Hardware Design to Compromise Between Area and Speed: CRYPTONII

Nirut Chalainanont
Department of Electrical & Computer Engineering,
Oregon State University, Corvallis, Oregon 97331.
E-mail: chalaini@ece.orst.edu

Abstract— **CRYPTON is a 128-bit block cipher submitted to the NIST as a candidate algorithm for the AES (Advanced Encryption Standard). CRYPTONII is the modified version of CRYPTON hardware model to compromise between area of the design and speed of encryption process. CRYPTONII resulting in up to 2.47 Gbps in data encryption rate with less than 40,000 gates.**

I. Introduction

CRYPTON is a SPN(substitution-permutation network)-type cipher based on the structure of SQUARE [1]. Hardware implementations of CRYPTON are efficient because the encryption and decryption use the same circuits, and no large logic for S-boxes required. The algorithm uses only exclusive-OR operations which can be done in parallel [3].

CRYPTON has the following features [2]:

• 12-round self-invertible cipher with 128-bit block size and key length up to 256 bits.

• Guaranteed security against existing attacks, such as di erential and linear cryptanalysis.

• High performance in SW: 362 cycles on Pentium III/450MHz (about 150 Mbps).

• Fast key scheduling: much faster than one-block encryption.

• Efficiency in HW implementation: a Gbps range with low-cost gate array implementation.

This paper is organized as follows. Section II briefly describe CRYPTON algorithm. Section III shows CRYPTONII hardware design and section IV is the summary of speed and area estimations of CRYPTON and CRYPTONII model.

II. CRYPTON Algorithm

CRYPTON represents each 128-bit data block 4 x 4 byte array and processes it using a sequence of round transformation. Each round transformation consists of four steps, byte-wise substitution, column-wise bit permutaion, column-to-row transposition and key addition.

• **Byte-wise substitution Υ:** Substitute 4 x 4 byte array using four 8 x 8 S-boxes S_i $(0 \leq i \leq 3)$. The four S-boxes are derived from one 8 x 8 involution S-box S (i.e., $S = S^{-1}$) and satisfy the following inverse relation :

$$S_2 = S_0^{-1}, S_3 = S_1^{-1}$$

Two different transformations are used, Υ_o in odd rounds and Υ_e in even rounds.They are defined as :

$$B = \Upsilon_o(A), b_{ij} = S_{i+jmod4}(a_{ij})$$

$$B = \Upsilon_e(A), b_{ij} = S_{i+j+2mod4}(a_{ij})$$

Four S-boxes are arranged so that the following holds for any 4×4 byte array A:

$$\Upsilon_o(\Upsilon_e(A)) = \Upsilon_o(\Upsilon_e(A)) = A$$

Figure 1 shows the byte-wise substitution Υ.

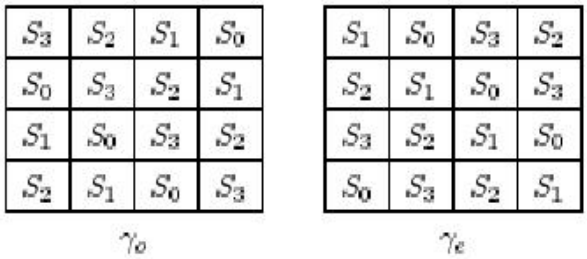

Figure 1: The byte-wise substitution Υ.

• **Column-wise bit permutaion π:** The bit permutation π bit-wise mixes each byte column of 4×4 byte array using four masking bytes m_i given by

$$m_0 = 0xfc, m_1 = 0xf3, m_2 = 0xcf, m_3 = 0x3f$$

Column permutations π_i $(0 \leq i \leq 3)$ are defined as

$$[b_3, b_2, b_1, b_0]^t = \Pi_i([a_3, a_2, a_1, a_0]^t)$$

$$b_j = xor_{k=0}^{3}(m_{i+j+kmod4} \triangle a_k)$$

We also use two different versions of bit permutation, π_o in odd rounds and π_e in even rounds. Let A^i be the i-th byte column of a 4×4 byte array A, i.e., $A^i = (a_{3i}, a_{2i}, a_{1i}, a_{0i})^t$. Then the bit permutations π_o and π_e respectively are defined as :

$$\pi_o(A) = (\pi_3(A^3), \pi_2(A^2), \pi_1(A^1), \pi_o(A^0))$$

$$\pi_e(A) = (\pi_1(A^3), \pi_0(A^2), \pi_3(A^1), \pi_2(A^0))$$

Figure 2 shows the column-wise bit permutation π.

