한국통신학회논문지(J-KICS) '13-11 Vol.38C No.11
http://dx.doi.org/10.7840/kics.2013.38C.11.972

AES를 이용한 스트림 암호 개발

김 성 기◆, 김 길 호*, 조 경 연°

Development of Stream Cipher using the AES

Sung-Gi Kim◆, Gil-Ho Kim*, Gyeong-Yeon Cho°

요 약

미래전의 양상은 네트워크중심전(NCW: Network Centric Warfare)으로 바뀌고 있으며, 다양한 전력 요소가 유무선 네트워크와 상호 유기적인 연결을 통한 첨단 정보통신기술의 활용이 네트워크중심전의 구현에 핵심 요소라고 할 수 있다. 그리고 네트워크중심전에 사용되는 다양한 정보들은 기밀성과 무결성이 우수하면서 신뢰성 있는 실시간 처리만이 빠른 상황판단을 할 수 있고, 이는 전쟁 승리의 핵심이 된다. 본 논문에서는 네트워크중심전 구현에 핵심 기술 중 하나인 암호 알고리즘으로서 128비트 출력 스트림 암호를 개발했다. 개발한 스트림 암호는 AES를 기반으로 변형된 OFB모드를 적용하여 기밀성과 무결성 그리고 하드웨어로 구현하여 실시간처리와 안전성이 우수하다.

Key Words : NCW, AES, Stream Cipher, Confidentiality, Integrity

ABSTRACT

Future aspects of the has turned into a network centric warfare(NCW). Organically combined wired and wireless networks in a variety of cross-of-the-art combat power factor utilization of information and communication technology is a key element of NCW implementation. At used various information in the NCW must be the confidentiality and integrity excellent then quick situation assessment through reliability the real-time processing, which is the core of winning the war. In this paper, NCW is one of the key technologies of the implementation of 128-bit output stream cipher algorithm is proposed. AES-based stream cipher developed by applying modified OFB mode the confidentiality and integrity as well as hardware implementation to the security and real-time processing is superior.

Ⅰ. 서 론

미래전의 양상은 네트워크중심전(NCW: Network Centric Warfare)[1]으로 바뀌고 있으며, 유무선 정보네트워크를 작전에 최대한 활용, 적보다 먼저 정보를 획득하고 해석하여 실시간 전파할 수 있는 최첨단 지휘통제체계의 구축이 승리의 핵심이 된다. 이 이론은 미 국방부의 전력변환에 있어 중요한 주제이며 우리 군의 경우에도 미래전쟁 환경에 대비한 군사력 발전 방향으로 인식하고 있다.

네트워크중심전 이론의 성공적인 구현을 위해서는 교리, 조직, 훈련, 자원(기술), 리더쉽 및 교육, 인력, 시설의 7대 영역들이 상호 병행 발전을 강조하고 있지만, 다양한 전력 요소의 유무선 네트워크를 통한 상호 유기적인 연결을 전제로 한 첨단 정보통신기술의 활용이라는 측면에서 기술적 환경(유무선 네트워크)

◆ First Author : 부경대학교 IT융합응용공학과 마이크로프로세서 연구실, heewookim@empal.com, 학생회원
° Corresponding Author : 부경대학교 IT융합응용공학과 마이크로프로세서 연구실, gycho@pknu.ac.kr, 정회원
* 부경대학교 IT융합응용공학과 마이크로프로세서 연구실, vnlqpcdd@hanmail.net, 정회원
논문번호 : KICS2013-08-377, 접수일자 : 2012년 8월 30일, 최종논문접수일자 : 2013년 10월 8일

의 구축이 네트워크중신전의 구현에 핵심 요소라고 할 수 있다.

미래 전쟁에서 네트워크중심전 구현을 위한 기술적 환경의 중요한 발전 방향은 각 전력 요소들간의 유무선 네트워킹을 위해서는 언제 어디서나 정보를 유통할 수 있는 기밀성(Confidentiality)과 무결성(Integrity)이 보장된 신뢰성 있는 실시간 처리 통로를 보장하는 것이다. 이는 열악한 전장 상황에서도 유기적인 네트워킹을 통한 전장상황인식의 공유 확대와 전력의 통합화는 정보를 저장, 유통, 활용, 관리하는 방식이 현재와는 다르거나 진보한 방식으로 발전 할 것으로 예상된다[2]. 이는 유무선 네트워크 활용 기술의 급격한 증가와 물리적, 논리적 분리된 기반 네트워크의 연동 및 통합이 확대 될 것이며, 또한 네트워크를 통해 전달되는 정보의 양(영상이나 사진 정보)이 급격히 늘어날 것이다. 이와 같은 신뢰성 있는 유무선 통신 환경의 변화와 발전은 정보보호 측면에서 다음과 같은 문제점과 요구사항을 가지고 있다.

첫째 휴대용 및 초소형 컴퓨팅 장치의 활용이 증가하므로 동적으로 변화하는 네트워크 환경으로 인해 일부 네트워크에 대한 공격 및 침해가 전체 네트워크로 쉽게 확산 될 수 있다. 둘째 전장에서 개별 장치들의 증가는 유통되는 정보에 대한 보안통제 및 관리가 적절히 수행되지 않을 수 있다. 셋째 정보의 실시간 처리가 전장 상황의 정확한 해석에 필수이며 이는 기밀성과 무결성이 보장된 정보에서 얻을 수 있다. 이와 같은 문제점들을 종합하면 기밀성과 무결성을 위한 암호기술은 대용량 및 다양한 유형의 데이터 처리와 동적으로 변하는 네트워크 환경에서 실시간 적용이 가능해야 한다.

특히 실시간 암호처리 기술은 스트림 암호만이 기능하지만 스트림 암호는 기밀성과 무결성 보장이 되지 않는 단점이 있다.

그래서 본 논문에서 네트워크중심전 구현을 위한 기술적 환경 요구사항을 바탕으로 개발한 스트림 암호는 128비트 블록 암호인 AES(Advanced Encryption Standard)[3]를 이용하여 소프트웨어보다 보안이 우수한 하드웨어 구현으로 128비트 처리 스트림 암호를 개발했다. 사용된 AES는 12라운드를 적용하지만 비밀 키는 128비트를 사용했다. 그리고 AES 알고리즘의 사용으로 생기는 장점은 하드웨어 구현에 필요한 많은 기반 기술들이 현재 개발되어 있으며, 또한 현재까지 AES의 안전성에 위협이 되는 문제가 없다는 점이다.

