JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지 Vol. 23, No. 7: 875~880, Jul. 2019

클러스터를 이용한 고성능 RC4 암호화 하드웨어 설계

이규희*

The Design of a High-Performance RC4 Cipher Hardware using Clusters

Kyu-Hee Lee*

*Technical Director, R&D Center, World Tech Inc., Wonju, 26408 Korea

요 약

RC4 스트림 암호화는 내부 구현이 간단하고 빠르게 암호화 할 수 있는 초경량 암호화 알고리즘으로 IEEE 802.11의 WEP와 IEEE 802.11i의 TKIP 등에 널리 이용되고 있다. RC4는 IoT 등의 제한적 자원을 갖는 시스템들에도 사용되지만 성능상 제약이 있다. RC4 암호화는 S-배열과 K-배열의 초기화 및 랜덤화를 수행하는 KSA(Key Scheduling Algorithm)와 랜덤화된 S-배열을 이용하여 암호문을 생성하는 PRGA(Pseudo-Random Generation Algorithm)의 두 단계로 구성된다. 본 논문에서는 KSA에서 발생하는 초기화 지연시간을 줄이기 위해, 랜덤화 과정에 초기화를 삽입하여 함께 처리한다. KSA의 랜덤화에서 교환(swap) 작업과 PRGA의 암호문 생성은 클러스터를 이용하여 매 클록마다 두 개의 교환 및 암호문이 생성되도록 하였다. 제안된 RC4 암호화 하드웨어 구조는 초기화 지연시간이 발생하지 않으며, 랜덤화와 키 스트림 생성율에서 다른 연구들과 비교하여 약 2배에서 6배의 성능이 향상되었다.

ABSTRACT

A RC4 stream cipher is widely used for security applications such as IEEE 802.11 WEP, IEEE 802.11i TKIP and so on, because it can be simply implemented to dedicated circuits and achieve a high-speed encryption. RC4 is also used for systems with limited resources like IoT, but there are performance limitations. RC4 consists of two stages, KSA and PRGA. KSA performs initialization and randomization of S-box and K-box and PRGA produces cipher texts using the randomized S-box. In this paper, we initialize the S-box and K-box in the randomization of the KSA stage to reduce the initialization delay. In the randomization, we use clusters to process swap operation between elements of S-box in parallel and can generate two cipher texts per clock. The proposed RC4 cipher hardware can initialize S-box and K-box without any delay and achieves about 2 times to 6 times improvement in KSA randomization and key stream generation.

키워드 : 스트림 암호화 알고리즘, RC4 하드웨어, 경량암호화 알고리즘, 클러스터 암호화

Keywords : Stream cipher algorithm, RC4 hardware, Light-weight cipher algorithm, cluster encryption

Received 27 May 2019, **Revised** 28 May 2019, **Accepted** 7 June 2019

*** Corresponding Author** Kyu-Hee Lee(E-mail:csys.epsilon@gmail.com, Tel:+82-33-746-4884)
Technical Director, R&D Center, World Tech Inc., Wonju, 26408 Korea

Open Access http://doi.org/10.6109/jkiice.2019.23.7.875 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

RC4 스트림 암호화는 소프트웨어로 구현될 수 있으며 내부 구현이 간단하고 빠르게 처리될 수 있는 초경량 암호화 알고리즘이다. RC4 암호화 알고리즘은 IEEE 802.11의 일부부인 무선랜 보안 프로토콜 WEP(Wired Equivalent Privacy), 오라클의 SQL 보안, IEEE 802.11i의 TKIP(Temporal Key Integrity Protocol) 등에 이용되고 있다[1][2]. 소프트웨어로 구현된 RC4 암호화 는 초기 네트워크 환경에 충분한 성능을 갖도록 설계되었지만, 현재 고속의 네트워크에서 일련의 스트림 암호화 과정은 성능 열화를 초래할 수 있다. 또한, 초경량 암호화 알고리즘들은 주로 제한적 자원을 갖는 IoT (Internet of Things)에도 사용되는데[1][3] 적은 자원만을 사용하여 구현되기 때문에, 성능에서 제한점을 갖는다.

