Journal of Korea Multimedia Society Vol. 16, No. 12, December 2013(pp. 1427-1438)
http://dx.doi.org/10.9717/kmms.2013.16.12.1427

# 기밀성과 무결성이 우수한 SHACAL-2 기반 스트림 암호 설계 및 구현

김길호†, 조경연††

## 요 약

5단계 파이프라인으로 구성된 기밀성과 무결성이 우수하고 실시간처리가 가능한 128비트 출력 스트림 암호를 개발했다. 개발한 스트림 암호는 ASR 277비트와 SHACAL-2를 통해 128비트 스트림을 만들고 이를 CFB모드 적용한 후 표백처리 과정을 통하여 최종적인 128비트 암호문을 만드는 스트림 암호 알고리즘이다. 스트림 암호의 하드웨어는 Verilog HDL을 사용하여 Modelsim 6.5d를 활용하여 기능을 검증하였고, 성능은 Quartus II 12.0을 활용하여 분석했고, Worst Case에서 Max Frequency는 33.34MHz(4.27Gbps)의 빠른 성능을 보여주었다. 이는 무선 인터넷과 센서 네트워크 및 DRM 환경의 속도를 충분히 만족함을 보여준다. 기밀성과 무결성이 우수한 스트림 암호 알고리즘 개발에 본 논문은 매우 유용한 아이디어를 제공하고 있다.

# Design and Implementation of Stream Cipher based on SHACAL-2 Superior in the Confidentiality and Integrity

Gil Ho Kim†, Gyeong Yeon Cho††

## ABSTRACT

We have developed a 128-bit stream cipher algorithm composed of the 5-stage pipeline, capable of real-time processing, confidentiality and integrity. The developed stream cipher is a stream cipher algorithm that makes the final 128-bit ciphers through a whitening process after making the ASR 277 bit and SHACAL-2 and applying them to the CFB mode. We have verified the hardware performance of the proposed stream cipher algorithm with Modelsim 6.5d and Quartus II 12.0, and the result shows that the hardware runs at 33.34Mhz(4.27Gbps) at worst case. According to the result, the new cipher algorithm has fully satisfied the speed requirement of wireless Internet and sensor networks, and DRM environment. Therefore, the proposed algorithm with satisfaction of both confidentiality and integrity provides a very useful ideas.

**Key words:** SHA-2, stream cipher(스트림 암호), Confidentiality(기밀성), Integrity(무결성), WSN(무선 센서 네트워크), DRM(디지털 저작권 관리)

## 1. 서 론

현대는 정보화 사회라고 말한다. 이는 정보의 가치를 인식하고, 이러한 정보들을 통신 기술의 발달로 많은 사람들이 이용하면서 정보의 가치를 더욱더 향상시켜 나가는 것이다. 이와 같은 정보화 사회에서는 신뢰(Confidence)를 바탕으로 믿을 수 있는 정보통신 시스템이 필요하다. 특히 무선 센서 네트워크

※ 교신저자(Corresponding Author) : , 주소 : 부산광역시 남구 용소로 45(대연캠퍼스) 마이크로프로세서연구실 2301호, 전화 : (051) 629-6262, FAX : (051) 629-6230, E-mail : gycho@pknu.ac.kr
접수일 : 2013년 5월 15일, 수정일 : 2013년 11월 13일
완료일 : 2013년 11월 18일
† 부경대학교 IT융합응용공학과 마이크로프로세서연구실 (E-mail: vnlqpcdd@hanmail.net)
†† 부경대학교 IT융합응용공학과 교수, 마이크로프로세서연구실

(WSN)는 다수의 센서 노드로 구성되어 주변의 사람, 사물 그리고 환경정보를 인식하여 현재 상황을 판단한다. 이 과정에서 각 센서 노드에서 처리되는 정보는 신뢰성이 매우 중요하며, 각 센서 노드는 물리적인 제한 때문에 데이터의 저장, 계산 능력, 배터리, 대역폭 등에 분명한 한계를 보이고 있다. 그래서 신뢰성과 안전성이 확보된 센서 네트워크 구축을 위해 추가적으로 정보의 저장 및 전송에 암호의 적용이 필요하며, 암호 알고리즘의 구현은 센서 노드에서 한정된 자원을 어떻게 효율적으로 활용할 것인가에 관한 연구가 반드시 필요하다.

현재 센서 노드 상에서 구현된 암호 알고리즘은 공개 키 암호[1]와 블록 암호 AES[2,3]가 있으나, 이들은 계산량과 계산에 필요한 메모리도 많이 필요로 한다. 이는 센서 노드의 물리적 한계점과 정면으로 배치되는 것으로 이를 극복할 대안으로 최근에 3단계로 진행된 eSTREAM[4,5] 프로젝트에서 소프트웨어와 하드웨어 부문에서 몇 개의 스트림 암호가 선택되었다. 센서 노드에서 스트림 암호의 적용은 공개 키 암호와 블록 암호 알고리즘보다 가볍기 때문에 구현이 쉬운 장점도 있지만 스트림 암화만으로는 기밀성(Confidentiality)과 무결성(Integrity)의 완전한 보장이 되지 않는 단점도 있다.

