Efficient Hardware Architecture of SEED S-box for Smart Cards

Joon-Ho Hwang

Abstract—This paper presents an efficient architecture that optimizes the design of SEED S-box using composite field arithmetic. SEED is the Korean standard 128-bit block cipher algorithm developed by Korea Information Security Agency. The non-linear function S-box is the most costly operation in terms of size and power consumption, taking up more than 30% of the entire SEED circuit. Therefore the S-box design can become a crucial factor when implemented in systems where resources are limited such as smart cards. In this paper, we transform elements in $GF(2^8)$ to composite field $GF(((2^2)^2)^2)$ where more efficient computations can be implemented and transform the computed result back to $GF(2^8)$. This technique reduces the S-box portion to 15% and the entire SEED algorithm can be implemented at 8,700 gates using Samsung smart card CMOS technology.

Index Terms—SEED, S-box, symmetric encryption, block cipher, composite field, smart card.

I. INTRODUCTION

SEED algorithm is a 128-bit symmetric key block cipher that has been developed by Korea Information Security Agency (KISA) and a group of experts in 1998 [1]. SEED is a national industrial association standard (TTAS KO-12.0004, 1999) and is widely used to protect electronic transactions, financial services, and electronic mails provided in Korea. SEED is a 16-round Feistel structure using 128-bit

message block and 128-bit key for operation. SEED utilize the S-boxes and permutations for high security level, and is known to be strong against differential cryptanalysis and linear cryptanalysis until now.

There are several factors to consider when implementing a block cipher such as SEED. There have been literatures to improve the performances [6], however there have been no publication on how to implement effectively on systems where resources are limited such as smart cards. In such systems, it is important to keep the gate count, as well as the power consumption to minimum since they have rigorous constraints on such factors. Another factor to consider is the vulnerability to side channel attacks. Smart cards have a characteristic that it can be easily probed and side channel attacks such as Differential Power Analysis (DPA) [7] can be performed. Hence, countermeasure against these side channel attacks is another challenging factor to consider when implementing cryptographic modules. One of the advantages of using composite field arithmetic for S-box implementation is that random masking techniques can be designed for SEED [8] to prevent side channel attacks. However, cryptanalysis is not the main subject of this paper and hence will not be mentioned any further.

The idea of using composite field arithmetic for S-box designs were first applied to AES (Advanced Encryption Standard) block cipher [2][3][4][5]. The main operation of AES S-box is an inverse operation for elements in $GF(2^8)$, whereas the main operation of SEED S-box is an modular exponentiation for elements in $GF(2^8)$. Also the primitive polynomial used for AES S-box is different from the primitive polynomial used for SEED S-box. Therefore the composite field transformation for AES S-box cannot be d irectly

SoC R&D Center, Samsung Electronics, Seoul, Korea
E-mail : joonho.hwang@samsung.com

applied to SEED S-box. In the remainder of this paper, we present an efficient architecture of implementing SEED S-box using composite field technique. The paper is organized as follows: In section 2, we give a brief introduction to SEED block cipher algorithm. Section 3 describes our new architecture for SEED S-box and section 4 gives some implementation results of our new architecture. Finally, our conclusion is given in section 5.

II. DESCRIPTION OF SEED ALGORITHM

This section gives a brief description on SEED block cipher algorithm. More detailed information can be obtained in [1].

1. Structure of SEED

SEED is a classical Feistel structure cipher with 16 rounds. Feistel structure ciphers have a common characteristic that an inverse function does not have to exist. The decryption process is exactly the same as the encryption process except that the round keys are arranged in reverse sequences. Therefore no distinguishment needs to be made between an SEED encryption circuit and a SEED decryption circuit. A 128-bit input is divided into two 64-bit blocks and the right 64-bit block is an input to the round function F with a 64-bit round key generated from the key scheduling. The structure of SEED is shown in Figure 1.
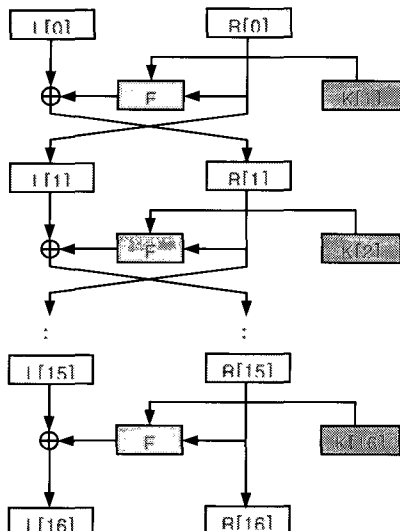
Fig. 1. Feistel structure of SEED

