情報保護學會論文誌
第18卷 第6(A)號, 2008. 12

자원 공유기법을 이용한 AES-ARIA 연산기의 효율적인 설계

구 본 석[1†], 유 권 호[1], 장 태 주[1], 이 상 진[2‡]
[1]한국전자통신연구원 부설연구소, [2]고려대학교 정보경영공학전문대학원

Design of an Efficient AES-ARIA Processor using Resource Sharing Technique

Bonseok Koo[1†], Gwonho Ryu[1], Taejoo Chang[1], Sangjin Lee[2‡]
[1]Attached Institute of ETRI, [2]Graduate School of Information Management & Security, Korea University

요 약

AES와 ARIA 블록암호 알고리즘은 각각 미국과 한국의 차세대 표준 블록암호 알고리즘으로 각광받고 있으며, 스마트 카드, 전자여권 등 기밀성이 요구되는 다양한 정보보호 분야에서 활용되고 있다. 본 논문에서는 최초로 AES와 ARIA의 효율적인 통합 하드웨어 연산기를 제안하고 0.25um CMOS 공정으로 구현한 결과를 제시한다. AES와 ARIA에 적용할 수 있는 확장 유한체 방식의 공통 S-box를 설계하고, 두 알고리즘의 확산 함수에서 공통항을 축출하여, 19,056 게이트 카운트의 소형 크기를 가지는 연산기를 설계하였다. 본 논문에서 제안하는 연산기는 AES와 ARIA의 개별 소형 연산기를 설계하는 방식에 비해 32% 감소된 크기를 가진다. 또한 제안하는 연산기는 128비트 한 블록에 대한 AES 암호화에는 11 클록 사이클, ARIA 암호화에는 16 클록 사이클을 사용하며, 이는 각각 1,047Mbps와 720Mbps의 성능을 나타난다.

ABSTRACT

AEA and ARIA are next generation standard block cipher of US and Korea, respectively, and these algorithms are used in various fields including smart cards, electronic passport, and etc. This paper addresses the first efficient unified hardware architecture of AES and ARIA, and shows the implementation results with 0.25um CMOS library. We designed shared S-boxes based on composite filed arithmetic for both algorithms, and also extracted common terms of the permutation matrices of both algorithms. With the 0.25-μm CMOS technology, our processor occupies 19,056 gate counts which is 32% decreased size from discrete implementations, and it uses 11 clock cycles and 16 cycles for AES and ARIA encryption , which shows 720 and 1,047 Mbps, respectively.

Keywords : AES, ARIA, S-box, Processor, Low-power

Ⅰ. 서 론

AES (Advanced Encryption Standard) 블록암호 알고리즘[1]은 지난 2001년 미국 NIST (National Institute of Standards and Technology, 국립 표준 기술원)에 의해 FIPS PUB 197 표준으로 채택되었으며, 기존의 DES (Data Encryption Standard) 알고리즘을 대체할 차세대 128비트 블록암호 알고리즘으로 각광받고

접수일 : 2008년 5월 30일; 채택일 : 2008년 10월 1일
† 주저자, bskoo@ensec.re.kr
‡ 교신저자, sangjin@korea.ac.kr

있다. 또한 ARIA 블록암호 알고리즘[2]은 지난 2004년 한국 산업 규격 (KS, Korean Standard) 표준으로 제정되었다. ARIA 알고리즘은 AES 알고리즘과 유사하게 128비트 SPN (Substitution Permutation Network) 구조를 특징으로 하며, 구현 속도나 알고리즘 안전성 측면에서 AES와 비견될 만한 성능을 자랑한다[3]. 이러한 AES와 ARIA, 두 국내외 표준 블록암호 알고리즘은 스마트카드, 전자여권, 서버급 암호장비 등 기밀성이 요구되는 다양한 정보보호 분야에서 사용되고 있다. 아울러, 현재 가장 널리 사용되고 있는 표준 보안 서비스 API (Application Program Interface) 중 하나인 RSA사의 PKCS (Public Key Cryptography Standard) #11은 AES 뿐만 아니라 ARIA 알고리즘에 대한 보안 매커니즘도 지원하고 있다[4].

AES와 ARIA 알고리즘은 SPN 구조를 가지는 128비트 블록암호 알고리즘이라는 점에서 유사한 구현 특징을 가진다. 두 알고리즘에서 사용하는 S-box를 정의하는 유한체는 동일하며, ARIA 알고리즘에서 사용하는 두 종류의 S-box 중 하나는 AES 알고리즘에서 사용하는 S-box와 동일하다. 한편 AES의 확산 계층은 암호화와 복호화에 따라 각기 다른 두 종류의 4 × 4 행렬 곱셈을 사용하며, 여기에는 유한체 $GF(2^8)$ 상의 곱셈이 적용된다. 반면에 ARIA의 확산 계층은 암호화와 복호화 시, 동일하게 하나의 이진 행렬 곱셈을 사용한다.

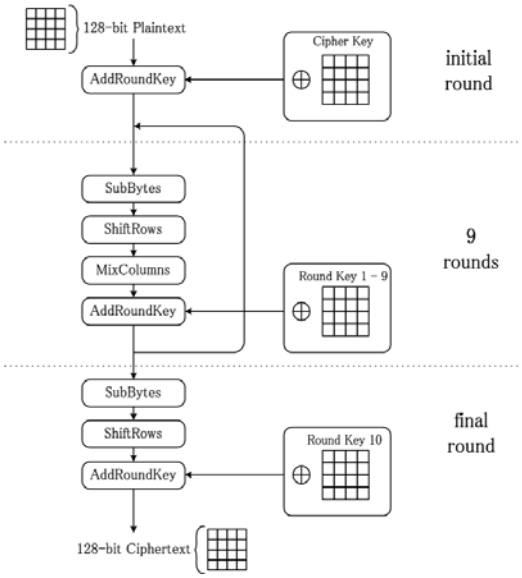
본 논문에서는 하드웨어 자원 공유 기법을 사용하여, AES와 ARIA 알고리즘을 모두 지원하는 통합 하드웨어 연산기를 제안하고, 0.25um CMOS 스탠다드 셀 공정을 이용하여 구현한 결과를 제시한다. 본 논문에서 연산기는 AES와 ARIA를 개별 소형 연산기로 구현하는 방식에 비해 32% 감소된 19,056 게이트 카운트의 작은 크기를 가진다. 따라서, 제안하는 연산기는 AES와 ARIA 알고리즘을 모두 지원하면서도, 하드웨어 자원이 한정된 저가의 스마트카드나 암호모듈의 개발에 매우 적합하다.

