Realization of combinatorial optimization of small-scale S-box

Yating Huang$^{1,2}$, Ming Mao$^1$, Yanjun Li$^{1,2*}$ and Weiguo Zhang$^1$

$^1$Beijing Electronic Science and Technology Institute, Beijing, 100070, China
$^2$Guangxi Key Laboratory of Cryptography and Information Security, Guilin University of Electronic Technology, Guilin, Guangxi, 541004, China
$^*$Corresponding author’s e-mail: lyj@besti.edu.cn

Abstract. This method describes a new technology for combinatorial logic optimization in detail, which can combine multiple criteria to reduce the hardware implementation area of the S-box. This technique can be achieved in two steps. The first is to optimize the non-linear part of the S-box. According to the optimization criterion of multiplication complexity, the quality of the S-box nonlinear part optimization can be judged, and the realization of the S-box with the smallest multiplication complexity can be obtained. The second step is to optimize the linear part of the S-box, and optimize on the basis of the results of the first step, focusing on reducing the number of XOR gates, and the optimization is performed through a heuristic-based algorithm. The above combinatorial logic optimization technology can be applied to any small S-box ($5 \times 5$ and below). Finally, the S-box of PRESENT algorithm and CTC2 algorithm are used as examples to illustrate the optimization effect, and the optimal realization under the minimum AND gate condition is obtained.

1. Introduction

In cryptography, a logical basis widely used to represent circuits is \{AND, XOR, NOT\}, that is, AND gates, XOR gates and NOT gates. Among them, reducing the number of AND gates is critical. The fewer the number of AND gates, the better the ability to resist rapid algebraic attacks. Meanwhile, when the number of AND gates increase in the technology of protecting side channel attacks, the cost of general protection will also increase.

With the booming Internet of Things, in resource-constrained situations, the requirements for information security are becoming stronger and stronger. Circuit area is a key criterion for lightweight cryptography in hardware, because smaller devices mean stronger area constraints. When people consider password security, they also focus on a smaller implementation area in hardware implementation. Therefore, many cryptographic schemes commonly use hardware implementation gates as one of the metrics. We also focus on minimizing the circuit area as much as possible. The S-box is the most expensive part of most lightweight implementations, so our goal is to reduce the consumption of the S-box.

In 2016, Stoffelen[1] used the SAT solver to optimize the S-box according to multiple optimization criteria such as multiplication complexity, bit slice gate complexity, and gate complexity, found the smallest implementation for the S-box of multiple cryptographic algorithms. Through observation, we found that there is no circuit that has both the best Boolean complexity and the best multiplication complexity. Therefore, the method we choose is divided into two steps. First, from the perspective of optimizing the multiplication complexity, we reduce the non-linearity of the circuit, and then optimize the linear component of the circuit. Such a two-step method usually produces a better circuit.
The combinatorial optimization method described in this article can use as few AND gates and XOR gates as possible in the realization of S-boxes, and this technology can be applied to any small S-boxes. Finally, we applied this technique to CTC2 algorithm[2] and PRESENT algorithm[3], and both obtained better circuit implementation results than the previous ones.

2. Optimization of non-linearity

2.1. Multiplication complexity

The definition of the multiplication complexity[4] is the minimum number of nonlinear gates required to calculate a function, where the gate is 2 inputs. And then, if we limit the implementation of S-box to AND, OR, XOR, and NOT operations, then the multiplication complexity refers to the number of AND gates and OR gates.

2.2. SAT solver

The SAT solver is the Boolean Satisfiability Problem solver. There are many SAT solvers that can be used directly, such as MiniSat[5] and CryptoMiniSat[6]. Give the SAT solver a logical proposition, including AND or NOR logical symbols and a number of Boolean variables, it can determine whether the logical proposition can be satisfied according to this expression, and then determine whether the proposition is true, which can help us solve various circuit complexity decision problems.

The SAT solution method also has limitations, mainly because there are too many combinations of SAT problems, so it is only suitable for small S-boxes. In addition, the answer obtained by the SAT solution does not give a general solution for the multiplication complexity of a given size S-box. In other words, even if two S-boxes are 4×4, their multiplication complexity may be different.

2.3. Steps to optimize multiplication complexity

Courtois[7,8] proposed a method for coding multiplicative complexity decision problems[9]. After solving this problem, the circuit with the optimal multiplication complexity is obtained.