RC4 스트림 암호화 알고리즘은 1987년 Ron Rivest에 의해 개발된 가변 길이의 키를 갖는 스트림 암호화로써, 8×256 크기의 S-배열(S-box)과 K-배열(K-box)에 대하여 초기화를 수행하고, 모듈로 연산, 덧셈, 교환(swapping)을 통해 랜덤화를 수행한다. 초기화 및 랜덤화가 수행된 S-배열의 임의의 값과 평문은 XOR 연산을 통해 바이트 단위의 암호문으로 생성된다.

본 논문에서는 S-배열과 K-배열의 초기화에 소비되는 지연시간을 최소화하고 교환 작업의 성능을 향상시키기 위해 병렬 처리를 이용한 고성능 RC4 암호화 방식을 제시하며 다양한 키 길이를 지원할 수 있는 유연한 하드웨어 구조를 제안한다.

본 논문은 2장에서 RC4 암호화 알고리즘의 동작방식과 관련연구들을 설명하고, 3장에서 제안된 고성능 RC4 암호화 하드웨어를 구성하기 위해 요구되는 문제점들과 하드웨어 구조를 제시한다. 4장에서는 제안된 RC4 암호화 하드웨어 구조의 복잡도 및 성능에 대한 비교평가를 수행하고 5장에서 결론을 맺는다.

Ⅱ. 관련연구

RC4 암호화 알고리즘은 구현의 간단성과 편의성 때문에 경량 암호화가 요구되는 시스템에서 널리 사용되고 있지만, S-배열과 K-배열의 초기화에 따른 지연시간은 성능 열화를 초래할 수 있다. 초기화 지연시간을 줄이기 위해 적은 수의 방법들이 연구되었다[4][5][6].

Key Scheduling Algorithm

```
// Initialization
1:  for i=0 to 256 begin
2:    S[i] = i
3:    K[i] = key(i mod key_length)
4:  end for
// Randomization
5:  j = 0
6:  for i=0 to 256 begin
7:    j = (j + S[i] + K[i]) mod 256
8:    swap(S[i], S[j])
9:  end for
```

Pseudo-Random Generation Algorithm

```
10:  i = 0, j = 0
11:  while (more bytes to encrypt) do
12:    i = (i + 1) mod 256
13:    j = (j + S[i]) mod 256
14:    swap(S[i], S[j])
15:    t = (S[i] + S[j]) mod 256
16:    ciphertext = plaintext ^ S[t]
17:  end while
```

Fig. 1 RC4 stream cipher algorithm

2.1. RC4 스트림 암호화 알고리즘

RC4 암호화 알고리즘은 블록 암호 방식과 같은 뒤섞임이 없는 대신에 메모리 내 데이터의 위치를 교환하여 뒤섞임과 같은 결과를 도출하며, 가변 길이(5-256바이트)의 키에 대하여 2의 지수승으로 모듈로 연산을 수행한다. 그림 1은 RC4 암호화 알고리즘의 두 가지 동작에 대한 의사생성코드(Pseudo-code)이다.

2.2.1. Key Scheduling Algorithm(KSA)

KSA는 랜덤화를 위한 S-배열의 초기화 및 입력되는 키 값을 K-배열에 저장하는 초기화 과정과 S-배열과 K-배열을 더하여 임의의 S-배열 위치에 저장하는 랜덤화 과정으로 나누어 수행된다. 초기화 단계에서 S-배열은 i 값을 1씩 증가하여 저장되며 K-배열에는 i 값을 키 길이로 모듈로 연산을 수행하여 주어진 키 값들이 나누어 저장된다. 만일, 키의 길이가 256 바이트 보다 작은 경우 키 값 전체를 K-배열에 저장하고 K-배열의 남은 부분을 키의 처음부터 순차적으로 채워나간다. S-배열과 K-배열의 초기화가 완료되면, j 값을 계산하고 S-배열의 i와 j 인덱스에 대한 값을 서로 교환하여 랜덤화를 수행한다.

이전의 연구들에서는 랜덤화 단계의 교환 및 모듈로 연산의 최적화를 중점적으로 연구하였으나, 초기화에 따른 지연시간들을 줄이기 위한 연구는 거의 수행되지 않았다. 이전의 모듈로 연산을 사용한 방식은[4] KSA의 초기화를 랜덤화 단계에 삽입하여 랜덤화와 동시에 수행되는 구조를 제안하여 초기화에 따른 지연시간을 최소화하였지만, 교환 및 모듈로 연산으로 인한 조합회로의 지연시간은 증가하였다.

