A Very Compact Masked S-Box for High-Performance Implementation of SM4 Based on Composite Field

Hailiang Fu[1], Guoqiang Bai[1,2](✉), and Xingjun Wu[1]

[1] Institute of Microelectronics, Tsinghua University, Beijing, China
fhl14@mails.tsinghua.edu.cn, baigq@mail.tsinghua.edu.cn
[2] Tsinghua National Laboratory for Information Science and Technology, Beijing, China

Abstract. Implementations of the SM4 algorithm, including different hardware applications with limited resources, are vulnerable to Side-Channel Attacks. This paper presents a countermeasure against such attacks by adding a random "mask" to the input plaintext and protect all variables through the whole encryption process. As is known to all, the unique nonlinear step in each round of SM4 algorithm is the "S-Box" and the previous works using lookup-table method to implement the S-Box always incur large area and high power. Here we give the compact design of masked S-Box using the normal basis in the composite field (consisting of a Galois inversion and several affine transformations). Then we compute the different masks diffused to all the steps in the SM4 algorithm process. The proposed design results in ultra-low cost of hardware and capability to resist first-order differential power analysis (DPA), which is suitable for the resource constrained devices. The synthesis result of masked S-Box shows that the area under the SMIC 0.13 μm is only about 978-gates, 46.8% fewer than the other works. Further, we apply the pipeline technique to our proposed "masked S-Box", thereby to the whole masked SM4 algorithm. The results of FPGA implementation present that our works have achieved an ultra-high speed with frequency nearly 551 MHz and the throughput over 70 Gbps.

Keywords: SM4 · S-Box · Mask · Pipeline · Composite field

1 Introduction

With the rapid development of the computer science and internet technology in the modern world, the information and data security have become more and more important. Thus, preventing the significant information from attacking by any other unauthorized parts is a challenging and essential task. There is no doubt that the security of hardware is the basis for data transmission, especially in the Wireless Local Area Network (WLAN). For this reason, plenty of methods about hardware cryptography (e.g. hiding, masking, etc.) have been come up with to protect the sensitive data and applied in different domains, such as embedded systems, wireless handsets and smart cards. In January 2006, the Office of State Commercial Cipher Administration of China (OSCCA) announced a specific encryption standard named SM4 block cipher, the purpose of which is to form the Wireless LAN Authentication and Privacy Infrastructure (WAPI) standard for our country [1]. Since then, there have been a large variety of researches focusing on improving the performance and security of SM4. On the other hand, some researchers try to seek the weakness of SM4 algorithm and do attacks on the specific hardware implementations. For example, smart cards may be vulnerable to first order side-channel attacks such as differential power analysis, which takes advantages of the leakage of information to do the physical analysis such as power consumption, electromagnetic radiation and so on, then to deduce the real secret key of the algorithm.

Due to the potential attacks above, this paper proposes a countermeasure against the first order side-channel attacks, applying the masking strategy to the nonlinear S-Box as well as the data path in the SM4 algorithm based on the composite field introduced by the previous work [2]. Compared to the other method to achieve the masking, this protection saves 46.8% area for the whole circuit. However, it incurs some other parts which slow down the encryption process. Thus we make use of the pipeline technique to accelerate the calculation, resulting in an ultra-high clock frequency up to 551 MHz and throughput over 70 Gbps for the masked SM4 algorithm.

The organization of this paper is as follows. In Sect. 2, we describe the SM4 block cipher and the algebraic description of S-Box very briefly. Section 3 shows the detailed masking strategy for S-Box, including masking the inversion and the affine transformation, and the reutilization of the masks. Section 4 presents the implementation of the SM4 algorithm using the masked S-Box. Also, the architecture of pipelined masked SM4 is designed and implemented in this part. Then we state the low-cost results of area using masking strategy and the high speed in pipeline scheme for SM4 in Sect. 5. At last, Sect. 6 concludes the paper.

2 Algebraic Description for S-Box

SM4 block cipher is a 32-round iterative algorithm with 128-bit input plaintext, secret key and output ciphertext. The input plaintext is first divided into four words and each word consists of 32 bits. Before encryption, the key for each round (rk_i) will be generated through the key expansion arithmetic, which is nearly identical with the encryption process, and the only difference between them is the linear part—round shifting left. With the rk_i, a new word, i.e. X_{i+4} will be produced in the *i-th* round of the encryption process by doing XOR, nonlinear substitution and round shifting left operations ($X_{i+4} = X_i \oplus T(X_{i+1} \oplus X_{i+2} \oplus X_{i+3} \oplus rk_i)(i = 0, ..., 31)$), shown in Fig. 1. Finally, the order of the last four words will be reversed to form the output ciphertext. The XOR and round shifting left operations are linear with respect to the data block, so it provides "diffusion"; While the S-Box is the only nonlinear step that provides "confusion".

