) \in (Z_2^{12})^4$$  \hfill (3)

$$ (K_0, K_1, K_2, K_3) = (MK_0 \oplus FK_0, MK_1 \oplus FK_1, MK_2 \oplus FK_2, MK_3 \oplus FK_3) $$  \hfill (4)

$$r_k = K_{i+4} = K_i \oplus T'(K_{i+1} \oplus K_{i+2} \oplus K_{i+3} \oplus CK_i)$$  \hfill (5)

Among them, $K_i$ belongs to the intermediate variable of calculation, and $MK_i$ belongs to the initial input key. The linear transformation formula of function $T'$ is: $L'(B)=B \oplus (B<<7) \oplus (B<<<13)$. 

Figure 1. Basic flowchart of encryption algorithm.
3. Selection and safety verification of S-box
As the only non-linear component in the algorithm, the S-box is an important way for block ciphers to achieve confusion and diffusion. However, due to the non-linear nature of the input and output of the S-box, it is extremely easy to be the target of side-channel attacks. Therefore, the security of the S-box plays a very important role in the security of the cryptographic algorithm. Based on the work of literature [5], the article selects the 4×4 type S-box selected by it. According to the requirements of the algorithm, the selected S-box truth table is shown in Table 1:

| input | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|-------|---|---|---|---|---|---|---|---|
| output| 0 | C | 9 | D | 3 | 5 | B | 1 |
| input | 8 | 9 | A | B | C | D | E | F |
| output| 6 | E | A | 8 | 7 | 4 | 2 | F |

Next, theoretically, the safety of the S-box will be analyzed and verified.

3.1. Theoretical Preparation

**Theorem 1** [8] Suppose S(x): $Z_2^n \rightarrow Z_2^n$, if for any $X \in Z_2^n$, S(x) $\in Z_2^n$, then S(x) satisfies injectivity, and if for any $X, X^* \in Z_2^n, X^* \neq X, S(x) \neq S(x^*)$, then S(x) satisfies bijection.

**Definition 1** [9] The algebraic number of the S-box can be represented by the algebraic number of its Boolean function $f$. The algebraic degree of the S-box refers to the degree of the highest term of the Boolean function of all the components of the S-box, which can be defined by the following formula:

$$\text{deg}(f) = \max_{u \in Z_2^n} \left\{ \sum_i u_i a_i \neq 0 \right\}$$  \hspace{1cm} (6)

The algebraic degree of S(x) can be defined by the formula:

$$\text{deg}(s) = \max_{(a,v) \in Z_2^n} \text{deg}(f)$$  \hspace{1cm} (7)

**Definition 2** [10] The non-linearity of the S-box should be the smallest non-linearity among all component functions, which can be calculated by the following formula:

$$NL_S = 2^{n-1} - \frac{1}{2} \max_{a \in Z_2^n} \left| W_S(a,v) \right|$$  \hspace{1cm} (8)

$$W_S(a,v) = \sum_{x \in Z_2^n} (-1)^{S(x) \cdot a} \cdot v, a, v \in Z_2^n$$  \hspace{1cm} (9)

Among them, $W_S$ is the Walsh spectrum of the function S, and $a \cdot v$ represents the dot product of a and b on $Z_2^n$, that is, $a \cdot b = \oplus_{i=1}^n a_i b_i$.

**Definition 3** [11] Given an n-order S-box, its linearity can be obtained by the following formula:

$$L_S = \max_{a \in Z_2^n} \left| W_S(a,v) \right|$$  \hspace{1cm} (10)

**Definition 4** [7] S(x) is a bijective function. Both input difference $\alpha$ and output difference $\beta$ satisfy $\alpha, \beta \in Z_2^n$, and a differential distribution table of S-box can be written based on this, namely $T_S(\alpha, \beta)$, for any differential pair, its value can be found in the differential table, the formula is as follows:
For a given Boolean function $S$, its differential uniformity can be obtained by the following formula:

$$T_s(\alpha, \beta) = \# \{ x \in F_2^n \mid S(x) \oplus S(x \oplus \alpha) = \beta \}$$

(11)

According to the various variables defined above, the following properties of $S(x)$ can be derived:

1. In the cryptographic algorithm, if the S-box function $S(x)$ satisfies the bijection, then $S(x)$ is said to be balanced.