개발한 스트림 암호의 전체 구성은 149비트 산술 쉬프트 레지스터(Arithmetic Shift Register)[4]를 통해 생성된 의사랜덤수열을 AES 알고리즘으로 128비트 스트림을 생성하고 생성된 스트림과 평문 128비트는 단순히 XOR연산을 수행하지 않고, 우수한 기밀성과 무결성을 위해 암호 운영모드인 변형된 OFB(Cipher FeedBack)를 적용하고 최종적으로 AES의 마지막 라운드 키와 XOR연산을 수행하여 128비트 암호문을 만든다. 개발한 스트림 암호는 5단계 파이프라인으로 구현했으며 Worst Case에서 Max Frequency는 32.3MHz(4.13Gbps)의 빠른 성능을 보여주었다. 이는 신뢰성을 보장하면서 실시간 처리가 필요한 네트워크 중심전의 기술적 환경에 필요한 속도를 충분히 만족함을 보여준다.

본 논문의 구성은 2장에서 간략한 AES 알고리즘 중심으로 설명하고, 3장에서 개발한 스트림 암호 알고리즘을 자세히 설명하고, 4장에서 스트림 암호의 하드웨어 구현, 5장에서 하드웨어 구현결과 분석, 6장에서 안전성에 대해 분석하고, 마지막으로 결론으로 끝맺는다.

Ⅱ. 관련연구

AES는 미국 국립 표준 기술 연구소(NIST)[5]에서 안전성에 심각한 문제가 있는 DES[6]를 대체하기 위해 전 세계적으로 블록 암호 알고리즘 공개 모집을 위한 프로젝트로, Joan Daemen과 Vincent Rijmen이 개발한 Rijndael을 2000년 10월에 AES 암호 알고리즘으로 선정하였다. 이후 AES는 전 세계적으로 가장 널리 사용되고 있는 대표 블록 암호 알고리즘이 된다. AES 알고리즘의 입력 평문의 길이는 128비트로 고정이고, 암호화에 사용되는 라운드 키의 길이는 128비트, 196비트, 256비트 중에서 선택해서 사용할 수 있고, 라운드 수는 키 길이에 따라 가변적으로 10, 12, 14라운드가 적용된다.

AES는 암호 및 복호과정에서 생성되는 중간 결과값 스테이트(State)를 바이트 단위로 4 * 4의 2차원 행렬로 간단히 표현할 수 있고, 4 * 4의 2차원 행렬은 기존의 행 우선이 아닌 열 우선으로 행렬의 순서를 표시한다. AES의 라운드 함수 내에 크게 4가지의 독립적인 함수가 있으며, 각각의 라운드 함수는 다음과 같다.

- SubBytes(SB): 8비트 Sbox를 이용한 비선형 바이트 치환 함수.
- ShiftRows(SR): 행 단위 왼쪽 회전 함수로 첫

번째 행은 변환하지 않으며, 두번째 행은 1바이트 회전 세번째 행은 2바이트 회전 네번째 행은 3바이트 회전을 적용하는 함수.

- MixColumns(MC): 열 단위로 혼합을 수행하는 32비트 선형 변환 함수.
- AddRoundKey(ARK): 라운드 키와 덧셈을 수행하는 함수.

AES의 암호화 과정에서 라운드 함수를 적용하기 전에 표백(Whitening) 단계로서 평문과 라운드 키 덧셈을 먼저 적용하고 4가지 라운드 함수를 수행한다. 그리고 마지막 라운드는 MixColumn() 함수를 제외하고 라운드 함수를 수행한다. 복호화 과정은 암호화 과정에서 수행한 4개의 개별 함수들의 역 변환 함수가 존재하고 그 역 변환 함수와 암호화 과정에서 사용된 라운드 키를 역순으로 적용한다.

AES의 키 스케줄링 알고리즘은 128비트 비밀 키 (K_0, K_1, K_2, K_3)로 라운드 키 K_4, K_5, ... ,K_{43}을 생성한다. 알고리즘은 다음과 같다.

```
for i = 4, i ≤ 43, i = i + 1,
  if i ≡ 0 mod 4,
    then Ki = Ki-4 ^ SB(RotByte(Ki-1))
         ^ Rcon(i/4),
    else Ki = Ki-4 ^ Ki-1,
  end if
end for
```

SB()는 Sbox 치환 함수이며, RotByte()는 1바이트 왼쪽 회전연산 함수이고, Rcon은 고정된 상수이다.

Ⅲ. 스트림 암호 설계

기밀성과 무결성이 우수하면서 실시간 처리가 필요한 네트워크중심전 구현을 위한 기술적 환경 요구사항에 적용할 128비트 출력 스트림 암호의 설계 목표는 다음과 같다.

- 5단계 파이프라인 구현하여 실시간처리.
- 메모리를 사용하지 않고 제어신호에 따른 라운드 키 계산과 Sbox 적용.
- 블록 암호운영 모드인 변형된 OFB 적용.

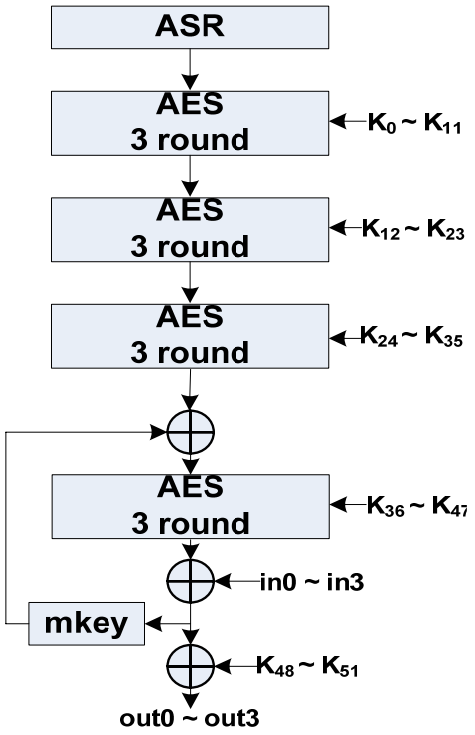

그림 1. 스트림 암호 흐름도
Fig. 1. Diagram of stream cipher

개발한 스트림 암호의 전체 진행 과정을 그림 1로 표현했으며 AES는 12라운드 적용과 비밀 키는 128비트를 사용했다. 그림 1에서 사용자가 입력한 비밀 키 128비트로 AES 12라운드 키를 생성하고 또한 IV 128비트로 ASR을 초기화 한 후, ASR 149비트의 의사 랜덤 수열을 생성한다. 생성된 수열 중 하위 128비트를 AES의 입력으로 사용한다. AES는 9라운드까지 수행 후 그 결과와 128비트 mkey와 XOR연산을 수행한 후 다시 3라운드 AES를 진행한다. 여기서 첫 128비트 mkey값은 사용자가 입력한 비밀 키 128비트 이다. 12라운드 AES 진행 결과와 평문 128비트(in0 - in3)와 XOR연산을 수행, 결과는 다음 128비트 mkey 값이 된다. 최종적인 128비트 암호문(out0 - out3)은 AES의 마지막 128비트 라운드 키(K_{48} - K_{51})와 XOR 연산을 통해 얻어진다.

