한국정보전자통신기술학회논문지(jkiiect)'17-02, Vol.10 No.1
논문 17-10-01-46 http://dx.doi.org/10.17661/jkiiect.2017.10.1.46

IoT/QR/전자태그용 저용량 메시지 데이터 암호화 적용을 위한 새로운 방식의 스트림 경량 암호화 알고리즘 모티브 제안

김정훈*

A new type of lightweight stream encryption algorithm motif for applying low capacity messaging data encryption for IoT / QR / electronic tags

Jung-Hoon Kim*

요 약 최근 IoT 기술의 확산이 본격화 되면서, 홈/가전/의료등 전 산업 분야에 적용되고 있는데, IoT의 저 사양, 저 전력 소모 특성과 통신 데이터 특성으로 인하여, 기존 암호화 알고리즘의 적용이 용이하지 않으며, 따라서 보안 위협에 대한 우려가 커지고 있다. 이에 대응하여 본 연구에서는, 기존의 고정비트에 대한 XOR연산을 이용한 스트림 암호화 방식에 비하여, 해당 기준 비트에서 상위 방향의 비트 패턴에 따라 불규칙적으로 결정되는 특정한 바이너리 클러스터를 기준으로 암호 키 값에 따라 자리내림, 자리올림 방식을 이용하여, 비트 값에 변화를 주는 일종의 가변 길이 비트 XOR연산 방식을 도입하여 암호화 및 복호화가 진행될 수 있음을 처음으로 제시하였다. 제안 알고리즘의 특징상 암호화 전후의 데이터 크기 변화가 없고, IoT 디바이스/QR코드/RFID/NFC가 빈번히 처리하는 짧은 메시지 데이터에 대해서도 암호화하는 실용성을 확인하였다.

Abstract Recently, the spread of IoT technology has been spreading, and it has been applied to all industrial fields such as home / home appliance / medical care. Due to the low specification, low power consumption characteristic and communication data characteristic of IoT, implementation of existing algorithm is difficult thing. From this reason, we have proposed for the first time that encryption and decryption can be proceeded by introducing a kind of variable length bit XOR operation method which changes a variable the bit length value by using carry up and carry down method. We confirmed the practicality of encrypting short message data frequently processed by IoT device / QR code / RFID / NFC without changing the size of data before and after encryption

Key Words : IoT, Light weight encryption, Message encryption, QR-code, RFID, Stream encryption algorithm

1. 서론

1.1 배경

1.1.1 경량암호화 알고리즘의 필요성

가트너(Gartner, Inc)에 따르면, 최근 사물 인터넷의 활성화에 따라 2020년까지 약 260 억 개의 사물이 인터넷과 연결될 것으로 추정되고 있다. 이는 우리의 실 생활과 밀접한 사물이 인터넷에 직접 연결되고, 이에 따라 기존 사이버 공간의 보안 위험이 일상현실로 확대 될 수 있음을 의미한다[1]. 그런데 IoT device들은 특성상 CPU나 메모리 자원이 제한적이며, 전력 소모가 작아야하는 특

This research was supported by 'Software Convergence Cluster Program', through the Minstry of Science, ICT and Future Planning/Gyeonggi Province(C-16-01).
*Corresponding Author : BinaryLab Co. Ltd. (jhkim@binarylab.co.kr)
Received January 13, 2017 Revised February 02, 2017 Accepted February 20, 2017

성이 있기 때문에[2], 기존의 AES, DES등 블록 암호화 알고리즘의 적용이 어렵다. 이에 따라 컴퓨팅 능력이 아주 적은 응용 분야에서는 주로 스트림(Stream) 암호가 사용되며, 스트림 암호중에 A5/1암호는 GSM 휴대폰 표준의 일부로 음성 암호화에 사용된다. 한편 RC4와 같은 스트림 암호는 인터넷 트래픽 암호에도 사용된다[3]. 한편, 국내에서는 IoT device적용을 위해 LEA등 경량블록 암호화를 개발하여 보급 중에 있으나[4], 경량 블록 암호화 알고리즘조차도 적용되기 어려운 초경량 IoT device 들은 여전히 존재하며 이들 또한 해킹이나 정보 탈취 시에 심각한 문제를 일으킬 수 있는 가능성이 높다[5]. 이러한 초경량 IoT device의 예로는 인체에 적용되는 심박 센서나 도로 교통 센서등 활용 특성에 따라 매우 제한적인 device 자원을 요구하는 경우 등이다.

이에 본 논문에서는 간단하고 단순한 연산만을 활용한 암호화 기술을 연구하였고, 바이너리 클러스터라는 가변적 비트수를 가진 이진수 박스에 대하여 자리올림/내리내림 연산을 통한 일종의 가변적인 XOR 블록을 이용한 스트림 암호화 방식을 제안하였다. 특히 스트림 암호화에서는 스트림 키의 예측 불가성이 매우 중요한 요소로 여기지고 있는데[3], 본 연구 논문은 스트림 키 이외에, 키 스트림을 이용하여 실제 평문을 암호화 과정에 있어서 바이너리 클러스터 기반의 자리올림, 자리내림 방식이라는 가변적인 XOR 방식을 새롭게 제안하여 기존 스트림 암호화에 있어서 보안성을 높일 수 있는 방법을 제시하였고, 특히 평문과 암호문의 크기가 변화하지 않아서 암호 특성상 암호문의 길이가 길 경우 기존 IoT device가 미리 고려한 정보량 특성을 넘쳐서 data overflow가 발생할 우려가 없을 것으로 기대된다.

2. 알고리즘의 기본 원리

2.1 용어의 정의

2.1.1 자리내림연산(Carry down)

자리내림 연산이란 복수의 비트로 이루어진 제1 이진수에 다른 제2이진수를 감산할 때, 제1 이진수의 어떤 자리에서의 덧셈결과 그 자리에서의 숫자가 모자라서 그 윗자리로 자리내림 현상이 발생하는 것을 의미한다.