2. For an $n$-order S-box, the optimal value of its algebraic number should be $n-1$, so that high-order cryptanalysis attacks can be prevented.

3. Because the 4th order S-box is used in the scheme, the highest degree term of the Boolean function expression should be the third degree term.

4. Generally speaking, the smaller the linearity of an S-box function, the greater the nonlinearity, and the stronger the ability to resist linear analysis.

5. The smaller the difference uniformity is, the stronger the S-box is against the difference analysis.

3.2. The construction principle of S-box

The S-box in the scheme is constructed based on the work of literature [5] by using a cyclic shift of 3 Boolean functions. For a given Boolean function $f$, the S-box construction formula is as follows:

$$S(x,y,z,w) = (f(x,y,z,w), f_1(x,y,z,w), f_2(x,y,z,w), f_3(x,y,z,w))$$

Suppose here that $S$ is a third-order Boolean function, and there are 1 third-order terms, 6 second-order terms, and 4 first-order terms in the expression. The coefficient before the third term is limited to 1, and the coefficients before the remaining terms are marked as $a_1$ to $a_{10}$. Then there are 1024 ($2^{10}$) possibilities. Among them, the Boolean function expression is as follows:

$$f = xyz + a_1xy + a_2xz + a_3xw + a_4yz + a_5yw + a_6zw + a_7 + a_8 + a_9 + a_{10}$$

The remaining 3-bit Boolean functions are obtained by shifting the variable value of the initial Boolean function by one bit to the left, two bits to the left, and three bits to the left. The final expression can be obtained as follows:

$$f_1 = yzw + a_1yz + a_2y + a_3yw + a_4y + a_5zw + a_6zw + a_7yw + a_8yw + a_9yw + a_{10}$$

$$f_2 = zwx + a_1zw + a_2xz + a_3zy + a_4wx + a_5wy + a_6wy + a_7wy + a_8wy + a_9wy + a_{10}$$

$$f_3 = wxy + a_1wx + a_2yw + a_3zw + a_4xy + a_5xz + a_6yz + a_7yw + a_8yw + a_9 + a_{10}$$

The $(x, y, z, w)$ is traversed and assigned from 0000 to 1111. In order to satisfy Theorem 1 in Section 3.1, the result should also be traversed from 0000 to 1111, and each input corresponds to a unique output, so The S-box function can be balanced.

According to the S-box truth table listed in the literature [5], one of the S-boxes is selected for this scheme. The truth table has been given at the beginning of Chapter 2. According to the truth table and programming, the Boolean function expression of the remaining 3 bits of the S-box can be obtained as follows:

$$f = xyz + yw + zw + z + w$$

$$f_1 = yzw + z + x + w + x$$

$$f_2 = zwx + wy + x + y$$

$$f_3 = wxy + x + y + z$$

Combine Section 3.1 and Section 3.2 to verify the security of the S-box.

3.3. Security verification of S-box

First, according to the S-box truth table and the Boolean function expression, it can be concluded that $S(x)$ satisfies the bijection, so the function satisfies the balance.
Secondly, the function’s algebraic times can be verified. Because the S-box in the scheme is implemented using a Boolean function, the optimal algebraic number of the fourth-order S-box should be 3. According to the Boolean function expression, the S-box satisfies the condition.

Next, analyze the linearity. A safety-qualified S-box should have the highest possible linearity and the lowest possible non-linearity. Even if \( S(x) \) is a non-linear component, the linear expression of \( S(x) \) can still be fitted to the greatest extent using the formula \( a \cdot x = v \cdot S(x) \). According to the definition of linearity and non-linearity in section 2.1, through matlab programming, the linearity value of the record S box can be obtained as shown in Table 2.