본 논문은 다음과 같이 구성된다. 2장에서는 AES와 ARIA 알고리즘의 개요를 서술하고, 3장과 4장에서는 두 알고리즘의 치환 계층과 확산 계층을 각각 하드웨어 자원 공유 기법을 이용하여 설계하는 방법을 설명한다. 또한 5장에서는 AES와 ARIA 통한 연산기의 하드웨어 구조와 설계에 대해 설명하고, 6장에서는 설계한 연산기의 구현 결과를 제시한다. 마지막으로 7장에서는 결론을 맺는다.

Ⅱ. AES와 ARIA 알고리즘의 개요

1. AES 알고리즘

AES 알고리즘은 128비트 평문에 대해 128비트 암호문을 출력하는 블록암호이며, 128/192/256 비트의 키 크기를 사용할 수 있다. 128비트 키 길이인 경우, AES 암호화는 [그림 1]과 같이 총 10 라운드의 연산과정으로 이루어지며, 각각의 라운드는 AddRoundKey, SubByte, ShiftRows, MixColumns의 4가지 원시 함수로 이루어진다. 각 원시 함수를 간략히 설명하면 다음과 같다.



[그림 1] AES 알고리즘의 암호화 과정 (128비트 키 길이)

먼저 AddRoundKey는 128비트의 중간 연산 결과값인 state와 라운드 키의 비트별 XOR 연산이며, SubBytes는 128비트 state의 각 16개 바이트에 대해 비선형 특성을 가지는 치환 연산을 수행한다. 이때 사용되는 S-box는 유한체 $GF(2^8)$상의 역원을 구하고, 아핀 변환을 취하는 구조를 가진다. 유한체 $GF(2^8)$를 정의하기 위한 기약 다항식 (irreducible polynomial) $m(x)$는 다음과 같다.

$$m(x) = x^8 + x^4 + x^3 + x + 1 \quad (1)$$

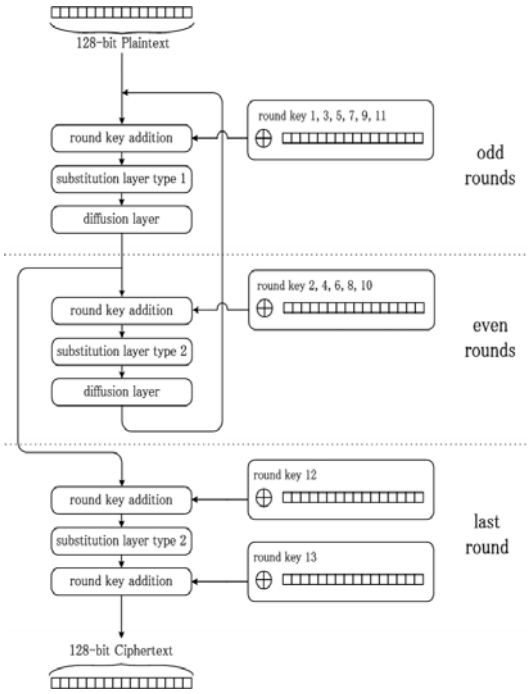
ShiftRows에서는 128비트 state를 바이트 단위로 환형 쉬프트 (cyclic shift) 시킨다. 이때 4 바이트의 각 행 (row)에 따라 쉬프트시키는 오프셋 (offset)을 달리한다. 마지막으로 MixColumns에서는 128비트 state를 4 바이트의 각 열 (column)에 다항식 곱셈 연산을 수행한다.

AES 알고리즘의 복호화 과정은 암호화 과정의 역순으로 진행된다. 또한 이때의 원시함수는 암호화 과정에서 사용된 각 원시함수의 역함수들인 AddRound Key, InvSubByte, InvShiftRows, InvMixColumns를 사용한다. 매 라운드에 사용되는 라운드 키는 초기 라운드 키로부터 확장된다. 암호화 시에는 각 라운드 함수를 수행하는 동시에, 해당 라운드 키를 계산하는 'on-the-fly' 방식의 키 확장이 가능하다. 반면에 복호화 시에는 암호화 과정과 역순으로 라운키가 사용되므로, 복호화 라운드 함수를 시작하기 전에 초기 라운드 키로부터 마지막 라운드 키를 계산한 다음, 복호화 과정을 수행하면서, 각 라운드 키를 역순으로 'on-the-fly' 키 확정을 수행할 수 있다.

2. ARIA 알고리즘

ARIA[3] 알고리즘은 AES와 동일하게 128비트의 데이터 블록을 처리하는 블록암호 알고리즘으로서 128/196/256 비트 길이의 키를 사용하며, AES와 유사한 정도의 암호학적 안전성을 가진다고 알려져 있다. ARIA 알고리즘은 Involutional SPN 구조를 가지며 128비트 키 길이인 경우, ARIA 암호화는 [그림 2]와 같이 총 12 라운드를 수행하며, 13개의 128비트 라운드 키를 사용한다. ARIA 알고리즘은 그림과 같이 홀수와 짝수 라운드에 각기 다른 치환 계층을 사용하며, 최종 라운드에서는 확산 계층 대신 라운드 키 덧셈 (round key addition)이 사용된다.

ARIA 알고리즘은 round key addition, sub stitution, diffusion의 3가지 원시 함수를 사용한다. 먼저 round key addition은 AES의 AddRoundKey와 동일한 비트별 XOR 연산이며, substitution에서는 AES의 SubBytes와 유사하게 128비트 데이터를 바이트 단위로 비선형 치환 연산을 수행한다. 이때 사용하는 S-box는 S_1과 S_2 두 가지가 있으며, 이 중 S_1은 AES에서 사용하는 S-box와 동일하지만, S_2는 S_1과 동일한 유한체 $GF(2^8)$를 사용하지만, x^{247}에 아핀변환을 취하는 형태로 정의된다. Diffusion은 16 × 16 이진 행렬을 이용한 바이트 간의 확산 함수를 사용하며, 이때 사용되는 행렬은 involution 행렬이므로, 암호화와 복호화 과정에서 동일한 연산이 사용되는 특징을 가진다.