예를 들어, 아래 그림 1(a)처럼 이진수 1011에서 이진수 11을 빼면 그 결과는 1000이 되는데, 이 경우 뺄셈 후의 1000은 뺄셈 전의 1011과 비교하여 자리내림 현상이 발생하지 않는다. 즉, 뺄셈을 위하여 아랫자리로 숫자를 물려주는 현상인 자리내림 연산 이벤트가 발생하지 않는다.

```
 1011
  -11
-----
 1000
```

그림 1(a). 캐리다운 이벤트가 발생하지 않는 예
Fig. 1(a). Example of carry down event NOT happening

반면, 아래처럼 동일한 이진수 1011에 이진수 111을 이용하여 빼면 그 결과는 0100이 되는데, 이 경우 뺄셈전의 1011은 뺄셈 후의 0100과 비교하여 자리내림 현상, 즉 자리내림 연산 이벤트가 발생한 것이다.

```
 1011
 -111
-----
 0100
```

그림 1(b). 캐리다운 이벤트가 발생하는 예
Fig. 1(b). Example of carry down event happening

이때 뺄셈의 기준이 되는 수인 제2이진수는 1,11,111,1111,... 과 같이 연속된 1로만 이루어진 이진수로서, 작은 수부터 차례로 뺄셈 연산을 적용하여 보면서 자리내림 현상이 최초로 일어날 때의 제2이진수를 캐리 키(Carry Key)라고 본 연구에서는 정의하였다. 위의 경우에서는 “11”이 캐리 키로 선택하여 뺄셈 연산을 수행하게 된다.

2.1.2 자리올림연산(Carry up)

자리올림 연산이란 복수의 자리(비트 수)로 이루어진 제 1 이진수에 다른 제 2 이진수를 가산할 때, 상기 제 1 이진수의 어떤 자리에서의 덧셈 결과 그 자리에서 숫자가 넘쳐서 그 윗자리로의 자리올림 현상이 발생하는 것을 의미한다.

예를 들어, 아래 그림 1(c)처럼 이진수 100에서 이진수 1을 더하면 그 결과는 101이 되는데, 이 경우 덧셈 후의 101은 덧셈 전의 100과 비교하여 자리올림 현상이 발생하지 않는다. 즉, 덧셈을 위하여 윗자리로 숫자를 올려주는 현상인 캐리 업 이벤트가 발생하지 않는다.

```
  100
   +1
 -----
  101
```

그림 1(c). 캐리업 이벤트가 발생하지 않는 예
Fig 1(c). Example of carry up event NOT happening

반면, 아래 그림 1(d)와 같이 이진수 100에 이진수 111을 더하면 그 결과는 1011이 되는데, 이 경우 덧셈 후의 1011은 덧셈 전의 100과 비교하여 자리올림 현상, 즉 자리올림 연산 이벤트가 발생한 것 이다.

```
  100
 +111
 -----
 1011
```

그림 1(d). 캐리업 이벤트가 발생하는 예
Fig 1(d). Example of carry up event happening

이때 덧셈의 기준이 되는 수인 제2이진수는 1,11,111,1111,... 과 같이 1로만 이루어진 이진수로서, 작은 수부터 차례로 덧셈연산을 적용하여 보면서 자리올림 현상이 최초로 일어 날 때의 제2이진수를 캐리 키라고 본 연구에서는 정의하였다. 위의 경우에서는 “111”이 캐리 키가 된다.

2.1.3 캐리 포인터(Carry pointer)

그림 2와 같이 캐리 포인터는 암호화중인 평문에 있어서, 자리올림 또는 자리내림 연산을 수행하기 위한 기준점이 되는 위치이다. 캐리 포인터로부터 최상위 비트 방향 쪽으로 자리올림 또는 자리내림 연산을 일으켜서, 암호화 및 복호화를 수행한다.

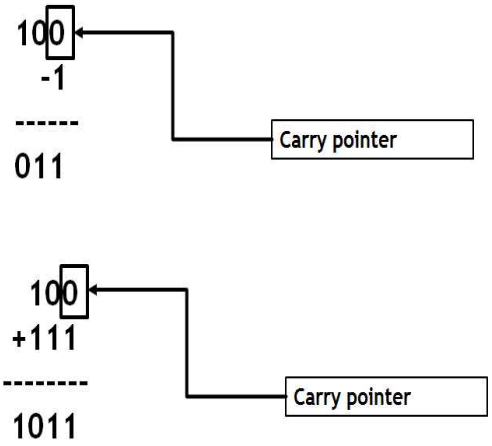


그림 2. 암호화가 진행되는 중의 평문에 대한 캐리포인터
Fig. 2. Carry pointer on plain text while encrypting

2.1.4 캐리 포인터 이동수(Initial Carry Pointer Movement number – Initial CPM)

캐리 포인터 이동 수는 평문에 대하여 최하위 비트에서부터 캐리 포인터가 이동하면서 암호화를 진행하는데, 캐리 포인터가 이동할 수 있는 횟수를 암호화시에 설정하는 값이다. 이 값이 클수록 평문을 여러 번 중복적으로 순환(rounding) 하면서 암호화를 수행하므로 보안성이 높아질 것이다. 이 캐리포인터 이동 수는 복호화시에도 꼭 알고 있어야 하는 일종의 암호 키로서도 작동한다. 이를 통해 다양한 경우의 수를 가진 암호화 결과가 나온다. 복호화 시에도 이 캐리 포인터 이동수를 정확히 알고 있어야 완벽히 복호화가 된다. 본 연구에서는 개발의 간소화를 위해 캐리 포인터 이동수를 평문의 비트 수 길이의 2배, 3배, 4배등으로 하였으나, 후술하겠으나 암호의 안전성 검증을 위해서 다양

한 임의(random)의 수로 수행하였다.

