Journal of the Korea Academia-Industrial cooperation Society
Vol. 19, No. 3 pp. 1-7, 2018

https://doi.org/10.5762/KAIS.2018.19.3.1
ISSN 1975-4701 / eISSN 2288-4688

IoT 환경에 적용 가능한 경량화 블록 암호알고리즘에 관한 연구

이선근
전북대학교 기계시스템공학부

A Study on Lightweight Block Cryptographic Algorithm Applicable to IoT Environment

Seon-Keun Lee
School of Mechanical System Engineering, Chonbuk National University

요 약 IoT 환경은 다양한 디바이스들과 네트워크를 이용하여 무한대의 서비스를 제공한다. 이러한 IoT 환경 발전은 비례적으로 보안의 중요성과 직결된다. 경량 암호는 보안, 높은 처리량, 낮은 전력 소비 및 소형을 제공하는 분야이기 때문에 IoT 환경에 적합하다. 그러나 경량 암호는 새로운 암호 체계를 형성해야 하고, 제한된 리소스 범위 내에서 활용되야 한다는 문제점을 가지고 있다. 그러므로 경량 암호는 다변화/다양화 등을 요구하는 IoT 환경에 최적의 솔루션이라고 단언할 수 없다. 그러므로 이러한 단점들을 없애기 위하여, 본 논문은 기존 블록 암호알고리즘을 경량화 암호알고리즘과 같이 사용할 수 있고, 기존 시스템(센싱부와 서버와 같은)을 거의 그대로 유지하면서 IoT 환경에 적합한 방법을 제안한다. 제안된 BCL 구조는 기존 유무선 센서 네트워크에서 다양한 센서 디바이스들에 대한 암호화를 경량 암호화 같이 수행할 수 있도록 한다. 제안된 BCL 구조는 기존 블록 암호알고리즘에 전/후처리부를 포함한다. BCL 전/후처리부는 흩어져 있는 각종 디바이스들을 데이지 체인 네트워크 환경에서 동작하도록 하였다. 이러한 특징은 분산된 센서시스템의 정보보호에 최적이며 해킹 및 크래킹이 발생하더라도 인접 네트워크 환경에 영향을 미치지 못한다. 그러므로 IoT 환경에서 제안된 BCL 구조는 기존 블록암호알고리즘을 경량화 암호알고리즘과 같이 사용할 수 있기 때문에 다변화되는 IoT 환경에 최적의 솔루션을 제공할 수 있다.

Abstract The IoT environment provides an infinite variety of services using many different devices and networks. The development of the IoT environment is directly proportional to the level of security that can be provided. In some ways, lightweight cryptography is suitable for IoT environments, because it provides security, higher throughput, low power consumption and compactness. However, it has the limitation that it must form a new cryptosystem and be used within a limited resource range. Therefore, it is not the best solution for the IoT environment that requires diversification.

Therefore, in order to overcome these disadvantages, this paper proposes a method suitable for the IoT environment, while using the existing block cipher algorithm, viz. the lightweight cipher algorithm, and keeping the existing system (viz. the sensing part and the server) almost unchanged. The proposed BCL architecture can perform encryption for various sensor devices in existing wire/wireless USNs (using) lightweight encryption. The proposed BCL architecture includes a pre/post-processing part in the existing block cipher algorithm, which allows various scattered devices to operate in a daisy chain network environment. This characteristic is optimal for the information security of distributed sensor systems and does not affect the neighboring network environment, even if hacking and cracking occur. Therefore, the BCL architecture proposed in the IoT environment can provide an optimal solution for the diversified IoT environment, because the existing block cryptographic algorithm, viz. the lightweight cryptographic algorithm, can be used.

Keywords : Bit permutation, Group operation, IoT, Lightweight cryptography, Serpent cryptographic algorithm

*Corresponding Author : Seon-Keun Lee(Chonbuk National Univ.)
Tel: +82-63-270-4778 email: caiserrisk@gmail.com
Received November 6, 2017 Revised (1st November 20, 2017, 2nd December 11, 2017)
Accepted March 9, 2018 Published March 31, 2018