[그림 2] ARIA 알고리즘의 암호화 과정 (128비트 키 길이)

ARIA의 키 스케줄링은 키 초기화와 키 확장의 두 단계로 이루어진다. 먼저 키 초기화 단계에서는 두 개의 128비트 입력 키 KL과 KR (128비트 키 길이인 경우는 0)로부터 3 라운드의 256비트 Feistel 구조를 거쳐서, 네 개의 128비트 키 W_0, W_1, W_2, W_3을 계산한다. 키 확장 단계에서는 초기화 단계에서 생성된 $W_i (i = 0, \ldots, 3)$을 입력받아 환형 쉬프트와 XOR 연산을 이용하는 간단한 방식으로, 각 라운드에 필요한 라운드 키를 생성한다. 따라서 ARIA의 키 스케줄링 방식도 AES처럼 'on-the-fly' 키 확장이 가능하다.

Ⅲ. 통합 치환 계층

치환 계층을 구성하는 S-box 연산은 일반적으로 룩업-테이블 (LUT, LookUp-Table) 방식으로 구현된다. 하지

만 이러한 방식으로 두 종류의 S-box 기능을 구현하는 경우, 두 배의 하드웨어 자원이 필요하므로, AES와 ARIA에서 공통으로 사용할 수 있는 S-box를 구현하는 데에는 적합하지 않다. 최근에는 이 보다 소형의 면적으로 S-box를 구현하기 위해 조합 논리 (combinational logic)를 이용하여 구현하는 기법들이 발표되어 왔다.[5-7] A. Satoh와 D. Canright 등은 유한체 $GF(2^8)$ 상의 8비트 원소를 복합체 $GF(((2^2)^2)^2)$ 상의 원소로 매핑하여 역원 계산을 수행함으로써, S-box 구현을 효율적으로 경량화하는 방법을 제안하였다[6-7].

앞서 설명한 것처럼, AES와 ARIA의 S-box는 동일한 유한체 $GF(2^8)$을 사용하며, AES의 S-box S_1은 8×8 이진 정칙 행렬 A와 벡터 b를 사용하여 다음 수식과 같이 표현될 수 있다.

$$S_1(x) = A \cdot x^{-1} \oplus b$$

$$= \begin{pmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{pmatrix} \cdot (x^{-1}) \oplus \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix},$$

$$S_1^{-1}(x) = (A^{-1} \cdot (x \oplus b))^{-1}$$
$$= (A^{-1} \cdot x \oplus A^{-1} \cdot b))^{-1}$$

$$= \left[\begin{pmatrix} 0&0&1&0&0&1&0&1 \\ 1&0&0&1&0&0&1&0 \\ 0&1&0&0&1&0&0&1 \\ 1&0&1&0&0&1&0&0 \\ 0&1&0&1&0&0&1&0 \\ 0&0&1&0&1&0&0&1 \\ 1&0&0&1&0&1&0&0 \\ 0&1&0&0&1&0&1&0 \end{pmatrix} \cdot x \oplus \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \right]^{-1}$$

ARIA에서 사용하는 또 하나의 S-box S_2는 x^{247}(=x^{-8})를 사용하는데, 유한체 $GF(2^8)$ 상의 임의의 원소 x에 대해 x^{2^i}(i는 임의의 정수)는 8 x 8 이진 행렬 M_i를 사용하여, $M_i \cdot x$로 표현될 수 있으므로, $x^{247} = M_3 \cdot x^{-1}$로 정리될 수 있다[9]. 따라서 S-box S_2는 8 x 8 이진 정칙 행렬 C, E, F와 벡터 d를 사용하여 다음 수식과 같이 표현될 수 있다.

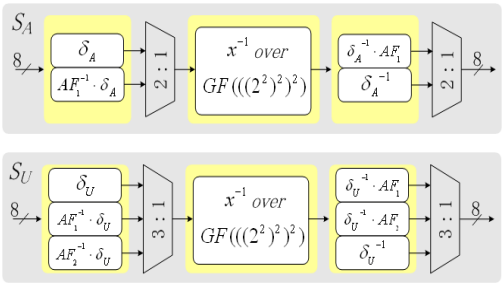
$$S_2(x) = C \cdot x^{-8} \oplus d = CM_3 \cdot x^{-1} \oplus d$$
$$= E \cdot x^{-1} \oplus d$$

$$= \begin{pmatrix} 0&1&0&1&0&1&1&1 \\ 0&0&1&1&1&1&1&1 \\ 1&1&1&0&1&1&0&1 \\ 1&1&0&0&0&0&1&1 \\ 0&1&0&0&0&0&1&1 \\ 1&1&0&0&1&1&1&0 \\ 0&1&1&0&0&0&1&1 \\ 1&1&1&1&0&1&1&0 \end{pmatrix} \cdot (x^{-1}) \oplus \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix},$$

$$S_2^{-1}(x) = (C^{-1} \cdot (x \oplus d))^{-32}$$
$$= (M_5 C^{-1} \cdot (x \oplus d))^{-1}$$
$$= (M_5 C^{-1} \cdot x \oplus M_5 C^{-1} \cdot d)^{-1}$$
$$= (F \cdot x \oplus F \cdot d)^{-1}$$

$$= \left[\begin{pmatrix} 0&0&0&1&1&0&0&0 \\ 0&0&1&0&0&1&1&0 \\ 0&0&0&0&1&0&1&0 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&0&1&1&0&0 \\ 0&1&1&0&1&0&1&1 \\ 1&0&1&1&1&1&0&1 \\ 1&0&0&1&0&0&1&1 \end{pmatrix} \cdot x \oplus \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \right]^{-1}$$

결과적으로 두 종류의 S-box S_1와 S_2는 유한체 $GF(2^8)$ 상의 역원 연산과 아핀 변환으로 표현될 수 있다. 따라서, 본 논문에서는 유한체 $GF(2^8)$상의 역원 연산 기능을 공유하면서, 각 S-box에서 사용하는 아핀 연산을 선택적으로 수행할 수 있도록 [그림 3]과 같은 두 종류의 공통 S-box, S_A와 S_U를 설계한다.



[그림 3] 공통 S-box S_A와 S_U의 내부 구조

그림에서 S_A는 S_1과 S_1^{-1} 기능을 선택적으로 수행할 수 있으며, S_U는 S_1, S_1^{-1}, S_2, S_2^{-1}의 네 가지 기능을 선택적으로 수행한다. 또한 δ_A와 δ_U는 $GF(2^8)$상의 8비트 입력을 $GF(((2^2)^2)^2)$ 상의 원소로 매핑시키는 동형

(isomorphism) 함수이고, AF_1과 AF_2은 각각 S_1과 S_2 S-box의 아핀 변환을 나타낸다.