2.1.5 스캔 포인터(Scan Pointer)

스캔 포인터는 캐리 포인터로부터 1비트 상위 비트에서 시작하여 상위 비트 방향으로 이동하면서, 암호 키 값에 따라 자리내림 또는 자리올림 연산이 일어날 수 있는 이진수의 범위인 바이너리 클러스터를 포착하는 포인터이다.

2.1.6 최종 암호키 포인터 값(final password key pointer value – final PSP)

암호화 키는 일정길이를 가진 이진수 비트열 인데, 본 연구에서는 캐리 포인터가 이동할 때 마다 암호화 키 포인터도 특정한 규칙에 따라 이동시키면서 해당 암호 키 포인터가 가리키고 있는 비트 값에 따라 자리내림 연산 또는 캐리 업 연산을 수행하도록 하였다. 본 연구에서는 간명하게, 캐리포인터 이동시마다 1비트씩 상위로 암호 키 포인터를 이동시켰고, 암호 키 포인터가 암호 키의 최 상위 비트에 도달 시 역시 순환하여 다시 암호 키의 최하위 비트를 가리키도록 하였으며, 캐리포인터 이동수가 설정된 값에 도달하면 그때 정확히 어떠한 비트 열을 가리키고 있는지를 알려주는 것이 암호 키 포인터 수이다. 이 값은 복호화에 있어 정확히 복호화 하기 위한 대단히 중요한 값이 된다.

2.1.7. 최종 캐리 포인터 값 (final carry pointer value – final CP)

캐리포인터는 최초 평문의 최하위 비트로부터 최상위 비트방향으로 이동 후, 최 상위 도달 시 다시 최하위로 순환 이동하는데, 설정된 캐리 포인터 이동수 횟수만큼 이동하면 암호화가 종료된다. 이때 최종적으로 암호화된 비트 열중에 몇 번째 비트 열을 가리키고 있었는지를 나타내는 값이 최종 캐리 포인터 값이다. 이 값 역시 최종 암호 키 포인터 값과 같이 복호 과정에서 중요한 역할을 차지한다.

2.1.8. 바이너리 클러스터(Binary cluster)

아래 그림 3과 같이 바이너리 클러스터는 캐리 포인터와 스캔 포인터 사이의 이진수 영역으로서, 자리내림 또는 자리올림 연산이 발생할 수 있는 최소한의 이진수 영역으로서, 자리 내림이 일어나기 위해서는 스캔 포인터로부터 “10”으로 시작하여 캐리 포인터까지의 이진수 영역에 해당하고, 자리 올림이 일어나기 위해서는 스캔 포인터로부터 “01”로 시작하여, 캐리 포인터까지의 이진수 영역에 해당하는 영역이다.

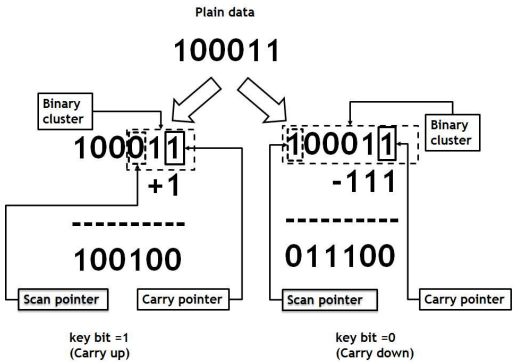


그림 3. 스캔포인터와 캐리포인터 사이에서 바이너리 클러스터를 설정하는 방법

Fig. 3. How to capture range of Binary cluster between scan pointer and carry pointer

2.1.9. 암호키(Password Key)

본 연구에서는 암호 키를 연구의 간소화를 위해 string type으로 지정하였고 이를 내부적으로는 각 string 문자를 구성하는 Ascii 코드의 이진화된 형태를 각 password 열의 비트로 활용하였다. 즉 8글자의 문자는 8*8=64비트의 암호 키 비트 열을 구성하는데, 바람직하게는 Password key에 대하여 별도로 암호화를 커지거나 Hash code로 변환하여 이를 본 연구의 암호 키로 활용하는 것이 보안성을 높일 것으로 기대된다.

2.2 암호화/복호화 기본원리

2.2.1. 자리올림 및 자리내림 연산을 이용한 암호화/ 복호화

상기 자리내림, 자리올림 연산을 이용하여, 그림 4 과 같이 암호 키의 값에 따라 평문을 자리내림 또는 자리올림 연산 중에 하나를 선택하는 형태로 암호화를 수행할 수 있는데, 복호화 시에도 암호화 키 값에 따라 암호문에 대하여 암호화시의 반대방향으로 복호화를 수행하는 원리를 이용하는 것이 본 연구에서 제안하는 암호화의 기본 원리이다. “100011” 이라는 이진수에 대하여, 최하위 비트에 캐리 포인터가 위치한 상태에서, 암호 키가 “1”인 경우, 암호화 시에는 자리올림 연산을 수행하여 “100100”으로 암호화하고, 복호화 시에는 암호 키 “1”을 이용하여 자리내림 연산을 수행하여 암호문에서 “100011” 평문으로 복호화한다.