### Table 2. The linearity value of S-box

| a\( x \) | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---------|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| 0       | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1       | 0 | -8 | 0 | 8 | 0 | 0 | 0 | 4 | 4 | -4 | 4 | -4 | 4 | 4 | 4 |
| 2       | 4 | 0 | 4 | -8 | -4 | 0 | 4 | 8 | 48 | -4 | 0 | 4 | 0 | 4 | 0 |
| 3       | -4 | 0 | 4 | 0 | 4 | 8 | 4 | 4 | 4 | -4 | 0 | 4 | -8 | 4 | 0 |
| 4       | 8 | 4 | -4 | 0 | 0 | 4 | 4 | -8 | 0 | -4 | 4 | 0 | 4 | 4 | 0 |
| 5       | 0 | 4 | -4 | 0 | 0 | 4 | -4 | 4 | 4 | 8 | 0 | -4 | 4 | 0 | 8 |
| 6       | 4 | -4 | 0 | 0 | -4 | 4 | 0 | 0 | 4 | 4 | -8 | 8 | 4 | 4 | 0 |
| 7       | 4 | 4 | 0 | 0 | -4 | 4 | 8 | 8 | 4 | 8 | 0 | -4 | 4 | 0 | 0 |
| 8       | -8 | 8 | 0 | 4 | -4 | -4 | 4 | 0 | 0 | 0 | 4 | 4 | 4 | 4 | 4 |
| 9       | 0 | 0 | 8 | 4 | -4 | 4 | 4 | -4 | 4 | 4 | 4 | 0 | 0 | -8 | 0 |
| 10      | 4 | 0 | 4 | 4 | -4 | 4 | 0 | 0 | -4 | 0 | -4 | 4 | 0 | -4 | 8 |
| 11      | 4 | 0 | -4 | 4 | 0 | -4 | 0 | 4 | 0 | 4 | 8 | 8 | 4 | 0 | -4 |
| 12      | 0 | 4 | 4 | -4 | 4 | 0 | -8 | 0 | 8 | -4 | 4 | 4 | 4 | 0 | 0 |
| 13      | 0 | 4 | -4 | 4 | 4 | 8 | 0 | 4 | -4 | 0 | 0 | 0 | 8 | -4 | -4 |
| 14      | 4 | 4 | 8 | 4 | 0 | 0 | -4 | 0 | -4 | 4 | 0 | -4 | 0 | 8 | -4 |
| 15      | -4 | -4 | 0 | -4 | 8 | 0 | 4 | -4 | 0 | 8 | 4 | 0 | 4 | 4 | 0 |

The linearity of the S-box is the maximum of all linearity values after traversing the values of a and v, which can be obtained from the above table. The linearity of the S-box selected in the scheme is 8, and the nonlinearity is 4. The S-box complies with the safety standard of 4-bit S box.

Then, use matlab for programming, and the difference uniformity value table of the S-box can be obtained as shown in Table 3.

### Table 3. The table of differential uniformity

| a\( x \) | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---------|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| 0       | 16 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 1       | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 2       | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 3       | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 4       | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 5       | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 6       | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 7       | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 8       | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 9       | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 10      | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
In the case of traversing $a$ and $v$, the difference uniformity should be the maximum value in the table (the case where both $a$ and $v$ are 0 is excluded), and the difference uniformity of the S-box can be obtained as 4. Meet the safety requirements for S-box.

It can be obtained from the above verification that the S-box selected by the scheme has the best cryptographic properties\cite{12}, and theoretically has the security against linear attacks.

4. The threshold scheme of S-box

4.1. The nature of the scheme

The threshold scheme is an important way for circuits to resist side-channel attacks. Among them, the variable $x$ is divided into $s$ parts, and each variable is marked as $x_i (1 \leq i \leq s)$, then $x$ can be obtained by XOR through $x_i$. For example: the Boolean function that needs to be shared is: $F(x,y,z,\ldots):GF(2^m)\rightarrow GF(2^n)$, take $x$ as an example, set the vector to $(x_1,x_2,\ldots,x_{i-1},x_{i+1},\ldots,x_s)$, that does not include the component $x_i$, in order to share the function $F$, a Boolean function $F_i$ with $s$ components needs to be constructed, and the function needs to satisfy the following properties\cite{13}:

**Property 1** Incompleteness All functions $F_i$ must be independent of all input variables $x, y, z, \ldots$, that is, in the Boolean function expression of $F_i$, components $x_i, y_i, z_i, \ldots$ are not allowed. This property makes the final circuit resistant to DPA attacks.

**Property 2** Correctness For the variables $x, y, z, x = \oplus x_i, y = \oplus y_i, z = \oplus z_i$ should be available, then the function $F$ should satisfy: $F(x,y,z,\ldots) = \oplus F_i(x,y,z,\ldots)$. This property can ensure that the grouping circuit can correctly execute the ungrouped circuit.