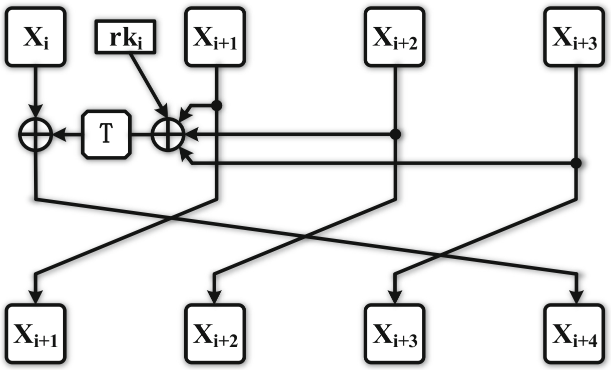

Fig. 1. SM4 round arithmetic

The S-Box can be implemented using the lookup tables, which occupies the majority of the cost in devices. In 2007, Liu et al. [3] gave the algebraic structure for SM4 algorithm, comprising two substeps: (i) regarding the byte as an element in the Galois Field $GF(2^8)$, get its inversion in this field (Note that the zero byte has no inversion, so it keeps unchanged); (ii) regarding the result of the inversion as a vector of bits in $GF(2^8)$, then multiply it by a given bit matrix and add a constant row vector, that is the procedure of an affine transformation. The inversion and affine transformation are shown below in Eq. (1):

$$S(\mathbf{X}) = I(\mathbf{X} \cdot A + \mathbf{C}) \cdot A + \mathbf{C}, \tag{1}$$

where the input of S-Box (S) is a 8-bit row vector $\mathbf{X} = \mathbf{X}_{7-0}$, and the cyclic matrix A in the algebraic expression is

$$A = \begin{pmatrix} 1\,1\,1\,0\,0\,1\,0\,1 \\ 1\,1\,1\,1\,0\,0\,1\,0 \\ 0\,1\,1\,1\,1\,0\,0\,1 \\ 1\,0\,1\,1\,1\,1\,0\,0 \\ 0\,1\,0\,1\,1\,1\,1\,0 \\ 0\,0\,1\,0\,1\,1\,1\,1 \\ 1\,0\,0\,1\,0\,1\,1\,1 \\ 1\,1\,0\,0\,1\,0\,1\,1 \end{pmatrix},$$

and the row vector C is

$$\mathbf{C} = \mathbf{C}_{7-0} = [1, 1, 0, 1, 0, 0, 1, 1].$$

For SM4 in the specific Galois Field, a byte represents a polynomial where the bits are coefficients of corresponding powers of x, and multiplication is modulo the irreducible primitive polynomial:

$$f(x) = x^8 + x^7 + x^6 + x^5 + x^4 + x^2 + 1.$$

We could consider the root of this polynomial as θ, then $f(\theta) = 0$ in $GF(2^8)$. Thus the bits of a byte could be related to the coefficients of powers of θ, e.g., $3 = \theta, 4 = \theta + 1, 9 = \theta^2$, etc. Therefore the bits make up a vector with respect to what is known as polynomial basis. However, we can change the representation of the polynomial basis in $GF(2^8)$ to a different one, named normal basis in composite field [4]. Instead of a vector of dimension 8 in $GF(2)$, we regard a byte as a vector of dimension 2 in $GF(2^4)$, where each 4-bit element is in turn a vector of dimension 2 in $GF(2^2)$, and each 2-bit element is a vector of dimension 2 in $GF(2)$. For each of these subfields, it has been introduced in details, referring to [5].

3 Masking Strategy

To convert the standard polynomial representation to the composite field representation, we need to choose the appropriate basis and build an isomorphic matrix. For more detailed information, please refer to [4]. In this paper, we try to add an additive mask to all the steps during the inversion, which will described below.

3.1 Inversion Without Masking

Now we apply the following convention: upper-case bold symbols stand for elements in the main field (e.g. $\mathbf{A} \in GF(2^8)$); upper-case italic symbols represent elements in the subfield (e.g. $A \in GF(2^4)$); lower-case bold symbols are for the sub-subfield (e.g. $\mathbf{a} \in GF(2^2)$); and lower-case italic symbols are used for single bits (e.g. $a \in GF(2)$).