Let \( f : 2^n \rightarrow 2^m \) be the S-box and \( k \) be the tested multiplication complexity. First, we build a set of equations \( F \) in ANF (algebraic normal form), which is composed of:

\[
q_i = a_i + \left( \sum_{j=0}^{\frac{n-1}{2}} a_{i+2j} \cdot x_j \right) + \left( \sum_{j=0}^{\frac{n+1}{2}} a_{i+2j+1} \cdot t_j \right), \quad i \in \{0, ..., 2k-1\}, \quad l = i(n+1) + \left[ \frac{i^2 - 2i + 1}{4} \right] \tag{1}
\]

\[
t_i = q_{2i} \cdot q_{2i+1}, \quad i \in \{0, ..., k-1\} \tag{2}
\]

\[
y_i = (\sum_{j=0}^{\frac{k-1}{2}} a_{s+2j} \cdot x_j) + (\sum_{j=0}^{\frac{k+1}{2}} a_{s+2j+1} \cdot t_j), \quad i \in \{0, ..., m-1\}, \quad s = 2k(n+1) + k(l-1) + i(n+k) \tag{3}
\]

Among them, \( x_i, y_i \) respectively represent the input and the output variable of the S-box; \( q_i, t_i \) respectively represent the input and the output variable of the AND gate; \( a_i \) represents the variable connected between the gates.

Equation (1) finds the input of the AND gate, which can be any XOR between the input of the S-box and the output of the previous AND gate, and the single \( a \) in the equation represents a NOT gate. Equation (2) is used to encode \( k \) AND gates. Equation (3) obtains the output of the S-box, which can be any linear combination of the input of the S-box and the output of the AND gate.

The next step is to build a set of equations \( F' \), which containing \( 2n \) copies of the equations in \( F \). In \( F' \), all \( x_i, y_i, q_i, t_i \) need to be renumbered, but all the \( a_i, b_i \) remain unchanged. \( f \) expands the truth table to \( 2^n (n+m) \) constant equations and adds them to \( F' \), so that each bit of the input and output of the S-box corresponds to a constant equation, then the problem description can be constrained.
From this we get $F'$, but $F'$ is in ANF. In order to use the SAT solver to solve the problems, we need to encode these problems into the input format required by the SAT solver, that is, to convert the ANF to the CNF (conjunctive normal form)[10]. Next, input the CNF expression into the SAT solver, then we can get the solution of the output problem.

In the CNF format, all variables are replaced by numbers. At this point, we need to convert the solution found by the SAT solver back to the original variable name. Finally, we can reconstruct the S-box from the original ANF and the output of the previous program. In this way, we have got the circuit realization with the least multiplication complexity.

3. Optimization of the number of XOR gates

After optimizing the multiplication complexity, we will get a result of XOR gates with a lot of redundancy. However, the cost of XOR gates in hardware implementation is very expensive. Improving this part will result in a large degree of savings in the resulting circuit. So next we need to find the maximum linear component of the circuit and reduce the number of XOR gates as much as possible.

3.1. The shortest linear program problem

Before optimizing the number of XOR gates, we first need to understand the shortest linear program (SLP) problem on $GF(2)$ [1]. Let $M$ be the constant matrix of $m \times n$ on $GF(2)$, and $x$ be the vector of $n$ variables on $GF(2)$. The SLP problem is to find the shortest path to calculate $Mx$, where each line of the program is in a definite form. Solving the SLP problem on $GF(2)$ is the process of finding a circuit with only XOR gates and then minimizing its number.

3.2. Heuristic Optimization Algorithm

Boyar, Matthews, and Peralta[11] pointed out that the SLP problem on $GF(2)$ is NP-hard. To this end, we choose a heuristic algorithm proposed by Boyar and Peralta[12], which can find a solution for the SLP problem instance on $GF(2)$. Blindly calculating XOR directly in the matrix representation is usually very inefficient, because there will be a lot of repeated calculations. At present, the basic idea to find the least XOR gate circuit is that we need to reuse the intermediate variables that have been calculated for other calculation steps. When considering a new basic element, assuming that the distance is 5, then only 4 combinations of the old and new basic elements need to be considered. This strategy is also the basis of the heuristic algorithm we are discussing.

Let $S$ be a basic vector set, which is a set of variables $x_1, x_2, \ldots, x_n$. The distance vector $Dist[i]$ is the minimum Hamming distance from each row in $M$ to $S$, that is, $Dist[i] = \delta(S, f_i)$, where $f_i$ is the $i$-th row of $M$ multiplied by the input vector $x$. The initial $Dist[i]$ is equal to the Hamming weight of the $i$-th row minus one. Then, execute the following loop:

- Combine two existing basic elements to form a new basic element and add it to $S$;
- Update according to the new $S$;
- Repeat process 1, 2 until $Dist[i] = 0$ for all $i$.