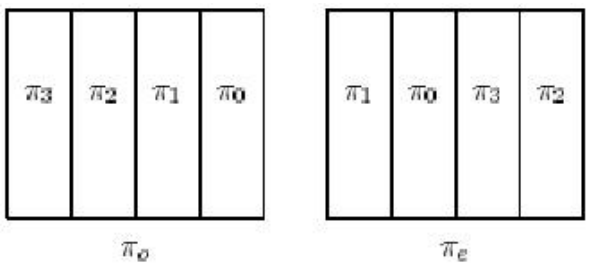

Figure 2: The column-wise bit permutation π.

• **Column-to-Row transposition τ:** The byte transposition rearranges a 4×4 byte array by moving the byte at the (i, j)-th position to the (j, i)-th position. See Figure 3 for $B = \tau(A)$. It is also involution, i.e., $\tau^{-1} = \tau$.

$A[0]$	a_{03}	a_{02}	a_{01}	a_{00}		$B[0]$	a_{30}	a_{20}	a_{10}	a_{00}
$A[1]$	a_{13}	a_{12}	a_{11}	a_{10}	τ	$B[1]$	a_{31}	a_{21}	a_{11}	a_{01}
$A[2]$	a_{23}	a_{22}	a_{21}	a_{20}	$\Longrightarrow$	$B[2]$	a_{32}	a_{22}	a_{12}	a_{02}
$A[3]$	a_{33}	a_{32}	a_{31}	a_{30}		$B[3]$	a_{33}	a_{23}	a_{13}	a_{03}

Figure 3: The column-to-row transposition τ.

• **Bit-wise key addition δ:** This process simply exclusive-or round keys with data words. For a round key $K = (K[3], K[2], K[1], K[0])^t, B = \delta_K(A)$ is defined by $B[i] = A[i] xor K[i]$ for $i = 0, 1, 2, 3$. And $\delta_K^{-1} = \delta_K$.

The encryption process repeats 12 iterations of the same transformation. However, 12-round encryption process needs 13 round keys, from round0 key to round12 key. Figure 4 shows top level structure of CRYPTON.

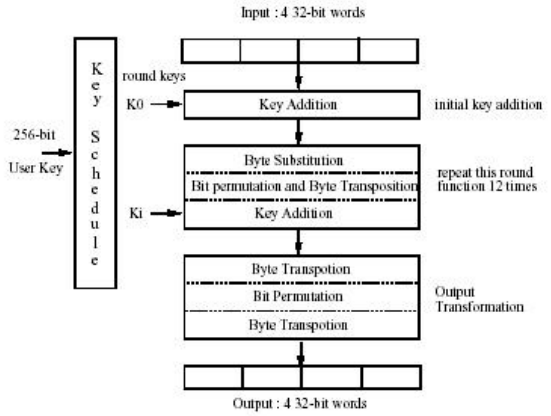

Figure 4: Top Level Structure of CRYPTON.

First, the 128-bit input data will be exclusive-OR with K_0 (round0 key). The result goes to round1 transformation. Then exclusive-OR the result from transformation with K_1 (round1 key). The result goes to round2 transformation. Repeat this process for 12 rounds then the result of round12 transformation exclusive-OR with K_{12} goes to Output transformation in the final step. Finally, 128-bit encrypted data is generated from output transformation. The decryption process is the same as encryption process using different key schedule.

The previous designs for CRYPTON Version 1.0 use two different models, Two-Round Model and Full-Round Model. For details, refer to [3]. Two-Round model uses small area of design by reuse the funtional blocks, but it results in longer encryption time. It needs 7 cycles to perform 128-bit data encryption. Figure 5 shows encryption model of Two-Round model. On the other hand, Full-Round model uses loop unrolling technique which resulting in large area design but faster in execution. Figure 6 shows encryption model of Full-Round model. CRYPTONII model is the compromise between these two models which resulting in high speed encryption with small area.

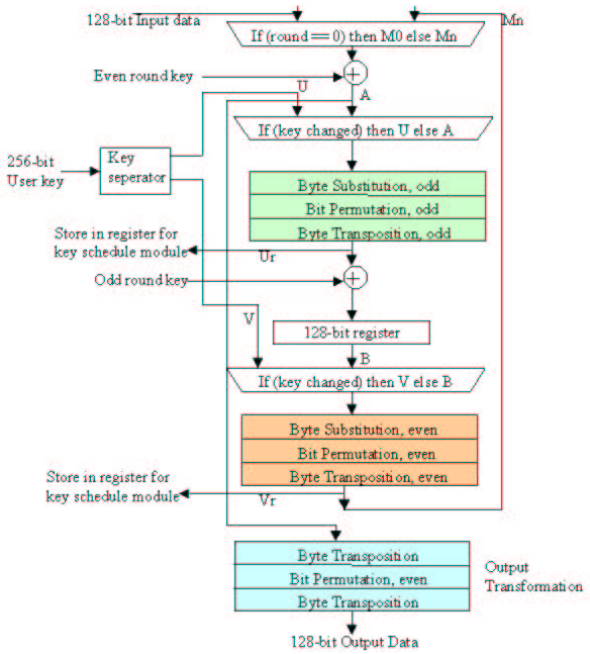

Figure 5: Encryption Model of Two-Round Model.