예를 들어 센서 노드에서 무선으로 송수신 하는 정보는 데이터의 양이 매우 작다. 그리고 평문과 단순히 스트림 키의 XOR연산은 생성된 암호문과 일대일 대응되므로 공격자는 암호문 전체를 공격하지 않고 특정 부분의 바이트, 또는 임의의 비트를 어떤 특정 값으로 변경함으로써 공격이 쉽게 이루어질 수 있다. 이를 방지하기 위해 블록 암호처럼 운영모드를 적용할 수 있지만, 센서 노드 상에서 송수신 되는 데이터의 양이 작기 때문에 모드의 적용이 쉽지 않고, 모드 적용을 위해서는 이전에 생성된 암호문이나 IV(Initial Vector) 값을 메모리에 저장하고 있어야 하므로 추가적인 메모리가 필요해진다. 이는 물리적 환경이 매우 제한적인 무선 센서 노드 구현을 어렵게 한다. 그리고 실시간 처리가 필요한 DMB(Digital Right Management) 방송 콘텐츠(세계적인 스포츠인 월드컵 경기 생중계 방송 등)는 블록 암호로 처리할 수 없으며, 생중계 방송의 콘텐츠는 워터마킹[6]과 스트림 암호만이 실시간으로 암호를 처리할 수 있다. 그리고 WSN에서 보안이 필요한 데이터들 역시 실시간 처리가 필요하다.

이에 본 논문에서는 256비트 블록 암호인 SHACAL-2[7] 알고리즘을 32라운드로 축소하고, 기밀성과 무결성이 우수한 블록 암호 운영모드(Operation Mode)를 적용하여 128비트 출력의 스트림 암호 알고리즘을 개발했다. 개발한 스트림 암호 알고리즘은 휴대폰과 같은 무선 인터넷 환경과 DRM와 같은 실시간처리가 필요한 분야와 무선 센서 네트워크, RFID 등과 같은 물리적 환경이 매우 제한적인 응용에 사용할 목적으로 소프트웨어 및 하드웨어 구현이 쉬운 스트림 암호이다. 개발한 스트림 암호의 구성은 277비트 산술 쉬프트 레지스터(Arithmetic Shift Register)[8]를 통해 생성된 의사랜덤수열을 SHACAL-2 알고리즘으로 스트림 키를 생성하고 생성된 키와 평문 128비트는 단순히 XOR연산을 수행하지 않고, 기밀성과 무결성이 우수한 암호 운영모드인 CFB(Cipher FeedBack)를 적용하고 최종적으로 표백(Whitening) 처리과정을 수행하여 128비트 암호문을 만든다. 개발한 스트림 암호는 5단계 파이프라인으로 구현했으며 Worst Case에서 Max Frequency는 33.34MHz (4.27Gbps)의 빠른 성능을 보여주었다. 이는 무선 인터넷과 센서 네트워크 및 DRM 환경의 속도를 충분히 만족함을 보여준다.

본 논문의 구성은 2장에서 SHACAL-2의 현제까지 안전성에 대한 문제점과 간략한 SHACAL-2 알고리즘을 설명하고, 3장에서 제안한 스트림 암호 알고리즘과 설계 목표를 설명하고, 4장에서 하드웨어 구현을 자세히 설명하고, 5장에서 스트림 암호의 하드웨어 구현 결과를 설명하고, 6장에서 개발한 스트림 암호 안전성에 대해 분석하고, 마지막으로 결론으로 끝맺는다.

## 2. 관련연구

SHACAL-2는 2000년 1월 유럽에서 시작된 NESSIE (New European Schemes for Signatures, Integrity and Encryption)[9] 프로젝트에서 블록 암호 분야에 제안된 256비트 암호 알고리즘으로 2003년에 3개(MISTY1[10], Camellia[11], AES[12])의 다른 블록 암호와 함께 256비트 암호로 선정되었다. SHACAL-2 알고리즘은 국제 표준 해시 알고리즘인 SHA-2 [13]의 압축함수를 기반으로 64라운드를 진행하고

라운드 마다 32비트 키를 사용해 총 2048비트 키를 사용하는 비대칭 Feistel 구조를 이루고 있어 암호와 복호가 다른 블록 암호 알고리즘이다. 현재까지 SHACAL-2의 안전성에 관한 분석은 많이 없지만 발표된 가장 효율적이고 성능이 좋은 분석 방법으로는 Biryukov 등의 2차 차분-충돌 분석(Second-Order Differential Collisions)[14]으로 SHACAL-2에 48라운드를 분석하였고, Fleischmann 등의 연관키 Boomerang 분석(Related-Key Boomerang)[15]은 메모리를 사용하지 않으면서 39라운드까지 분석했으며, Wang의 연관키 Rectangle 분석(Related-Key Rectangle)[16]은 43라운드 까지 분석 했으며, 홍석희 등의 불능 차분 분석(Impossible Differential)[17]은 30라운드 공격이며, 신영섭 등의 차분 선형 분석(Differential-Linear)[18]은 32라운드까지 공격했다.

이와 같은 분석 방법들은 축소된 라운드까지의 전수조사보다 좋은 성능으로 SHACAL-2를 공격하지만 전체 64라운드 분석이 아니라 축소된 라운드 공격 방법이다. 그리고 2차 차분-충돌 공격, 연관키 Boomerang 공격, 연관키 Rectangle 공격, 불능 차분 공격, 차분 선형 공격 등은 단순한 공격이 아니라 2가지의 기본 공격을 결합한 복합공격 방법이며, 반드시 선택 평문(Chosen Plaintexts) 또는 고정된 연관키 선택 평문(Related-Key Chosen Plaintexts)을 사용한 공격이다. 그래서 아직까지는 SHACAL-2에 대한 전체 64라운드 공격이 이루어지지 않았으므로 안전성에 대한 문제가 없는 것으로 추정된다.