그림 1의 진행과정은 암호화 과정으로 복호는 그림 1과 조금 다른 진행과정을 보이며, 자세한 내용은 다음 장에서 상세히 설명한다. 그리고 개발한 스트림 암호는 Visual Studio 2010 C 컴파일러를 사용하여 알고리즘 테스트용 프로그램을 만들었고 암호와 복호가 정상적으로 수행하는 것을 확인 했다.

Ⅳ. 하드웨어 구현

그림 2는 개발한 스트림 암호의 5단계 파이프라인으로 구성한 모습을 그림으로 표현했다. ASR-149와 AES 블록은 128비트 D-latch를 포함한 순차회로이고 나머지 블록은 메모리가 없는 조합회로이다.

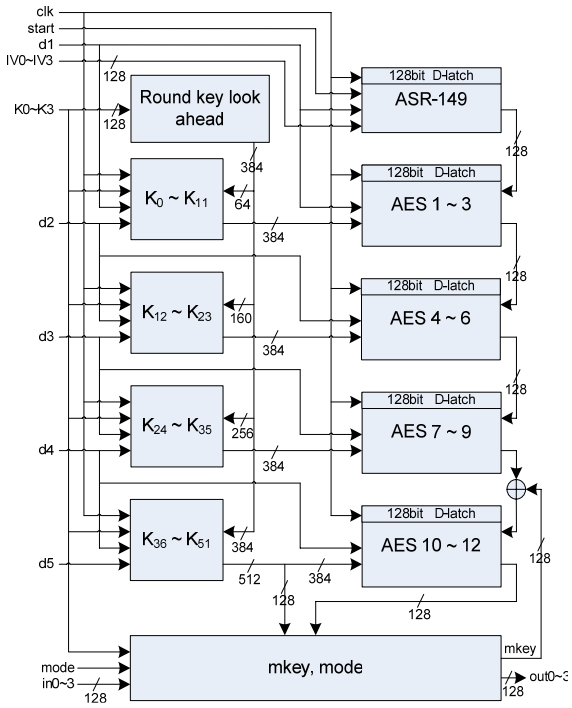

그림 2. 스트림 암호 하드웨어 구성도
Fig. 2. Hardware configuration of stream cipher

스트림 암호의 진행과정은 1비트 신호 입력으로 clk, start, d1 - d5, mode이며 32비트 단위 128비트 입력은 각각 K_0 - K_3과 IV_0 - IV_3이고, 32비트 단위 128비트 평문(in0 - in3)을 입력 받아 5단계 파이프라인을 통과한 후 32비트 단위 128비트 암호문(out0 - out3)을 출력한다.

4.1. ASR-149

PRNG로 사용한 ASR을 개발한 스트림 암호에서는 GF(2^{149})상에서 특성다항식은 16진수로 0x00200000 0x00000001 0x00000001 0x00000001 0x00000091, D = 2^{21}을 적용한다. ASR 알고리즘은 수식 (1)과 같다.

$$
\begin{aligned}
W &= ASR^{i}_{4} \\
ASR^{i+1}_{4} &= (ASR^{i}_{3} \gg 11) \\
ASR^{i+1}_{3} &= ((ASR^{i}_{3} \ll 21) \mid (ASR^{i}_{2} \gg 11)) \hat{} W \\
ASR^{i+1}_{2} &= ((ASR^{i}_{2} \ll 21) \mid (ASR^{i}_{1} \gg 11)) \hat{} W \\
ASR^{i+1}_{1} &= ((ASR^{i}_{1} \ll 21) \mid (ASR^{i}_{0} \gg 11)) \hat{} W \\
ASR^{i+1}_{0} &= (ASR^{i}_{0} \ll 21) \hat{} (W \ll 7) \hat{} (W \ll 4) \hat{} W \quad (1)
\end{aligned}
$$

수식 (1)에서 W는 32비트 임시 변수이고, ASR의 위 첨자 i는 현재 상태를 나타내고 i+1은 다음 상태를 나타낸다. ≪, ≫는 각각 32비트 단위 왼쪽, 오른쪽 시프트 연산이고, |, ^는 32비트 단위 OR, XOR연산이다.

그림 3은 ASR의 다음 상태 변환을 수식 (1)보다 쉬운 그림으로 표현한 것이다.

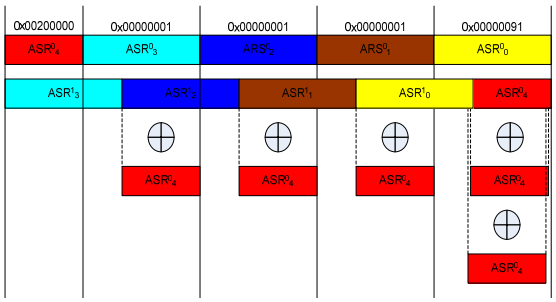

그림 3. ASR의 다음 상태도
Fig. 3. ASR flowcharts next state

4.2. 라운드 키 생성

개발한 스트림 암호에서는 AES를 12 라운드 적용하므로 32비트 라운드 키 52개(K_0 - K_{51})가 필요하다. 52개의 라운드 키는 메모리에 저장 후 사용하는 것이 아니라 제어입력 신호에 따라 라운드 키를 생성하는 방법을 사용 했다.

그림 2에서와 같이 4단계 파이프라인 적용에 맞추어 단계별 라운드 키 12개씩 라운드 키 생성블록에서 제어 신호에 의해 생성되며 마지막 라운드 키 생성 블록은 16개의 라운드 키를 생성한다. Round Key look ahead는 비밀 키 128비트로 4단계의 라운드 키 생성 블록에 384비트(rk0 - rk11)의 Sbox연산, 왼쪽 1바이트 회전연산, Rcon값을 미리 계산하여 각각 입력으로 연결된다. Round Key look ahead 알고리즘은 수식 (2)와 같다.

$$
\begin{aligned}
rk0 &= SB(RotByte(K_3)) \hat{} 0x01 \\
rk1 &= SB(RotByte(rk0 \hat{} K_3 \hat{} K_2 \hat{} K_1 \hat{} K_0)) \hat{} 0x02 \\
rk2 &= SB(RotByte(rk1 \hat{} K_3 \hat{} K_1)) \hat{} 0x04 \\
rk3 &= SB(RotByte(rk2 \hat{} K_3 \hat{} K_2)) \hat{} 0x08 \\
rk4 &= SB(RotByte(rk3 \hat{} K_3)) \hat{} 0x10 \\
rk5 &= SB(RotByte(rk4 \hat{} rk0 \hat{} K_3 \hat{} K_2 \hat{} K_1 \hat{} K_0)) \hat{} 0x20 \\
rk6 &= SB(RotByte(rk5 \hat{} rk1 \hat{} K_3 \hat{} K_1)) \hat{} x040 \\
rk7 &= SB(RotByte(rk6 \hat{} rk2 \hat{} K_3 \hat{} K_2)) \hat{} 0x80 \\
rk8 &= SB(RotByte(rk7 \hat{} rk3 \hat{} K_3)) \hat{} 0x1b \\
rk9 &= SB(RotByte(rk8 \hat{} rk4 \hat{} rk0 \hat{} K_3 \hat{} K_2 \hat{} K_1 \hat{} K_0)) \hat{} 0x36 \\
rk10 &= SB(RotByte(rk9 \hat{} rk5 \hat{} rk1 \hat{} K_3 \hat{} K_1)) \hat{} 0x6c \\
rk11 &= SB(RotByte(rk10 \hat{} rk6 \hat{} rk2 \hat{} K_3 \hat{} K_2)) \hat{} 0xd8 \quad (2)
\end{aligned}
$$

수식 (2)에서 생성된 rk0 - rk11 값은 메모리가 아닌 선(wire)이다.