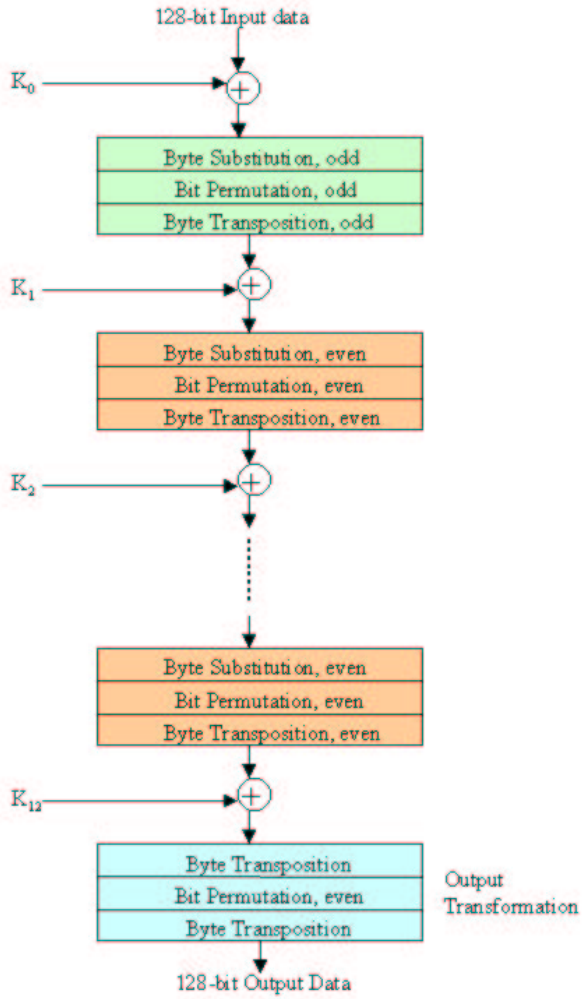

Figure 6: Encryption Model of Full-Round Model.

III. Hardware Design of CRYPTONII

CRYPTONII combines the advantages of Two-Round model by re-use the functional block to reduces the area of the design with the loop unrolling technique to reduces number of cycles used in encryption process. By loop unrolling of 4, CRYPTONII performs 4 iterations per cycles. Each cycle contains 2 odd rounds and 2 even rounds. Figure 7 shows the encryption model of CRYPTONII.

The first multiplexer selects between the input data in round0 and the data from previous cycle (M_n) in round1 to round12. Because CRYPTON uses only simple operations, we can reduce area of the design by share some logic blocks. For example, CRYPTON has Byte Substitution operation both in key scheduling module and encryption module. Thus, the Byte Substitution modules are shared. These modules are used for key expansion in case of the user key has been changed. The multiplexers will select U/V as their input to the Byte Substitution module and get the expanded key (U_r/V_r) as the output. These expanded key will be stored in 128-bit register for key schedule module to read and generate round keys. In the encryption process, the multiplexers will select the data from previous round (A/B) to be input to Byte Substitution module. This sharing scheme can reduce area of the design, but it needs one more clock cycle for key expansion process when the key has been changed.