경량 구현을 위해 복합체 $GF(((2^2)^2)^2)$ 상의 역원 연산을 사용하였으며, $AF_1^{-1}\cdot\delta_A$와 같이 동형 함수와 아핀 변환 함수를 하나로 복합 (combined) 시킨 함수를 사용하였다. S-box의 하드웨어 면적을 최소화하기 위해, S_A의 구현는 현재까지 알려진 가장 작은 하드웨어 면적 결과를 보이는 D. Canright의 방식을 사용하였다. 또한 S_U에 대해서는 모든 가능한 동형 함수 δ_U를 조사하여, δ_U, δ_U^{-1}, $AF_1^{-1}\cdot\delta_U$, $AF_2^{-1}\cdot\delta_U$, $\delta_U^{-1}\cdot AF_1$, $\delta_U^{-1}\cdot AF_2$ 의 6개 함수가 가장 작은 '1'의 개수를 가지도록 구현하였다.

$$\delta_U(x)=\begin{pmatrix}1&1&1&1&0&0&1&0\\1&0&0&0&0&1&1&0\\1&0&0&0&0&0&0&0\\1&1&0&1&1&0&0&1\\1&0&0&0&0&1&1&1\\1&1&0&0&0&1&1&0\\1&0&0&0&1&1&1&0\\1&1&1&0&0&1&1&1\end{pmatrix}\cdot x,$$

$$AF_1^{-1}\cdot\delta_U(x)=\begin{pmatrix}1&1&0&0&1&1&1&0\\1&0&0&1&1&0&0&0\\0&0&1&0&0&1&0&1\\0&0&0&0&1&0&1&1\\1&1&0&1&0&0&1&0\\0&0&0&0&1&0&1&0\\1&1&0&0&1&0&1&0\\0&0&0&0&1&0&0&1\end{pmatrix}\cdot x\oplus\begin{pmatrix}0\\1\\1\\1\\1\\1\\1\\0\end{pmatrix},$$

$$AF_2^{-1}\cdot\delta_U(x)=\begin{pmatrix}0&1&1&0&1&0&1&0\\1&1&0&0&1&1&1&0\\0&0&0&1&1&0&0&0\\1&0&1&0&0&0&1&0\\0&1&0&1&1&1&0&1\\1&1&1&0&1&0&0&0\\0&0&1&0&0&0&1&0\\0&1&1&1&0&0&0&1\end{pmatrix}\cdot x\oplus\begin{pmatrix}0\\1\\0\\1\\1\\1\\1\\0\end{pmatrix},$$

$$\delta_U^{-1}(x)=\begin{pmatrix}0&0&1&0&0&0&0&0\\0&1&0&0&0&1&0&0\\0&1&0&0&1&1&0&1\\0&1&1&1&1&1&1&0\\0&1&0&0&0&0&1&0\\1&0&1&1&0&1&1&1\\1&1&0&1&0&1&1&1\\0&1&0&0&1&0&0&0\end{pmatrix}\cdot x,$$

$$\delta_U^{-1}\cdot AF_1(x)=\begin{pmatrix}0&1&0&0&1&0&1&0\\0&1&0&0&1&1&0&0\\1&0&1&1&0&1&1&0\\0&0&0&1&1&1&1&1\\0&0&0&1&0&1&0&1\\1&0&0&0&0&0&1&0\\0&0&0&1&0&0&0&1\\0&0&0&1&0&1&0&0\end{pmatrix}\cdot x\oplus\begin{pmatrix}1\\1\\0\\0\\0\\1\\1\\0\end{pmatrix},$$

$$\delta_U^{-1}\cdot AF_2(x)=\begin{pmatrix}0&0&0&1&0&0&1&0\\0&1&0&1&1&0&0&1\\1&0&0&1&0&1&0&0\\1&1&1&1&1&0&1&1\\1&1&0&1&1&0&1&1\\0&1&0&0&0&1&1&0\\1&0&0&1&0&1&1&0\\0&0&1&1&0&1&1&1\end{pmatrix}\cdot x\oplus\begin{pmatrix}0\\1\\0\\0\\0\\1\\1\\1\end{pmatrix}.$$

Synopsys Design Compiler툴[17]과 0.25um CMOS 공정을 이용하여 공통 S-box를 구현한 결과를 [그림 5]와 [그림 6]에 나타내었고, [표 1]에 정리하였다.S_U는 4가지 종류의 S-box 기능을 수행할 수 있는데도 불구하고, S_A에 비해 면적은 37%, 지연 시간은 7% 증가된 373 게이트 카운트와 6.63ns의 결과를 가진다.

```
****************************************
Report : area
Design : Sbox_A
Version: W-2004.12-SP2-1
Date   : Fri Jul 21 09:13:51 2006
****************************************

Library(s) Used:

    anam (File: /RAID1/user/dwlee/WORK/LIBI

Number of ports:                17
Number of nets:                 98
Number of cells:                68
Number of references:            6

Combinational area:        272.699890
Noncombinational area:       0.000000
Net Interconnect area:      undefined  (No
```

[그림 4] 공통 S-box S_A의 합성 결과

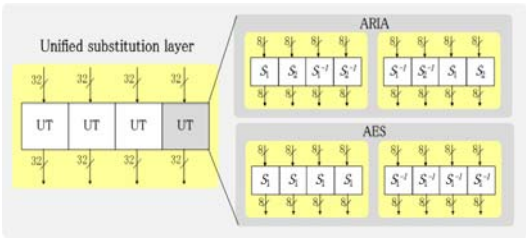
```
****************************************
Report : area
Design : Sbox_T
Version: W-2004.12-SP2-1
Date   : Fri Jul 21 09:12:36 2006
****************************************

Library(s) Used:

    anam (File: /RAID1/user/dwlee/WORK/LIB

Number of ports:                18
Number of nets:                156
Number of cells:               139
Number of references:           10

Combinational area:        373.400269
Noncombinational area:       0.000000
Net Interconnect area:      undefined  (No
```

[그림 5] 공통 S-box S_U의 합성 결과

[표 1] S_A와 S_U의 하드웨어 복잡도 및 지연 시간

S-box 기능	게이트 카운트	지연 시간 (ns)
$S_A(S_1, S_1^{-1})$	272	6.17
$S_U(S_1, S_1^{-1}, S_2, S_2^{-1})$	373	6.63

본 논문에서는 이러한 공통 S-box S_A와 S_U를 이용하여, (S_A, S_U, S_A, S_U)로 구성되는 통합 변환 (Unified Transformation, UT) 블록을 구성하고, 최종적으로 AES와 ARIA 알고리즘용 통합 치환 계층을 [그림 6]과 같이 설계하였다. [그림 3]의 공통 S-box 블록들의 다중화기의 (MUX) 입력 선택 신호를 변경하여, 통합 변환 (UT) 블록은 ARIA의 경우는 (S_1, S_2, S_1^{-1}, S_2^{-1})나 (S_1^{-1}, S_2^{-1}, S_1, S_2)로 동작하며, AES의 경우는 네 개의 S_1이나 S_1^{-1}로 동작하게 된다.