**Property 3** Uniformity The input and output values of the combination circuit should satisfy a uniform distribution, namely: $\Pr(u = \overline{U} | \mu = \oplus U_i)$. Is a constant. Among them, $u = F(x)$. When property 1 and property 2 are satisfied, the function may not satisfy property 3 at the same time, so property 3 must be verified separately.

4.2. The specific content of the scheme

The maximum number of Boolean function expressions of the S-box output value is 3, which will cause the hardware resource overhead and power consumption to increase significantly when the circuit is implemented. Before the threshold scheme is implemented, the component expression is first processed to reduce the number of terms. The Boolean function $f$ whose algebraic degree is 3 is composed of three subfunctions $g_1, g_2,$ and $g_3$, whose degree is 2. The final results of reduction times are as follows:

\[
\begin{align*}
    f &= xyz + yw + zw + z + w \\
    g_1(x,y,z) &= xy + xz + x + y \\
    g_2(y,z,w) &= zw + y + z \\
    g_3(x,y) &= xy + y + w \\
    f(x,y,z,w) &= g_1g_2 + g_2g_3 + g_3 \\
    f &= g_1g_2 + g_2g_3 + g_3 = g_1g_2g_3
\end{align*}
\]

In the hardware implementation, the hardware resources consumed by the circuit laid by the Boolean function expression with the highest degree of 2 are significantly reduced compared to the previous\cite{6}, and the share of sub-functions and input variables is set to $3^{14}$.

\[
\begin{align*}
    \lambda &= x_1 + x_2 + x_3 \\
    j &= y_1 + y_2 + y_3
\end{align*}
\]
\[ z = z_1 + z_2 + z_3 \]
\[ w = w_1 + w_2 + w_3 \]
\[ g_1 = g_{11} + g_{12} + g_{13} \]
\[ g_2 = g_{21} + g_{22} + g_{23} \]
\[ g_3 = g_{31} + g_{32} + g_{33} \]

Finally, the threshold scheme with a shared share of 3 can be designed as follows:

\[ g_{11} = x_2y_2 + x_3y_3 + x_1y_2 + x_4z_2 + x_5z_1 + x_7z_1 \]
\[ g_{12} = x_3y_3 + x_1y_1 + x_4y_1 + x_4z_2 + x_7z_1 + x_7z_1 \]
\[ g_{13} = x_1y_1 + x_3y_3 + x_2y_2 + x_4z_2 + x_7z_1 + x_7z_1 \]
\[ g_{21} = z_2w_2 + y_2 + z_3 \]
\[ g_{22} = z_3w_3 + y_3 + z_1 \]
\[ g_{23} = z_1w_1 + y_1 + z_2 \]
\[ g_{31} = x_2y_2 + x_3y_3 + x_1y_1 + x_4w_2 + y_2w_3 \]
\[ g_{32} = x_3y_3 + x_1y_1 + x_4w_2 + y_2w_3 + y_3w_1 + y_3w_1 \]
\[ g_{33} = x_1y_1 + x_3y_3 + x_2y_2 + y_2w_3 \]

The scheme meets the three properties required by the threshold scheme, so the threshold scheme can be implemented, and the S-box has the ability to resist side channel attacks.

Since the Boolean functions of the remaining three bits of the S-box are obtained by shifting \( f \), the threshold scheme of the remaining bits can also be obtained by shifting the scheme of the function \( f \) directly.

4.3. The hardware implementation of S-box based on threshold scheme

The logical function expression for each bit of the 4-bit S-box output value has been given in the previous article. According to the Karnaugh map, the simplified form of the expression can be obtained. The simplified logical function expression form is as follows:

\[ f = yz + yz + xz + zw \]
\[ f_1 = zw + zw + yw + yw \]
\[ f_2 = wx + wx + xy + zx \]
\[ f_3 = x + yz + zw + wyz \]

Since the expressions of the logic functions are all the same in the form of hardware implementation, the functions \( f_1, f_2, f_3 \) can be obtained by shifting the input value by one, two, and three bits to the left. Now design the gate circuit constructed by the function expression \( f \), as shown in Figure 2.

![Figure 2. The gate circuit design drawing of S-box.](image-url)
For the purpose of further improve the overall security of the algorithm, the scheme proposes to transform the 4-bit input of the S-box to the compound domain $GF((2^2)^2)$ for calculation, further reducing the consumption of hardware resources. The architecture diagram of the composite domain is shown in Figure 3. The multiplier provides operations on the field $GF(2^2)$, and Sq.sc is the square counter.