1. 서론

현재 네트워크 환경 및 정보통신 기기의 발달에 따라, 글로벌 IT 및 ICT 기업들은 IoT(Internet of Things) 분야에 대한 다양한 서비스 및 이에 적합한 플랫폼 개발에 집중하고 있다[1-3]. 이러한 서비스 및 플랫폼은 대량 분포 및 배포에 따른 엄격한 비용 제약으로 인해 메모리, 전력관리 및 컴퓨팅 성능에서 매우 제한된 리소스를 사용하게 된다[4]. 그러므로 IoT 플랫폼과 연결된 다양한 디바이스들은 항상 이러한 제약을 극복해야만 한다. 이와 더불어 IoT의 발달은 일반적으로 사용자들의 편리성 및 유용성 등을 제공하기 위하여 기후변화 DB, 음성/영상 DB, 각종 생활정보 등과 같은 빅 데이터 등을 이용하기 위하여 특정 서버의 정보를 필요로 하거나 하나의 정보제공을 수행하게 된다. 이러한 과정에서 다양한 형태의 해킹 및 크래킹 노출에 대한 방어능력 배양은 당연한 당면과제이다.

IoT 분야에 사용되는 많은 응용 프로그램들이 사용자들에게 민감한 상태 모니터링[5] 또는 생체 인식 데이터[6]와 같은 센싱 정보를 처리하므로, IoT 분야에 사용되는 암호화 방식은 효율적으로 구현할 수 있으며 실시간 처리가 가능할 수 있는 암호화 구성 요소에 대한 요구가 강하게 증가되고 있다. 이러한 요구 및 목적에 적합한 암호 방식이 경량 암호이다[7-8].

IoT 환경에 적합하도록 경량 암호를 설계할 경우, 보안, 비용 및 성능 사이에서 항상 절충을 수행해야 한다. 즉 IoT 환경은 보안, 비용 그리고 성능 측면을 항상 고려해야 한다. 그러나 IoT와 같이 한정된 리소스에 의한 제한된 조건에서 이러한 면을 모두 만족할 수 없다.

그러므로 본 논문에서는 보안성 유지, 비용 절감 및 IoT 환경의 적응성을 목적으로 비트 순열 명령 GRP (Group Operation)[9] 개념과 AES[10] 중의 하나인 SPN (Substitution-Permutation Network) 구조를 가지는 Serpent[11-12]의 내부구조를 적절하게 대체시킨 BCL 구조 (Block Cryptographic for Lightweight structure)를 제안하였다.

Serpent 암호알고리즘은 블록 암호알고리즘이지만 제안된 BCL 구조를 적용하면, 분산되어 동작하는 다양한 디바이스들에 대한 보안 등의 기능을 경량화 암호화 같이 수행할 수 있다. 그러므로 제안된 BCL 구조는 리소스가 많이 필요한 다양한 블록 암호알고리즘을 경량화 암호알고리즘 대용으로 사용할 때 경량화 암호알고리즘들과 같은 특징을 가질 수 있기 때문에 IoT 환경 등에 적용가능하다.

2. GRP, Serpent 알고리즘

2.1 GRP(Group Operation) 알고리즘

GRP는 일반적으로 보안성을 강조하기 위하여 암호분야에서 사용되어지는 가장 복잡한 비트 순열 명령어 중의 하나이다. 또한 GRP 속성은 하드웨어의 소형 구현을 달성하는 데 매우 유용하다. GRP는 n비트 순열에 대하여 $\log_2(n)$을 수행한다.

```
j = 0;
  for (i = 0; i < n; i ++)
      if (R2[i]      == 0)
        R3[j ++]   = R1[i]          (1)
  for (i = 0; i < n; i ++)
      if (R2[i]      == 1)
        R3[j ++] = R1[i]
```

여기에서 $R1$, $R2$는 소스 레지스터이고, $R3$은 최종 레지스터이다. GRP 명령의 기본 개념은 식 (1), 그림 1과 같이 소스 $R1$의 비트를 $R2$의 비트에 따라 두 개의 그룹으로 분리한다. 이때 $R1$의 각 비트에 대해 $R2$의 해당 비트를 확인한다. 그리고 $R2$의 비트가 0이면 현재 비트를 $R1$에 첫 번째 그룹에 넣는다. 그리고 $R2$ 비트가 1이면 $R1$의 값을 두 번째 그룹에 넣는다. 이러한 연산에서 동일한 그룹 내의 비트들의 상대적 위치는 변경되지 않는다[9].