The criterion for selecting a new basic element is to choose an element to minimize the sum of the new distance. When we select different elements but produce the same distance, then we choose the one that maximizes the Euclidean norm of the new vector. Specific examples will be given in 4.
4. Combination Optimization: Optimize the S-box of PRESENT and CTC2

4.1. S-box Optimization of PRESENT Algorithm

The first step is to optimize the multiplication complexity. Through the method described in Chapter 1, we found a solution with a multiplication complexity of 4, as shown below:

\[ q_0 = -(x_i \oplus x_i) \quad t_i = q_i \land q_i \]
\[ q_i = -(x_0 \oplus x_2) \quad q_i = x_0 \oplus x_1 \oplus x_2 \oplus t_2 \]
\[ t_0 = q_0 \land q_i \quad q_i = -(x_0 \oplus x_1 \oplus x_2 \oplus x_3 \oplus t_0) \]
\[ q_1 = x_0 \oplus x_1 \oplus x_2 \oplus t_0 \quad t_i = q_i \land q_i \]
\[ q_i = x_0 \oplus x_2 \oplus t_0 \quad y_0 = x_0 \oplus x_1 \oplus t_i \]
\[ t_i = q_i \land q_i \quad y_i = x_0 \oplus x_1 \oplus t_0 \oplus t_i \]
\[ q_i = -(x_i \oplus x_i) \quad y_i = x_0 \oplus x_1 \oplus t_i \]

Observing the above results, we can find that some XOR operations are redundant. We divide the XOR operation according to the before and after of the AND gate. Next, we take the entire process of optimizing \( x_i \) as an example to describe in detail the optimization of the heuristic algorithm.

First, we convert the XOR involving \( x_i \) into a matrix \( M \).

\[
M^T = \begin{bmatrix}
0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\
1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}
\]

The initial distance vector is \( D = [1,1,2,1,0,1,2,3,1,1,1,2] \) corresponding to \([1,0,0,0,0,1,0,0,0,0,1,0]\) respectively. We need to find two basic vectors from them. When the sum of these two vectors is added to the basic vector, a new minimum distance vector will be obtained. The next step has been to proceed in this way. We will enumerate the whole process of optimization.

Step 1: \( \omega_1 = x_0 + x_1 \), \( D = [1,1,1,1,0,1,2,2,0,1,1,1] \);
Step 2: \( \omega_2 = x_0 + x_2 \), \( D = [1,0,1,0,1,0,1,2,0,1,1,1] \);
Step 3: \( \omega_3 = x_0 + x_3 \), \( D = [1,0,1,0,0,1,1,1,2,0,1,1] \);
Step 4: \( \omega_4 = x_0 + x_2 \), \( D = [0,0,0,0,1,0,1,1,1,0,1,1] \);
Step 5: \( \omega_5 = x_0 + x_3 \), \( D = [0,0,0,0,0,1,1,1,0,1,1,1] \);
Step 6: \( \omega_6 = x_2 + x_1 \), \( D = [0,0,0,0,0,0,0,1,1,0,1,0] \);
Step 7: \( \omega_7 = x_0 + x_1 \), \( D = [0,0,0,0,0,0,1,1,1,0,0,0] \);
Step 8: \( \omega_8 = x_0 + x_6 \), \( D = [0,0,0,0,0,0,0,0,1,0,0,0] \);
Step 9: \( \omega_9 = x_7 + x_1 \), \( D = [0,0,0,0,0,0,0,0,0,0,0,0] \);

When the result that the distance vector \( D \) is all 0 is obtained, it means that the optimization is over. After the above optimization, we can get the following realization:

\[
\begin{align*}
 z_0 &= x_1 \oplus x_2 & t_i &= q_i \land q_i \\
 q_0 &= \neg z_0 & z_2 &= x_2 \oplus x_3 \\
 q_i &= -(x_0 \oplus x_i) & z_i &= x_0 \oplus z_2
\end{align*}
\]
Observing the above implementation, we can find that $y_1$ and $y_2$ have the same $zt$, and one more XOR can be reduced. From this we can draw the S-box implementation as shown in the figure 1, and the results of 4 AND gates, 19 XOR gates and 3 NOT gates are obtained, which is one XOR gate less than the optimized result of Courtois'[8].

### 4.2. S-box optimization of CTC2 algorithm

We used the same method to optimize the S-box of CTC2, and got the results of 3 AND gates, 7 XOR gates, and 5 NOT gates. Compared with the optimized result of Courtois'[8], our result has 2 fewer XOR gates, which reduces the area of hardware implementation. The realization of the S box of CTC2 is shown in figure 2.

$$t_0 = q_0 \land q_1$$
$$q_0 = z_0 \oplus t_2$$
$$q_1 = x_0 \oplus z_0 \oplus t_0$$
$$q_2 = z_1 \oplus t_o$$
$$q_3 = t_0 \land q_1$$
$$t_1 = q_2 \land q_1$$
$$z_0 = x_0 \oplus x_1$$
$$y_0 = x_0 \oplus x_1 \oplus t_1$$
$$y_1 = z_1 \oplus t_0 \oplus t_2$$

4.3. Comparison of optimization results

The specific implementation results are compared in table 1. It should be noted that the area required for different circuit components is different, and we use the equivalent gate (GE) to measure it. Under the SMIC 130nm process library, a NOT gate requires 0.67GE, and an XOR gate requires 2.33GE.
Therefore, although our method slightly increases the number of NOT gates, due to the decrease in the number of XOR gates, the total area of one S-box used by PRESENT and CTC2 algorithm is still reduced 0.99 and 3.99 respectively compared with previous methods.