2. Round function F

The round function F divides the 64-bit input block into two 32-bit blocks (C, D) and goes through 4 phases: a mixing phase with two 32-bit round key blocks (K0[i], K1[i]) and 3 layers of function G with additions for mixing two 32-bit blocks. The round function F is shown in Figure 2.
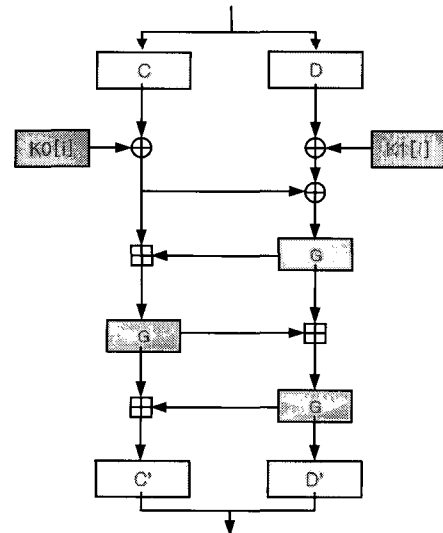
Fig. 2. Structure of round function F

The $\oplus$ denotes a bitwise exclusive OR operation and the boxed $+$ denotes an addition in modular 2^{32}.

3. Function G

The function G has two layers: a layer of two 8×8 S-boxes and a layer of block permutation of sixteen 6-bit sub-blocks.
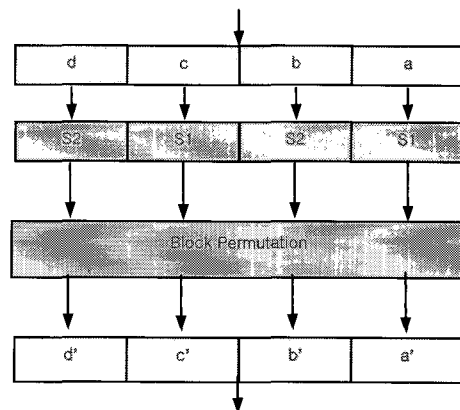
Fig. 3. Structure of function G

The first layer of two S-boxes is generated from the following equation.

$$S_1 : GF(2^8) \to GF(2^8) , \; S_1(x) = A^{(1)} \cdot x^{247} \oplus 169 \quad (1)$$

$$where \quad A^{(1)} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

$$S_2 : GF(2^8) \to GF(2^8) , \; S_2(x) = A^{(2)} \cdot x^{251} \oplus 56 \quad (2)$$

$$where \quad A^{(2)} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

primitive polynomial, $p(x) = x^8 + x^6 + x^5 + x + 1$

4. Key scheduling

The description of the SEED key schedule will be omitted in this paper because our proposed architecture is independent of the structure of the SEED key schedule.

III. SEED S-BOX IN COMPOSITE FIELD

In general the $GF(2^8)$ modular exponentiation calculation in equation (1) and (2) is very complicated to implement and hence is usually implemented as a lookup table or as SOP (Sum-of-Product) logic circuits generated by CAD tools. In this section we present an efficient way to implement equation (1) and (2), where $GF(2^8)$ elements are transformed to composite field elements $GF(((2^2)^2)^2)$ for computation.

1. Modification of S-box equations

Since the inputs and outputs of S-box equations are all elements of $GF(2^8)$, the following congruence is true.

$$x^{-1} \equiv x^{254} \bmod p(x) \quad (3)$$

The following congruence can be derived from (3)

$$(x^{-1})^4 \equiv x^{251} \bmod p(x) \quad (4)$$
$$(x^{-1})^8 \equiv x^{247} \bmod p(x)$$

Therefore, S-box $S_1(x)$ can be modified as follows.

$$S_1(x) = A^{(1)} \cdot (x^{-1})^8 \oplus 169 \quad (5)$$

Equivalently, S-box $S_2(x)$ can be modified as follows.

$$S_2(x) = A^{(2)} \cdot (x^{-1})^4 \oplus 56 \quad (6)$$

As seen in equation (5) and (6), the S-box equations for SEED have been modified to $GF(2^8)$ inverse operation with additional squaring operations. However, squaring in $GF(2^8)$ is merely matrix transformations and eventually can be merged with inverse isomorphic transformation and affine transformation at the final stage, requiring no additional hardware resources. Therefore, most part of S-box operation will be concentrated in inverse calculation.