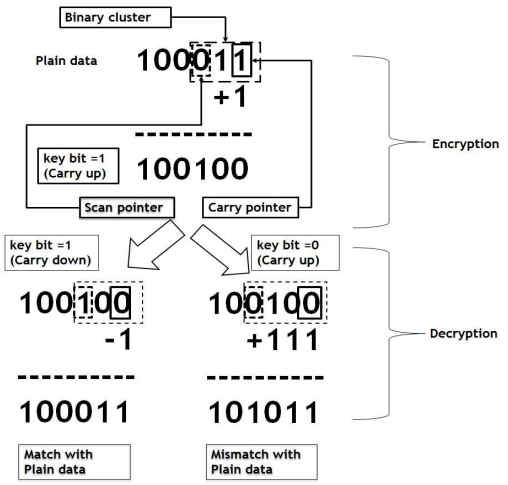


그림 4. 암호 키에 따른 캐리 업 및 캐리 다운 이벤트를 사용한 암호 및 복호 과정

Fig. 4. Encoding and decoding using carry up or carry down based on key value

2.2.2 암호화 알고리즘의 반복적용

자리올림 연산과 자리내림 연산의 기준이 되는 비트를 캐리 포인터라고 본 연구에서는 명명하였는데, 캐리 포인터를 상위 방향으로 1비트씩 이동하면서, 본 연구에서 제안한 자리올림 연산 및 자리내림 연산을 반복적으로 적용할 수 있다. 아래 그림 5와 같이 특정한 캐리 포인터에 있어서, 암호 키 값이 0 일 경우, 자리 내림 연산을 수행하고, 1 비트 상위 방향으로 캐리 포인터를 이동한 뒤, 키 값에 따라 자리 올림 또는 자리 내림 연산을 반복하는 방법을 통해 암호의 보안성을 향상 시킬 수 있다. 아래 그림은 암호 키 값이 0으로서 자리 내림 연산을 수행한 뒤, 캐리 포인터를 1비트 앞으로 이동하여 다시 바이너리 클러스터를 포착하여 암호화하는 모습을 보이고 있다.

2.2.3 자리올림 연산 및 자리내림 연산의 순환적 적용

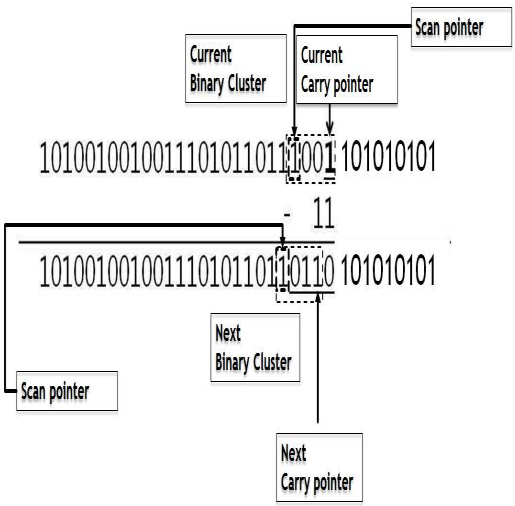


그림 5. 캐리포인터를 상위비트 방향으로 순환 이동을 통한 반복적 암호화

Fig. 5. Iterative encoding method by moving forward carry point to the upper bit of plain text

2.1.1에서 설명한 바와 같이 스캔 포인터를 상위 방향으로 1비트씩 이동하면서, 자리올림 연산 이벤트를 일으켜야 할 때는 캐리 포인터로부터 ″10″을 처음 만나는 범위까지를 바이너리 클러스터라고 하고, 캐리 다운 이벤트를 일으켜야 할 때는 캐리 포인터로부터 “01”을 처음 만날 때까지의 범위를 바이너리 클러스터라고 한다. 한편 자리올림 연산

이벤트를 일으켜야 하는 상황에서 스캔 포인터가 최 상위 비트까지 이동하면서,"10"을 만나지 못할 경우, 스캔 포인터는 다시 최하위로 이동하여 상위 방향으로 순환하여 "10"을 찾고, 찾았다면 그때까지의 이진수를 가상의 바이너리 클러스터로 하여 암호화를 수행한다. 이렇게 스캔 포인터가 계속 암호화 대상 데이터를 순환하여 암호화할 경우, 순환 전의 암호화 문의 패턴에 따라 이후 순환 시에는 암호화 결과가 계속 변하게 되어 암호의 해독을 어렵게 할 수 있다. 이러한 캐리 포인터 이동 수는 암호화 시에 특정한 인수로 초기에 설정할 수 있다. 아래 그림 6에서는 암호 키 값에 따라 자리올림 연산 이벤트를 일으키기 위해 캐리 포인터로부터 스캔 포인터가 1비트씩 최 상위 비트까지 이동하였으나, “01” 을 찾지 못하였고, 이 때 최 상위 비트에 도달한 스캔 포인터가 다시 최하위 비트로 이동하여 상위방향으로 순환 이동하면서, 포착한 “01”으로부터 현재 캐리 포인터 위치까지의 이진수 “0100000”을 바이너리 클러스터로 간주하여 자리 올림 연산을 수행하고 있는 예시를 보였다.

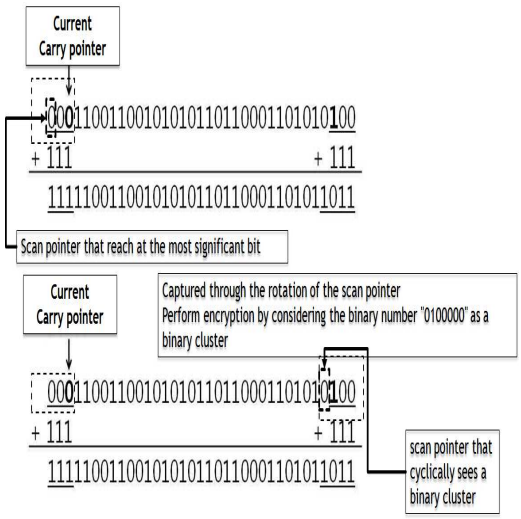


그림 6. 스캔 포인터를 순환적으로 적용함으로써 재 암호화하는 방법

Fig. 6. How to re-encrypt scan pointers in a cyclic manner over ciphertext

3. 제안 알고리즘

3.1. 암호 함수 입력 인수(Parameter)

본 제안 알고리즘은 평문, CPM으로 정의된 캐리포인터 이동 수, 암호 키라는 3가지 input을 투입하여 구현된다. 캐리 포인터 이동수가 클수록 암호화는 중첩적으로 이루어져 보안성이 높아질 수 있으며, 암호 키의 길이는 무제한이므로 암호화 목적에 맞추어 입력한다. 구현 함수의 원형은 다음과 같다.

```
public static byte[] Encode(byte[] plain
_text_byte, string password, int CPM)
```

캐리포인터 이동수를 평문의 비트길이에서 산출되는 함수 값으로 표현하면 캐리포인터 이동수를 input하지 않아도 되나 보안성은 그만큼 떨어질 수 있다.