To begin with, we don't concern about the mask. So the inversion in $GF(2^8)/GF(2^4)$ (this expresses the representation of $GF(2^8)$ as vectors in $GF(2^4)$ using a normal basis $[\mathbf{Y}^{16}, \mathbf{Y}]$), where $\mathbf{Y}^{16}$ and $\mathbf{Y}$ are the roots of $\mathbf{Y}^2 + \mathbf{Y} + N$ and $N \in GF(2^4)$ is the norm ($N = Y^{16} \cdot Y$), is given as [4]:

$$\mathbf{A} = A_h\mathbf{Y}^{16} + A_l\mathbf{Y}(\texttt{known}), \tag{2}$$

$$B = N \otimes (A_h \oplus A_l)^2 \oplus A_h \otimes A_l, \tag{3}$$

$$\mathbf{A}^{-1} = (A_l \otimes B^{-1})\mathbf{Y}^{16} + (A_h \otimes B^{-1})\mathbf{Y}(\texttt{result}). \tag{4}$$

Here we make a agreement on the meaning of the operators above: $\oplus$ and $\otimes$ denote addition and multiplication in Galois Field, respectively. The expression $A_h\mathbf{Y}^{16} + A_l\mathbf{Y}$ is an algebraic method using the normal basis to denote the vector $[A_h, A_l]$ (i.e. $[A_h, A_l] = [\mathbf{A}_{7-4}, \mathbf{A}_{3-0}]$). To achieve the inversion in $GF(2^8)$, it requires the inversion, addition, multiplication and the combined square-scaling operation ($N \otimes A^2$) in the subfield $GF(2^4)$. In the same way, the inversion in $GF(2^4)/GF(2^2)$ which uses a normal basis $[X^4, X]$, where the X^4 and X are the roots of $X^2 + X + \mathbf{n}$ (and $\mathbf{n} \in GF(2^2)$ is the norm ($\mathbf{n} = X^4 \cdot X$)), is given as:

$$B = \mathbf{b}_h X^4 + \mathbf{b}_l X(\mathtt{known}), \tag{5}$$

$$\mathbf{c} = \mathbf{n} \otimes (\mathbf{b}_h \oplus \mathbf{b}_l)^2 \oplus \mathbf{b}_h \otimes \mathbf{b}_l, \tag{6}$$

$$B^{-1} = (\mathbf{b}_l \otimes \mathbf{c}^{-1})X^4 + (\mathbf{b}_h \otimes \mathbf{c}^{-1})X(\mathtt{result}). \tag{7}$$

However, finding the inversion in the sub-subfield $GF(2^2)$, using the normal basis $[\mathbf{w}^2, \mathbf{w}]$, where $\mathbf{w}^2$ and $\mathbf{w}$ are the roots of $\mathbf{w}^2 + \mathbf{w} + 1$ (and here we define the norm as 1), is very easy. It is equivalent to the squaring operation, shown as a bit swap:

$$\mathbf{c} = c_h \mathbf{w}^2 + c_l \mathbf{w}(\mathtt{known}), \tag{8}$$

$$\mathbf{c}^{-1} = c_l \mathbf{w}^2 + c_h \mathbf{w}(\mathtt{result}). \tag{9}$$

All the steps above are used to obtain the inversion in $GF(2^8)$ without masking. In the following, we will detail the steps about how to mask the inversion.

3.2 Masking the Inversion

As is mentioned above, additive mask becomes our preference due to its resistance to zero-value attacks. It has been analyzed in [2] that the statistical distribution of masks is uniform over the field by adding a random mask. Therefore the operands appear randomly, uncorrelated to either the input plaintext or the secret key. Thus the data leaked from the side channel is independent of the chosen input plaintext, might regarded as noise, and the key in this way will be protected against first-order differential power attacks. To ensure the correct process from the input mask to the output mask, we apply the masking strategy as follows.

In $GF(2^8)$, we express the masked byte with a tilde (i.e. $\tilde{\mathbf{A}}$), and similarly for the other masked variables. Now we use the mask ($\mathbf{M}$) to mask the input plaintext.

$$\mathbf{M} = M_h \mathbf{Y}^{16} + M_l \mathbf{Y}; \tag{10}$$

$$\tilde{\mathbf{A}} = \mathbf{A} \oplus \mathbf{M} = \tilde{\mathbf{A}}_h \mathbf{Y}^{16} + \tilde{\mathbf{A}}_l \mathbf{Y} \tag{11}$$

Then let

$$\tilde{B} = N \otimes (\tilde{A}_h \oplus \tilde{A}_l)^2 \oplus \tilde{A}_h \otimes \tilde{A}_l \oplus \tilde{A}_h \otimes M_l \oplus \tilde{A}_l \otimes M_h \oplus M_h \otimes M_l, \tag{12}$$

$$M_2 = N \otimes (M_h \oplus M_l)^2, \tag{13}$$

Here the result $\tilde{B}$ is B above in Eq. (3) masked by M_2 (i.e. $\tilde{B} = B \oplus M_2$). Note that the products in Eq. (12) must be added in turn to make all the intermediate results uniformly distributed and masked, so that the information about the original data will not be leaked out.