지금부터는 간략히 SHACAL-2 알고리즘을 설명한다. SHACAL-2의 256비트 평문 P는 8개의 32비트로 나누어 처리하며 $P = A_0 \| B_0 \| C_0 \| D_0 \| E_0 \| F_0 \| G_0 \| H_0$이고 64라운드 진행 후 256비트 암호문 $C = A_{64} \| B_{64} \| C_{64} \| D_{64} \| E_{64} \| F_{64} \| G_{64} \| H_{64}$가 된다. i번째 라운드의 암호 진행은 다음과 같다.

$$T^1_{i+1} = H_i + \Sigma_1(E_i) + ch(E_i, F_i, G_i) + Y_i + K_i$$
$$T^2_{i+1} = \Sigma_0(A_i) + maj(A_i, B_i, C_i)$$
$$H_{i+1} = G_i$$
$$G_{i+1} = F_i$$
$$F_{i+1} = E_i$$
$$E_{i+1} = D_i + T^1_{i+1}$$
$$D_{i+1} = C_i$$
$$C_{i+1} = B_i$$
$$B_{i+1} = A_i$$
$$A_{i+1} = T^1_{i+1} + T^2_{i+1} \quad (1)$$

(수식-1)에서 i는 0 - 63까지이고 +는 32비트 덧셈 연산이며, $K_i$는 32비트 라운드 키이고 $Y_i$는 32비트 라운드 상수이고, $T^1$, $T^2$는 32비트 임시 저장 메모리이다. i라운드 암호 진행에서 사용된 함수의 정의는 다음과 같다.

$$ch(X, Y, Z) = (X \& Y) \hat{} (\tilde{}X \& Z)$$
$$maj(X, Y, Z) = (X \& Y) \hat{} (X \& Z) \hat{} (Y \& Z)$$
$$\Sigma_0(X) = ROTR_2(X) \hat{} ROTR_{13}(X) \hat{} ROTR_{22}(X)$$
$$\Sigma_1(X) = ROTR_6(X) \hat{} ROTR_{11}(X) \hat{} ROTR_{25}(X) \quad (2)$$

(수식-2)에서 &는 32비트 AND연산이고 ^는 32비트 XOR연산이며 ~는 32비트 NOT연산이다. 그리고 $ROTR_i$는 32비트 값 X를 i비트만큼 오른쪽 순환(Rotation) 연산이다.

SHACAL-2에서 사용되는 라운드 키는 최소 128비트에서 최대 512비트를 사용하여 라운드마다 32비트 1개씩 총 2048비트를 만들어 사용한다. 만일 키 스케줄링에 사용되는 키가 512비트 보다 작을 경우 0으로 512비트까지 채워(padding) 키 스케줄링 한다. 512비트 키 $K = K_0, K_1, \ldots, K_{15}$를 다음과 같은 알고리즘으로 확장한다.

$$K_i = \sigma_1(K_{i-2}) + K_{i-7} + \sigma_0(K_{i-15}) + K_{i-16} \quad (3)$$

(수식-3)에서 i는 16 - 63까지 이며 키 스케줄링 알고리즘에서 사용된 함수의 정의는 다음과 같다.

$$\sigma_0(X) = ROTR_{17}(X) \hat{} ROTR_{19}(X) \hat{} SR_{10}(X)$$
$$\sigma_1(X) = ROTR_7(X) \hat{} ROTR_{18}(X) \hat{} SR_3(X) \quad (4)$$

(수식-4)에서 $SR_i$는 32비트 값 X를 i비트만큼 오른쪽 시프트(Shift) 연산이다.

## 3. 스트림 암호 알고리즘 설계

기밀성과 무결성이 우수하면서 실시간 처리가 필요한 분야와 제한적인 하드웨어 응용인 WSN의 구현 등에 적용할 128비트 출력 스트림 암호 하드웨어 구현을 위한 고려사항은 수행속도와 하드웨어 면적이다.

### 3.1 설계 고려 사항

• 수행속도는 SHACAL-2의 각 라운드 전체를 파

이프라인 구현기술로 수행속도를 향상할 수 있지만 개발한 스트림 암호는 암호운영모드를 적용하므로 전체 파이프라인이 아닌 8라운드 단위로 나누어진 파이프라인을 적용했다. 그리고 파이프라인 단계를 8라운드로 나눈 이유는 SHACAL-2 알고리즘이 32비트 단위 8개로 나누어 라운드를 진행하며 전체 256비트가 확산되기 위해서는 최소 8라운드 진행이 필요하기 때문이다.

• 하드웨어 면적을 최소로 만들기 위해 안전성에 문제가 없다면 라운드 수를 줄이는 방법이 최선이다. 그래서 SHACAL-2의 라운드 수를 반으로 줄인 32라운드를 적용했다. 안전성에 대해서는 6장에서 자세히 설명한다. 그리고 하드웨어 구현 시 메모리가 가장 많은 면적을 차지한다. 그래서 개발한 스트림 암호는 생성된 32비트 32개의 라운드 키를 메모리에 저장한 후 사용하는 방식이 아니라 제어신호와 파이프라인 단계에 따라 그때그때 생성하는 방식으로 구현했다.

### 3.2 세부 설계 내용

그림 1은 개발한 스트림 암호의 암호화 과정을 표현한 것으로 사용자가 입력한 각각 128비트 비밀 키와 IV를 사용하여 4.2절의 32비트 32개의 라운드 키를 생성한 다음, ASR-277은 128비트 IV를 이용하여 초기화 한 후 ASR-277을 수행하여 생성된 277비트 중 하위 256비트를 사용하여 SHACAL-2 24라운드를 적용한다. 24라운드 결과 값 256비트는 마지막 표백을 위해 128비트로 나누어 XOR연산을 수행하여 저장하고 SHACAL-2를 32라운드까지 진행한다. 32라운드 결과 256비트를 128비트로 나누어 XOR연산을 수행한 후 128비트 스트림과 128비트 평문을 XOR연산을 수행하고 이 결과는 다음번 25-32라운드 키 생성을 위한 값으로 사용된다. 최종적인 128비트 암호문은 128비트 표백 값과 XOR연산을 수행한 값이 된다. 그림 1은 암호화 과정만을 표현한 것으로 기밀성과 무결성을 위해 CFB모드를 적용하므로 개발한 스트림 암호는 암호와 복호가 서로 다른 알고리즘을 보인다. 자세한 내용 4.4절에서 설명한다.