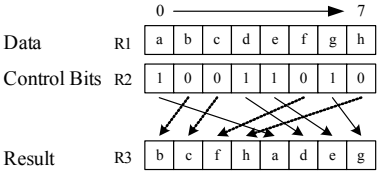

Fig. 1. GRP instruction on 8-bit systems

이러한 일련의 과정과 같이, $R2$는 비트 위치를 결정짓는데 사용되며 블록 암호시스템에서 키 생성 레지스터로 사용된다.

2.2 Serpent 암호알고리즘

블록 암호알고리즘은 비도를 위한 충분한 라운딩 횟수를 가지며 128 비트 처리를 수행할 수 있고 외부로부터 해킹 및 크래킹을 방지하기 위한 재구성(reconfigurable)이 가능 할 수 있도록 구현되어야 한다. 이와 같은 조건을 만족시키는 암호알고리즘은 AES 중에서 Serpent가 가장 적합하다. 그러나 원래 Serpent 암호알고리즘은 알고리즘 자체에 다음과 같은 문제점들을 가지고 있기 때문에 제한된 리소스를 가지는 IoT 환경에 바로 적용하기에 부담스러운 점이 있다[13].

첫째, LT 처리부분이 매우 복잡한 wired-logic으로 구현될 수 밖에 없다. 둘째, 충분한 비도 유지를 위하여 Serpent 암호알고리즘은 라운딩 횟수를 32번 수행한다. 마지막으로 키 스케쥴러에서 서브키를 발생시키기 위한 부분이 32 비트씩 132개로 구성된 4,224 비트에 대한 처리를 수행해야 하기 때문에 LT 부분과 비슷한 단점을 가진다. 마지막으로, 제한된 리소스를 이용하여 정보를 전송하는 센서 입장에서 많은 리소스를 필요로 하기 때문에, 결론적으로 블록암호알고리즘은 IoT 환경에 적합하지 않다.

그러나 Serpent 암호알고리즘은 SPN 구조를 가지는 S 박스를 사용하는 특징을 가지기 때문에 다양한 차동공격에 대한 저항력을 강화할 수 있다는 장점이 있다[12].

Serpent 암호알고리즘은 초기 치환(IP) 및 마지막 치환(FP), 32 라운드 연산, LT 및 S 변환, 키 스케쥴러(KS)로 분류된다.

3. BCL구조를 갖는 Serpent 암호알고리즘

Serpent 암호알고리즘의 경우, 리소스를 많이 차지하기 때문에 한정된 리소스를 사용하여 정보를 보호해야 하는 IoT 환경에 부적합하다. 그러나 SPN 구조를 가지기 때문에 Serpent 암호알고리즘은 보안측면에서 우수한 특성을 자랑한다[13].

그러므로 본 논문에서는 보안측면에서 우수한 특성을 가지는 Serpent 암호알고리즘과 데이지 체인 네트워크를 형성하는 분산된 디바이스들에 대하여 GRP 기법의 특성을 결합하여 IoT 환경에 적합한 경량화 암호알고리즘의 특징을 가지며, 네트워크 환경에 적합할 수 있는 BCL 구조를 제안하였다.

3.1 BCL 알고리즘

제안된 BCL 구조(BCL: Block Cryptographic for Lightweight structure)는 GRP 기법에서와 같이 제어신호를 이용하여 특정 데이터를 특정 위치에 위치시켜 분산된 디바이스(DD: Distributed Device)의 정보를 이용한 암호화가 가능하도록 한 구조이다.

그림 2는 BCL 구조를 적용하기 위한 데이지 체인 네트워크에 존재하는 다양한 DD들의 구성이다. 여기에서 n 비트인 DDA(distributed data)는 DD들에게 분산되어질 데이터들이며, 이에 대한 제어는 DCK(distributed clock)를 이용한다. DCK는 DD들과 서버사이에서 동기식으로 동작하며, 데이터들(DDA)을 DD($D_0,\cdots,D_{k-1}$)들의 레지스터에 순차적으로 전송(write)하거나 수신(Read)하는 제어를 수행한다.