For the inversion in $GF(2^4)$, say $\tilde{B} = \tilde{\mathbf{b}}_h X^4 + \tilde{\mathbf{b}}_l X$ and $M_2 = \mathbf{m}_h X^4 + \mathbf{m}_l X$, then let

$$\tilde{\mathbf{c}} = \mathbf{n} \otimes (\tilde{\mathbf{b}}_h \oplus \tilde{\mathbf{b}}_l)^2 \oplus \tilde{\mathbf{b}}_h \otimes \tilde{\mathbf{b}}_l \oplus \tilde{\mathbf{b}}_h \otimes \mathbf{m}_l \oplus \tilde{\mathbf{b}}_l \otimes \mathbf{m}_h \oplus \mathbf{m}_h \otimes \mathbf{m}_l, \tag{14}$$

$$\mathbf{p} = \mathbf{n} \otimes (\mathbf{m}_h \oplus \mathbf{m}_l)^2, \tag{15}$$

and $\tilde{\mathbf{c}}$ is $\mathbf{c}$ above in Eq. (6), masked by $\mathbf{p}$ (say $\mathbf{p} = p_h\mathbf{w}^2 + p_l\mathbf{w}$, and let $\mathbf{q} = \mathbf{p}^2 = \mathbf{n}^2 \otimes (\mathbf{m}_h \oplus \mathbf{m}_l) = p_l\mathbf{w}^2 + p_h\mathbf{w}$). Above we employ the convention of the inversion as squaring in the sub-subfield $GF(2^2)$, so

$$\tilde{\mathbf{c}}^{-1} = (\mathbf{c} \oplus \mathbf{p})^{-1} = (\mathbf{c} \oplus \mathbf{p})^2 = \mathbf{c}^2 \oplus \mathbf{p}^2 = \mathbf{c}^{-1} \oplus \mathbf{q}, \tag{16}$$

Therefore $\tilde{\mathbf{c}}^{-1}$ (say $\tilde{\mathbf{c}}^{-1} = \tilde{c}_l W^2 + \tilde{c}_h W$) is $\mathbf{c}^{-1}$ above in Eq. (9) masked by another mask $\mathbf{q}$.

Now we introduce a new 4-bit mask $S = \mathbf{s}_h X^4 + \mathbf{s}_l X$, and let

$$\begin{aligned}
\tilde{\mathbf{b}}_h^{-1} &= \mathbf{s}_h \oplus \mathbf{b}_h^{-1} = \mathbf{s}_h \oplus (\mathbf{b}_l \otimes \mathbf{c}^{-1}),\\
&= \mathbf{s}_h \oplus [(\tilde{\mathbf{b}}_l \oplus \mathbf{m}_l) \otimes (\tilde{\mathbf{c}}^{-1} \oplus \mathbf{q})],\\
&= \mathbf{s}_h \oplus \tilde{\mathbf{b}}_l \otimes \tilde{\mathbf{c}}^{-1} \oplus \tilde{\mathbf{b}}_l \otimes \mathbf{q} \oplus \mathbf{m}_l \otimes \tilde{\mathbf{c}}^{-1} \oplus \mathbf{m}_l \otimes \mathbf{q}, \qquad (17)\\
\tilde{\mathbf{b}}_l^{-1} &= \mathbf{s}_l \oplus \mathbf{b}_l^{-1} = \mathbf{s}_l \oplus (\mathbf{b}_h \otimes \mathbf{c}^{-1})\\
&= \mathbf{s}_l \oplus [(\tilde{\mathbf{b}}_h \oplus \mathbf{m}_h) \otimes (\tilde{\mathbf{c}}^{-1} \oplus \mathbf{q})]\\
&= \mathbf{s}_l \oplus \tilde{\mathbf{b}}_h \otimes \tilde{\mathbf{c}}^{-1} \oplus \tilde{\mathbf{b}}_h \otimes \mathbf{q} \oplus \mathbf{m}_h \otimes \tilde{\mathbf{c}}^{-1} \oplus \mathbf{m}_h \otimes \mathbf{q}, \qquad (18)
\end{aligned}$$

thus the result $\tilde{B}^{-1} = \tilde{\mathbf{b}}_h^{-1} X^4 + \tilde{\mathbf{b}}_l^{-1} X$ is B^{-1} above in Eq. (7) masked by S.