Table 1. S-box realization result

| Cryptographic algorithm | Multiplication complexity | XOR gate | Not gate | reference       |
|-------------------------|---------------------------|----------|----------|----------------|
| PRESENT                 | 4                         | 20       | 1        | [8]            |
| PRESENT                 | 4                         | 19       | 3        | This article   |
| CTC2                    | 3                         | 9        | 4        | [8]            |
| CTC2                    | 3                         | 7        | 5        | This article   |

5. Summary and outlook

We have optimized the S-boxes of PRESENT algorithm and CTC2 algorithm respectively. With the smallest multiplication complexity, we have obtained the smallest S-box implementation at present, which reduces the area required for hardware implementation. Among them, the S-box implementation of CTC2 requires 3 AND gates, 7 XOR gates, and 5 NOT gates, and the S-box of the present algorithm requires 4 AND gates, 19 XOR gates and 3 NOT gates. Next, we will optimize the heuristic algorithm and think about whether it can be used in a larger-scale S-box to reduce its implementation area.

Acknowledgments

This work is supported by the Fundamental Research Funds for the Guangxi Key Laboratory of Cryptography and Information Security (GCIS201912).

References

[1] Stoffelen, K. (2016) Optimizing S-Box Implementations for Several Criteria Using SAT Solvers. In: International Conference on Fast Software Encryption, FSE 2016, Bochum. pp. 140–160.

[2] Courtois, Nicolas, T. (2007) CTC2 and Fast Algebraic Attacks on Block Ciphers Revisited. IACR Cryptology ePrint Archive, 2007: 2007/152. https://eprint.iacr.org/2007/152.pdf

[3] Bogdanov, A., Knudsen, L.R., Leander, G., et al. (2007) PRESENT: An ultra-lightweight block cipher. In: Cryptographic Hardware and Embedded Systems—CHES 2007. Vienna. pp. 450–466.

[4] Boyar, J., Peralta, R., Pochuev, D. (2000) On the Multiplicative Complexity of Boolean Functions over the Basis \( \land, \lor, 1 \). Theoretical Computer Science, 235(1): 43–57.

[5] Sörensson, N., Een, N. (2005) MiniSat v1.13 - a SAT solver with conflict-clause minimization. In: International Conference on Theory and Applications of Satisfiability Testing. St. Andrews. 53:1-2.

[6] Soos, M. (2009) CryptoMiniSat 2.5.0 SAT , Berlin: Springer Press, 2009: Soos, M., Nohl, K., Castelluccia, C. (2009) Extending SAT Solvers to Cryptographic Problems. In: International Conference on Theory and Applications of Satisfiability Testing. Swansea. pp. 244-257.

[7] Fuhs, C., Schneider-Kamp, P. (2010) Synthesizing Shortest Linear Straight-Line Programs over GF(2) Using SAT. In: International Conference on Theory and Applications of Satisfiability Testing. Edinburgh. pp. 71–84.

[8] Courtois, N.T., Mourouzis, T.D., Hulme. (2013) Exact Logic Minimization and Multiplicative Complexity of Concrete Algebraic and Cryptographic Circuits. International Journal on Advances in Intelligent Systems, 6: 165–176.

[9] Mourouzis, T. (2015) Optimizations in Algebraic and Differential Cryptanalysis. Doctoral Thesis, UCL (University College London), London.

[10] Bard, G.V., Courtois, N.T., Jefferson, C., (2007) Efficient Methods for Conversion and Solution of Sparse Systems of Low-Degree Multivariate Polynomials over GF(2) via SAT-Solvers. IACR Cryptology ePrint Archive, 2007:2007/024. https://eprint.iacr.org/2007/024.pdf
[11] Boyar, J., Matthews, P., Peralta, R. (2008) On the shortest linear straight-line program for computing linear forms. In: Mathematical Foundations of Computer Science. Toruń. pp. 168–179.

[12] Boyar, J., Peralta, R. (2010) A new combinational logic minimization technique with applications to cryptology. Experimental Algorithms. SEA 2010. 6049: 178–189.