K_0 - K_{11}블록은 1비트 제어신호 clk, d1, d2, 비밀키 128비트, Round Key look ahead로부터 64비트(rk0, rk1)를 입력으로 받아 32비트 12개의 라운드 키를 생성한다. 생성 알고리즘은 수식 (3)과 같다.

$$
\begin{aligned}
K_0 &= K_0 \\
K_1 &= K_1 \\
K_2 &= K_2 \\
K_3 &= K_3 \\
K_4 &= rk0 \hat{} K_0 \\
K_5 &= rk0 \hat{} K_1 \hat{} K_0 \\
K_6 &= rk0 \hat{} K_2 \hat{} K_1 \hat{} K_0 \\
K_7 &= rk0 \hat{} K_3 \hat{} K_2 \hat{} K_1 \hat{} K_0 \\
K_8 &= rk1 \hat{} rk0 \hat{} K_0 \\
K_9 &= rk1 \hat{} K_1 \\
K_{10} &= rk1 \hat{} rk0 \hat{} K_2 \hat{} K_0 \\
K_{11} &= rk1 \hat{} K_3 \hat{} K_1 \qquad (3)
\end{aligned}
$$

K_{12} - K_{23}블록은 1비트 제어신호 clk, d2, d3, 비밀키 128비트, Round Key look ahead로부터 160비트(rk0 - rk4)를 입력으로 받아 32비트 12개의 라운드 키를 생성한다. 생성 알고리즘은 수식 (4)와 같다.

$$
\begin{aligned}
K_{12} &= rk2 \hat{} rk1 \hat{} rk0 \hat{} K_0 \\
K_{13} &= rk2 \hat{} rk0 \hat{} K_1 \hat{} K_0 \\
K_{14} &= rk2 \hat{} rk1 \hat{} K_2 \hat{} K_1 \\
K_{15} &= rk2 \hat{} K_3 \hat{} K_2 \\
K_{16} &= rk3 \hat{} rk2 \hat{} rk1 \hat{} rk0 \hat{} K_0 \\
K_{17} &= rk3 \hat{} rk1 \hat{} K_1 \\
K_{18} &= rk3 \hat{} rk2 \hat{} K_2 \\
K_{19} &= rk3 \hat{} K_3 \\
K_{20} &= rk4 \hat{} rk3 \hat{} rk2 \hat{} rk1 \hat{} rk0 \hat{} K_0 \\
K_{21} &= rk4 \hat{} rk2 \hat{} rk0 \hat{} K_1 \hat{} K_0 \\
K_{22} &= rk4 \hat{} rk3 \hat{} rk0 \hat{} K_2 \hat{} K_1 \hat{} K_0 \\
K_{23} &= rk4 \hat{} rk0 \hat{} K_3 \hat{} K_2 \hat{} K_1 \hat{} K_0 \qquad (4)
\end{aligned}
$$

K_{24} - K_{35}블록은 1비트 제어신호 clk, d3, d4, 비밀키 128비트, Round Key look ahead로부터 256비트(rk0 - rk7)를 입력으로 받아 32비트 12개의 라운드 키를 생성한다. 생성 알고리즘은 수식 (5)와 같다.

$$
\begin{aligned}
K_{24} &= rk5 \hat{} rk4 \hat{} rk3 \hat{} rk2 \hat{} rk1 \hat{} rk0 \hat{} K_0 \\
K_{25} &= rk5 \hat{} rk3 \hat{} rk1 \hat{} K_1 \\
K_{26} &= rk5 \hat{} rk4 \hat{} rk1 \hat{} rk0 \hat{} K_2 \hat{} K_0 \\
K_{27} &= rk5 \hat{} rk1 \hat{} K_3 \hat{} K_1 \\
K_{28} &= rk6 \hat{} rk5 \hat{} rk4 \hat{} rk3 \hat{} rk2 \hat{} rk1 \hat{} rk0 \hat{} K_0 \\
K_{29} &= rk6 \hat{} rk4 \hat{} rk2 \hat{} rk0 \hat{} K_1 \hat{} K_0 \\
K_{30} &= rk6 \hat{} rk5 \hat{} rk2 \hat{} rk1 \hat{} K_2 \hat{} K_1 \\
K_{31} &= rk6 \hat{} rk2 \hat{} K_3 \hat{} K_2 \\
K_{32} &= rk7 \hat{} rk6 \hat{} rk5 \hat{} rk4 \hat{} rk3 \hat{} rk2 \hat{} rk1 \hat{} rk0 \hat{} K_0 \\
K_{33} &= rk7 \hat{} rk5 \hat{} rk3 \hat{} rk1 \hat{} K_1 \\
K_{34} &= rk7 \hat{} rk6 \hat{} rk3 \hat{} rk2 \hat{} K_2 \\
K_{35} &= rk7 \hat{} rk3 \hat{} K_3 \qquad (5)
\end{aligned}
$$

K_{36} - K_{51}블록은 1비트 제어신호 clk, d4, d5, 비밀키 128비트, Round Key look ahead로부터 384비트(rk0 - rk11)를 입력으로 받아 32비트 16개의 라운드 키를 생성한다. 생성 알고리즘은 수식 (6)과 같다.