Similarly, apply the output 8-bit mask $\mathbf{T} = T_h\mathbf{Y}^{16} + T_l\mathbf{Y}$ to the output $\mathbf{A}^{-1}$, and let:

$$\tilde{A}_h^{-1} = T_h \oplus \tilde{A}_l \otimes \tilde{B}^{-1} \oplus \tilde{A}_l \otimes S \oplus M_l \otimes \tilde{B}^{-1} \oplus M_l \otimes S, \tag{19}$$

$$\tilde{A}_l^{-1} = T_l \oplus \ \tilde{A}_h \otimes \tilde{B}^{-1} \oplus \tilde{A}_h \otimes S \oplus M_h \otimes \tilde{B}^{-1} \oplus M_h \otimes S \tag{20}$$

So the result $\tilde{\mathbf{A}}^{-1} = \tilde{A}_h^{-1}\mathbf{Y}^{16} + \tilde{A}_l^{-1}\mathbf{Y}$ is the original inversion $\mathbf{A}^{-1}$ above in Eq. (4) masked by the output mask $\mathbf{T}$:

$$\tilde{\mathbf{A}}^{-1} = \mathbf{A}^{-1} \oplus \mathbf{T}. \tag{21}$$

3.3 Reutilization of Masks

Canright and Batina [2] shows the re-using of the masks to make the implementation more vulnerable to the higher-order differential side channel attacks and save the cost of the same operations. Firstly, by replacing the mask $\mathbf{q}$ by $\mathbf{m}_l$ or $\mathbf{m}_h$, we can modify the expression as follows:

$$\tilde{\mathbf{c}}^{-1} = (\tilde{\mathbf{c}}_l\mathbf{w}^2 + \tilde{\mathbf{c}}_h\mathbf{w}) \oplus \mathbf{m}_h \oplus \mathbf{q} \quad (\texttt{masked by } \mathbf{m}_h), \tag{22}$$

$$\tilde{\mathbf{b}}_h^{-1} = \mathbf{m}_{1h} \oplus \tilde{\mathbf{b}}_l \otimes \tilde{\mathbf{c}}^{-1} \oplus \underline{\tilde{\mathbf{b}}_l \otimes \mathbf{m}_h} \oplus \mathbf{m}_l \otimes \tilde{\mathbf{c}}^{-1} \oplus \underline{\mathbf{m}_l \otimes \mathbf{m}_h}, \tag{23}$$

$$\tilde{\mathbf{c}}_2^{-1} = \tilde{\mathbf{c}}^{-1} \oplus (\mathbf{m}_l \oplus \mathbf{m}_h) \quad (\texttt{masked by } \mathbf{m}_l), \tag{24}$$

$$\tilde{\mathbf{b}}_l^{-1} = \mathbf{m}_{1l} \oplus \tilde{\mathbf{b}}_h \otimes \tilde{\mathbf{c}}_2^{-1} \oplus \underline{\tilde{\mathbf{b}}_h \otimes \mathbf{m}_l} \oplus \mathbf{m}_h \otimes \tilde{\mathbf{c}}_2^{-1} \oplus \underline{\mathbf{m}_h \otimes \mathbf{m}_l}, \tag{25}$$

where the underlined products had already been calculated in Eq. (14), so here we can re-use these results. Now the result $\tilde{B}^{-1} = \tilde{\mathbf{b}}_h^{-1} X^4 + \tilde{\mathbf{b}}_l^{-1} X$ is B^{-1} above, but here masked by $M_h = \mathbf{m}_{1h}X^4 + \mathbf{m}_{1l}X$, which is the upper nibble of the input mask $\mathbf{M}$. In the same way, we get the updated masked $\mathbf{A}^{-1}$ in the following:

$$\tilde{A}_h^{-1} = T_h \oplus \tilde{A}_l \otimes \tilde{B}^{-1} \oplus \underline{\tilde{A}_l \otimes M_h} \oplus M_l \otimes \tilde{B}^{-1} \oplus \underline{M_l \otimes M_h}, \tag{26}$$

$$\tilde{B}_2^{-1} = \tilde{B}^{-1} \oplus M_l \oplus M_h \quad (\texttt{masked by } M_l), \tag{27}$$

$$\tilde{A}_l^{-1} = T_l \oplus \tilde{A}_h \otimes \tilde{B}_2^{-1} \oplus \underline{\tilde{A}_h \otimes M_l} \oplus M_h \otimes \tilde{B}_2^{-1} \oplus \underline{M_h \otimes M_l}, \tag{28}$$

the underlined products are re-used and the output $\tilde{\mathbf{A}}^{-1}$ is still $\mathbf{A}^{-1}$ above masked by output mask $\mathbf{T}$ (which might be same with the input mask $\mathbf{M}$ or not):