개발한 스트림 암호는 Visual Studio 2010 C 컴파일러를 사용하여 알고리즘 테스트용 프로그램을 만들었고 암호와 복호가 정상적으로 수행하는 것을 확인했다.

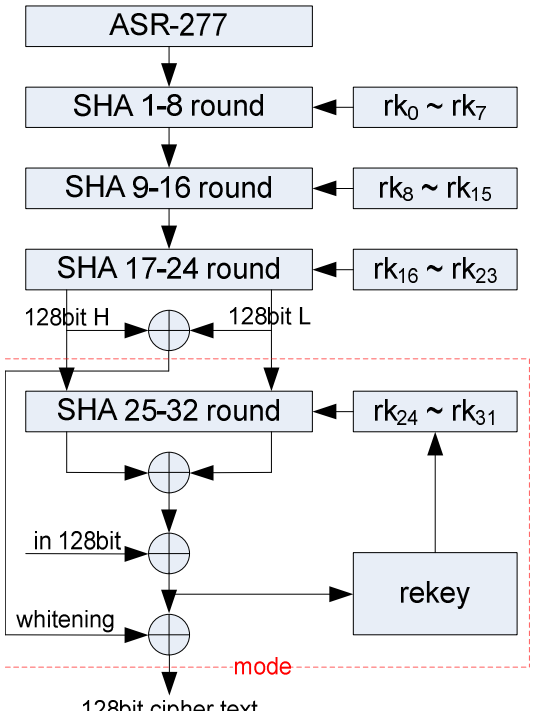


그림 1. SHACAL-2를 이용한 스트림 암호 흐름도

## 4. 하드웨어 구현

그림 2는 개발한 스트림 암호의 5단계 파이프라인으로 구성된 모습을 그림으로 표현했다. Round key look ahead와 rekey, mode 블록을 제외한 사각형 박스로 표시된 블록은 256비트 D-latch를 포함한 순차회로이다. 스트림 암호의 진행과정은 1비트 신호 입력으로 clk, start, d1 - d5, mode이며 32비트 단위 128비트 입력은 각각 K0 - K3과 IV0 - IV3이고, 32비트 단위 128비트 평문 in0 - in3을 입력 받아 5단계 파이프라인을 통과한 후 32비트 단위 128비트 암호문 out0 - out3을 만든다.

다음 절부터 그림 2에 있는 각 블록의 상세한 내용을 진행 순서대로 상세히 설명한다.

### 4.1 ASR-277

PRNG로 사용할 수 있는 산술 쉬프트 레지스터(ASR)는 GF($2^n$)상에서 0이 아닌 초기 값에 0 또는 1이 아닌 임의의 수 D를 곱하는 수열로 정의한다. ASR의 i번째 값(상태) $ASR^i$는 $ASR^0 \cdot D^i$가 된다. $D^k$ = 1이 되는 t가 $t = 2^n - 1$로 유일하게 되는 기약 다항식(Irreducible Polynomial)이 ASR의 특성다항식

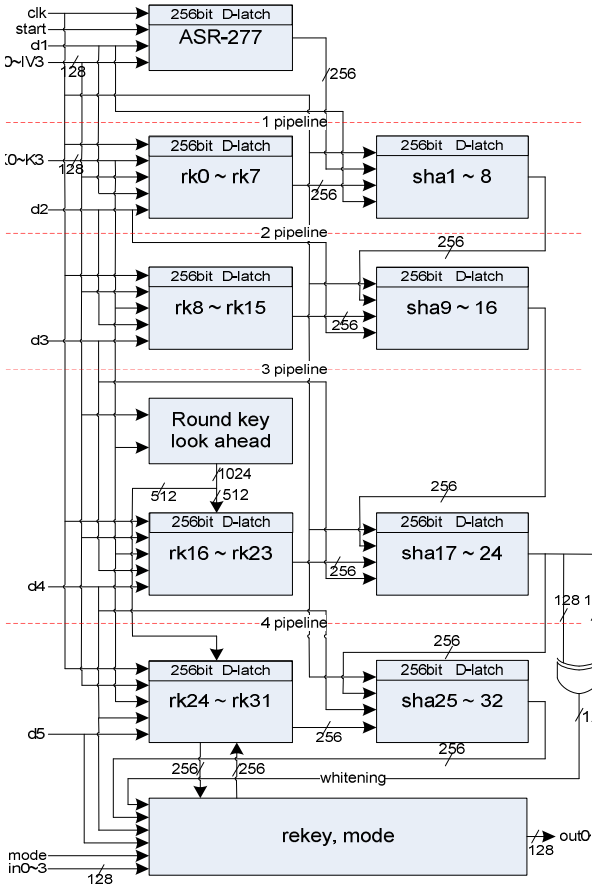


그림 2. SHACAL-2를 이용한 스트림 암호 전체 하드웨어 구성도

(Characteristic Polynomial)이며, ASR의 주기는 $2^n$ - 1로 최대 주기를 가진다. 그리고 ASR의 선형 복잡도는 기존의 LFSR의 선형 복잡도 보다 높아서 안전도가 높고, 다음 상태 갱신을 위한 연산이 32비트 단위로 처리되므로 수행속도도 기존의 LFSR보다 빠르다.

개발한 스트림 암호에서는 GF($2^{277}$)상에서 특성다항식은 16진수로 0x00200000 0x00000001 0x00000001 0x00000001 0x00000001 0x00000001 0x00000001 0x00000002 0x00000015, D = $2^{21}$을 적용한다. ASR-277 블록 알고리즘이 그림 3이다.