2.2.2. Pseudo-Random Generation Algorithm(PRGA)

PRGA는 KSA로부터 랜덤화된 S-배열을 이용하여 새로 계산되는 인덱스 i와 j를 통해 교환 작업을 수행한다. 마지막으로 S-배열의 i 및 j 인덱스 값을 더하여 평문과 XOR 연산으로 최종 암호문을 생성한다. 복호화는 암호문이 생성되는 과정의 반대로 수행하면 평문을 얻을 수 있다.

대부분의 이전 연구들은 모듈로(Modular) 연산을 위해 카운터를 이용하여 간단하게 구현되었지만[7], 몇몇 연구들은 카운터의 동기화 등의 이유로 수학적 이론에 기반한 모듈로 연산을 수행하였다[4][8]. Gupta[6] 등은 RC4 암호화의 성능 향상을 위해 PRGA에서 교환 작업을 두 개의 레지스터를 할당하여 처리하였다. 그러나 Gupta 등의 구조는 동시에 두 개의 교환을 위해 K-배열이 없는 PRGA에서만 병렬처리를 수행하였고 KSA 내에 존재하는 교환은 고려하지 않았으며, 초기화 지연시간이 고려되지 않아 암호화 하드웨어의 전체적인 성능은 크게 향상되지 않았다.

본 논문에서는 KSA에서 초기화에 따른 지연시간을 최소화하기 위한 방법을 제시하고 KSA 및 PRGA에서 S-배열 및 K-배열에 사용되는 i와 j 인덱스의 계산과 교환 작업을 클러스터를 이용하여 병렬로 수행할 수 있는 고성능의 RC4 암호화 하드웨어를 제안한다.

Ⅲ. 클러스터 기반 RC4 하드웨어 구조

RC4 암호화는 KSA 단계에서 S-배열 및 K-배열의 초기화와 랜덤화가 이루어지는데, 이전 연구들에서는 초기화 지연시간을 언급하지 않았지만, 본 논문에서는 초기화에 따른 지연시간을 감소시키기 위하여 초기화 과정을 KSA의 랜덤화 과정에 삽입하여 동시에 처리하도록 한다.

3.1. 클러스터를 이용한 인덱스 생성

본 논문에서는 고성능의 RC4 암호화를 수행하기 위해 병렬처리가 가능한 별도의 클러스터들을 구성한다. 사용되는 클러스터 수는 데이터의 의존성에 따라 결정된다. 만일, 클러스터의 수가 4일 때, 최악의 경우 총 4개의 인덱스에서 모두 충돌이 발생할 수 있는데 이는 데이터의 병렬성을 저하시키고 하드웨어의 복잡도와 소비되는 클록 수를 증가시킨다. 따라서, 본 논문에서는 클러스터의 수를 2로 결정하여 데이터 의존성을 최소로 한다.

RC4 알고리즘은 교환을 위하여 두 개의 인덱스 i와 j를 생성하는데 i는 간단히 구할 수 있지만 j 값을 구하기 위해 이전의 j 값을 기반으로 계산해야 한다. 만일, 두 개의 클러스터를 사용할 때, j는 다음과 같이 구할 수 있다.

$$j_0 = (j_{init} + i_0 + K[i_0]) \\ j_1 = (j_0 + i_1 + K[i_1]) \quad (1)$$

식(1)의 j_0을 풀어서 기재하면 j_1에 대한 식은 다음과 같다.

$$j_1 = ((j_{init} + i_0 + K[i_0]) + i_1 + K[i_1]) \quad (2)$$

위의 식(1)과 식(2)를 이용하여 S-배열의 교환 작업을 수행할 수 있는데, 두 개의 i는 동일한 값을 가질 수 없지만, 각 i 값과 j 값들 사이에는 동일한 값이 생성될 수 있다. 이는 S-배열의 동일한 곳을 지칭하는 것으로 서로 간의 의존성 때문에 S-배열의 교환 작업을 병렬로 처리할 수 없다.