$$\tilde{\mathbf{A}}^{-1} = \mathbf{A}^{-1} \oplus \mathbf{T}. \tag{29}$$

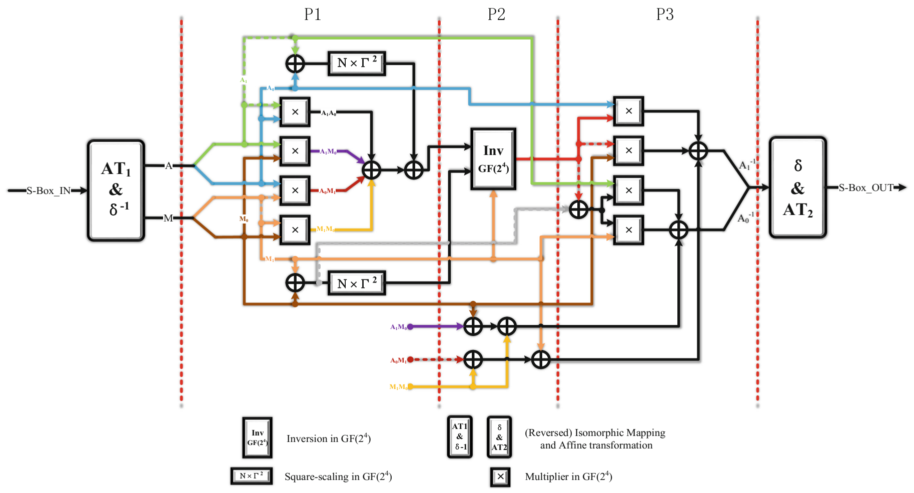

Fig. 2. Architecture of masked S-Box: (a) Single cycle (without the red dash line); (b) Pipeline (the red dash line is the pipeline registers) (Color figure online)

3.4 Mask Transformation

Equation (1) shows the algebraic expression of unmasked S-Box. Here we make some changes to the mathematical relationship and deduce the correct mask transformation from input to output, where the function I stands for inversion process in $GF(2^8)$ and the function Inv represents inversion process in the "tower field", i.e. $GF(((2^2)^2)^2)$. Note the matrix δ is the isomorphic mapping from the normal basis in composite field to the standard polynomial basis (and δ^{-1} is the reversed mapping).

$$
\begin{aligned}
S(\mathbf{X}+\mathbf{M}) &= I[(\mathbf{X}+\mathbf{M})\cdot A+\mathbf{C}]\cdot A+\mathbf{C} && (30)\\
&= A^T\cdot I[A^T\cdot(\mathbf{X}+\mathbf{M})+\mathbf{C}^T]+\mathbf{C}^T\\
&= A^T\cdot I[(A^T\mathbf{X}+C^T)+A^T\mathbf{M}]+\mathbf{C}^T\\
&= A^T\delta\cdot Inv[\delta^{-1}(A^T\mathbf{X}+\mathbf{C}^T)+\delta^{-1}A^T\mathbf{M}]+\mathbf{C}^T && (31)
\end{aligned}
$$

where A^T is the transposition of A (also similar with $\mathbf{C}^T$). Here we can learn from Eq. (29) that $Inv(\hat{\mathbf{A}}+\hat{\mathbf{M}}) = Inv(\hat{\mathbf{A}})+\hat{\mathbf{M}}$ only if the output mask is equal to the input mask: $\mathbf{S} = \mathbf{M}$ (this is the conclusion in $GF(((2^2)^2)^2)$. With this assumption, Eq. (31) in $GF(2^8)$ could be modified as follows:

$$
\begin{aligned}
S(\mathbf{X}+\mathbf{M}) &= (31)\\
&= A^T\cdot I(A^T\mathbf{X}+\mathbf{C}^T)+\mathbf{C}^T+A^TA^T\mathbf{M}\\
&= S(\mathbf{X})+A^TA^T\mathbf{M} && (32)
\end{aligned}
$$

If the input mask of S-Box is $\mathbf{M}$, the Eq. (32) shows the correct output mask of S-Box in $GF(2^8)$, i.e. $A^TA^T\mathbf{M}$, which is the "confusion" of the mask. Until now, we have achieved the masking process using the normal basis in composite field. Figure 2 gives the complete hardware implementation of the masked S-Box, depending on all the mathematical computing above.

4 Implementation of Masked SM4

In this section, we apply our "masked S-Box" to the encryption process and illustrate the architecture of the SM4 round arithmetic in two different directions: (i) use the iterative architecture and make all the steps of SM4 secure, the purpose of which is to decrease the cost of the area; (ii) insert some registers inside the S-Box appropriately to increase the clock frequency and improve the throughput, which will be very useful in high-speed applications.

4.1 Iterative Architecture of Masked SM4

In Fig. 3. The rk_i is well prepared in RAM or it can be produced by the iterative architecture presented in [4] before each round. Here the latter is our preference and we just concentrate on the masked encryption. For instance, we choose a 32-bit mask $\mathbf{M}$ for our design. Before the first round of encryption, the mask is produced and extended to 128 bits (e.g. $\{4\{\mathbf{M}\}\}$), which is XORed to the input 128-bit plaintext to obtain the masked input $\underline{\mathbf{X}} = (\underline{X}_0, \underline{X}_1, \underline{X}_2, \underline{X}_3)$ for the first round. It is obvious that all the variables before the function "Masked S-Box" are masked by $\mathbf{M}$. As is shown above, the outputs of the "Masked S-Box" are masked by $A^TA^T\mathbf{M}$. So we do the same round shifting left to $A^TA^T\mathbf{M}$, then add the outputs together and XOR the $\underline{X}_0$ simultaneously. Thus the mask $A^TA^T\mathbf{M}$ is eliminated and the output $\underline{X}_4$ has been already masked by $\mathbf{M}$, which is diffused from $\underline{X}_0$. In this way, we redo the arithmetic for 32 rounds, then reverse the last four words and finally XOR the output with $\{4\{\mathbf{M}\}\}$, we can certainly get the right result of ciphertext.