$$
\begin{aligned}
K_{36} &= rk8 \hat{} rk7 \hat{} rk6 \hat{} rk5 \hat{} rk4 \hat{} rk3 \hat{} rk2 \hat{} rk1 \hat{} rk0 \hat{} K_0 \\
K_{37} &= rk8 \hat{} rk6 \hat{} rk4 \hat{} rk2 \hat{} rk0 \hat{} K_1 \hat{} K_0 \\
K_{38} &= rk8 \hat{} rk7 \hat{} rk4 \hat{} rk3 \hat{} rk0 \hat{} K_2 \hat{} K_1 \hat{} K_0 \\
K_{39} &= rk8 \hat{} rk4 \hat{} rk0 \hat{} K_3 \hat{} K_2 \hat{} K_1 \hat{} K_0 \\
K_{40} &= rk9 \hat{} rk8 \hat{} rk7 \hat{} rk6 \hat{} rk5 \hat{} rk4 \hat{} rk3 \hat{} rk2 \hat{} rk1 \hat{} rk0 \hat{} K_0 \\
K_{41} &= rk9 \hat{} rk7 \hat{} rk5 \hat{} rk3 \hat{} rk1 \hat{} K_1 \\
K_{42} &= rk9 \hat{} rk8 \hat{} rk5 \hat{} rk4 \hat{} rk1 \hat{} rk0 \hat{} K_2 \hat{} K_0 \\
K_{43} &= rk9 \hat{} rk5 \hat{} rk1 \hat{} K_3 \hat{} K_1 \\
K_{44} &= rk10 \hat{} rk9 \hat{} rk8 \hat{} rk7 \hat{} rk6 \hat{} rk5 \hat{} rk4 \hat{} rk3 \hat{} rk2 \hat{} rk1 \hat{} rk0 \hat{} K_0 \\
K_{45} &= rk10 \hat{} rk8 \hat{} rk6 \hat{} rk4 \hat{} rk2 \hat{} rk0 \hat{} K_1 \hat{} K_0 \\
K_{46} &= rk10 \hat{} rk9 \hat{} rk6 \hat{} rk5 \hat{} rk2 \hat{} rk1 \hat{} K_2 \hat{} K_1 \\
K_{47} &= rk10 \hat{} rk6 \hat{} rk2 \hat{} K_3 \hat{} K_2 \\
K_{48} &= rk11 \hat{} rk10 \hat{} rk9 \hat{} rk8 \hat{} rk7 \hat{} rk6 \hat{} rk5 \hat{} rk4 \hat{} rk3 \hat{} rk2 \hat{} rk1 \hat{} rk0 \hat{} K_0 \\
K_{49} &= rk11 \hat{} rk9 \hat{} rk7 \hat{} rk5 \hat{} rk3 \hat{} rk1 \hat{} K_1 \\
K_{50} &= rk10 \hat{} rk7 \hat{} rk6 \hat{} rk3 \hat{} rk2 \hat{} rk1 \hat{} K_2 \\
K_{51} &= rk11 \hat{} rk7 \hat{} rk3 \hat{} K_3 \qquad (6)
\end{aligned}
$$

수식 (2, 3, 4, 5, 6)에서 사용자가 입력한 비밀 키 128비트(K_0 - K_3)로 4단계 파이프라인에서 요구된 각각의 라운드 키를 XOR 연산만으로 구현할 수 있다.

4.3. AES

AES는 ASR로부터 128비트를 입력으로 받아 3라운드 씩 4단계 파이프라인을 통해 128비트 스트림 키를 만든다. 그림 1에서 AES는 9라운드 실행 결과와 128비트 mkey와 XOR연산한 결과를 나머지 3라운드

AES에 적용한다. 그리고 변형된 OFB모드 적용으로 갱신된 mkey가 계속해서 AES에 적용된다.

AES의 하드웨어 구현에 관한 많은 연구 결과물이 있으며, 개발한 스트림 암호는 AES의 하드웨어 구현 기술 중 네트워크중심전의 기술적 환경을 구현하는데 가장 보편적인 기술을 적용 했다.

4.3.1. SubBytes(SB)

일반적으로 AES에 사용된 Sbox의 구현은 미리 계산된 8비트 Sbox값 256바이트를 메모리에 저장한 다음 사용하는 LookUp Table: LUT 방식을 사용한다. 개발한 스트림 암호는 네트워크중심전에서 사용되는 휴대용장비나 계산능력, 메모리, 전력, 대역폭 등이 제한적인 장비에 적용하기 위해 LUT방식이 아닌 제어신호에 따라 합성체 산술계산(Composite Field Arithmetic)[7,8,9]으로 실시간 계산하는 방식으로 구현했다. 합성체를 이용한 Sbox구현은 4단계 과정으로 다음과 같다.

1. 동형사상(Isomorphism) δ 함수를 사용하여 $GF(2^8)$을 합성체로 변환.
2. 합성체에서 곱셈의 역원 계산.
3. 역 동형사상 δ^{-1} 함수로 합성체에서 $GF(2^8)$으로 재변환.
4. Affine 변환.

합성체에서 곱셈의 역원 계산이 가장 많은 계산이 요구되며, 이는 $GF(2^8)$을 GF(2)로 반복 적용하여 구성할 수 있다. 이때 적용된 기약 다항식(Irreducible Polynomial)은 수식 (7)과 같다.

$$\begin{aligned} GF(2^2) &: x^2 + x + 1 \\ GF((2^2)^2) &: x^2 + x + \phi \\ GF(((2^2)^2)^2) &: x^2 + x + \lambda \end{aligned} \quad (7)$$

수식 (7)을 기반으로 합성체를 이용한 곱셈의 역원(Multiplicative Inverses)을 계산하는 알고리즘을 그림 4로 표현했다.

그림 4에서 8비트 입력은 S이고 출력 S^{-1}는 입력 S의 곱셈의 역원이다. 그리고 4각형 박스는 부분함수로, 첫 x^2은 S의 상위 4비트를 제곱하는 부분 함수로 수식 (8)과 같다.

$$C_3 = S_7$$

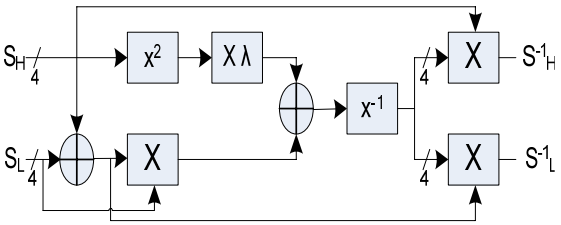

그림 4. 합성체를 이용한 곱셈 역원 계산
Fig. 4. Multiplicative inverse calculated using the composites field

$$\begin{aligned} C_2 &= S_7 \hat{} S_6 \\ C_1 &= S_6 \hat{} S_5 \\ C_0 &= S_7 \hat{} S_5 \hat{} S_4 \end{aligned} \quad (8)$$

수식 (8)에서 변수의 아래 첨자는 비트 순서번호이고 제곱한 결과 4비트는 C이다.

x^2의 결과와 4비트 상수곱셈기 λ 부분함수에서 상수 $\lambda = \{1100\}_2$ 이며, 수식 (9)과 같다.

$$\begin{aligned} D_3 &= C_2 \hat{} C_0 \\ D_2 &= C_3 \hat{} C_2 \hat{} C_1 \hat{} C_0 \\ D_1 &= C_3 \\ D_0 &= C_2 \end{aligned} \quad (9)$$

수식 (9)에서 곱셈기 λ의 결과는 D이다.

그림 4에서 4비트 곱셈기 부분 함수는 X 이다. X 부분함수는 그림 5와 같다.