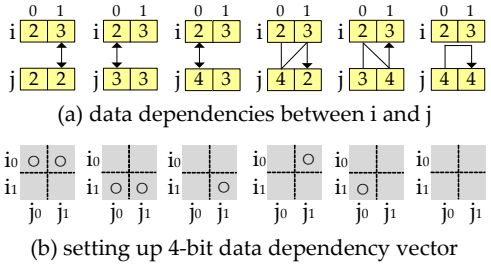

Fig. 2 data dependency check

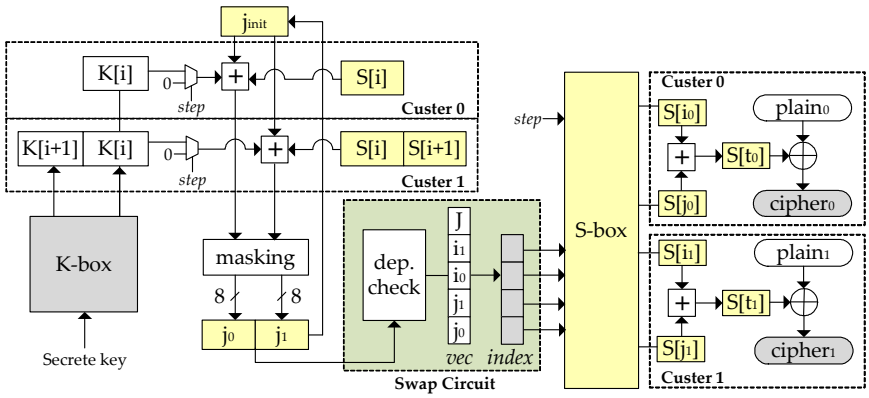

Fig. 3 RC4 hardware architecture using two clusters

3.2. 클러스터 간 데이터 의존성 검사

본 논문에서는 S-배열 요소 간의 교환을 위해 두 개의 클러스터를 사용하는데, 동일한 인덱스가 생성되는 경우 데이터의 의존성이 발생한다. 인덱스 간 서로 동일한 값을 가질 확률은 1/256으로 매우 적지만, 교환 작업의 병렬 처리를 위해 데이터 의존성은 반드시 해결되어야 한다.

그림 2(a)는 두 개의 클러스터를 이용하여 교환 작업을 할 때 발생할 수 있는 모든 경우의 수를 나타내었다. 병렬 처리를 위해 본 논문에서는 그림 2(b)와 같이 4개의 각 인덱스를 비교하여 동일한 값에 대한 사분면의 비트를 세트하여 의존성을 표시한다. 데이터 의존성 벡터(vec)는 각 사분면을 1비트씩 할당하고, j_0과 j_1의 값이 동일한지를 체크하는 비트를 J로 표기하여 총 5비트로 구성한다. 데이터 의존성 벡터는 교환 회로에서 S-배열에 대한 인덱스 정보로 사용된다.

그림 2(a)에서 i 위의 0과 1은 세로로 나누었을 때, 클러스터 0과 클러스터 1을 나타낸다. 데이터 의존성이 발생되었을 때 S-배열의 교환 작업은 화살표와 같은 절차로 수행되며, 이는 회로에서 병렬로 처리된다. 그림 2(b)는 그림 2(a)의 데이터 의존성이 있는 경우 데이터 의존성 벡터에 세트되는 사분면을 나타낸다.

데이터 의존성 검사는 교환 작업 수행 이전에 추가 클록을 소비하여 성능 저하의 원인이 될 수 있지만, 제안된 구조는 추가 클록의 소비로 인한 성능저하를 막기 위해 클록의 상승에지와 하강에지를 모두 사용하는 구조로 설계되었다. 두 개의 에지를 모두 사용하는 경우 최대 클록 주파수는 감소하지만 클록 당 처리량은 증가하여 전체 성능은 향상된다.

3.3. 제안된 RC4 하드웨어 구조

두 개의 클러스터를 이용한 i와 j 인덱스의 생성 및 데이터 의존성 검사 회로를 포함한 RC4 암호화 하드웨어의 구조는 그림 3과 같다. 입력되는 가변 길이(5~256바이트)의 비밀 키는 K-배열로 메모리에 저장되고 각 클러스터에 공급된다. i를 통해 S-배열와 K-배열의 값을 인출하여 업데이트된 j와 더하면 새로운 j가 생성된다. 각 클러스터에서 생성된 i 및 j 값은 masking을 통해 모듈로 연산(mod 256)을 수행하고, 데이터 의존성 검사를 위해 교환 회로(swap circuit) 내에 의존성 검사기를 통과하여 생성된 인덱스들을 기반으로 S-배열 내의 요소들을 교환한다.