그림 3에서 ASR0 - ASR8은 32비트 단위 D-latch이며, 최상위 ASR8은 21비트만 유효한 값이 된다. 그리고 >>는 32비트 ASR을 오른쪽 시프트 연산이고 <<는 32비트 ASR을 왼쪽 시프트 연산이다. 그림 2의 ASR-277 블록의 제어 신호는 start신호가 0 그리고 d1이 1일 때 128비트 IV0 - IV3을 사용하여 ASR-277 블록을 초기화 시키고, start신호가 1 그리고 d1이 0이 되면 clk 신호에 따라 계속해서 다음상태로 갱신된다.

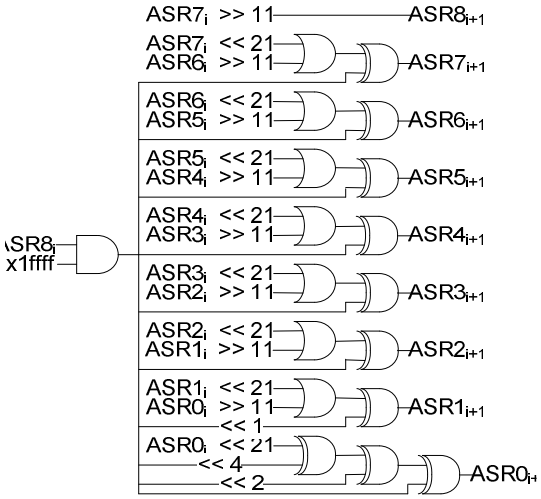


그림 3. ASR-277 블록 알고리즘

ASR-277블록이 스트림 암호의 첫 번째 파이프라인이 되고 clk 신호에 따라 새롭게 생성된 277비트 중 상위 21비트를 제외한 256비트가 sha1-8 블록의 입력으로 사용 된다.

### 4.2 라운드 키 생성

개발한 스트림 암호에서는 SHACAL-2를 32 라운드 적용하므로 32 비트 라운드 키 32개(rk0-rk31)가 필요하다. 32개의 라운드 키는 메모리에 저장 후 사용하는 것이 아니라 제어입력 신호에 따라 라운드 키를 생성하는 방법을 사용했다. 라운드 키 생성은 (수식-3)의 알고리즘을 그대로 적용하여 생성하지만 그림 2에서와 같이 4번에 나누어 단계별 라운드 키 8개씩 각각의 라운드 키 생성블록에서 제어 신호에 의해 생성된다. 그리고 블록 rk0-7, rk8-15, rk16-23에서 생성된 라운드 키는 암호 또는 복호가 끝날 때까지 고정된 라운드 키를 생성하지만, 블록 rk24-31은 제어신호에 따라 계속해서 이전 값과 현재 값을 가지고 갱신되는 라운드 키를 생성한다.

rk0-7블록의 제어신호 입력은 clk, d1, d2이고 8개의 라운드 키 생성을 위한 128비트 비밀 키(K0 - K3)와 128비트 IV(IV0 - IV3)를 입력으로 받는다. 제어신호 d1이 0 그리고 d2가 1일 때 clk 신호에 따라 고정된 라운드 키(rk0 - 7)가 생성된다. 생성 알고리즘은 블록 입력 값 K0 - K3 연속해서 IV0 - IV3을 라운드 키 rk0 - rk7로 각각 순서대로 생성한다.

rk8-15블록의 제어신호 입력은 clk, d2, d3이고 8개의 라운드 키 생성을 위한 비밀키(K0 - K3)와 IV (IV0 - IV3)를 입력으로 받는다. 제어신호 d2이 0 그리고 d3가 1일 때 clk 신호에 따라 고정된 라운드 키 (rk8 - 15)를 만든다. SHACAL-2에서는 512비트보다 작은 비밀 키 일 경우 0으로 512비트까지 채우지만, rk8-15블록은 0으로 채우지 않고 (수식-5)와 같은 알고리즘으로 8개의 라운드 키를 생성한다.

$$\begin{aligned} rk8 &= K0 \hat{} IV0 \\ rk9 &= K1 \hat{} IV1 \\ rk10 &= K2 \hat{} IV2 \\ rk11 &= K3 \hat{} IV3 \\ rk12 &= K0 \hat{} \tilde{}IV0 \\ rk13 &= K1 \hat{} \tilde{}IV1 \\ rk14 &= K2 \hat{} \tilde{}IV2 \\ rk15 &= K3 \hat{} \tilde{}IV3 \end{aligned} \quad (5)$$

그림 2에서 라운드 키 생성 블록 중간에 Round key look ahead블록이 있다. 이 블록은 rk16-23와 rk24-31블록에서 생성할 각각 8개의 라운드 키를 위해 (수식-3)의 라운드 키 확장 알고리즘의 $\sigma_0(X)$와 $\sigma_1(X)$를 미리 계산하는 조합회로이다. 입력으로는 비밀키(K0 - K3)와 IV(IV0 - IV3)이며, 출력은 rk16-23 블록에 $\sigma_0(X)$와 $\sigma_1(X)$ 각각 8개의 512비트와 rk24-31 블록에 $\sigma_0(X)$와 $\sigma_1(X)$ 각각 8개의 512비트 그래서 총 1024비트를 출력한다. rk16-23블록 입력을 위한 512비트 출력 알고리즘은 (수식-6)과 같다.