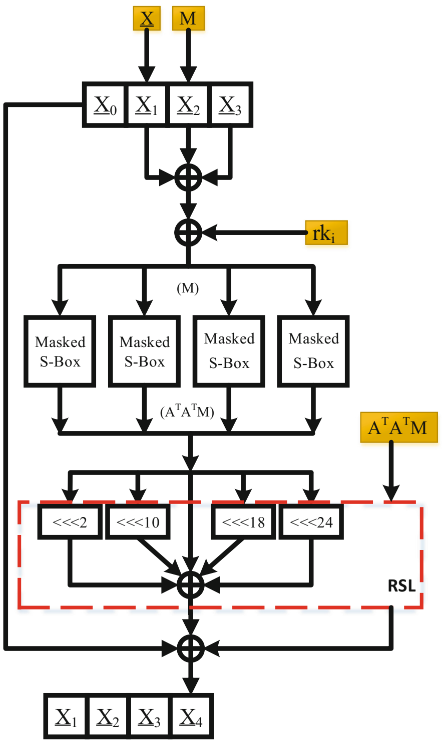

Fig. 3. Architecture of masked SM4 round arithmetic using the single cycle "Masked S-Box" (the underlined elements in this picture are masked variables and (·) in the architecture shows the transient mask at that step. The red dash rectangle is the round shifting left function) (Coloe figure online)

4.2 Pipelined Architecture of Masked SM4

For the pipeline scheme, we use the synchronous technique to adjust the structure of the round arithmetic of SM4. However, one key problem is to balance the pipeline stages. So we divide the round arithmetic into several periods to achieve one round encryption in order to ensure the approximate executing time for each part, which means to decrease the time of the critical-path. The registers being inserted into the S-Box are shown as red dash line in Fig. 2. Although the pipelined S-Box is well designed, the SM4 round arithmetic needs to be seriously considered according to the linear parts execution. Here we present the optimized architecture for pipelined masked SM4, given as Fig. 4. To keep the variables of each period secure, we add the random mask to all the input elements and transfer them to their corresponding buffers in each pipeline stage (shown as the yellow registers in Fig. 4). In this way, all the elements in the round encryption are securely masked and can be parallel implemented.

Figure 4 is just one round encryption for the SM4 algorithm, which contains five levels for the pipeline. As we expect, we implement 32 same structures and finally do the reversion function. The right results are realized and it will be shown in next section.

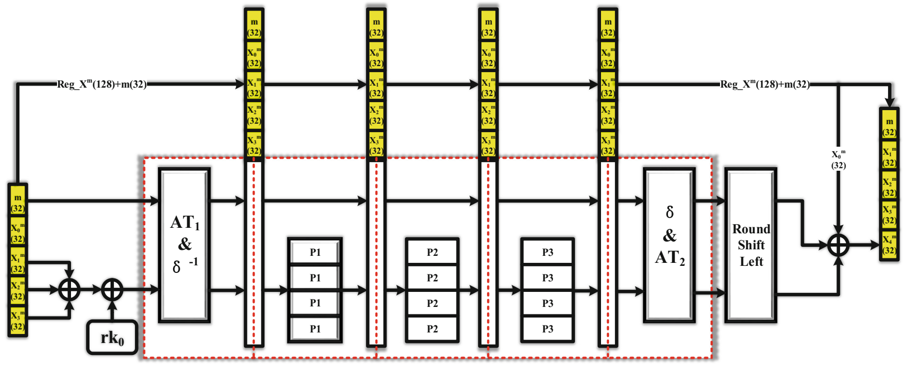

Fig. 4. Pipelined round arithmetic of masked SM4 (the red dash rectangle is pipelined masked S-Box described above in Fig. 2, similarly with the modules P1, P2 and P3) (Color figure online)

5 Results

The proposed SM4 algorithm implemented in two different ways based on "Masked S-Box" in composite field has been realized by Verilog HDL and simulated in Modelsim software. All the input plaintexts have achieved correct output ciphertexts.