PRGA는 KSA 단계와 거의 동일하며, j를 구하기 위해 K-배열 값이 필요 없다. KSA 단계에서 그림 3의 step 신호는 1이 공급되어 K-배열 값을 사용할 수 있도록 설계되었으며, PRGA 단계에서 step 신호는 0으로 리셋되어 K-배열 값을 요구하지 않는다. PRGA에서 교환 작업 후에 i와 j에 대한 덧셈을 수행하여 평문과 XOR 연산을 거치면 최종 암호문이 생성된다.

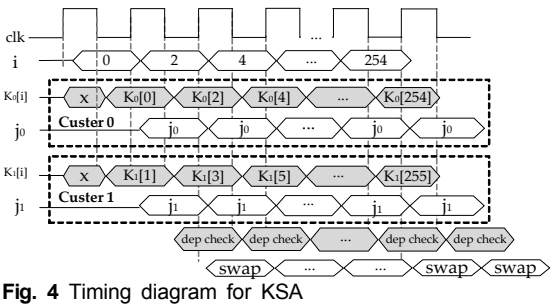

Fig. 4 Timing diagram for KSA

3.4. 제안된 RC4 하드웨어 타이밍도

제안된 RC4하드웨어의 KSA와 PRGA에 대한 타이밍도를 그림 4와 그림 5에 각각 나타내었으며, 중요한 신호 위주로 기술하였다. 본 논문에서는 각 단계에서 소비되는 클록의 수를 감소하기 위해 클록의 상승에지와 하강에지를 모두 사용하였다. 그림 4와 그림 5에서 음영으로 표기된 부분은 클록의 하강에지에서 동작하는 것을 의미한다. 암호문의 생성은 초기 2클록 이후에 매 클록마다 두 개씩 생성된다.

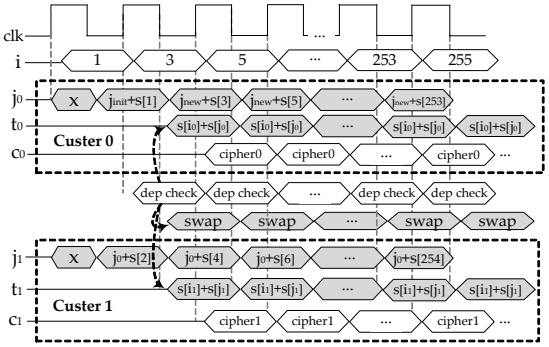

Fig. 5 Timing diagram for PRGA

Ⅳ. 평 가

제안된 RC4 암호화 하드웨어는 성능 향상을 위해 클러스터를 사용하여 KSA와 PRGA 단계에서 두 개 단위로 교환 및 키 스트림을 생성하도록 설계되었다. 본 논문에서 제안된 RC4 암호화 하드웨어는 소비되는 클록수를 감소하기 위하여 클록의 상승/하강에지를 모두 사용하였다.

제안된 구조의 하드웨어 설계는 Altera Quartus Prime 16.1의 합성도구를 이용하여 CycloneV FPGA로 합성되었다. 합성결과는 표 1에 제시되었으며, 메모리는 S-배열과K-배열의 정보를 제공하는데 사용된다. ALM(Adaptive Logic Module)은 8-입력 ALUT (Adaptive Look-up table)와 4개의 전용 레지스터로 구성된 기본생성 블록이며, Fmax는 최대 처리 속도를 나타낸다.

Table. 1 The synthesis report of RC4 hardware

Device	ALUT	Memory	Fmax
CycloneV	2,652	4,096(bits)	102.8 MHz

이전 연구들과의 키 스트림 생성율의 비교를 위해 식 (3)을 이용한다[4][5][6][7]. 식 (3)에서 f는 하드웨어의 최대 처리 주파수를 나타낸다.

$$Throughput\ of\ RC4\ processor(bps) = \frac{8 \times f}{\nu m\ of\ clock\ per\ 8\ bit\ output} \quad (3)$$

표 2는 다른 연구들에서 제시한 RC4 하드웨어들과 정량적인 성능 비교를 기술하였다. 제안된 구조는 Lee[4], Kitsos[5]의 구조와 마찬가지로 S-배열 초기화를 위한 지연시간을 소비하지 않으며, Choi[8]는 초기화를 언급하지 않았다. Kitsos의 경우 초기화를 위해 256×8의 레지스터를 사용하여 하드웨어 복잡도는 증가한다. 랜덤화에 대한 속도는 Lee 및 Gupta[6]의 구조가 다른 구조에 비해 약 3배 정도 적은 클록을 소비하는 것으로 나타나는데, 제안된 구조는 그들의 구조보다 약 2배 더 향상된 구조를 갖는다. 또한 키 스트림 생성율은 Lee의 구조와 비교하여 약 2배, 다른 구조들과 비교하여 약 6배의 향상되었다.