2. Isomorphic transformation

In order to optimize the inverse calculation, we transform $GF(2^8)$ elements to composite field $GF(((2^2)^2)^2)$ elements defined by the following irreducible polynomials.

$$Original \; field \quad GF(2^8) : x^8 + x^6 + x^5 + x + 1$$

$$Composite \; field \; \begin{cases} GF(2^2) : x^2 + x + 1 \\ GF((2^2)^2) : x^2 + x + \{10\} \\ GF(((2^2)^2)^2) : x^2 + x + \{\{11\}\{00\}\} \end{cases}$$

The isomorphic transformation of $GF(2^8)$ to $GF(((2^2)^2)^2)$ using the above irreducible polynomial can be given by the following matrix transformation δ.

$$f : GF(2^8) \xrightarrow{\;\delta\;} GF(((2^2)^2)^2)$$

$$\delta = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

3. Inverse calculation in composite field

Once the element is transformed into a composite field element the inverse calculation can be done with the following circuit.
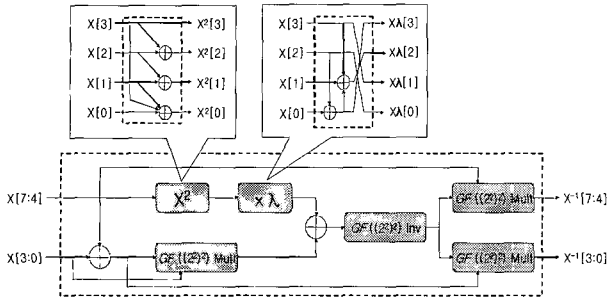

Fig. 4. $GF(((2^2)^2)^2)$ inverse calculation circuit

The $GF(((2^2)^2)^2)$ inverse calculation circuit is constructed with three $GF((2^2)^2)$ multipliers and a $GF((2^2)^2)$ inverse calculation. The $GF((2^2)^2)$ multiplier circuit is give in the following diagram.
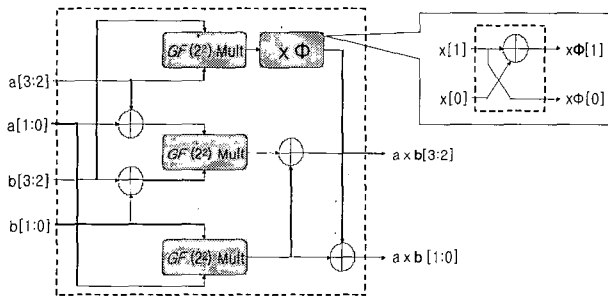

Fig. 5. $GF((2^2)^2)$ multiplier circuit

The $GF((2^2)^2)$ multiplier circuit is constructed with three $GF(2^2)$ multipliers and exclusive OR operations. Breaking down more into lower field, the $GF(2^2)$ multiplier circuit is constructed with bitwise AND operations and exclusive OR operations. The structure for $GF(2^2)$ multiplier circuit is given in Figure 6. The $GF((2^2)^2)$ inverse calculation can be either constructed as a pre-computed lookup table, since it is simple enough by having only 16(=4x4) cases, or by breaking down more into lower field multiplication in the similar manner. The described circuits above calculate the inverse value of an element in $GF(((2^2)^2)^2)$ and this is much more efficient than calculating the inverse value of an element in $GF(2^8)$.
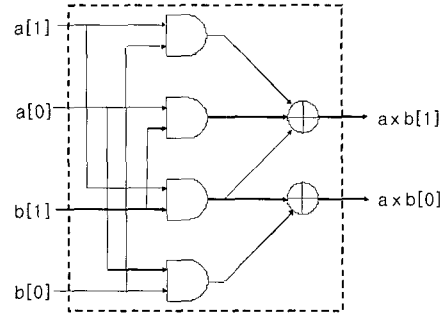