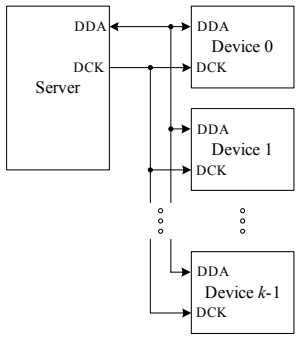

Fig. 2. Distributed Devices in the Daisy chain

즉, 서버는 DCK를 이용하여 데이터들을 순차적으로 전송한다. 이때 각 디바이스들은 순차적으로 데이터들을 받아들여지거나(write) 서버로 보내주는(read) 역할을 수행한다.

$$d_i^{n-1},\ \begin{cases} 0 < k-1 \text{ is DD No.} \leq \infty \\ 0 < i(\text{Reg. width}) \leq 7 \end{cases} \tag{1}$$

여기에서 k는 분산된 디바이스 수, i는 분산된 디바이스의 레지스터 크기이다. DDA는 식 (1)과 같은 분포를 가지며 이를 이용하여 데이터들에 대한 분산처리를 수행한다.

그림 3은 식 (1)을 이용하여 BCL 구조가 동작되는 타이밍 차트이다. 8비트씩 소그룹을 만들고 이를 각 디바

이스에 순차적으로 전송하게 된다. 이때 DDA의 마지막 비트인 *ACK*(r_w)의 값에 따라 해당 디바이스 레지스터에 데이터를 읽을지 쓸지를 결정하게 된다.

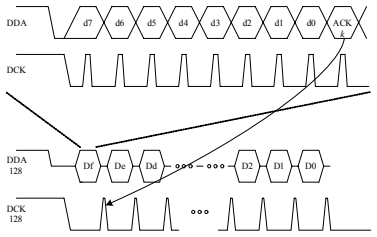

Fig. 3. BCL DCK and DDA operation

DDA의 전체 크기가 128비트인 경우, 16개의 소그룹이 형성된다. 소그룹은 식 (2)와 같다.

$$D_s = d_f, d_e, d_d, \cdots d_1, d_0 \tag{2}$$

식 (2)의 D_s는 전체 DDA를 의미하며 여기에서는 DDA_n이 된다. 그리고 식 (2)의 구성요소들($d_f, \cdots, d_0$)은 분산되어 있는 각각의 디바이스들을 의미한다. m번째 디바이스에서, 특정 디바이스 데이터 d^m은 디바이스 m에 할당된 데이터 d_i^m을 디바이스 레지스터 R_{mt}를 이용하여 저장하고, 이 데이터를 m번째 디바이스의 IV(initial vector)와 논리합 연산을 수행한다. 그리고 그 결과 값을 식 (3)과 같이 디바이스 m의 레지스터 $R_{m(t+1)}$에 저장하고, 다음 명령(ACK)을 기다린다.

$$\begin{aligned} d^m: \; d_i^m &\leftarrow IP_i^m \oplus IV_m \\ R_{m(t+1)} &\leftarrow R_{mt}(d^m) \end{aligned} \tag{3}$$

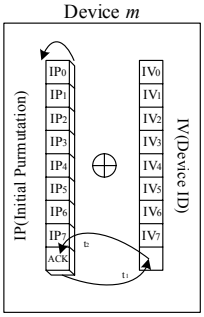

Fig. 4. Device m operation

식 (3)과 그림 4는 DDA의 일부 데이터를 디바이스 m에서 처리하는 것을 표현하고 있다. 디바이스 m번째로 입력되어지는 데이터 d^m 8비트는 디바이스 m 내부 데이터인 IV와 논리합을 수행한 후, 일정 시간(t_1에서 t_2) 후, ACK 값을 토글링 한 후 서버로부터 데이터 요청신호가 올 때까지 대기상태에 있게 된다.

서버로부터 데이터 요청신호가 들어오면 특정 디바이스 m을 포함한 모든 디바이스 k개는 각각의 디바이스에서 연산처리되어 저장되어 있던 비트들을 read로 변환하고 t_2 이후 일괄적으로 서버로 전송된다. 이러한 일괄 전송 데이터는 $DDA_n = DDA_{128}$을 의미하며 이는 식 (4)와 같다.