[그림 6] 제안하는 통합 치환 계층의 내부 구조

Ⅳ. 통합 확산 계층

AES의 확산 계층에서는 MixColumns와 InvMix Columns 연산을 암호화와 복호화 시에 각각 수행한다. 각 연산은 $GF(2^8)$ 상의 덧셈과 곱셈을 수반하며, 4 × 4 행렬 곱셈으로 표현될 수 있다[2]. 또한 InvMix Columns 연산의 행렬은 MixColumns의 행렬을 포함하도록 정리가 가능하므로, MixColumns와 InvMix Columns 연산은 하드웨어 자원을 공유하여 효율적으로 구현될 수 있다[6]. 반면에, ARIA의 확산 계층 함수는 16 × 16 involutional 이진 행렬을 사용한다. 이러한 이진 행렬의 각 행 (row)은 7 개의 '1'을 가지므로, 일반적인 방법으로 ARIA의 확산 함수를 구현하기 위해서는 총 768개 (6×16×8)의 2 입력 XOR 게이트가 필요하다.[2]

AES와 ARIA의 통합 확산 계층을 효율적으로 설계하기 위해, AES의 MixColumns와 InvMixColumns, 그리고 ARIA의 확산 행렬이 공통으로 사용하는 2 입력 XOR 항을 조사하여 공유하는 방법을 사용한다. 16 바이트 입력 $(x_0, x_1, ..., x_{14}, x_{15})$에 대해, 다음과 같은 1 바이트 공유항 α_i와 β_j를 찾을 수 있다.

$$\alpha_0 = x_0 \oplus x_1,\ \alpha_1 = x_1 \oplus x_2,$$
$$\alpha_2 = x_2 \oplus x_3,\ \alpha_3 = x_3 \oplus x_0,$$
$$\alpha_4 = x_4 \oplus x_5,\ \alpha_5 = x_5 \oplus x_6,$$
$$\alpha_6 = x_6 \oplus x_7,\ \alpha_7 = x_7 \oplus x_4,$$
$$\alpha_8 = x_8 \oplus x_9,\ \alpha_9 = x_9 \oplus x_{10},$$
$$\alpha_{10} = x_{10} \oplus x_{11},\ \alpha_{11} = x_{11} \oplus x_8,$$
$$\alpha_{12} = x_{12} \oplus x_{13},\ \alpha_{13} = x_{13} \oplus x_{14},$$
$$\alpha_{14} = x_{14} \oplus x_{15},\ \alpha_{15} = x_{15} \oplus x_{12},$$

$$\beta_0 = x_0 \oplus x_2,\ \beta_1 = x_1 \oplus x_3,$$
$$\beta_2 = x_4 \oplus x_6,\ \ \beta_3 = x_5 \oplus x_7,$$
$$\beta_4 = x_8 \oplus x_{10},\ \beta_5 = x_9 \oplus x_{11},$$
$$\beta_6 = x_{12} \oplus x_{14},\ \beta_7 = x_{13} \oplus x_{15}.$$

MixColumns 연산의 출력 $(y_0, y_1, ..., y_{14}, y_{15})$는 공유항 α_i를 이용하여 다음과 같이 정리될 수 있다.

$$u_0 = 2\alpha_0 \oplus \alpha_2 \oplus x_1,\ u_1 = 2\alpha_1 \oplus \alpha_3 \oplus x_2,$$
$$u_2 = 2\alpha_2 \oplus \alpha_0 \oplus x_3,\ u_3 = 2\alpha_3 \oplus \alpha_1 \oplus x_0,$$
$$u_4 = 2\alpha_4 \oplus \alpha_6 \oplus x_5,\ u_5 = 2\alpha_5 \oplus \alpha_7 \oplus x_6,$$
$$u_6 = 2\alpha_6 \oplus \alpha_4 \oplus x_7,\ u_7 = 2\alpha_7 \oplus \alpha_5 \oplus x_4,$$
$$u_8 = 2\alpha_8 \oplus \alpha_{10} \oplus x_9,\ u_9 = 2\alpha_9 \oplus \alpha_{11} \oplus x_{10},$$
$$u_{10} = 2\alpha_{10} \oplus \alpha_8 \oplus x_{11},\ u_{11} = 2\alpha_{11} \oplus \alpha_9 \oplus x_8,$$
$$u_{12} = 2\alpha_{12} \oplus \alpha_{14} \oplus x_{13},\ u_{13} = 2\alpha_{13} \oplus \alpha_{15} \oplus x_{14},$$
$$u_{14} = 2\alpha_{14} \oplus \alpha_{12} \oplus x_{15},\ u_{15} = 2\alpha_{15} \oplus \alpha_{13} \oplus x_{12}.$$

또한 InvMixColumns 연산의 출력 $(u_0, u_1, ..., u_{14}, u_{15})$는 u_k와 공유항 β_j를 사용하여 다음과 같이 정리될 수 있다.