함수원형은 아래와 같이 C#으로 구현할 경우 암호화 결과 값을 바이트 배열로 return하는 함수로 표현된다.

3.2 암호화과정

3.2.1 캐리포인터 이동

CPM 값 하나를 차감할 때마다 캐리포인터는 상위방향으로 1비트씩 이동하면서 바이너리 클러스터를 확정하는 기준이 된다.

3.2.2 바이너리 클러스터 확정

캐리포인터 위치의 비트로부터 상위비트 방향에서 암호화 킷값에 따라 “01” 또는 “10”을 만날때까지의 범위를 바이너리 클러스터로 확정한다.

3.2.3. 바이너리 클러스터 확정의 예외

예외적으로 최상위 비트까지 이동하여도 바이너리 클러스터를 확정할 수 없을 때는 최하위로부터 순환하여 바이너리 클러스터를 가상으로 확정한다.

3.2.3 자리올림/자리내림 연산

확정된 바이너리 클러스터에 대해 암호키 값에 따라 자리내림 또는 자리올림 연산을 수행한다. 패스워드 포인트도 이때 상위방향으로 1비트 이동하고 최상위까지 이동시 다시 순환한다.

3.2.4 다음 캐리포인터 이동

3.2.1과정으로 돌아가서 CPM 이 0 이 될 때까지 계속 순환한다.

3.3. 복호 함수 입력 인수

복호화를 위해서는 암호문과 최종 캐리포인터 위치(final CP), 최종 암호 키 포인터 값(final PSP) 이라는 3가지 결과 값을 동적으로 생성하여 리턴하는 함수로 구성되어 있다.

복호화 측에서는 final CP 와 final PSP, 패스워드를 모두 알고 있어야만 복호화가 가능한데, 캐리포인터 이동수를 평문의 비트수의 배수로 설정하면 최종 캐리 포인터 위치는 항상 0 이 되어, 좀 더 복호화에서 구현을 용이하게 할 수 있다. 또한 final PSP는 어떤 경우이든 암호 키의 비트열의 크기보다 크지 않으므로, 암호문의 특정위치에 고정 비트 열로 final_PSP를 삽입하면, 복호화 측에서 final PSP를 알고 있지 않아도 암호문에서 확인할 수 있기 때문에 final CP, final PSP는 용도에 따라 반드시 알지 않아도 복호화가 가능하도록 보안 수준을 조절할 수 있다

복호화 함수의 경우도 Encode함수에서 생성된 암호문 바이트 배열과, 패스워드, CPM, final CP, final PSP라는 4가지 인수를 받아서 암호 해독 결과를 바이트 배열로 리턴하는 함수로 구현된다.

```
public static byte[] Decode(byte[] encrypted_result, string password, int CPM, int final_CP, int final_PSP)
```

3.4 복호화 과정

3.4.1 복호화 시작위치 결정

암호문에서 복호화 시작위치 결정을 위해 final CP를 이용하고, 패스워드 키의 시작 값을 결정하기 위해 final PSP값을 이용하여 결정한다.

3.4.2 암호화 과정의 역 과정으로 복호화

암호화 과정의 역 과정으로서 캐리포인터 수 CPM만큼을 암호화의 반대방향으로 캐리포인터를 이동하게 되며, 패스워크 값도 패스워드 포인터의 이동방향의 반대방향으로 이동하면서 암호화 키 값을 취한다.

특히 킷값에 따라 자리내림 및 자리올림도 암호화와 반대로 진행하게 된다.

4. 결과

4.1 암호화 알고리즘의 구현 결과

본 연구에서는 상기와 같은 암호화 알고리즘을 Visual Studio Community 2015를 이용하여 Visual C# 으로 실제구현 하여 암호화 및 복호화를 수행하여 암호알고리즘의 알고리즘으로서의 가능성을 테스트 하였다.

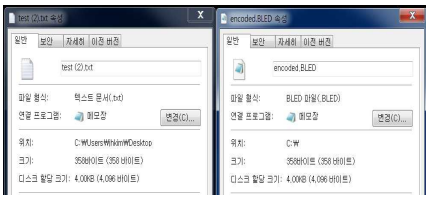
실제 구현사례로서 표1 에는 평문에 대한 암호화 결과를 상세히 보이고 있다. 암호화에 적용되었던 암호 키는 “binarylab2015”였으며, 캐리 포인터 이동 수(CPM)는 4000, 최종 암호 키 포인터 값(final PSP)은 47, 최종 캐리 포인터 값(final CP)는 0을 적용시키면 암호 키의 47번째 비트부터 암호 키를 읽어서 암호화의 역방향으로 이동하면서 키의 값을 적용하고, 암호문에 있어서도 캐리 포인터의 위치변화는 암호화의 반대인 하위비트방향으로 이동하면서 순환하면서 캐리 포인터 이동 수 4000 이 0이 될 때까지 수행한다. 따라서 ①최종 캐리 포인터 값(final CP) ②최종 암호 키 포인터 값(final PSP) ③암호화 키 및 ④ 캐리 포인터 이동 수(initial CPM)가 복호화 단계에서 필요하다. 이중 ①, ② 는 평문의 패턴, 암호키 값, 캐리 포인터 이동 수에 따라 암호화 과정 중에 동적으로 생성되는 값이다.