$$DDA_{128} = d^0 \,\&\, d^1 \,\&\, \cdots d^m \cdots \,\&\, d^{k-1} \tag{4}$$

식 (4)는 서버로부터 DD로 전송되어진 데이터가 각각의 DD에서 연산처리 후 제어신호에 의하여 서버로 입력되어진 데이터를 의미하며 이는 식 (2) 이후의 업데이트 된 데이터 수열들이다. $(n-1)/i = k$개의 소그룹들은 i 배수에 대한 특정값인 k 만큼의 디바이스들을 의미한다. 그러므로 데이지 체인 네트워크를 구성할 때 사용되는 DD는 $0 \leqq k \leqq \infty$의 값을 가질 수 있고 각 디바이스에서 별도의 제어 메카니즘이 없이 서버측 DCK에 의한 제어에 의하여 모든 메카니즘이 제어된다. 그러므로 본 논문에서 제안한 BCL 구조는 다양한 센서 데이터들에 대한 분산된 디바이스들을 암호화하거나 인증기능을 가질 수 있게 도와주는 구조이므로, 리소스를 많이 소비하는 블록 암호알고리즘도 IoT 환경에 매우 적합할 것이라 생각된다.

3.2 BCL-Serpent 암호알고리즘

그림 5는 IoT 데이지 체인 네트워크이다. 여기에서 Ⓢ는 센서들을, Ⓐ는 엑츄에이터를 의미한다. 즉, 그림 5는 서버를 기준으로 센서와 엑츄에이터를 모두 포함하는 데이지 체인 네트워크이다.

제안된 BCL 구조를 IoT 환경에 적용하기 위하여 Serpent 암호알고리즘에 적용시키기 위하여 그림 6과 같이 BCL-Serpent 암호알고리즘을 제안하였다.

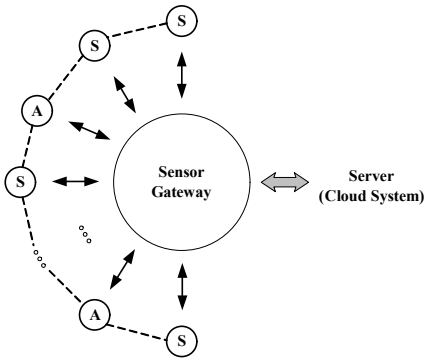

Fig. 5. BCL-Serpent IoT network

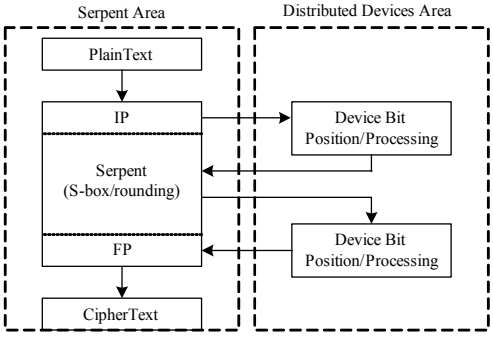

Fig. 6. BCL-Serpent IoT network

그림 6에서 Serpent Area는 기존 Serpent이며, Serpent 영역의 일부와 DD 영역의 일부가 BCL 구조를 가진다. 서버는 기존 Serpent 암호알고리즘을 수행하고, Serpent IP/FP와 DD 영역은 제안된 BCL 구조 기능을 수행한다.

식 (5)는 식 (3)에서 수행한 연산에 대한 역연산이다. 식 (5) 및 그림 7은 Serpent IP/FP 만을 위한 BCL 구조이다. 나머지 Serpent 구조는 그대로 두어 기존 Serpent 알고리즘과 BCL 구조만으로 경량화 특성을 가질 수 있다는 것을 보여준다.

$$\begin{aligned} d^{-m}: \; d_i^{-m} &\leftarrow FP_i^m \oplus IV_m \\ R_{m(t+1)}^{-1} &\leftarrow R_{mt}(d^{-m}) \end{aligned} \tag{5}$$

그림 7은 그림 6에 대한 BCL-Serpent 전체 블록도이다. 그림에서 보는 바와 같이, 기존 Serpent 암호알고리즘은 그대로 유지된 상태에서 서버에서 동작한다. 그리고 Serpent의 일부분인 IP/FP 영역은 분산 배치된 디바이스들을 위한 비트 치환 처리를 수행하기 위하여 BCL 구조를 갖는다.

