

# Design and Implementation of a Crypto Processor and Its Application to Security System

HoWon Kim<sup>1</sup>, YongJe Choi<sup>1</sup> and MooSeop Kim<sup>1</sup>

<sup>1</sup> Department of Information Security Basic,  
Electronics and Telecommunications Research Institute(ETRI)  
161 Gajeong-Dong YuSeong-Gu, DaeJeon, 305-350, KOREA  
Tel : +82-42-860-6228, / FAX : +82-42-860-5611  
e-mail :khw@etri.re.kr

**Abstract:** This paper presents the design and implementation of a crypto processor, a special-purpose microprocessor optimized for the execution of cryptography algorithms. This crypto processor can be used for various security applications such as storage devices, embedded systems, network routers, etc. The crypto processor consists of a 32-bit RISC processor block and a coprocessor block dedicated to the SEED and triple-DES (data encryption standard) symmetric key crypto (cryptography) algorithms. The crypto processor has been designed and fabricated as a single VLSI chip using 0.5  $\mu\text{m}$  CMOS technology. To test and demonstrate the capabilities of this chip, a custom board providing real-time data security for a data storage device has been developed. Testing results show that the crypto processor operates correctly at a working frequency of 30MHz and a bandwidth of 240Mbps

## 1. Introduction

The expansion of the worldwide communication network such as the internet and the increased dependency on digitized information in our society makes information more vulnerable to abuse. If there are security problems in these information systems, users will fear that their sensitive information may be monitored and business secrets stolen. For these reasons, it is important to make information systems secure by protecting data and resources from malicious acts --- crypto algorithms are the core of such security systems[1].

By encoding a message using crypto algorithms, users can make information transmitted over communication systems almost impossible to read, even if such information is intercepted for malicious purposes. It is fairly easy to implement crypto algorithms in software, but such algorithms are typically too slow for real-time applications, such as storage devices, embedded systems, network routers, etc. For this reason, it becomes necessary to implement crypto algorithms in hardware. In our crypto processor implementation, the dedicated crypto block of the crypto processor permits fast execution of encryption, decryption, and key scheduling operations for triple-DES[14,12] and SEED[13] private key crypto algorithms. Also, the 32-bit RISC processor block can execute other crypto algorithms such as RSA and ECC (the Elliptic Curve Cryptography algorithm) and control the dedicated crypto block and I/O buffers.

This paper is organized as follows. In Section 2, the architecture of the crypto processor is briefly described; this includes the dedicated crypto block for SEED and triple-

DES and the 32-bit RISC processor. In Section 3, the detailed VLSI design methodology of the crypto processor is described. In Section 4, the simulation and verification of the crypto processor design is reported. Section 5 presents the application of the crypto processor as a means of providing real time data security for a storage device. Finally, concluding remarks are presented in Section 6..

## 2. The Crypto Processor Architecture

### 2.1 The architecture of the Crypto Processor

The block diagram of our crypto processor is shown in Fig. 1. This single chip crypto processor has a crypto controller and a dedicated crypto block for the triple-DES and SEED algorithms. The 32-bit RISC type crypto controller controls the dedicated crypto block and performs the interface operations with external devices such as memory and an I/O bus interface controller. It can also execute various crypto algorithms such as RSA and ECC and other application programs such as a user authentication program and an IC card interface program.

The dedicated crypto block executes encryption, decryption and key scheduling operations for the SEED and triple-DES algorithms. The 128-bit plain text data streams entered into the 128-bit input register are encrypted with a proper key and control signals based on the SEED algorithm. After plain text data streams are encrypted, the 128-bit cipher texts are output to the 128-bit output register. The decryption process is the same as the encryption process except for the control signals. For the DES algorithm, 64-bit plain text data streams and 64-bit key values with 8-bit parity bits are necessary for encryption and decryption. Our crypto processor supports four operation modes: ECB(Electronic CodeBook), CBC(Cipher Block Chaining), OFB(Output FeedBack) and CFB(Cipher FeedBack) for the SEED and triple-DES algorithms.