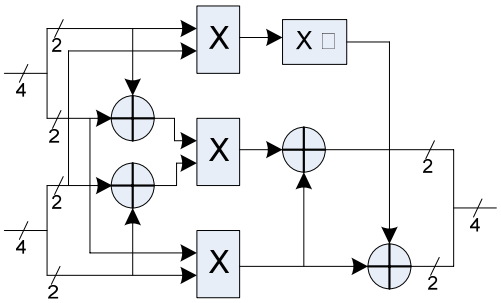

그림 5. $GF(2^2)^2$) 곱셈기
Fig. 5. $GF((2^2)^2)$ multiplier

그림 5에서 X는 2비트 곱셈기 부분함수이며, 수식 (10)와 같다.

$$\begin{aligned} M_1 &= (A_0 \,\&\, B_0) \hat{} ((A_1 \hat{} A_0) \,\&\, (B_1 \hat{} B_0)) \\ M_0 &= (A_0 \,\&\, B_0) \hat{} (A_1 \,\&\, B_1) \end{aligned} \quad (10)$$

수식 (10)에서 2개의 2비트 피연산자는 각각 A, B이고 곱셈 결과는 M이다. 마지막으로 2비트 상수곱셈

기 ϕ 부분함수에서 ϕ = {10}$_2$이다. 2비트 곱셈 결과와 ϕ 곱셈은 수식 (11)과 같다.

$$E_1 = M_1 \hat{} M_0$$
$$E_0 = M_1 \quad (11)$$

합성체를 이용한 AES의 Sbox 구현에서 곱셈의 역원을 계산하는 수식 (8, 9, 10, 11)은 간단한 논리함수로만 이루어져 있으며 이는 하드웨어 구현을 매우 쉽게 할 수 있다. 마지막으로 그림 4의 8비트 입력 전 8 * 8의 동형사상 δ와 출력 결과에 대한 역 동형사상 δ^{-1}는 참고문헌[9]를 그대로 사용했다.

4.3.2. ShiftRows(SR)

SR()는 바이트 단위 왼쪽 회전연산으로 이는 하드웨어 구현에서 간단히 선(wire)으로 연결해서 구현할 수 있다.

4.3.3. MixColumns(MC)

MC()는 GF(2^8) 행렬곱셈 연산으로 선형변환을 통해 암호의 특성을 확산 시키는 기능을 한다. GF(2^8) 곱셈에서 고정된 다항식(fixed polynomial) a(x)를 사용하며, a(x) = {03}x^3 + {01}x^2 + {01}x + {02}이다. 개발한 스트림 암호는 참고문헌[10]의 알고리즘을 사용했다.

MC()의 행렬곱셈은 수식 (12)와 같다.

$$H(x) = \begin{bmatrix} 02\,03\,01\,01 \\ 01\,02\,03\,01 \\ 01\,01\,02\,03 \\ 03\,01\,01\,02 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix} \quad \text{for } 0 \le c \le 3 \quad (12)$$

S는 AES의 128비트 state를 바이트 단위 4 * 4행렬을 나타내며 S의 아래첨자는 바이트 순서 번호이다. 수식 (12)를 전개하여 표현하면 다음과 같다.

$$H_{0,c} = \{02\}S_{0,c} \hat{} \{03\}S_{1,c} \hat{} S_{2,c} \hat{} S_{3,c}$$
$$H_{1,c} = S_{0,c} \hat{} \{02\}S_{1,c} \hat{} \{03\}S_{2,c} \hat{} S_{3,c}$$
$$H_{2,c} = S_{0,c} \hat{} S_{1,c} \hat{} \{02\}S_{2,c} \hat{} \{03\}S_{3,c}$$
$$H_{3,c} = \{03\}S_{0,c} \hat{} S_{1,c} \hat{} S_{2,c} \hat{} \{02\}S_{3,c} \quad (13)$$

수식 (13)에서 8비트 {02}, {03}의 곱셈은 참고문헌[3]의 xtime()를 사용하여 구현했다.

$$\{02\}S_{x,c} = \text{xtime}(S_{x,c})$$
$$\{03\}S_{x,c} = \text{xtime}(S_{x,c}) \hat{} S_{x,c} \quad (14)$$

xtime() 알고리즘은 수식 (15)와 같다.

$$a_7 = b_6$$
$$a_6 = b_5$$
$$a_5 = b_4$$
$$a_4 = b_7 \hat{} b_3$$
$$a_3 = b_7 \hat{} b_2$$
$$a_2 = b_1$$
$$a_1 = b_7 \hat{} b_0$$
$$a_0 = b_7 \quad (15)$$

수식 (15)에서 xtime()의 8비트 입력을 b로 표시하고 8비트 결과는 a로 표시 했으며 아래첨자는 비트 순서번호이다.

4.3.4. AddRoundKey(ARK)

ARK()는 XOR연산으로 하드웨어 구현을 할 수 있다.

4.4. mkey, mode

그림 2의 블록 mkey, mode는 최종적인 128비트 암호문을 만드는 조합회로이다. 입력으로 제어신호 mode, 평문 128비트 in0 - in3, AES10 - 12블록으로부터 128비트, 라운드 키(K_{48} - K_{51})을 입력으로 받아 최종적인 암호문 128비트 out0 - out3과 새롭게 갱신된 mkey 128비트를 생성한다. 만일 평문 입력이 128비트보다 작을 경우 공백으로 128비트를 채워서 mkey, mode블록을 진행한다.

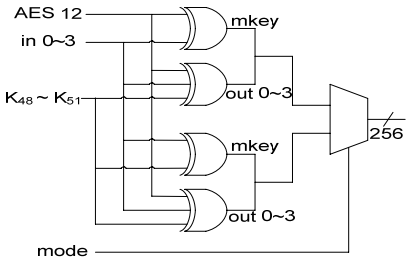

그림 6. mkey, mode 알고리즘
Fig. 6. mkey, mode algorithm

그림 6은 mkey, mode블록의 하드웨어 구조를 설명하고 있다. 그림 6에서 최종적인 출력 결과 out는 mode 신호에 의해 2 * 1 선택기(MUX)를 통해 256비트 값을 출력한다. mode가 0이면 암호문과 갱신된 mkey를 출력하고, 1이면 평문과 갱신된 mkey를 출력한다. mode가 0인 경우 AES 10 - 12 블록 128비트를 128비트 평문(in0 - in3)과 XOR연산을 수행하여 128비트 mkey를 만들고, 최종적인 암호문은 mkey에 128

비트 라운드 키(K_{48} - K_{51})를 XOR연산으로 생성한다.

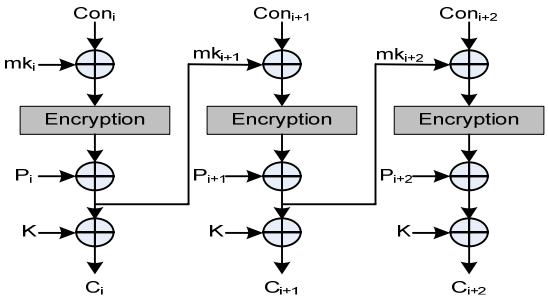

그림 7. 암호 진행도
Fig. 7. Encryption flowchart

평문으로 복호는 mode 신호가 1이며 수행과정은 128비트 암호문과 128비트 라운드 키를 XOR연산하여 mkey를 생성한 후 그 결과와 128비트 AES 10 - 12블록 결과와 XOR연산을 통해 최종적인 평문 128비트를 만들다. 이와 같이 스트림 암호이지만 암호와 복호가 다른 이유는 스트림 암호에 기밀성과 무결성 보장을 위해 암호운영 모드인 변형된 OFB모드를 적용했기 때문이다. 그림 7과 그림 8은 변형된 OFB모드를 적용한 스트림 암호의 암호와 복호 진행을 그림으로 표현 했다.