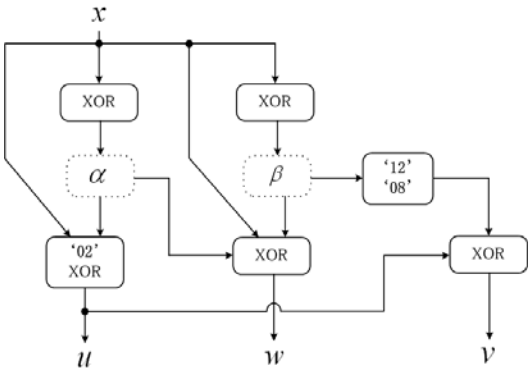
$$v_0 = 12\beta_0 \oplus 8\beta_1 \oplus u_0,\ v_1 = 12\beta_1 \oplus 8\beta_0 \oplus u_1,$$
$$v_2 = 12\beta_0 \oplus 8\beta_1 \oplus u_2,\ v_3 = 12\beta_1 \oplus 8\beta_0 \oplus u_3,$$
$$v_4 = 12\beta_2 \oplus 8\beta_3 \oplus u_4,\ v_5 = 12\beta_3 \oplus 8\beta_2 \oplus u_5,$$
$$v_6 = 12\beta_2 \oplus 8\beta_3 \oplus u_6,\ v_7 = 12\beta_3 \oplus 8\beta_2 \oplus u_7,$$
$$v_8 = 12\beta_4 \oplus 8\beta_5 \oplus u_8,\ v_9 = 12\beta_5 \oplus 8\beta_4 \oplus u_9,$$
$$v_{10} = 12\beta_4 \oplus 8\beta_5 \oplus u_{10},\ v_{11} = 12\beta_4 \oplus 8\beta_5 \oplus u_{11},$$
$$v_{12} = 12\beta_6 \oplus 8\beta_7 \oplus u_{12},\ v_{13} = 12\beta_7 \oplus 8\beta_6 \oplus u_{13},$$
$$v_{14} = 12\beta_6 \oplus 8\beta_7 \oplus u_{14},\ v_{15} = 12\beta_7 \oplus 8\beta_6 \oplus u_{15}.$$

마지막으로, ARIA 확산 함수의 출력 $(w_0,w_1,\ldots,w_{14},w_{15})$는 공유항 α_i와 β_j를 이용하여 다음과 같이 정리된다.

$$w_0 = x_3 \oplus \beta_2 \oplus \alpha_8 \oplus \alpha_{13},\ w_1 = x_2 \oplus \beta_3 \oplus \alpha_8 \oplus \alpha_{15},$$
$$w_2 = x_1 \oplus \beta_2 \oplus \alpha_{10} \oplus \alpha_{15},\ w_3 = x_0 \oplus \beta_3 \oplus \alpha_{10} \oplus \alpha_{13},$$
$$w_4 = x_5 \oplus \beta_0 \oplus \alpha_{11} \oplus \alpha_{14},\ w_5 = x_4 \oplus \beta_1 \oplus \alpha_9 \oplus \alpha_{14},$$
$$w_6 = x_7 \oplus \beta_0 \oplus \alpha_9 \oplus \alpha_{12},\ w_7 = x_6 \oplus \beta_1 \oplus \alpha_{11} \oplus \alpha_{12},$$
$$w_8 = x_{10} \oplus \beta_7 \oplus \alpha_0 \oplus \alpha_7,\ w_9 = x_{11} \oplus \beta_6 \oplus \alpha_0 \oplus \alpha_5,$$
$$w_{10} = x_8 \oplus \beta_7 \oplus \alpha_2 \oplus \alpha_5,\ w_{11} = x_9 \oplus \beta_6 \oplus \alpha_2 \oplus \alpha_7,$$
$$w_{12} = x_{12} \oplus \beta_5 \oplus \alpha_1 \oplus \alpha_6,\ w_{13} = x_{13} \oplus \beta_4 \oplus \alpha_3 \oplus \alpha_6,$$
$$w_{14} = x_{14} \oplus \beta_5 \oplus \alpha_3 \oplus \alpha_4,\ w_{15} = x_{15} \oplus \beta_4 \oplus \alpha_1 \oplus \alpha_4.$$

위 식에서 각 w_k는 세 개의 공유항 (하나의 β_j와 두 개의 서로 다른 α_i)를 사용하므로, 자원 공유 기법을 이용하여 총 384개의 (3 공유항 × 16 행 × 8 비트) 2 입력 XOR 게이트를 절약할 수 있다. [그림 7]에는 공통항을 이용하여, AES의 MixCloumns와 Inv MixColumns, 그리고 ARIA의 확산 함수를 선택적으로 수행하는 통합 확산 계층의 구조를 개념적으로 나타내었다.

공통항을 이용하여 통합 확산 계층을 설계하고 0.25um CMOS 공정으로 합성한 결과를 그림 7과 8에 나타내었고, [표 2]에 정리하였다. AES의 확산 함수인 MixColumns와 InvMixColumns 기능에 공통항을 사용하여 ARIA 확산 함수를 추가적으로 구현한 경우, 하드웨어 자원이 49% 증가하게 된다. 반면에 InvMix Columns 함수의 출력 v_k가 최대 지연 경로 (critical path)를 발생시키므로 전체적인 확산 함수의 지연 시간은 증가하지 않는다.



[그림 7] 제안하는 통합 확산 계층의 구조

```
****************************************
Report : area
Design : MixnInvMixColnDiff
Version: W-2004.12-SP2-1
Date   : Mon Apr  2 13:43:57 2007
****************************************

Library(s) Used:

    anam (File: /RAID1/user/dwlee/WORK/LIB

Number of ports:              512
Number of nets:              1088
Number of cells:              120
Number of references:           2

Combinational area:       1715.988647
Noncombinational area:       0.000000
Net Interconnect area:      undefined  (No
```

[그림 8] MixCol.+InvMixCol. 로직의 합성 결과

```
****************************************
Report : area
Design : MixnInvMixColnDiff
Version: W-2004.12-SP2-1
Date   : Mon Apr  2 13:40:52 2007
****************************************

Library(s) Used:

    anam (File: /RAID1/user/dwlee/WORK/LIB

Number of ports:              512
Number of nets:              1472
Number of cells:              168
Number of references:           2

Combinational area:       2560.770020
Noncombinational area:       0.000000
Net Interconnect area:      undefined  (No
```

[그림 9] MixCol.+InvMixCol.+ARIA Diff. 로직의 합성 결과

[표 2] 공통항을 이용한 통합 확산 계층의 하드웨어 성능

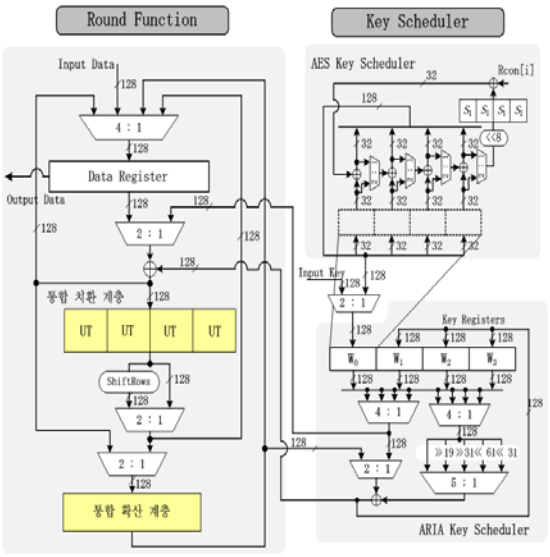
확산 기능	2입력 XOR 갯수	게이트 카운트	지연 시간 (ns)
MixCol. + InvMixCol.	780	1,715	1.64
MixCol. + InvMixCol. + ARIA Diff.	1,164	2,560	1.64