Figure 3. The composite domain implementation architecture diagram of S-box.

Based on the work of Liang[15] and Li[16], the specific scheme of compound domain realization is described here:

In the first stage, three groups of 4-bit data $a_1$, $a_2$, and $a_3$ are input to the register. After mapping, they become 6 groups of 2-bit data, which are respectively $b_1$, $b_2$, $b_3$, $b_4$, $b_5$, $b_6$ in order. Divide 6 2-bit data into two groups, namely $S_1$ ($b_1$, $b_2$, $b_3$) and $S_2$ ($b_4$, $b_5$, $b_6$). After $S_1$ and $S_2$ are subjected to 3 exclusive OR operation, they are input to the $GF(2^2)$ squarer, and then the data is input to the $GF(2^2)$ multiplier. After obtaining 3 groups of 2 bit information $c_1$, $c_2$, $c_3$ and 3 groups of 2 bit information $d_1$, $d_2$, $d_3$, the 3 groups of information are XORed to obtain $e_1$, $e_2$, $e_3$, and the obtained values are input into the register.

Figure 4. The first stage of the scheme.

In order to meet the uniformity of the threshold scheme, the method of adding a dummy value can be used[17], adding a 4 bit dummy value $z$ when entering the multiplier, and dividing $z$ into 3 groups of 2 bit information $z_1$, $z_2$, $z_3$, and $b_1$, $b_2$, $b_3$, $b_4$, $b_5$, $b_6$ enter the 9×3 multiplier at the same time, and get $d_1$, $d_2$, $d_3$. The final calculation is as follows:
After entering the second stage, the decomposition method is introduced to find the inversion, and the obtained 3 groups of 2 bit information are input into the register and then regrouped to obtain 4 groups of 4 bit information, and then input to the inverter, and the inverter finally runs as follows:

\[
\begin{align*}
\text{Inv}(m_1^*, m_2^*, m_3^*, m_4^*) &= (f_1, f_2, f_3, f_4) \\
\end{align*}
\]

\[
\begin{align*}
\text{Inv}(m_1^*, m_2^*, m_3^*, m_4^*) &= (f_1, f_2, f_3, f_4) \\

f_1^* &= m_2^* \oplus m_3^* \oplus m_1^* m_3^* \oplus m_2^* m_3^* \oplus m_2^* m_3^* m_4^* \\
f_2^* &= m_4^* \oplus m_1^* m_3^* \oplus m_2^* m_3^* \oplus m_1^* m_2^* m_4^* \oplus m_1^* m_2^* m_3^* \\
f_3^* &= m_1^* \oplus m_2^* \oplus m_1^* m_3^* \oplus m_1^* m_4^* \oplus m_1^* m_2^* m_4^* \oplus m_1^* m_2^* m_3^* \\
f_4^* &= m_2^* \oplus m_1^* m_3^* \oplus m_1^* m_4^* \oplus m_2^* m_4^* \oplus m_2^* m_3^* \oplus m_2^* m_3^* m_4^* \\
\end{align*}
\]

Input \(f_1^*, f_2^*, f_3^*, \) and \(f_4^* \) into the register to retrieve 3 sets of 2 bit information \((f_1, f_2, f_3)\), and output them to the third stage.

**Fig. 5 The second stage of the scheme.**

**Fig. 6. The third stage of the scheme.**
In the third stage, \( b_1, b_2, b_3 \) and \( f_1, f_2, f_3 \) are input to the \( GF(2^2) \) multiplier, and the virtual value method is continued to obtain three sets of 2 bit data \( (h_1, h_2, h_3) \). And \( b_4, b_5, b_6 \) also perform the same operation as \( f_1, f_2, f_3 \) to get three sets of 2 bit data \( (h_4, h_5, h_6) \). The obtained 6 sets of 2 bit data are combined in pairs to obtain 3 sets of 4 bit data \( (k_1, k_2, k_3) \), and the obtained 4 bit data is input into the affine and isomorphic mapping part to obtain the final 3 sets of 4 bit data \( (l_1, l_2, l_3) \), add the three sets of data to get the final output result.

After outputting the 4-bit data, it is put into the gate circuit to generate the final bit value.