전체 평문의 비트 수 길이를 캐리포인터 이동수로 나눈 값을 라운드 횟수로 정의 할 수 있으나

본 암호화 알고리즘은 비트단위로 이루어진 것으로서 라운드 횟수는 암호화의 중첩적 적용의 정도 및 암/복호화 속도를 예상할 수 있는 인덱스로 볼 수 있다.

또한 2000비트의 평문에 대해 4000 의 캐리 포인트 이동수를 적용하였으므로, 2 round 로 암호화된 암호문의 복호화를 수행한 결과를 보이고 있다. 완벽하게 암호문이 복호화 됨을 알 수 있다.

표 1. 암호화 알고리즘 전후 데이터 크기 변화
Table 1. Data size change before and after encryption algorithm

Plain text (UTF-8 encoded)
최근 사물인터넷의 활성화에 따라 가트너에 따르면 2020년까지 약 260억 개의 사물이 인터넷과 연결될 것으로 추정되고 있다. 이는 우리의 실생활과 밀접한 사물이 인터넷에 직접 연결되고, 이에 따라 기존 사이버공간의 보안위험이 일상현실로 확대될 수 있음을 의미한다.
Comparison size of plain text and encrypted text


4.2 암호화 및 복호화 처리성능

FPGA(Field Programmable Gate Array)로 LEA(Lightweight Encryption Algorithm) 경량암호화 알고리즘을 구현한 경우 성능 검증을 위해 평문을 암호문으로 바꾸고 이를 다시 평문으로 복원하는 과정을 통해 검증[4]을 하는 방식을 활용하였다. 이와 같은 선행 연구를 참고하여, 본 연구에서는 새롭게 제안된 암호화 알고리즘의 암호화 및 복호화가 완전하게 수행되는지를 확인하는 방법으로서 MKT(몬테카를로 검사)를 활용하였다. 먼저 임의의 16,438 개의 고정길이 평문에 동일 패스워드에 대해 동일한 캐리포인터 이동 수를 적용하여 암/복호화가 완벽히 재연되는지 확인하였고, 두 번째로, 44192개의 다양한 길이와 내용을 가진 원문 텍스트에 대해 다양한 암호 키와 다양한 캐리포인터 이동수를 적용하여 암호화 하였다가 다시 복호화 하는 과정을 통해 그 결과 값이 완전히 일치하는지를 확인하는 방식으로 알고리즘을 테스트 하였다.

4.2.1 고정길이 평문에 대해 동일한 암호키 및 동일한 캐리포인터 이동수를 적용

동일한 방식을 차용하여 먼저, 100바이트의 고정길이 평문에 대해 동일한 암호 키를 이용하여, CPM은 평문길이의 1배수로 하여 16438가지의 경우의 수로부터 나온 16438건의 평문 메시지의 암호화 / 복호화에 대하여 암호화 및 복호화가 정상적으로 되었다.

4.2.2 가변길이 평문에 가변길이 암호키 및 가변길이 캐리포인터 이동수 적용

한 번의 시도에서 임의로 20~220 byte의 가변길이의 평문에 대하여, 암호 키 또한 임의로 8비트~160비트 길이를 적용하고, CPM 또한 가변길이 원문의 1~4배로 하고 이 값에 추가로 0~50회를 적용한 44,192건의 경우에 대하여 암호화 및 복호화 또한 완벽히 진행되었음을 확인했다.

4.2.3 알고리즘의 속도 테스트

본 알고리즘의 암호화/복호화의 처리속도 성능 평가는 Intel CORE i7-6500U CPU(2.5Ghz), 8GB RAM, SSD 시스템, 64bit Windows 7에서 진행하였으며, Visual Studio 2015 C#으로 개발, 컴파일(Compile)한 알고리즘을 이용하여 성능을 테스트하였다.

암호화 알고리즘의 성능은 주로 대규모의 암호화 대상 평문에 대하여, 배치연산을 통한 시간, 리소스 점유율 등을 측정, 비교하는 경우가 주류를 이루고 있었다[6-9], 이와 같은 선행 연구들을 참고하여 본 연구에서도 유사한 방식으로 제안된 알고리즘의 성능을 테스트 하였고, 표2에 그 결과를 보였으며, 그림7은 표2를 도식화하여 표현한 것이

다. 결과 데이터는 마이크로소프트사(Microsoft)의 개발 툴인 비주얼스튜디오(Visual Studio) 의 라이브러리인 “System.Security.Cryptography” 내의 AES 알고리즘과 비교했을 때, 본 알고리즘은 매우 짧은 데이터에서부터 비교적 큰 데이터까지 고른 암호화 속도를 보이는 것을 알 수 있다.

표 2. 제안된 알고리즘의 성능
Table 2. Performance of proposed algorithm

No.	plain text size (byte)	key size (bit)	Carry Pointer Movement number	total trial count	Avr. encoding time	Avr. decoding time
1	100	128	1000	180	1.22	1.56
2	500	128	5000	10542	3.45	3.61
3	100000	128	1000000	52	755.94	759.98

한편, 표3는 동일한 평문데이터 셋에 대하여 AES-128 알고리즘으로 암·복호화한 시간을 측정한 결과 값이며, 복호화 속도는 본 제안 알고리즘보다 2배정도 빨랐으나, 암호화 속도는 전반적으로 본 제안 알고리즘보다 느리거나 비슷하다고 볼 수 있으나, 처리해야할 평문 데이터가 커질수록 3배~30배까지 급격하게 느려짐을 알 수 있었다. 이는 비주얼스튜디오의 AES 알고리즘 라이브러리의 문제일 가능성이 크나, 닷넷 라이브러리를 사용할 경우에 본 제안 알고리즘을 쓴다면 빠른 암호화시에 보다 더 나은 처리속도를 나타낼 수 있을 것으로 예상된다.