$$\begin{aligned} \sigma_0 16 &= \sigma_0(K1) \\ \sigma_1 16 &= \sigma_1(K2 \hat{} \tilde{}IV2) \\ \sigma_0 17 &= \sigma_0(K2) \\ \sigma_1 17 &= \sigma_1(K3 \hat{} \tilde{}IV3) \\ \sigma_0 18 &= \sigma_0(K3) \\ \sigma_1 18 &= \sigma_1(\sigma_1 16 + (K1 \hat{} IV1) + \sigma_0 16 + K0) \\ \sigma_0 19 &= \sigma_0(IV0) \\ \sigma_1 19 &= \sigma_1(\sigma_1 17 + (K2 \hat{} IV2) + \sigma_0 17 + K1) \\ \sigma_0 20 &= \sigma_0(IV1) \\ \sigma_1 20 &= \sigma_1(\sigma_1 18 + (K3 \hat{} IV3) + \sigma_0 18 + K2) \\ \sigma_0 21 &= \sigma_0(IV2) \\ \sigma_1 21 &= \sigma_1(\sigma_1 19 + (K0 \hat{} \tilde{}IV0) + \sigma_0 19 + K3) \\ \sigma_0 22 &= \sigma_0(IV3) \\ \sigma_1 22 &= \sigma_1(\sigma_1 20 + (K1 \hat{} \tilde{}IV1) + \sigma_0 20 + IV0) \\ \sigma_0 23 &= \sigma_0(K0 \hat{} IV0) \\ \sigma_1 23 &= \sigma_1(\sigma_1 21 + (K2 \hat{} \tilde{}IV2) + \sigma_0 21 + IV1) \end{aligned} \quad (6)$$

$rk^i$24-31블록 입력을 위한 512비트 출력 알고리즘은 (수식-7)과 같다.

$$\begin{aligned} \sigma_0 24 &= \sigma_0(K1 \hat{} IV1) \\ \sigma_1 24 &= \sigma_1(\sigma_1 22 + (K3 \hat{} \tilde{}IV3) + \sigma_0 22 + IV2) \\ \sigma_0 25 &= \sigma_0(K2 \hat{} IV2) \\ \sigma_1 25 &= \sigma_1(\sigma_1 23 + (\sigma_1 16 + (K1 \hat{} IV1) + \sigma_0 16 + K0) \\ &\quad + \sigma_0 23 + IV3) \\ \sigma_0 26 &= \sigma_0(K3 \hat{} IV3) \\ \sigma_1 26 &= \sigma_1(\sigma_1 24 + (\sigma_1 17 + (K2 \hat{} IV2) + \sigma_0 17 + K1) \\ &\quad + \sigma_0 24 + (K0 \hat{} IV0)) \\ \sigma_0 27 &= \sigma_0(K0 \hat{} \tilde{}IV0) \\ \sigma_1 27 &= \sigma_1(\sigma_1 25 + (\sigma_1 18 + (K3 \hat{} IV3) + \sigma_0 18 + K2) \\ &\quad + \sigma_0 25 + (K1 \hat{} IV1)) \\ \sigma_0 28 &= \sigma_0(K1 \hat{} \tilde{}IV1) \\ \sigma_1 28 &= \sigma_1(\sigma_1 26 + (\sigma_1 19 + (K0 \hat{} \tilde{}IV0) + \sigma_0 19 + \\ &\quad K3) + \sigma_0 26 + (K2 \hat{} IV2)) \\ \sigma_0 29 &= \sigma_0(K2 \hat{} \tilde{}IV2) \\ \sigma_1 29 &= \sigma_1(\sigma_1 27 + (\sigma_1 20 + (K1 \hat{} \tilde{}IV1) + \sigma_0 20 + \\ &\quad IV0) + \sigma_0 27 + (K3 \hat{} IV3)) \\ \sigma_0 30 &= \sigma_0(K3 \hat{} \tilde{}IV3) \\ \sigma_1 30 &= \sigma_1(\sigma_1 28 + (\sigma_1 21 + (K2 \hat{} \tilde{}IV2) + \sigma_0 21 + \\ &\quad IV1) + \sigma_0 28 + (K0 \hat{} \tilde{}IV0)) \\ \sigma_0 31 &= \sigma_0(\sigma_1 16 + (K1 \hat{} IV1) + \sigma_0 16 + K0) \\ \sigma_1 31 &= \sigma_1(\sigma_1 29 + (\sigma_1 22 + (K3 \hat{} \tilde{}IV3) + \sigma_0 22 + \\ &\quad IV2) + \sigma_0 29 + (K1 \hat{} \tilde{}IV1)) \end{aligned} \quad (7)$$

(수식-6), (수식-7)의 알고리즘 왼쪽의 $\sigma_0N$, $\sigma_1N$은 (N은 선 번호) 메모리가 아닌 32비트 선(wire)이다.

rk16-23블록의 제어신호 입력은 clk, d3, d4이고 8개의 라운드 키 생성을 위한 비밀키(K0 - K3)와 IV (IV0 - IV3)를 입력 받고 Round key look ahead 블록으로부터 512비트를 입력 받는다. 제어신호 d3이 0 그리고 d4가 1일 때 clk 신호에 따라 고정된 라운드 키(rk16 - 23)를 만든다. 알고리즘은 (수식-8)과 같다.

$$\begin{aligned} rk16 &= \sigma_1 16 + (K1 \hat{} IV1) + \sigma_0 16 + K0 \\ rk17 &= \sigma_1 17 + (K2 \hat{} IV2) + \sigma_0 17 + K1 \\ rk18 &= \sigma_1 18 + (K3 \hat{} IV3) + \sigma_0 18 + K2 \\ rk19 &= \sigma_1 19 + (K0 \hat{} \tilde{}IV0) + \sigma_0 19 + K3 \\ rk20 &= \sigma_1 20 + (K1 \hat{} \tilde{}IV1) + \sigma_0 20 + IV0 \\ rk21 &= \sigma_1 21 + (K2 \hat{} \tilde{}IV2) + \sigma_0 21 + IV1 \\ rk22 &= \sigma_1 22 + (K3 \hat{} \tilde{}IV3) + \sigma_0 22 + IV2 \end{aligned}$$

$$rk23 = \sigma_1 23 + (\sigma_1 16 + (K1 \hat{} IV1) + \sigma_0 16 + K0) + \sigma_0 23 + IV3 \quad (8)$$