5. The standard of lightweight cryptographic algorithm

After the concept of lightweight cryptographic algorithm was proposed, it has not been strictly defined so far. When designing, designers are based on specific implementation environment and requirements\(^{[18]}\). Most designers first consider the hardware implementation area of the algorithm. The size of the hardware implementation area is used as a measure of the "weight" of the algorithm. It also needs to consider parameters such as energy consumption, throughput, and delay. According to the consideration of different scholars, different lightweight standards can be proposed according to the implementation environment of the algorithm. The implementation environment is divided into: hardware implementation environment and software implementation environment\(^{[19]}\).

5.1. Lightweight standards in the hardware environment

The lightweight standard in the hardware environment is mainly the consumption of hardware resources. The specific standard is the number of logic gates required for algorithm implementation. This variable is based on GE (Gate Equivalence) as a unit. One GE means that one NAND gate is consumed in the algorithm implementation. Another important criterion is the data throughput (Throughout), that is, the number of data bits per second. Among them, in the field of side-channel attacks, the power consumption output of the cryptographic chip during operation should also be considered to resist power consumption attacks. Different designers have different considerations when designing. But under normal circumstances, the standard for lightweight algorithms is to obtain the largest data throughput with the smallest possible hardware implementation area, and to reduce the power output to a minimum in the process.

5.2. Lightweight standards in the software environment

The operating weight in the software environment mainly considers two indicators: time and storage complexity. Use the number of clock cycles required to encrypt or decrypt a byte of data as a criterion for judging time complexity. However, due to the problem of experimental accuracy, this method is not very practical. In most cases, the overall delay of the algorithm is considered, that is, the number of clock cycles required for the algorithm to run completely. The storage complexity mainly considers the amount of RAM and Flash space occupied by the algorithm operation.

5.3. Summary

The purpose of the lightweight scheme proposed in this paper is to implement hardware, so try to design algorithms based on lightweight standards in the hardware environment. Among them, in the threshold scheme, the Boolean function that implements the S-box is reduced from 3 times to 2 times, which greatly reduces the complexity and overhead of circuit implementation. In the 4-bit S-box, the format of the Boolean function expression of the first bit and the remaining three bits is the same, only the input value is different. Therefore, the same circuit can be reused, reducing resource consumption during hardware implementation.

6. Hardware implementation of algorithm

According to the lightweight password algorithm scheme proposed in this paper, the hardware implementation\(^{[20]}\) is carried out. In Verilog programming, the output of S-box is no longer realized by the original Look-Up Table method (LUT), but by the circuit scheme proposed in Chapter 3, which
greatly saves the hardware resources. After the program is implemented in code, it is put into Quartus software for compilation\cite{21}, under the TSMC 60 nm process, the constrained frequency is 50MHz. Among them, GE, as the basic logic unit on the chip, is an important indicator to measure hardware resources. Among them, the S-box module diagram after software packaging is shown in Figure 3.

![Figure 3. The circuit diagram of S-box after packaging.](image)

Table 4. Comparison of hardware resource consumption

| Scheme            | Number of logical units used | Down  |
|-------------------|------------------------------|-------|
| Literature [20]   | 8223                         | 65.91%|
| Literature [22]   | 5268                         | 46.79%|
| This article scheme | 2803                         |       |

Because this article implements the lightweight algorithm of SM4 algorithm, it is normal to reduce hardware resource consumption, but compared with the normal SM4 algorithm hardware implementation, the reduction of hardware resources is not only due to the reduction of the number of iterations and the number of input and output bits. In the process of lightweight algorithm, each output value is represented by a Boolean function, and a sequential shift method is used to complete the output of each output value. And the Boolean function can significantly reduce the consumption of hardware resources after reducing the highest number of algebras from 3 to 2 digits. It proves that the lightweight implementation of SM4 algorithm is successful in hardware.

7. Conclusion

According to the data obtained in the above table, it can be concluded that under the condition of ensuring the security of the cryptographic algorithm, the hardware resources required to implement the lightweight algorithm are 2803 logic units, which meets the standards of lightweight cryptographic algorithms in the hardware environment. And the program can be implemented in hardware. The research has not verified the various indicators of the cryptographic algorithm in the software environment, and the security of the solution against side-channel attacks has not been verified by experiments, and further verification is needed in the next step. Compared with the literature [3], this paper applies it to specific algorithms based on the S-box safety standard proposed by it, and completes the hardware implementation. Cryptographic chips using lightweight algorithms will also be applied to the construction of side-channel platforms in the next step, and used to study side-channel attacks and defense solutions against lightweight cryptographic algorithms.
Acknowledgements
This paper is supported by the National Social Science Foundation project "Network Anomaly Comment Mining Method and Application Research", project number: 16BTJ033