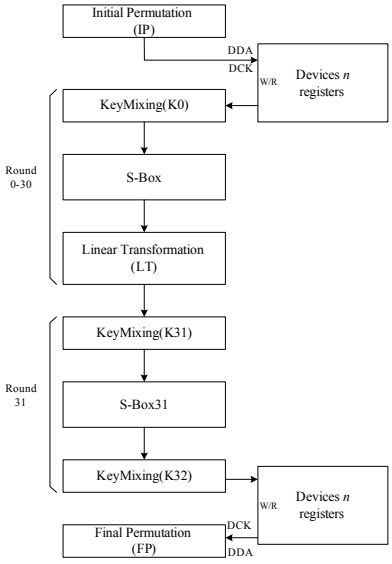

Fig. 7. BCL-Serpent architecture

4. BCL-Serpent 암호시스템 모의실험

BCL-Serpent 암호시스템을 구축하기 위하여 VHDL을 사용하여 코딩을 수행하였고, QuartusII 12.1/ModelSim 10.1b를 사용하여 회로합성 및 모의실험을 수행하였다. $i=7$, $k=16$, $n=128$을 사용하였으며, 모의실험한 결과는 그림 8과 같다. 그림 8에서 보는 바와 같이, 입력 데이터를 로드한 후 이를 8비트씩 16 블록으로 분리하고, 분리된 데이터를 시리얼 통신으로 각각의 디바이스들에 전송되어진다. 분산된 디바이스들로 전송된 데이터들은 r_w 신호에 의하여 각각의 디바이스에 저장되거나 업로드된다. r_w 신호가 '1'의 값을 가지면 서버에서 각 디바이스로 데이터가 전송됨과 동시에 각 디바이스 레지스터에 저장된다.

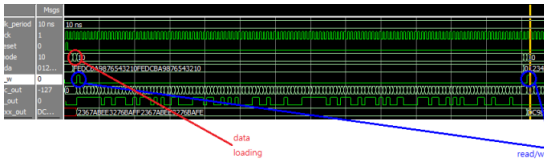

Fig. 8. BCL-Serpent architecture simulation wave

그리고 각 디바이스들은 ID를 IV로 사용하여 입력된 특정 데이터들과 배타적 논리합 연산을 수행한 후, 서버

의 명령을 기다린다. 서버에서 데이터 복귀 명령이 수행(read)되면 데이터들은 복귀하여 Serpent 암호기능을 수행한 후 일반적인 Serpent 암호시스템 기능을 수행하게 된다.

제안된 BCL 구조를 블록 암호알고리즘인 Serpent 알고리즘에 적용한 BCL-Serpent 시스템을 FPGA를 이용하여 구현해 본 결과, 표 1과 같은 특성을 가지는 것을 확인하였다.

Table 1. existing and proposed system

@50MHz	existing Serpent[14]	proposed BCL-Serpent
throughput	350Mbps/4.9Gbps	452Mbps
gatecounting [CLB Slices]	7,100	7,500
round counting	32/4/1	32/4/1
device No.	1	16
architecture	IL/ILP/ FLU	FLU

표 1에서 보는 바와 같이, 기존 Serpent 암호알고리즘에 비하여 제안된 BCL 구조가 전반적인 효율면에서 떨어지는 점이 있다. 그러나 BCL 구조는 분산된 디바이스들을 모두 포함하여 동작한다는 점에서, IoT 환경에 기존 블록 암호알고리즘을 적용할 수 있다는 점을 보여주고 있다. 향후, 제안된 BCL 구조를 더 개선하거나 기존 대칭형 암호알고리즘 및 경량화 암호알고리즘에 적용할 경우에 성능면에서 우수한 특징을 보여줄 것으로 기대된다.

5. 결론

생활 속의 편리함은 더욱 더 IoT 환경을 발전시킬 것이다. 이러한 흐름에 각각의 디바이스(센서/엑츄에이터 등)들에 대한 보안은 더욱 중요한 위치를 차지하게 될 것이다. 이러한 흐름에 맞게 경량화 암호알고리즘이 발전하고 있다. 그러나 센서 시스템 레벨이 아닌 센서들과 많은 수의 링크를 가진 IoT 네트워크인 경우, 경량화 암호알고리즘의 한계가 나타나기 마련이다.

그러므로 본 논문에서는 기존 블록 암호알고리즘을 IoT에 적용하기 위하여 BCL 구조를 제안하였고 Serpent 암호알고리즘에 적용한 BCL-Serpent 암호시스템을 설계한 후 이를 검증하였다.