Ⅴ. AES-ARIA 통합 연산기의 하드웨어 설계

본 논문에서 최종 제안하는 AES-ARIA 통합 하드웨어 연산기의 구조를 [그림 10]에 나타내었다. 제안하는 연산기는 크게 라운드 함수 블록과 키 스케줄러 블록으로 구성된다. 라운드 함수 블록은 앞서 설명한 통합 치환 계층과 통합 확산 계층을 중심으로, 128비트 데이터

저장용 레지스터, 데이터 경로 선택을 위한 다중화기 등으로 구성된다. 한편 AES와 ARIA의 키 스케줄링 과정은 상이하므로, 그림과 같이 라운드 키 저장용 128비트 레지스터를 공유하는 방식으로 설계한다. 즉, ARIA 키 스케줄링에서 사용하는 네 개의 128비트 초기키 값 중 하나인 W_0를 저장하는 레지스터와 AES의 128비트 라운드 키를 저장하는 레지스터를 공유하여 사용한다.

한편 AES 키 스케줄링 과정에는 4개의 S_1 S-box 연산이 필요한데, 이를 라운드 함수 내부의 S-box를 이용하여 수행하는 경우는 매 라운드마다 별도의 클록 사이클이 소요된다. 또한 전체 연산기의 크기에 비해 4개의 S_1 S-box가 차지하는 면적은 5% 미만이므로 본 연산기는 [그림 8]과 같이 별도의 S_1 S-box들을 사용하였다. 따라서, 본 연산기의 모든 동작은 128비트 단위로 이루어지며, 동시에 라운드 키 확장은 'on-the-fly' 방식으로 수행된다. 결과적으로 제안하는 연산기는 128비트 한 블록에 대한 AES 암호화는 11 클록 사이클을 소모하며, 복호화 시에는 초기 라운드 키로부터 마지막 라운드 키를 계산하는 데 10 클록 사이클이 필요하므로 총 21 클록 사이클이 소모된다. 한편, ARIA 암호화 및 복호화 시에는 라운드 키 초기화 과정에 3 클록 사이클이 사용되고 13 라운드를 수행하므로, 총 16 클록 사이클이 128비트 한 블록을 처리하는 데 소모된다.



[그림 10] 제안하는 AES-ARIA 통합 연산기의 구조

Ⅵ. 구현 결과 및 성능

제안하는 AES-ARIA 통합 연산기는 Verilog HDL (Hardware Description Language)를 이용하여 구현하였으며, Synopsys 사의 Design Compiler와 0.25um CMOS 스탠다드 라이브러리를 이용하여 합성하고, 그 결과를 [그림 11]에 나타내었다. 제안하는 연산기는 19,056 게이트 카운트의 비교적 소형 크기를 가지고, 최대 90MHz 클록 속도로 동작할 수 있다.

기존 블록암호 연산기들과의 성능 비교 결과를 [표 3]에 나타내었다. 제안하는 연산기는 128비트 구조를 가지며 AES와 ARIA 블록암호의 암호화 및 복호화가 가능하면서도, 현재까지 알려진 가장 소형의 128비트 구조 AES 연산기[4]에 비해 53%만 증가된 크기를 나타낸다. 또한, 비교를 위해 별도로 구현한 소형 ARIA 연산기에 비해 23% 증가된 크기를 가진다. 한편, Satoh[9] 등은 AES와 Camellia 블록암호를 수행할 수 있는 통합 연산기를 제안하였고, 이 연산기는 제안하는 연산기에 비해 22% 감소된 크기를 가진다. 하지만, 이 연산기는 64비트 구조를 가지므로, 본 논문에서 제안하는 연산기에 비해 두 배 이상의 클록 사이클을 소모하게 된다. 결과적으로 제안하는 AES-ARIA 통합 연산기는 개별 구현 결과에 비해 하드웨어 자원 측면에서 32% 정도 절약되고, AES와 ARIA 암호화에 대해 각각 1,047Mbps와 720Mbps의 우수한 성능을 가진다.

[표 3] 기존 블록암호 연산기들과의 성능 비교

디자인 (um)	게이트 카운트	기능	클록 사이클	성능 (Mbps)
제안하는 연산기 (0.25)	19,056	ARIA	16	720
		AES (E/D)	11/21	1,047/548
ARIA (0.25)	15,496	ARIA	16	816
AES［6］ (0.18)	12,454	AES	11	1,691
AES-Camellia［9］ (0.18)	14,918	Camellia	22	661
		AES	31	469

Reference	Library	Unit Area	Count	Total Area	Attributes
AESB32		896.399963	1	896.399963	d, h
FF128b		832.000000	4	3328.000000	d, h, n
FF128b_sr		907.500000	1	907.500000	d, h, n
MUX32b_2in		70.400002	4	281.600006	d, h
MUX128b_2in		281.600006	4	1126.400024	d, h
MUX128b_4in		661.299988	3	1983.899902	d, h
MUX128b_5in		775.900024	1	775.900024	d, h
MixColnDiff		2560.400146	1	2560.400146	d, h
RCON_ROM		25.500000	1	25.500000	d, h
ShiftRows		230.399994	1	230.399994	d, h
USB32		1292.199951	4	5168.799805	d, h
AX003D1	anam	7.300000	1	7.300000	
...		...	...	.	...
UnifiedAESnARIA_DW01_inc_7_0		19.700001	1	19.700001	h
Total 46 references				19056.699219	

[그림 11] 제안하는 AES-ARIA연산기의 합성 결과

Ⅶ. 결 론

본 논문에서는 AES와 ARIA 블록암호의 통합 하드웨어 연산기를 효율적으로 설계하는 방법을 제안하고, 그 구현 결과를 제시하였다. 복합체 $GF(((2^2)^2)^2)$ 상의 역원 연산을 이용하여 자원 효율적인 공통 S-box를 설계하였고, 두 알고리즘의 확산 함수 행렬의 공통항을 축출하여 하나의 통합 확산 계층을 설계하였으며, 이를 이용하여 자원 효율적인 128비트 구조의 AES-ARIA 통합 연산기를 설계하였다. 제안하는 연산기는 AES와 ARIA 블록암호를 모두 수행하는 최초의 연산기이며, 0.25um CMOS 공정을 기준으로, 개별 연산기 구현 방법보다 32% 감소된 19,056 게이트 카운트의 크기를 가진다. 또한 최대 90MHz 클록 속도를 기준으로 AES와 ARIA 암호화를 각각 1,047 Mbps와 720Mbps의 속도로 수행할 수 있다.