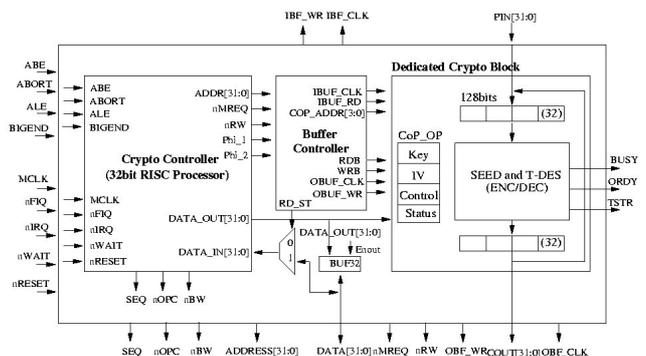


Figure 1. Block diagram of the Crypto processor

## 2.2 The dedicated crypto block for the SEED algorithm

The SEED algorithm[13] is a block cipher that operates on 128-bit blocks of data and uses a 128-bit key. It has a 16 rounded Feistel structure. A Feistel structure takes a block of length  $n$  and divides it into two halves of length  $n/2$ , a left and right block. It is an iterated block cipher in which the output of the  $i$ -th round is determined from the output of the  $(i-1)$ -th round[11]. The SEED algorithm uses two 8 X 8 S-boxes (for substitution), permutations, rotations, and basic modulo-arithmetic operations such as modulo-2 addition (exclusive OR) and modulo-2<sup>32</sup> addition. As with other Feistel ciphers, the SEED algorithm has an  $F$  function, which takes a 64-bit data value and 64-bit key values as shown in Fig.3.

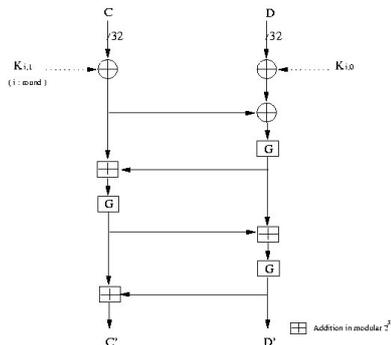


Figure 3. Block diagram of the  $F$  Function of the SEED algorithm

To implement the SEED algorithm, we have instantiated one stage and iterated the data through this stage 16 times. We could also have 16 or more pipeline stages. But in this case, we would have had high performance in a non-feedback mode such as ECB, but no performance gains and much excessive hardware redundancy for feedback modes such as CBC, OFB, and CFB. Because we wanted to design a crypto processor with equally high performance for various modes, we have selected this iterated method. The key values for encryption and decryption are pre-computed and stored in internal buffers. These stored key values are used for encryption or decryption of the data sequences that follow.

## 2.3 The dedicated crypto block for a triple-DES algorithm

DES(Data Encryption Standard) [10], an encryption algorithm developed in the 1970's by the National Bureau of Standards and IBM Corporation, uses a 56-bit key. In the DES algorithm, there are 16 rounds of identical operations such as non-linear substitutions and permutations. In each round, 48-bit subkeys are generated, and substitutions using S-box, bitwise shift, and XOR (exclusive-OR) operations are performed. The 56-bit key length is relatively small by today's standards. For increased security, the DES operation can be performed three consecutive times, which expands the effective key length to 112 bits [11]. Using DES in this manner is referred to as triple-DES.

Fig. 4 shows one round of the DES algorithm. The left and right halves of each 64-bit input data operand are treated as separate 32-bit data operands,  $L_{i-1}$  and  $R_{i-1}$ . The

32-bit right halves of the data are passed to the next left halves of the data ( $L_{i-1} = R_{i-1}$ ), and the 32-bit left halves of the data are processed in the following manner:  $R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$ . As shown in Fig. 4, the  $F$  function of the DES algorithm is composed of an expansion permutation table (block E), modulo-2 addition with the  $i$ -th round key ( $K_i$ ), substitution with the S-box, and permutation with the P table(block P). Because one round of the DES algorithm is simpler than the SEED algorithm, we have made 4 rounds of the DES algorithm executable in one clock cycle. Most of the latency in one round of the DES algorithm is due to the S-box operation.