모의실험 결과, BCL-Serpent 암호시스템 전체 성능은 크기, 소비전력 등은 기존 Serpent와 거의 비슷하지만, 경량화 암호시스템들에 비하면 매우 저조하다. 그러나 서버를 기준으로 디바이스들의 이상유무(부수적 특징인 인증기능)를 바로 발견할 수 있으며, 디바이스의 수와 상관없이 암호화를 수행할 수 있다는 점은 다른 시스템들이 가지지 못한 특징들이다.

그러므로 제안된 BCL 구조는 기존 블록 암호알고리즘을 IoT 분야까지 확대시킬 수 있다는 특징을 가지기 때문에 블록 암호알고리즘 뿐만 아니라 스트림 암호알고리즘 그리고 경량화 암호알고리즘에 적용할 경우, IoT를 포함하는 다양한 분야에 적용 가능할 것이라 생각된다.

References

[1] CISCO, "IoT", https://www.cisco.com/c/en/us/solutions/internet-of-things/overview.html#~stickynav=1(accessed Jan., 10, 2017)

[2] Microsoft, "iot", https://www.microsoft.com/en-us/internet-of-things/(accessed Feb., 13, 2017)

[3] IBM, "iot", https://www.ibm.com/internet-of-things/(accessed Feb., 13, 2017)

[4] Ye, J., Dobson, S., McKeever, S., "Situation identification techniques in pervasive computing", *Pervasive and Mobile Computing*, vol. 8, no. 1, pp. 36-66, Feb. 2012.
DOI: https://doi.org/10.1016/j.pmcj.2011.01.004

[5] S.D.T. Kelly, N.K. Suryadevara, S.C. Mukhopadhyay, "Towards the Implementation of IoT for Environmental Condition Monitoring in Homes", *IEEE Sensors Journal*, vol. 13, no. 10, 2013.
DOI: https://doi.org/10.1109/JSEN.2013.2263379

[6] Tapalina Bhattasali, Khalid Saeed, Nabendu Chaki, Rituparna Chaki, "A Survey of Security and Privacy Issues for Biometrics Based Remote Authentication in Cloud", *IFIP International Conference on Computer Information Systems and Industrial Management CISIM* 2014, pp. 112-121, 2014.
DOI: https://doi.org/10.1007/978-3-662-45237-0_12

[7] Lightweight Cryptography definition, https://www.cryptolux.org/index.php/Lightweight_Cryptography(accessed Aug., 03, 2017)

[8] NIST, Lightweight Cryptography, https://www.nist.gov/programs-projects/lightweight-cryptography(accessed Aug., 17, 2017)

[9] Z. Shi and R. B. Lee, "Bit permutation instructions for accelerating software cryptography," *In Proceedings of the IEEE International Conference on Application Specific Systems, Architectures and Processors* (ASAP 2000), pp. 138-148, July 2000.
DOI: https://doi.org/10.1109/ASAP.2000.862385

[10] J. Guo, I. Nikolic, T. Peyrin, and L. Wang, “ryptanalysis of Zorro,” In IACR Cryptology ePrint Archive, 2013, 713.

[11] Eli Biham. "Serpent: A New Block Cipher Proposal", *Lecture Notes in Computer Science*, 1998. DOI: https://doi.org/10.1007/3-540-69710-1_15

[12] Mansoor Ebrahim, Shujaat Khan, Umer Bin Khalid, “Symmetric Algorithm Survey: A Comparative Analysis”, International Journal of Computer Applications (0975 – 8887), vol. 61, no. 20, Jan. 2013.

[13] Serpent, http://www.cl.cam.ac.uk/~rja14/serpent.html

[14] AJ Elbirt, C Paar, "An FPGA Implementation and Performance Evaluation of the Serpent Block Cipher", *Field-Programmable Gate Arrays, International ACM Symposium on (2000)*, pp. 33-40, Feb. 11, 2000. DOI: https://doi.org/10.1145/329166.329176

이 선 근(Seon-Keun Lee) **[종신회원]**

- 1997년 8월 : 원광대학교 전자공학과(공학석사)
- 2003년 2월 : 원광대학교 전자공학과(공학박사)
- 2006년 4월 ~ 2008년 2월 : 원광대학교 전자공학과 교수
- 2017년 3월 ~ 현재 : 전북대학교 기계시스템공학부 강의전담교수

<관심분야>
IoT, 임베디드시스템, H/W 암호시스템, 프로세서설계