Fig. 6. $GF(2^2)$ multiplier circuit

4. Inverse isomorphic transformation

The element in $GF(((2^2)^2)^2)$ must be transformed back to the original field $GF(2^8)$ once inverse is computed. The inverse isomorphic transformation δ^{-1} is simply the inverse matrix of isomorphic transformation δ. However, the inverse matrix must be merged with squaring transformation matrix and affine transformation matrix to get the S-box result in equation (5) and (6). This is depicted in Figure 7.
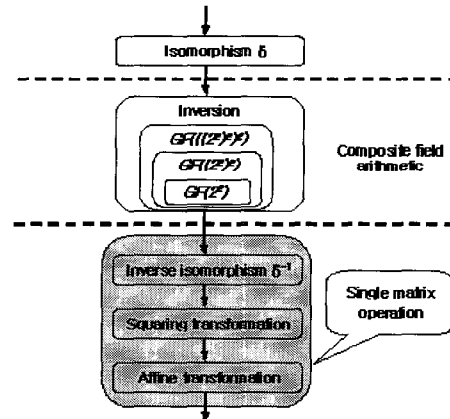

Fig. 7. Inverse isomorphic transformation process

IV. IMPLEMENTATION RESULTS

Table 1. Implementation results of SEED

		Our method	Conventional
Area (Gates)	S-box	1,266 (15%)	3,154 (30%)
	Total	8,742	10,630
Critical Path		32 ns	25 ns
Throughput		16.98 Mbps @ 15 MHz	

We implemented SEED with Verilog-HDL using the presented architecture. Our implementation used the

shared G-function scheme, since it was optimized for area and power, and therefore requires 7 clock cycles for each round. We simulated our implementation with Cadence NC-Verilog and synthesized it with Synopsys Design-Complier. The implementation was simulated and synthesized with Samsung smart-card library (smart130), which is a 0.18μm CMOS technology. The results are summarized in Table 1.

V. CONCLUSION

In this paper, we presented an efficient hardware architecture for SEED S-box implementation using composite field arithmetic. This architecture is applicable to systems where resources, such as area and power, are limited, as in smart cards or mobile devices. The reduction of complexity for S-box results in a very compact and secure hardware architecture of SEED block cipher algorithm.

REFERENCES

[1] Korea Information Security Agency (KISA), SEED Algorithm Specification, available at http://www. kisa.or.kr.

[2] National Institute of Standards and Technology (NIST), "Advanced Encryption Standard (AES)," FIPS Publication 197, Nov. 2001.

[3] A. Rudra et al, "Efficient Rijndael encryption implementation with composite field arithmetic," Proc. CHES 2001, LNCS Vol. 2162, pp. 175-188, 2001.

[4] A. Satoh, S. Morioka, K. Takano, and S. Munetoh, "A Compact Rijndael Hardware Architecture with S-Box Optimization," Advances in Cryptology - ASIACRYPT 2001, LNCS Vol. 2248, pp. 239-254, 2001.

[5] S. Morioka and A. Satoh, "An Optimized S-Box Circuit Architecture for Low Power AES Design," Proc. CHES 2002, LNCS Vol. 2523, 172-186, 2002.

[6] D.W. Kim, Y.H. Seo, J.H. Kim, and Y.J. Jung,

"Hardware Implementation of 128-Bit Symmetric Cipher SEED," IEEE AP-ASIC 2000, 2000.

[7] P. Kocher, J. Jaffe and B. Jun, "Differential Power Analysis," Advances in Cryptology - CRYPTO 1999, LNCS Vol. 1666, pp. 388-397, 1999.

[8] Y.J. Baek and J.H. Hwang, "Improved Algorithms for converting between Boolean Mask and Arithmetic Mask," to appear in Fourth Conference on Security in Communication Networks(SCN) '04, 2004

Joon-Ho Hwang received his B.S. and M.S. degrees in electronic and electrical engineering from Pohang University of Science and Technology (POSTECH), Korea, in 1999 and 2001 respectively. He was a member of the information security and telecommunication laboratory during the M.S. course, where he concentrated research on cryptography and PKI. He is currently a research engineer at Samsung Electronics, SoC R&D Center. His current research interests include hardware architecture of cryptographic algorithms, side-channel attacks and countermeasure schemes for such attacks.