표 3. AES알고리즘의 성능
Table 3. Performance of AES-128

No	plain text size(byte)	password key size(bit)	total trial count	average encoding time per trial	average decoding time per trial
1	100	128	52892	1.91	0.51
2	500	128	10542	11.31	1.71
3	100000	128	52	21043	348.75

아래 그림 7,그림 8은 표2의 제안 알고리즘의 속도 평가 결과와 표3의 AES 알고리즘의 속도평가 결과를 암호화 및 복호화 각각의 기준에서 비교하여 본 그림이다. 주의할 것은 비주얼 스튜디오의 “System.Security.Cryptography” 에 탑재된 AES 알고리즘과의 비교이므로, 같은 알고리즘이라도 구현방식에 따라 성능 차이가 있을 수 있어서 절대적 비교 기준이 되기가 어려운 점이 고려되어야 한다.

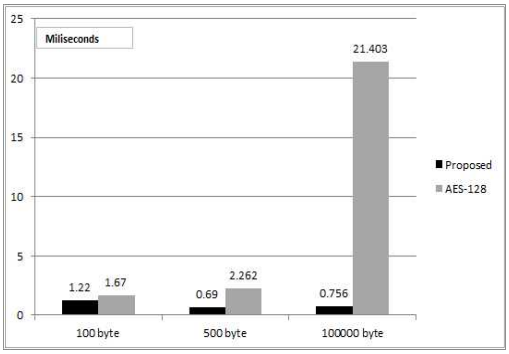


그림 7. AES-128 알고리즘과 비교한 평균 암호화 시간
Fig. 7. Average encoding time comparing with Proposed Algorithm with AES-128

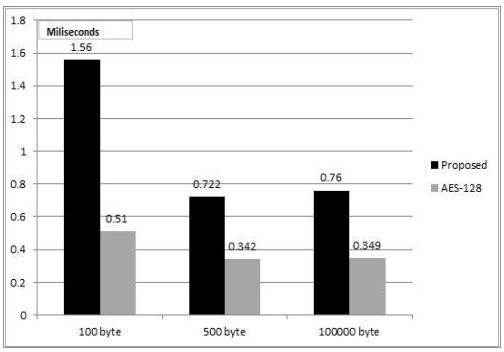


그림 8. AES-128알고리즘과 비교한 평균 복호화 시간
Fig. 8. Average decoding time comparing with Proposed Algorithm with AES-128

4.3 메시지 데이터 암호화 실제 응용

QR코드와 RFID는 많은 IoT device에서 읽히는 대표적인 소규모 정보 저장 매개이다. 이러한 데이터는 그 특징상 길이가 짧고, 암호화할 경우 더 크기가 커지며, 따라서 암호문의 크기가 정보저장 한계를 넘어서는 경우가 있기 때문에, 기존 암호화 알고리즘을 적용하기가 용이하지 않으며, 저사양성으로 인하여 암호화 복호화 자체가 쉽지 않다. 본

연구의 제안 알고리즘을 실제로 저자가 개발한 QR코드 압축 알고리즘인 QR2x 알고리즘을 생성한 QR코드를 스캐닝하고 해독하는 모바일 앱에 적용하여 활용하였다.

4.3.1 QR2x 모바일 앱에 암호화 기능 탑재

아래 그림9는 www.qr2x.co.kr 에 실제로 구현된 QR2x 바코드에 본 제안 알고리즘이 탑재한 뒤 암호화 압축 QR코드를 생성하고 있는 웹페이지 화면이다. 본 암호화알고리즘은 이러한 단문 메시지 암호화 및 암호문의 크기 불변으로 QR코드에 표현된 정보 내에서 암호화가 될 수 있어서 제한 없이 암호화가 가능하다.

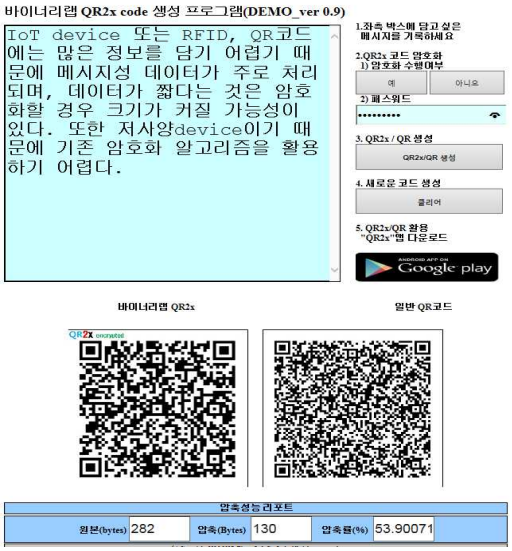


그림 9. 단문 메시지 QR코드 데이터에 대한 제안 암호화 알고리즘 탑재 사이트(www.qr2x.co.kr)
Fig. 9. Short Message QR Code Data Encryption Algorithm Suggested Site

단문 메시지를 넣고 암호 키를 “binarylab” 이라는 72비트의 암호 키로 암호화한 결과의 암호화를 수행하지 않고 생성한 QR 코드 비교를 아래 그림 10에 나타내었다.