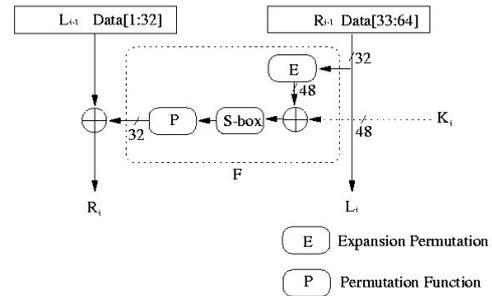


Figure 4. One round of the DES algorithm

## 2.4 The 32-bit RISC processor block

The block diagram of the 32-bit RISC type crypto controller is shown in Fig.5 [3]. This controller controls the operation of the dedicated crypto block during encryption, decryption and key scheduling, and also performs the operations required to interface with external devices such as the input FIFO, output FIFO, memory, and system I/O bus(address and data bus). Since the crypto controller block is fully programmable, it can execute various crypto algorithms, protocols and application programs with a high degree of freedom. The crypto controller is a 32-bit processor with a RISC architecture and a 3-stage pipeline. It has features (such as a barrel shifter, a Booth multiplier block, register file, and a 16-bit and 32-bit data memory architecture) that enable it to achieve high performance and savings in memory when executing crypto algorithms.

The codes for crypto controller generate the control signals for a dedicated crypto block based on a memory-mapped method. The crypto controller generates control signals for the key and initial vector (which are required to execute the SEED and triple-DES algorithms), an algorithm selection signal, and a mode selection signal. It also performs other miscellaneous tasks such as *done* signal generation for the encryption or decryption operations. Then, when the plain text data becomes available, the dedicated crypto block receives the data and encrypts it with a proper mode and algorithm. When the encryption operations are done, the encrypted cipher texts are output to an output register and the corresponding control signals are set. Our crypto controller is fully compatible with ARM7TM [3] and described using Verilog HDL.

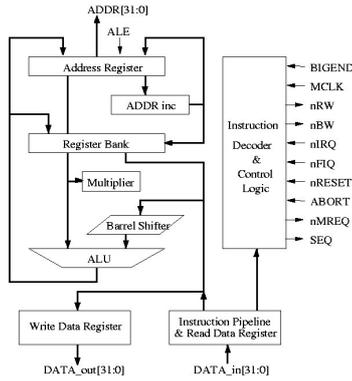


Figure 5. Block diagram of the 32-bit RISC controller block

### 3. The VLSI Implementation of the Crypto processor

Our crypto processor was modeled using Verilog HDL (Hardware Description Language) and implemented as an ASIC chip. Modeling the processor using Verilog HDL facilitates quick prototyping and modification of the target design while considering various possible trade-offs in different implementations of the crypto algorithms with differing speed and area characteristics. Next, the crypto processor's HDL model was simulated using ModelSim HDL compiler and simulator [9]. Then, Synopsys Design Analyzer and Compiler [12] was used to synthesize the HDL models into gate level designs, and the SDF files were simulated using Cadence's SimWave [5]. Because the SDF file includes fairly accurate delay and load information, the simulation results are comparable to actual measurement results after the circuit is fabricated in silicon. The target process technology is Hynix's  $0.5 \mu m$  CMOS technology.

### 4. The Simulation and Verification of the Crypto Processor

Simulation was used to validate the Verilog HDL model of the crypto processor. After validation, the HDL model was synthesized into a gate level design with a target CMOS process technology library

Static timing analysis is, however, required in combination with formal verification to achieve complete ASIC verification. Thus, we have also performed static timing analysis from the SDF files. After simulation and verification of our design, we have laid out and fabricated the crypto processor using Hynix's  $0.5 \mu m$  CMOS technology. Fig. 6 shows a photograph of the crypto processor, and Table 2 summarizes the main features of the crypto processor. Note that a photograph of the layout is not presented as the circuit was synthesized using a standard cell library.