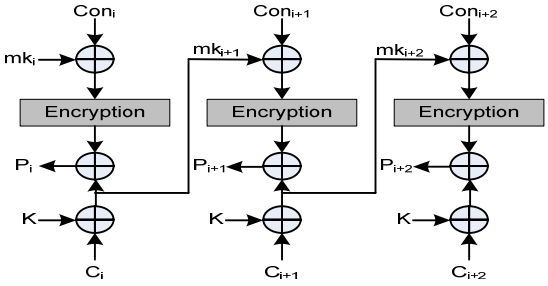

그림 8. 복호 진행도
Fig. 8. Decryption flowchart

그림 7, 8에서 Con은 AES 10 - 12블록 수행 결과 128비트이며, P는 평문, C는 암호문 K는 라운드 키, mk는 mkey이다.

네트워크중심전에서 개별 전력요소들 간의 실시간 데이터 전송이 전장 상황을 빠르고 정확하게 해석할 수 있다. 그러나 파일 단위로 암호와 복호를 수행하는 블록 암호는 실시간 처리가 불가능하고 실시간 암호처리는 스트림 암호만이 가능하지만 스트림 암호는 기밀성과 무결성이 보장되지 않는 단점이 있다. 그래서 그림 7,8과 같이 스트림 암호에 블록 암호 운영모드를 적용함으로서 기밀성과 무결성이 우수하면서 실시간 처리가 가능한 스트림 암호를 설계했다.

Ⅴ. 하드웨어 구현 결과

하드웨어 구현은 Verilog HDL을 사용하여 Modelsim 6.5d를 활용하여 스트림 알고리즘의 기능을 검증하였고 성능은 Quartus II 12.0을 활용하여 분석한 결과, Altera Cyclone Ⅳ GX - 32bit FPGA에서 Compile한 결과 Total Logic Elements : 35,759 / 109,424(33%), Total Registers : 0, Total Pins: 392 / 508(77%) 이다. 표 1은 개발된 스트림 암호의 성능을 요약한 표이다. 그리고 Worst Case에서 Max Frequency는 32.3MHz(4.13Gbps)의 빠른 성능을 보여주었다.

그리고 네트워크중심전 구현을 위한 핵심 기술 중 하나인 전술데이터링크(Tactical Data Link: TDL) 기술은 CDL(Common Data Link), MP-CDL(Multi-Platform Common Data Link) 등이 있으며, 초기에는 간단한 문자나 소리 등의 전송을 위해 낮은 전송속도(10.71/45Mbps)를 보였지만 최근에는 대용량의 멀티미디어 데이터(사진, 동영상 등) 전송을 위해 548/1096Mbps까지 구현할 예정이다. 그래서 개발한 스트림 암호는 네트워크중심전 구현을 위한 기술적 환경 수행속도[11]를 충분히 만족함을 보여준다.

표 1. 하드웨어 구현 요약
Table. 1 Hardware implementation summary

Algorithm	Device	Data-rate (Gbps)	Area(slices)
AES[12]	XCV2000E	20.3 / 1.82	5810, 100B RAM
This paper	EP4CGX110DF 31C7	4.13	35,759

그리고 표 1의 AES[12]는 5단계 펼쳐진(Unrolling) 전체 파이프라인과 5단계 반복적(Iterative) 파이프라인 구현 결과이다. 참고문헌[12]과 개발한 스트림 암호의 하드웨어 개발 환경이 다르며 5단계 펼쳐진 전체 파이프라인 구현은 CTR모드로 구현한 것이므로 본 논문과의 비교는 적절치 않다. 그래서 반복적 파이프라인 구현과 비교에서 하드웨어 실행속도(Data-rate)가 많이 개선된 것을 볼 수 있다.

개발한 스트림 암호의 하드웨어 구현은 그림 2의 전체 구성도와 같이 5단계 파이프라인으로 구현 했다. 이를 간략히 표현한 것이 그림 9이다.

그림 9에서 ASR은 ASR-149블록을 표시한 것이

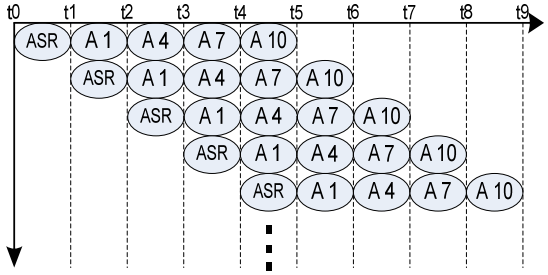

그림 9. 5단계 파이프라인 흐름도
Fig. 9. 5-stage pipeline flowchart

고, A1, A4, A7, A10는 각각 AES 1 - 3, AES 4 - 6, AES 7 - 9, AES 10 - 12블록을 표시한 것이다. 처음 스트림 암호의 시작이 t0일 때 첫 암호문 128비트의 생성은 시간이 t5가 되어야 생성된다. 그 후 매 t시간마다 128비트 암호문이 만들어지므로 대기시간(latency)이 t5시간이 된다.

Ⅵ. 안전성 분석

개발한 스트림 암호의 안전성 분석은 그림 1에서 ASR 블록은 상태변화에 따라 의사랜덤 수열을 계속해서 생성하고 있다. 따라서 ASR 블록의 전체적인 선형 복잡도 분석 확률을 먼저 구하면, ASR 블록의 최소 선형 복잡도는 149비트이다. 즉 298개 출력 값을 알면 ASR 블록의 모든 상태를 분석할 수 있다. 1/298의 확률은 약 2^{-8}이다. 이는 ASR 블록의 안전성이 약 2^{-8}이라고 추정할 수 있다. 그리고 ASR 블록은 0을 제외한 최대 주기 수열을 생성한다. 이는 ASR의 특정 비트 또는 비트열(연속적이거나 연속적이지 않는 모든 경우)은 0을 제외한 모든 값들이 동일한 출현 빈도수를 보여줌으로 특정한 값을 추정한 어떤 공격[13]에도 내성이 있다.

개발한 스트림 암호는 블록 암호 알고리즘인 AES의 12 라운드 출력 결과를 스트림 키로 사용하는 AES를 기반으로 한다. 그래서 AES의 안전성이 개발한 스트림 암호의 안전성에 직접적인 영향을 미치고 있다. 현재까지 AES에 대한 많은 확률적인 축소된 라운드에 대한 안전성[14-17]은 증명이 되어 있으며, 아직까지는 AES 전체 라운드 분석 방법이 없으므로 AES 암호 알고리즘은 안전한 것으로 추정 할 수 있다. AES의 증명된 안전성을 개발한 스트림 암호가 그대로 물려받았으므로 AES의 안전성만큼 개발한 스트림 암호도 안전할 것으로 판단된다.