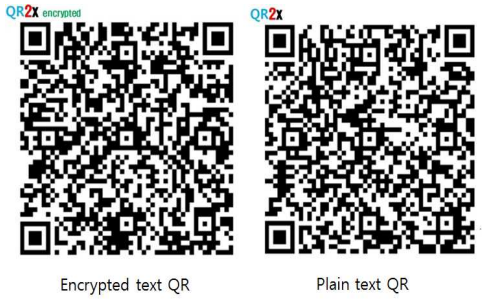


그림 10. 암호화 QR코드와 비 암호화 QR코드간의 데이터 차이로 발생하는 QR코드 이미지 차이 비교
Fig. 10. Comparison of QR code image differences caused by data difference between encrypted QR and non-encrypted QR

5. 결론

5.1 새로운 암호화 모티브 제시

본 연구에서는 기존의 스트림 암호화 알고리즘에서 주로 쓰이는 고정 비트에 대한 XOR연산 방식[3]과 달리 , 해당 기준 비트에서 상위 방향의 비트 패턴에 따라 불규칙적으로 결정되는 특정한 바이너리 클러스터를 기준으로 암호 키 값에 따라 자리내림, 자리올림 방식을 이용하여 비트 값에 변화를 주는 일종의 가변길이에 대한 XOR연산을 통해서도 암호화 및 복호화가 진행될 수 있음을 처음으로 제시하였으며, 특정 비트 위치에서의 비트 패턴 뿐만 아니라, 암호화가 일어나는 범위가 달라지는 패턴과 범위라는 두 가지 불규칙성을 추가로 스트림 암호화 알고리즘에 부여할 수 있게 되었다.

5.2 제안 알고리즘의 응용 가치

이러한 특징으로 인해, IoT device장치의 특성상 암호화된 생성데이터가 서버 등 고 사양 장비로 전송된 뒤, 고 사양 장비에서 복호화되는 경우가 많을 것으로 사료되므로, IoT device의 암호 경량화의 요구목표로는 복호화 성능보다는 암호화 성능이 보다 더 중요한 요소가 될 것으로 판단된다.

따라서 이러한 부분을 통해 본 연구에서 제안하는 암호화 알고리즘의 IoT device적용에 대하여

경량성을 확인할 수 있었으며, 향후 본 제안 알고리즘의 최적화를 고려한다면, 충분히 실용적이 가치가 있을 것으로 판단된다.

또한 본 연구의 암호화 알고리즘은 저사양의 저효율 디바이스에 사용될 수 있는 암호화 알고리즘으로서 암호화와 복호화 결과 평문과 암호문의 길이의 변화가 일어나지 않는 점이 특징적이다. 특히 10~200 byte 정도의 짧은 길이의 메시지를 암호문의 크기 증가 없이 암호화 할 수 있으므로, QR 코드/RFID등 한정된 저장 공간에서 다양한 단문 메시지만을 저장할 수 있는 매체의 정보를 크기 확장의 우려 없이 암호화 할 수 있게 되었다.

5.3 한계점

이러한 장점에도 불구하고, 본 제안 알고리즘의 보안적 안전성이 정량적이고 수학적으로 엄밀히 입증하여야 할 부분이 있으며 이는 추후 지속적인 연구개발을 통해 확인할 예정이다. 특히 암호화 알고리즘의 안전성이 어느 정도인지에 따라 활용될 수 있는 용도와 적용 가능한 산업분야가 명확해질 것으로 판단된다.

REFERENCES

[1] Kyeoung-Ju Ha., Chang-Ho Seo, Dae-Youb Kim, “Design of Validation System for a Crypto-Algorithm Implementation“, The Journal of Korea Information and Communications Society, 39(B), pp. 242-250, Apr. 2014

[2] Jianying Zhou and Dieter Gollmann, “Evidence and Non-repudiation,” Journal of Network and Computer Applications, Vol. 20, No. 3, pp. 267 - 281, Jul. 1997

[3] Dong-Ho Won and Young-Suk Lee, Ji-Yeon Kim, "Understanding Cryptography, A Textbook for Students and Practitioners", Greenpress, pp. 37-41, Feb. 2016

[4] Mi-Ji Sung and Kyung-Wook Shin,”An Efficient Hardware Implementation of Lightweight Block CipherLEA-128/192/256 for IoT Security Applications”, J. Korea Inst. Inf. Commun. Eng. Vol. 19, No. 7,pp 1608-1616, Jul. 2015

[5] Ministry of Science,ICT and Future Planning, "Roadmap of IoT Information Security ", p 68, Oct. 2014

[6] Je-Seong Jeong, Kyung-min Kim, HakJu Kim,Joon-jeong Park, Soo-Hyun, Ahn, Dongsoo Lee,Rakyong Choi, Kwangjo Kim,Daeyoung Kim, Shin,”Configuration of IoT Secure SNAIL Platform Using Lightweight Crypto primitives (I)“, CISC-W'14, v.0, Dec. 2014

[7] When encrypting personal information of large capacity OLTP system System Performance Improvement Plan (Focused on Public A System), INNOVATION STUDIES, Vol. 9, No. 1, pp. 115-118, Jun. 2014

[8] Kyuwoon Kim and Hyunwoo Kim and Huijeong Kim and Taeyoung Huh and Sanghyuk Jung, and Yong Ho SongAn "Analytical Model for Performance Prediction of AES on GPU Architecture", Journal of The Institute of Electronics Engineers of Korea Vol. 50, NO. 4, pp. 852-854, April 2013

[9] Jang-Hyun Kim, Hyo-Joong Suh, "Implementation of Verification and Evaluation Testbed of WiMax2 PKMv2 Encryption Layer", The Journal of the Institute of Internet, Broadcasting and Communication, Vol. 15 No.1, pp.80-81. Feb. 2015

저자약력

김 정 훈(Jung Hoon Kim) **[정회원]**

- 2002년 2월 : 서울대학교 약학과 (약학석사)
- 2010년 2월 : 서울대학교 보건대학원 (보건학석사)
- 2013년 2월 : 서울대학교 보건대학원 (보건학박사 수료)
- 2015년 4월~현재 : 바이너리랩 주식회사(대표이사)

<관심분야>

압축알고리즘, QR바코드, 경량암호화, 약학정보화