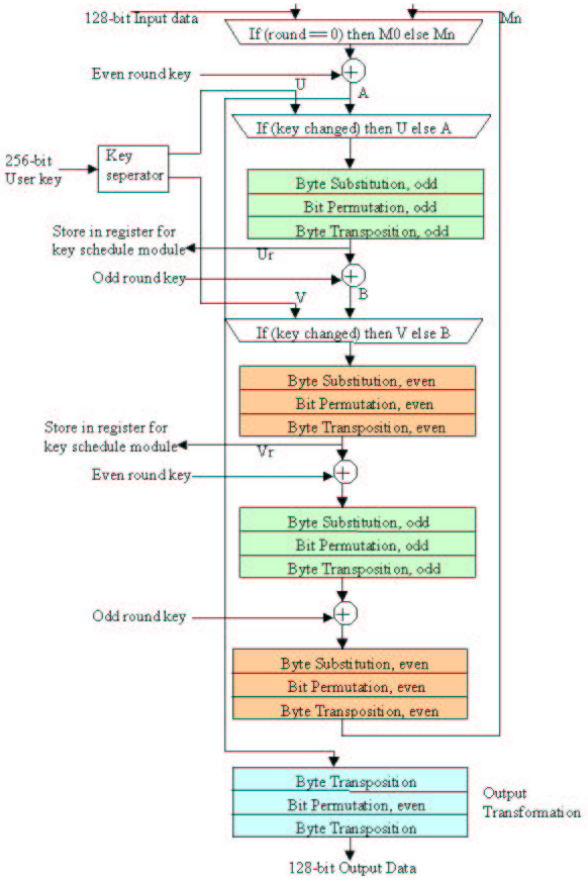

Figure 7: Encryption Model of CRYPTONII.

Four keys needed in each cycle, 2 even round keys and 2 odd round keys. These keys are generated by key schedule module. Figure 8 shows the structure of key scheduling module.

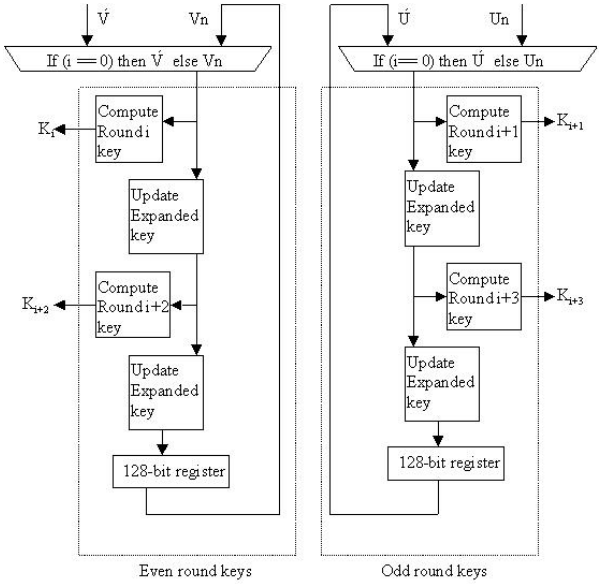

Figure 8: Key Scheduling Module for iteration i^{th}.

From Figure 8 there are two main parts in key scheduling module. The left part generates 2 keys for even round i and i+2. The right part generates 2 odd round keys for round i+1 and i+3, where i = 0,4,8 and 12. Initially, when i=0 the multiplexer will select U'/V' which is the output from 128-bit register that store expanded key Ur/Vr. Then U'/V' are fed into key computation module to compute K_0 and K_1. These values also used for updating expanded key for using in next round key computation.

IV. Results

These are estimated results based on Synopsys Design Compiler and Hyundai 0.35 μm gate array library. Table 1 shows area and speed of CRYPTONII compare with two-round model and full-round model [3]. These results are optimized in term of encryption speed.

Table 1: Estimated area and speed for Two-round model, Full-round model and CRYPTONII.

Model	Total gates	Minimum clock period (ns)	Time to encrypt one block (ns)	Throughput (Gbps)
Two-round	28,179	10.23	71.61	1.66
Full-round	93,929	44.30	44.30	2.69
CRYPTONII	**38,348**	**12.05**	**48.20**	**2.47**

From the table, total gates indicates number of gates in the whole system, encryption module and key scheduling module. Gate means a 2-input NAND gate that equivalent to 4 transistors. Minimum clock period is the minimum time required for one cycle of process. Time to encrypt one block is the longest path delay from data input to the encrypted data output.

The results show that CRYPTONII model has only 10,000 gates more than Two-round model, but it has much higher encryption rate, upto 2.47 Gbps, which is almost equal to the speed of Full-round model.

References

[1] J.Daemen, L.Knudsen and V.Rijmen, "The block cipher Square, In Fast Software Encryption," in *Lecture Notes in Computer Science, No.1267*, Springer-Verlag, Ed. 1997, pp. 149–171.

[2] C.H. Lim, "Specification and Analysis of CRYPTON Version 1.0," 1999.

[3] E. Hong, J.-H. Chung, and C. H. Lim, "Hardware design and performance estimation of the 128-bit block cipher CRYPTON," in *Lecture Notes in Computer Science, No. 1717*, Ç. K. Koç and C. Paar, Ed. 1999, pp. 49–60.

[4] E.Biham, "A Note on Comparing the AES Candidates," *Proceedings of the Second Advanced Encryption Standard Candidate Conference*, Rome, Italy, 1999.

[5] C.H. Lim, "A Revised Version of CRYPTON - CRYPTON Version 1.0," *Proceedings of the 1999 Fast Software Encryption Workshop*, 1999,