Figure 6: Photograph of the crypto processor.

Table 2 : Main features of the crypto processor.

Technology	$0.5 \mu m$ CMOS
Package Type	PQFP
Gate Counts	200K(with I/O PADS)
Chip Size	$8.1mm \times 8.1mm$
Bandwidth	240Mbps(SEED), 160Mbps(triple-DES)
Operating Frequency	30MHz
The number of I/O pins	176 pins
VDD and VSS	5V and 0V

To validate the usability of the 32-bit RISC type crypto controller in our crypto processor for various security systems, we have implemented the ECDSA [8] and ECDH [6] protocols. The ECC algorithm we have implemented is defined over the field  $GF(2^{163})$ , which is a SEC-2 recommendation [7], with this field being defined by the field polynomial  $F(x) = x^{163} + x^7 + x^6 + x^3 + 1$ . The timing results are shown in Table 3. As shown in Table 3, most of the latency was due to the scalar multiplications  $kG$  in Algorithm 1. The latency of the ECDSA signature verification algorithm is asymptotically twice the latency of the signature generation algorithm. The latencies of the modular reduction and inversion processes are also negligible when compared to scalar multiplication.

We have also implemented the ECDH key agreement protocol for the crypto controller. To obtain a common key for the two participants Alice and Bob, Alice secretly chooses a random integer  $k_A$  and computes the factor  $k_A G$ , which she sends to Bob. Likewise, Bob secretly chooses a random integer  $k_B$ , computes  $k_B G$ , and sends it to Alice. The common key is  $P = k_B k_A G$ . As shown in Table 3, the performance of the crypto controller in the crypto processor is suitable for embedded system applications, where high flexibility and performance are a must.

#### Algorithm 1. ECDSA Signature Generation Algorithm

To sign a message  $m$ , a signer  $A$  does the following:

Select a random integer  $k$  from  $[1, n - 1]$

Compute  $kG = (x_1, y_1)$  and  $r = x_1 \bmod n$

Compute  $k^{-1} \bmod n$

Compute  $e = \text{SHA-1}(m)$

Compute  $s = k^{-1}\{e + dr\} \bmod n$

If  $s = 0$  then go to step 1

$A$ 's signature for the message  $m$  is  $(r, s)$

Where,  $G$  is a base point on  $E(GF(2^m))$ .

$d$  is a random integer from  $[1, n - 1]$  and  $A$ 's private key.

Table 3 : Performance of the ECDSA and ECDH algorithms when executed on the crypto controller.

Method	Timing
Scalar Multiplication	1.004 sec
ECDSA signature generation	1.032 sec
ECDSA signature generation	2.255 sec
EC Diffie-Hellman	1.920 sec
SHA-1(for 163bit data size)	11.24 $\mu sec$

## 5. A Crypto Processor Application: Real-time Data Security for a Storage Device

To evaluate the usability of the crypto processor, we have developed an RTDS (Real Time Data Security) system for storage devices. The RTDS system is composed of control and monitoring software with a GUI(Graphical User Interface) environment, a device driver, and an RTDS board. Fig.7 shows the block diagram of the RTDS system, and Fig.8 shows a photograph of the RTDS board with the crypto processor. The main operations of the RTDS system are described as follows.

- A user process wants to write data into the secure area of a hard disk (a)
- The CPU reads data from a certain area of the memory and sends it to the hard disk via the I/O bus (b).
- The device driver, which is a part of a RTDS system, catches the hard disk write event, and forwards data to the crypto processor (c).
- In the crypto processor, an encryption task is performed in real-time (d).
- The crypto processor, which has completed its encryption task, sends the encrypted data to the hard disk (e).
- The hard disk receives the encrypted data and completes the write procedure (f).