그리고 그림 1에서 평문과 파이프라인 4단계의 AES 블록에서 생성된 스트림 키와 XOR연산 결과에 마지막 라운드 키에 의한 표백 값으로 다시 XOR 연산을 수행 하는 이유는 출력된 암호문은 완전히 노출되는 값이며, 이 값은 변형된 OFB모드 적용을 위해 다음 128비트 mkey 정보를 포함하고 있다. 이 정보를 감추기 위해 마지막 라운드 키로 표백하여 정보를 희석시킨 후 최종적인 암호문을 만든다. 생성된 암호문으로 평문을 찾기 어렵고, 다음 단계 128비트 mkey를 찾기도 매우 어렵다.

그리고 개발한 스트림 암호에 암호운영 모드의 적용은 스트림 암호의 안전성에는 영향이 없지만 기밀성과 무결성 보장을 위해 모드를 적용했다. 이 모드의 적용으로 현재 출력된 128비트 암호문 중에서 1문자 변경 또는 공격으로 인해 다음 128비트 암호문에서는 해독이 불가능한 확산(Diffusion)이 발생한다. 이와 같은 확산 때문에 무결성이 보장될 수 있다. 그리고 데이터 전송과정에서 발생한 오류에 의한 확산도 발생한다.

Ⅶ. 결 론

실시간 적용이 가능하며 기밀성과 무결성이 우수한 대용량 및 다양한 유형의 데이터 처리가 가능한 미래 전장에 적용할 기반 기술인 128비트 출력의 스트림 암호 알고리즘을 개발 했다. 개발한 스트림 암호는 AES 알고리즘을 하드웨어로 구현하여 보안이 우수하며 네트워크중심전의 기술적 요구사항을 만족하고 있다.

개발한 스트림 암호 알고리즘을 유무선으로 연결된 각 개별 전력 요소에 적용하여 네트워크중심전의 실질적인 적용과 테스트가 다음 연구의 목표이다.

References

[1] H. No and T. S. Sohn, "NCW," *J. KIDA*, vol. 5, no. 19, pp. 1046(05-19), May. 2005.

[2] I. S. Choi, "Defense information security in NCW environments," *J. Korean Inst. Inform. Sci. Eng. (KIISE)*, vol. 25, no. 9, pp. 81-88, Aug. 2007.

[3] NIST, "*Advanced Encryption Standard(AES)*," FIPS PUB 197, Nov. 2001.

[4] C. S. Park and G. Y. Cho, "Generalization of Galois linear feedback register," *J. Inst. Electron. Eng. Korea (IEEK)*, vol. 43C1, no. 1, pp. 1-8, Jan. 2006.

[5] NIST, *Papers Submitted to AES3*, retrieved Aug, 10, 2013, from http://csrc.nist.gov/archive/aes/round2/conf3/aes3papers.html

[6] National Bureau of Standards, "*Data Encryption Standard*," FIPS PUB 46, Jan. 1977.

[7] A. Rudra, P. K. Dubey, C. S. Jutla, V. Kumar, J. R. Rao, and P. Rohatgi, "Efficient Rijndael encryption implementation with composite field arithmetic," *Lecture Notes Comput. Sci.*, vol. 2162, pp. 171–184, May 2001.

[8] A. Satoh, S. Morioka, K. Takano, and S. Munetoh, "A compact Rijndael hardware architecture with S-Box optimization," *Lecture Notes Comput. Sci.*, vol. 2248, pp. 239–254, Dec. 2001.

[9] S. Morioka and A. Satoh, "An optimized S-Box circuit architecture for low power AES design," *Lecture Notes Comput. Sci.*, vol. 2523, pp. 172–186, Aug. 2003.

[10] J. M. Grando-Criado, M. A. Vega-Roriguez, J. M. Sanchez-Perez, and J. A. Gomez-Pulido, "A new methodology to implement the AES algorithm using partial and dynamic reconfiguration," *Integration, the VLSI J.*, vol. 43, no. 1, pp. 72-80, Jan. 2010.

[11] H. K. Baek, J. S. Lim, and S. M. Jeong, "Technology trend of tactical data link for NCW," *The Journal of The Korea Institute of Communication Sciences*, vol. 28, no. 7, pp. 59-69, Sep. 2010.

[12] G. Saggese, A. Mazzeo, N. Mazzocca, and A. Strollo, "An FPGA-based performance analysis of the unrolling, tiling, and pipelining of the AES algorithm," *Lecture Notes Comput. Sci.*, vol. 2778, pp. 292-302, Sep. 2003.

[13] P. Hawkes and G. Rose, "Guess-and-determine attacks on SNOW," *Lecture Notes Comput. Sci.*, vol. 2595, pp. 37-46, Aug. 2002.

[14] J. Daemen and V. Rijmen, "*The Design of Rijndael*," Springer-Verlag, Sep. 2002.

[15] J. Cheon, M. Kim, K. Kim, J. Lee, and S. Kang, "Improved impossible differential cryptanalysis of Rijndael and crypton," *Lecture Notes Comput. Sci.*, vol. 2288, pp. 39-49, Dec. 2001.

[16] A. Birynkov, "The boomerang attack on 5 and 6-round reduced AES," *Lecture Notes Comput. Sci.*, vol. 3373, pp. 42-57, May. 2004.

[17] S. Hong, S. Lee, J. Lim, J. Sung, D. Cheon, and I. Cho, "Provable security against differential and linear cryptanalysis for the SPN structure," *Lecture Notes Comput. Sci.*, vol. 1978, pp. 273-283, Apr. 2001.

김 성 기 (Sung-Gi Kim)

1977년 2월 울산대학교 응용물리학과 졸업

1994년 2월 부경대학교 전자계산학과 석사

2004년 3월~현재 부경대학교 IT융합응용공학과 박사과정

<관심분야> 전산기 구조, 통신공학, 암호 알고리즘

김 길 호 (Gil-Ho Kim)

2000년 2월 한국방송통신대학교 전자계산학과 졸업

2002년 2월 부경대학교 컴퓨터공학과 석사 졸업

2010년 2월 부경대학교 컴퓨터공학과 박사 졸업

<관심분야> 암호 알고리즘, 통신공학, 컴퓨터 구조

조 경 연 (Gyeong-Yeon Cho)

1990년 2월 인하대학교 전자공학과(공학박사)

1996년~1991년 삼보컴퓨터 기술연구소 책임연구원

1998년~2003년 에이디칩스 기술고문

1991년~현재 부경대학교 IT융합응용공학과 교수

<관심분야> 전산기 구조, 반도체 회로 설계, 암호 알고리즘