$rk^i 24$-31블록의 제어신호 입력은 clk, d4, d5이고 8개의 라운드 키 생성을 위한 비밀키(K0 - K3)와 IV(IV0 - IV3)를 입력 받고 Round key look ahead 블록으로부터 512비트를 입력 받는다. 제어신호 d4가 0 그리고 d5가 1일 때 clk 신호에 따라 라운드 키($rk^i 24$ - 31)를 만든다. 그리고 d4가 0 그리고 d5가 0이면 clk 신호에 따라 매번 새로 갱신된 8개의 라운드 키를 생성한다.

$$rk^0 24 = \sigma_1 24 + (\sigma_1 17 + (K2 \hat{} IV2) + \sigma_0 17 + K1) + \sigma_0 24 + (K0 \hat{} IV0)$$
$$rk^0 25 = \sigma_1 25 + (\sigma_1 18 + (K3 \hat{} IV3) + \sigma_0 18 + K2) + \sigma_0 25 + (K1 \hat{} IV1)$$
$$rk^0 26 = \sigma_1 26 + (\sigma_1 19 + (K0 \hat{} \tilde{}IV0) + \sigma_0 19 + K3) + \sigma_0 26 + (K2 \hat{} IV2)$$
$$rk^0 27 = \sigma_1 27 + (\sigma_1 20 + (K1 \hat{} \tilde{}IV1) + \sigma_0 20 + IV0) + \sigma_0 27 + (K3 \hat{} IV3)$$
$$rk^0 28 = \sigma_1 28 + (\sigma_1 21 + (K2 \hat{} \tilde{}IV2) + \sigma_0 21 + IV1) + \sigma_0 28 + (K0 \hat{} \tilde{}IV0)$$
$$rk^0 29 = \sigma_1 29 + (\sigma_1 22 + (K3 \hat{} \tilde{}IV3) + \sigma_0 22 + IV2) + \sigma_0 29 + (K1 \hat{} \tilde{}IV1)$$
$$rk^0 30 = \sigma_1 30 + (\sigma_1 23 + (\sigma_1 16 + (K1 \hat{} IV1) + \sigma_0 16 + K0) + \sigma_0 23 + IV3) + \sigma_0 30 + (K2 \hat{} \tilde{}IV2)$$
$$rk^0 31 = \sigma_1 31 + (\sigma_1 24 + (\sigma_1 17 + (K2 \hat{} IV2) + \sigma_0 17 + K1) + \sigma_0 24 + (K0 \hat{} IV0)) + \sigma_0 31 + (K3 \hat{} \tilde{}IV3) \quad (9)$$

(수식-9)는 제일 처음 생성되는 라운드 키($rk^0 24$ - 31)이고 제어입력 신호에 따라 새로 갱신되는 알고리즘은 4.4절에서 자세히 설명한다.

### 4.3 SHACAL-2

SHACAL-2 알고리즘에서 (수식 1)의 라운드 상수 $Y_i$를 제외하고 [그림 2]에서와 같이 4단계로 나누어 각각 8번 SHACAL-2 알고리즘을 수행한다. 첫 번째 블록 sha1-8의 제어입력 신호는 clk, d1이고, ASR-277 블록으로부터 256비트, rk0-7 블록으로부터 라운드 키 8개를 입력으로 받는다. sha1-8블록의 진행 과정은 그림 4와 같다. 그림 4는 1번 SHACAL-2 알고리즘을 수행하는 것으로 이를 8번 직렬로 연결

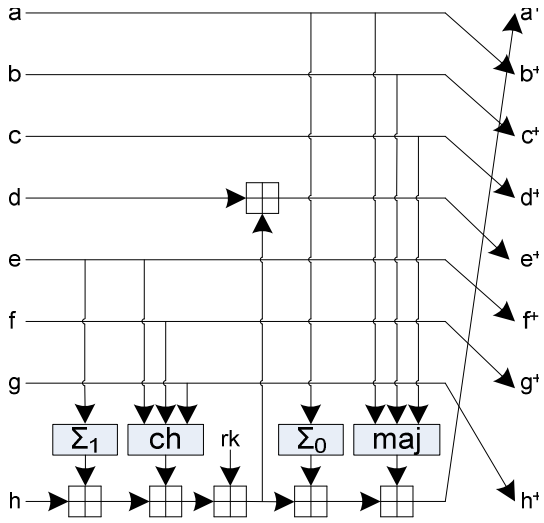


그림 4. SHACAL-2 1라운드 구조도

되어 있다고 생각하면 된다. d1이 0 그리고 clk이 인가되면 sha1-8 블록의 출력이 생성된다.

블록 sha9-16의 제어입력 신호는 clk, d2이고, sha1-8블록으로부터 256비트, rk8-15블록으로부터 라운드 키 8개를 입력으로 받고 d2가 0 그리고 clk이 인가되면 sha9-16블록의 출력이 생성된다. 블록 sha17-24의 제어입력 신호는 clk, d3이고, sha9-16블록으로부터 256비트, rk16-23블록으로부터 라운드 키 8개를 입력으로 받고 d3이 0 그리고 clk이 인가되면 sha17-24블록의 출력이 생성된다. sha17-24블록의 256비트 출력은 그림 2에서 128비트 표백(whitening)을 만든다. 256비트 출력 중 상위 128비트와 하위 128비트를 XOR연산을 통해 128비트 표백을 만들고 이 값은 rekey, mode 블록의 입력으로 사용된다. 표백의 사용은 4.4절에서 자세히 설명한다.