참고문헌

[1] FIPS Pub. 197: Specification for the AES, Nov. 2001, available at: http://csrc.nist.gov /publications/fips/fips197/fips-197.pdf.

[2] NSRI: ARIA Algorithm Specification, http://www.nsri.re.kr/ARIA/doc/ARIA-specification.pdf.

[3] D. Kwon, J. Kim, S. Park, S. Sung, Y. Sohn, J. Song, Y. Yeom, E. Yoon, S. Lee, J. Lee, S. Chee, D. Han and J. Hong, “New block cipher: ARIA,” Proceeding ICISC2003, pp. 432-445, Nov. 2003.

[4] PKCS#11 v2.20 Amendment 3 Rev. 1: Additional PKCS#11 Mechanisms, available at: ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-11/v2-20/pkcs-11v2-20a3.pdf.

[5] Vincent Rijmen, “Efficient implementation of the Rijndael S-box,” available at: http://www.esat.kuleuven.ac.be/~rijmen/rijndael/sbox.pdf, 2001.

[6] A. Satoh, S. Mroioka, K. Takano, and Seiji Munetoh, “A compact Rijndael hardware architecture with S-box optimization,” Advan ces in Cryptology-ASIACRYPT 2001, LNCS 2248, pp. 239-254, Springer-Verlag, 2001.

[7] D. Canright, “A Very Compact S-Box for AES,” CHES'05, LNCS 3659, pp. 441-455, Springer-Verlag, 2005.

[8] 전기원, 전은아, 지재덕, 박익수, 정석원, 서재현, 오병균, “유비쿼터스 환경에 적합한 블록암호 ARIA의 S-Box FPGA 구현,” CISC 2005, pp. 621-625, Jun. 2005.

[9] A. Satoh and S. Morioka, “Unified Hardware Architecture for 128-Bit Block Ciphers AES and Camellia,” CHES'03, LNCS 2779, pp. 304-318, Springer-Verlag, 2003.

[10] H. Kuo and I. Verbauwhede, “Architectural Optimization for a 1.82Gbits/sec VLSI Implementation of the AES Rijndeal Algorithm,” CHES'01 pp. 53-67, Springer-Verlag, 2001.

[11] I. Verbauwhede, P. Schaumont, and H. Kuo, “Design and Performance Testing of a 2.29 Gb/s Rijndael Processor,” IEEE Journal of Solid-State Circuits, vol. 38, no. 3, pp. 569-572, March 2003.

[12] M. Feldhofer, S. Dominikus, and J. Wolkerstorfer, “Strong Authentication for RFID Systems Using the AES Algorithm,” CHES'04, LNCS 3156, pp. 357-370, Springer-Verlag, 2004.

[13] J. Park, Y.-D. Kim and Y. You, “A Scale Down Structure of ARIA Processor,” Journal of the Research Institute for Computer and

Information Communication, pp. 79-84, Dec. 2005.

[14] J. Park, Y. Yun, S. Kim, Y.-D. Kim, and Y. You, “A crypto-processor for security PDA systems,” Proceeding ISOCC2005, pp. 11-14, Oct. 2005.

[15] J. Park, Y.-D. Kim, S. Yang, and Y. You, “Low power compact design of ARIA block cipher,” Proceeding ISCAS2006, pp. 313-316, May 2006.

[16] A. Biryukov, C. D. Canniére, J. Lano, S. B. Ors, and B. Preneel: Security and performance analysis of ARIA, http://www.nsri.re.kr/ ARIA/ doc/ARIA-COSICreport. pdf, Jan. 2003.

[17] www.synopsys.com

[18] 윤영상, 조광두, 김용대, 한선경, 유영갑, “네트워크 환경에 적합한 AES 암호프로세서 구조 분석,” 정보보호학회논문지, 제15권 5호, pp. 3-11, 2005.

[19] 구본석, 유권호, 양상운, 장태주, “RFID 태그를 위한 초소형 AES 연산기의 구현,” 정보보호학회논문지, 제16권 5호, pp. 67-77, 2006.

[20] 유권호, 구본석, 양상운, 장태주, “경량화된 확산계층을 이용한 32-비트 구조의 소형 ARIA 연산기 구현,” 정보보호학회논문지, 제16권 6호, pp. 15-24, 2006.

<著者紹介>

구 본 석 (Bonseok Koo) 정회원
1998년 2월 : 경북대학교 전자공학과 학사
2000년 2월 : 포항공과대학교 전자공학과 석사
2000년 2월~2000년 9월 : LG 정보통신 중앙 연구소 연구원
2000년 10월~현재 : ETRI부설 연구소 선임연구원
<관심분야> 암호칩 설계, 공개키 암호, 부채널 공격

유 권 호 (Gwonho Ryu) 정회원
1999년 2월 : 포항공과대학교 전자공학과 학사
2001년 2월 : 포항공과대학교 전자공학과 석사
2002년 9월~현재 : ETRI부설 연구소 선임연구원
<관심분야> 암호칩 설계, 블록 암호, 부채널 공격

장 태 주 (Taejoo Chang) 정회원
1982년 2월 : 울산대학교 전기공학과 학사
1990년 : 한국과학기술원 전기 및 전자공학과 석사
1998년 : 한국과학기술원 전기 및 전자공학과 공학박사
1982년~2000년 : 국방과학연구소 선임연구원
2000년~현재 : ETRI 부설연구소 책임연구원
<관심분야> 암호칩 설계, 정보보호, 통계학적 신호처리

이 상 진 (Sangjin Lee) 종신회원
1987년 2월 : 고려대학교 수학과 학사
1989년 2월 : 고려대학교 수학과 석사
1994년 2월 : 고려대학교 수학과 박사
1989년 2월~1999년 2월 : 한국전자통신연구원 선임연구원
1999년 2월~2001년 8월 : 고려대학교 자연과학대학 조교수
2000년 2월~현재 : 고려대학교 정보보호대학원 부교수
<관심분야> 대칭키 암호의 분석 및 설계, 정보은닉이론, 컴퓨터 포렌식