Besides, the area reports of the masked S-Box under the SMIC 0.13 µm in the Synopsys Design Compiler indicate that the equivalent amount of gates is 978, at least 46.8% fewer than 1,840 in [6] (where the area has been divided by 9.79, which is the area of one NAND2X1 cell under the SMIC 0.18 µm). We firmly believe that our compact masked S-Box has occupied the lowest area and it certainly contributes to the low-cost iterative masked SM4 algorithm very much.

In addition, comparing to the other designs, we implement it in different FPGA boards and show the results in Table 1. Because of the large cost for masking, our design has reached a very suitable and satisfying resource usage and it is still much lower than some other works without anti-attack methods (Table 2).

Furthermore, by employing the pipelined masked S-Box to the SM4 algorithm, the pipeline SM4 round arithmetic shown in Fig. 4 achieve the ultra-high clock frequency up to 551 MHz under Xilinx FPGAs, resulting in the ultra-high throughput over 70 Gbps. To our knowledge, this is the highest speed and throughput to date.

Table 1. Resources comparison

Work	FPGA devices	Area	Anti-attack
[4]	Cyclone-II EP2C35F672C6	1657 LEs	NO
[7]		3406 LEs	NO
Ours		**2255** LEs	**YES**
[4]	Stratix-II EP2S15F484C3	535 ALMs	NO
[8]		1150 ALMs	NO
[9]		1552 ALMs	NO
Ours		**1321** ALMs	**YES**

Table 2. Performance comparison

Work	Platform	Frequency (MHz)	Throughput (Gbps)	Anti-attack
[10]	Virtex-6 XC6VLX240T	253	0.253	NO
	Virtex-5 XC5VLX110T	203	0.203	
	Virtex-4 XC4VLX100	211	0.211	
Ours	Virtex-6 XC6VLX240T	**551**	**70.5**	**YES**
	Virtex-5 XC5VLX110T	**462**	**59.1**	
	Virtex-4 XC4VLX100	**368**	**47.1**	

6 Conclusion

In this paper, the process to design a very compact "Masked S-Box" has been described clearly using normal basis in composite field at first. Second, we analyze the "diffusion" and "confusion" of the mask through the whole SM4 algorithm, and make sure every variable during encryption has been securely masked. Third, we implement the masked S-Box in two architecture: iteration and pipeline. Then we simulate all the designs and get the correctly inspiring results. The synthesis results in different devices have been compared with other works. As far as we know, our proposed work has reached the lowest area of masked S-Box, which leads to the lowest cost for masked SM4 implementation. What's more, the proposed pipelined S-Box has been implemented to construct a pipelined masked SM4 architecture, which achieves the highest speed to date. We believe this work has developed a good countermeasure to the side-channel attacks and will be widely used in resource constrained devices and speed demanded area in the future.

Acknowledgment. This work was supported by the National Natural Science Foundation of China (Grants 61472208), and by the National Key Basic Research Program of China (Grant 2013CB338004).

References

1. Office of State Commercial Cipher Administration of China. Block Cipher for WLAN products-SMS4 (2006). http://www.oscca.gov.cn/UpFile/200621016423197990.pdf
2. Canright, D., Batina, L.: A Very Compact Perfectly Masked S-Box for AES. Springer, Berlin (2008)
3. Liu, F., Ji, W., Hu, L., Ding, J., Lv, S., Pyshkin, A., Weinmann, R.-P.: Analysis of the SMS4 block cipher. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 158–170. Springer, Heidelberg (2007). doi:10.1007/978-3-540-73458-1_13
4. Fu, H., Bai, G., Wu, X.: Low-cost hardware implementation of SM4 based on composite field. In: IEEE Information Technology, Networking, Electronic and Automation Control Conference, pp. 260–264. IEEE (2016)
5. Canright, D.: A very compact Rijndael S-box (2004)
6. Niu, Y., Jiang, A.: The low power design of SM4 cipher with resistance to differential power analysis. In: 2015 16th International Symposium on Quality Electronic Design (ISQED) (2015)
7. Yuan-Yang, Z.: Area-efficient IP core design of block cipher SMS4. Electr. Technol. Appl. **23**, 127–129 (2007)
8. Husen, W., Shuguo, L.: High performance FPGA implementation for SMS4. In: Wu, Y. (ed.) ICHCC 2011. CCIS, vol. 163, pp. 469–475. Springer, Heidelberg (2011). doi:10.1007/978-3-642-25002-6_66
9. Gao, X., Lu, E., Xian, L., Chen, H.: FPGA implementation of the SMS4 block cipher in the Chinese WAPI standard. In: International Conference on Embedded Software and Systems Symposia, ICESS Symposia 2008, pp. 104–106. IEEE (2008)
10. Shang, M., Zhang, Q., Liu, Z., Xiang, J.: An ultra-compact hardware implementation of SMS4. In: 2014 IIAI 3rd International Conference on Advanced Applied Informatics (IIAIAAI), pp. 86–90 (2014)