References
[1] Liu B Z. (2016) Design and analysis of S-box, a lightweight block cipher algorithm. Shanghai Jiaotong University:1-9
[2] National Commercial Password Management Office. (2016) SMS4 password algorithm used by wireless LAN products. http://www.oscca.gov.cn/UpFile/200621016423197990.pdf.
[3] Yang W, Wan W N, Chen Y, Zhang Y T. (2014) Overview of lightweight cryptography for restricted devices. Computer Applications, 34(07): 1871-1877.
[4] Zhang J Z. (2020) Design and analysis of lightweight block cipher algorithm for IoT applications. University of Electronic Science and Technology of China:16-27
[5] Li F, Zhang W Y, Xing C H. (2020) Design of a safe and lightweight S-box suitable for software and hardware implementation. Computer Application Research. 37(12):3717-3721.
[6] Ghoshal A, Sadhukhan R, Patranabis S, et al. (2018) Lightweight and Sidechannel Secure 4× 4 S-Boxes from Cellular Automata Rules. IACR Transactions on Symmetric Cryptology: 311-334.
[7] Canteaut A, Duval S, Leurent G. (2015) Construction of lightweight S-boxes using Feistel and MISTY structures. In: International Conference on Se-lected Areas in Cryptography. Berlin. 373-393.
[8] Naji A W, Hameed S A, Zaidan B B, et al. (2009) Novel framework for hidden data in the image page within executable file using computation between advanced encryption standard and distortion techniques. Computer Science, 106 (2746): 135-40.
[9] Boss E, Grosso V, Güneysu T, et al. (2017) Strong 8-bit Sboxes with efficient masking in hardware ex-tended version. Journal of Cryptographic En-gineering, 7 (2): 149-165.
[10] Picek S, Mariot L, Yang B, et al. (2017) Design of S-boxes defined with cellular automata rules. In: Pro-ceedings of the Computing Frontiers Conference. New York. 409-414.
[11] Mariot L, Picek S, Leporati A, et al. (2019) Cellular au-tomata based S-boxes. Cryptography and Communications, 11 (1): 41-62.
[12] Leander G, Poschmann A. (2017) On the classification of 4 bit s-boxes. International Workshop on the Arithmetic of Finite Fields. Berlin.159-176.
[13] Reparaz O, Bilgin B, Nikova S, et al. (2019) Consolidat-ing masking schemes. Annual Cryptology Conference. Berlin. 764-783.
[14] Kutzner S, Nguyen P H, Poschmann A. (2013) Enabling 3-share threshold implementations for all 4-bit s-boxes. International Conference on Infor-mation Security and Cryptolo-gy. Cham. 91-108.
[15] LIANG H, WU L, ZHANG X, et al. (2014) Design of a masked S-box for SM4 based on composite field. In: Proceedings of Tenth International Conference on Computational Intelligence and Security. IEEE. 2014: 387–391.
[16] LI X C, ZHONG W D, ZHANG S W, YANG X Y. (2018) A new threshold implementation of the S-box in SM4. Journal of Cryptologic Research, 5(6): 641–650.
[17] BILGIN B, NIKOVA S, NIKOVA V, et al. (2015) Threshold implementations of small S-boxes. Cryptography and Communications, 7(1): 3-33.
[18] Chen Huafeng. (2018) Research on the design of lightweight block cipher algorithm. Information Technology and Network Security, 37(11): 25-29.
[19] BIRYUKOVA, PERRIN L. On Lightweightness (2018) https: //www.Cryptolux.org /index. php /On_Lightweightness#cite_note-SE12-1.
[20] ZHANG L H, WU S, JIANG T F, JIANG P P. (2018) SMS4 algorithm implementation and online verification based on FPGA. Journal of East China Jiaotong University,35(05):111-116.
[21] HUANG J Y, PAN S (2018) EDA Technology Practical Course-Verilog HDL Edition (Sixth Edition). Science Press, Beijing.

[22] CAI Y Y, QU Y J. (2016) Research and design of SMS4 encryption chip based on single-round loop structure. Electronic Design Engineering, 24(22):39-42+46.