Table. 2 Comparison with other works

other works	S-box initialization	S-box randomization	key stream generation rate	key length
Lee [4]	0	1+256	$1+8\times f$	5-256 bytes
Kitsos [5]	0	768	$\frac{8\times f}{3}$	1-128 bytes
Gupta [6]	1	257	$1+8\times f$	5-256 bytes
Tsoi [7]	256	768	$\frac{8\times f}{3}$	5 bytes
Choi [8]	-	768	$\frac{8\times f}{3}$	5/128 bytes
Ours	0	2+128	$2+16\times f$	5-256 bytes

Ⅴ. 결 론

본 논문에서는 초경량 스트림 암호화 방식의 RC4 하드웨어를 설계하였으며, KSA에서 초기화 지연시간을

감소시키고, 처음 2 클록 이후 매 클록마다 두 개의 클러스터에서 동시에 교환이 수행될 수 있는 고속의 하드웨어를 제안하였다.

제안된 RC4 암호화 하드웨어는 두 개의 클러스터를 이용하여 랜덤화 및 암호문 생성에 사용하였으며, 이때에 발생할 수 있는 데이터 간의 의존성을 해결하는 모델을 함께 제시하였다. 제시된 RC4 하드웨어 구조는 KSA의 초기화에 지연시간을 갖지 않으며, 랜덤화 및 키 스트림 생성율에서 약 2배에서 6배 정도의 성능 향상이 있었다. 또한 RC4에서 제공하는 가변 길이의 키(5~256바이트)를 모두 지원할 수 있다.

최근 RC4 암호화 알고리즘은 TLS 세션 탈취 및 쿠키의 복호화가 이루어지는 등의 공격에 취약점을 가지고 있는 것으로 분석되어, 향후 RC4 알고리즘을 기반으로 추가 단계를 삽입 또는 변경을 통해 취약점을 해결하는 연구가 필요하다.

References

[1] P. Jindal, and B. Singh, “A Survey on RC4 Stream Cipher,” *International Journal of Computer Network and Information Security*, vol. 7, pp. 37-45, Jun. 2015.

[2] P. Jindal, and B. Singh, “RC4 Encryption-A Literature Survey,” *Procedia Computer Science*, pp.697-705, vol.46, 2015.

[3] E. Kartikadarm, T. Listyorini, and R. Rahim, “An Android mobile RC4 simulation for education,” *World Transactions on Engineering and Technology Education*, pp.75-79, vol.16, no.1, 2018.

[4] K. H. Lee, “A Modulo Arithmetic Technique for RC4 Hardware Design,” *Journal of The Institute of Electronics and Information Engineers*, pp.119-124, vol. 55, no. 7, Jul. 2018.

[5] P. Kitos, G. Kostopoulos, N. Sklavos, and O. Koufopavlou, “Hardware Implementation of the RC4 Stream Cipher,” *2003 46th Midwest Symposium on Circuits and Systems*, pp. 1363-1366, vol. 3, 2004.

[6] S. S. Gupta, K. Sinha, S. Maitra, and B. P. Sinha, “One Byte per Clock: A Novel RC4 Hardware,” *Progress in Cryptology – INDOCRYPT 2010*, pp. 347-363, 2010.

[7] K. H. Tsoi, K. H. Lee, and P. H. W. Leong, “A Massively Parallel RC4 Key Search Engine,” *Proceedings. 10th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, pp. 13-21, 2002.

[8] B. Y. Choi, J. H. Lee, and H. S. Cho, “FPGA Implementation and Performance Analysis of High Speed Architecture for RC4 Stream Cipher Algorithm,” *Journal of the Korea Institute of Information Security and Cryptology*, vol. 14, pp. 123-134, Aug. 2004.

이규희(Kyu-Hee Lee)

연세대학교 컴퓨터정보통신공학부 공학사
연세대학교 전산학과 이학석사
연세대학교 전산학과 이학박사
※관심분야 : 암호화 알고리즘, 사물인터넷, FPGA 설계