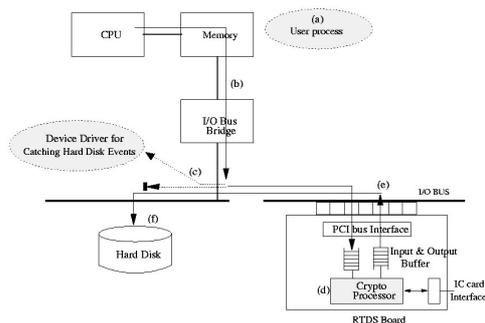


Figure 7: Block diagram of the Real Time Data Security System for storage devices.

The RTDS board, shown in Fig. 8, is mainly composed of a PCI interface controller, an SRAM buffer, an IC card interface controller, and a crypto processor. An Altera FPGA chip is used for the PCI interface controller, and the ASIC chip, located in the right upper part of the board, is the crypto processor. The performance of the crypto processor and the PCI interface controller is high --- 240 Mbps and 1056 Mbps, respectively --- and the average access time of the hard disk (a Quantum FireBall 15 device) is low --- 12 msec in our system. Therefore, the RTDS system operates in real-time.



Figure 7: Photograph of the RTDS board.

## 6. Concluding Remarks

In this paper, we have presented the design and implementation of a crypto processor composed of a 32-bit RISC processor and a coprocessor block dedicated to the triple-DES and SEED algorithms. The dedicated block of the crypto processor accelerates private key crypto algorithms and the programmability of the crypto controller makes possible fast execution of various crypto algorithms (such as RSA, ECC, etc.) and security applications. The crypto processor was implemented as an ASIC chip using Hynix's 0.5  $\mu\text{m}$  CMOS technology. Simulations, formal verification, and static timing analysis were used to fully verify the ASIC design before fabrication. The fabricated chip was found to have a 30MHz operating frequency and a data rate of 240Mbps for all modes of operation (ECB, CBC, OFB, CFB) of the SEED algorithm. The crypto processor was evaluated by constructing an RTDS (Real-Time Data Security) system for storage devices. This application board was used to thoroughly test and verify the functionality of the crypto processor. The crypto processor in the RTDS system performs data encryption and decryption in real-time. The high performance and high flexibility of the crypto processor design makes it applicable to various security applications such as storage devices, embedded systems, network routers, firewalls, etc.

## References

- [1] Paul C. van Oorschot Alfred J. Menezes and Scott A. Vanstone, Handbook of applied cryptography, CRC press Inc., Florida, 1996.
- [2] Analog Devices, VMS115 IPsec Coprocessor Data Sheet, Rev. 2.0, January 1999.
- [3] ARM corp., ARM7 Data Sheet, 1996.
- [4] H.B. Bakoglu, Circuits, interconnects, and packaging for VLSI, Addison-Wesley Publishers Ltd., 1990.
- [5] Cadence Corp., SimWave, may 1999.
- [6] Certicom Corp., SEC 1: Elliptic curve cryptography, September 2000.
- [7] Certicom Corp., SEC 2: Recommendation elliptic curve domain parameters, September 2000.
- [8] Don B. Johnson, Alfred J.Menezes, and Scott Vanstone, Elliptic curve digital signature algorithm(ECDSA), available at <http://www.certicom.com>
- [9] Modeltech Corp., Modelsim Compiler, May 1999.
- [10] National Institute of Standards and Technology, FIPS publication 46-2: Data Encryption Standard, MD, USA, December 1993.
- [11] Bruce Schneier, Applied cryptography(2<sup>nd</sup> ed. ), John Wiley and Sons, Inc., New York, 1996.
- [12] Synopsys Corp., Design Compiler Reference Manual, February 1998.
- [13] TTA, 128-bit Symmetric Block Cipher(SEED), Telecommunications Technology Association(TTA), Seoul, Korea, June 1999.