블록 sha25-32의 제어입력 신호는 clk, d4이고, sha17-24블록으로부터 256비트, $rk^i 24$-31블록으로부터 라운드 키 8개를 입력으로 받고 d4이 0 그리고 clk이 인가되면 sha25-32블록의 256비트 출력이 생성된다.

### 4.4 rekey, mode

그림 2의 블록 rekey, mode는 최종적인 128비트 암호문을 만드는 조합회로이다. 입력으로 제어신호 d4, d5, mode, 평문 128비트 in0-3, sha17-24블록으로부터 128비트 표백, sha25-32블록으로부터 256비트, $rk^i 24$-31블록으로부터 8개의 라운드 키를 입력으

로 받아 최종적인 암호문 128비트 out0-3과 새롭게 갱신된 $rk^i$24-31블록의 8개 라운드 키를 생성한다. 만일 평문 입력이 128비트보다 작을 경우 0으로 128비트를 채워서 rekey, mode블록을 진행한다.

128비트 암호문과 갱신된 8개의 라운드 생성 알고리즘은 (수식 10)과 같다.

$$\begin{aligned} &out = sha32H \hat{} sha32L \hat{} in \hat{} whitening \\ &rk^{i+1}24\text{-}27 = sha32H \hat{} sha32L \hat{} in \hat{} rk^i24\text{-}27 \\ &rk^{i+1}28\text{-}31 = \tilde{}(sha32H \hat{} sha32L \hat{} in) \hat{} rk^i28\text{-}31 \end{aligned} \quad (10)$$

(수식 10)의 각 변수들은 128비트이고 위첨자 i = 0이며 out는 128비트 암호문이고 in은 128비트 평문이다. sha32H와 sha32L은 256비트 SHACAL-2를 32라운드 진행 후 각각 상위 128비트와 하위 128비트를 나타낸다. 초기 $rk^0$24-31의 값은 (수식 9)이고 다음 32비트 8개의 라운드 키는 (수식 10)과 같다.

128비트 평문과 갱신된 8개의 라운드 키 생성 알고리즘은 (수식 11)과 같다.

$$\begin{aligned} &out = whitening \hat{} in \hat{} sha32H \hat{} sha32L \\ &rk^{i+1}24\text{-}27 = whitening \hat{} in \hat{} rk^i24\text{-}27 \\ &rk^{i+1}28\text{-}31 = \tilde{}(whitening \hat{} in) \hat{} rk^i28\text{-}31 \end{aligned} \quad (11)$$

(수식 11)에서 out는 128비트 평문이며 in은 128비트 암호문이다. 나머지 변수들은 (수식 10)과 같다.

그림 5는 (수식 10,11)에 대한 rekey, mode블록의 하드웨어 구조를 설명하고 있다.

그림 5에서 최종적인 출력 결과 out는 mode 신호에 의해 2*1 선택기(MUX)를 통해 384(128+256)비트 값을 출력한다. mode가 0이면 암호문과 갱신된 8개의 라운드 키를 출력하고, 1이면 평문과 갱신된 8개의 라운드 키를 출력한다. 이와 같이 스트림 암호이지만 암호와 복호가 다른 이유는 스트림 암호에 우수한 기밀성과 무결성을 위해 암호운영 모드인 CFB모드를 적용했기 때문이다.

그림 6와 그림 7은 CFB모드를 적용한 스트림 암호의 암호와 복호 진행을 그림으로 표현 했다.

그림 6에서 평문과 sha25-32 블록에서 생성된 스트림 키와 XOR연산 결과에 sha17-24 블록에서 생성된 표백 값으로 다시 XOR 연산을 수행 하는 이유는 출력된 암호문은 완전히 노출되는 값이며, 이 값은 CFB모드 적용을 위해 다음 sha25-32블록에 사용될

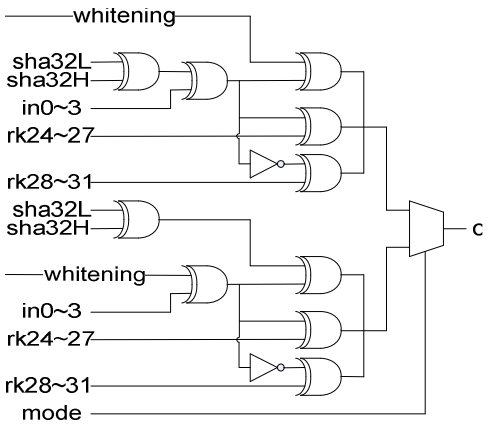


그림 5. rekey, mode 구조도

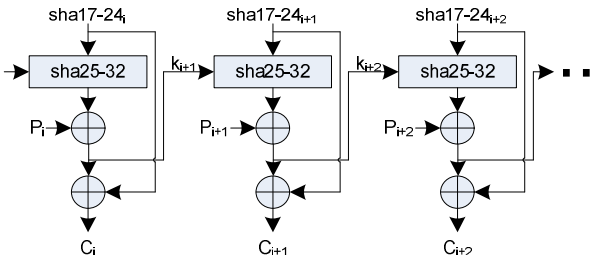


그림 6. 암호 수행 CFB모드 적용 흐름도

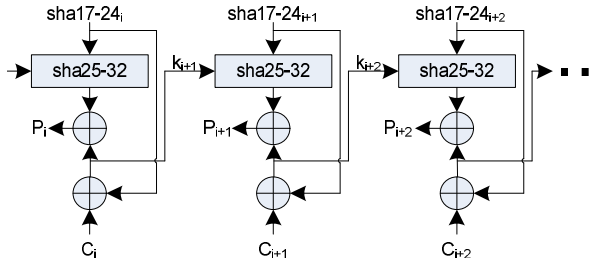


그림 7. 복호 수행 CFB모드 적용 흐름도

8개의 라운드 키 생성 정보를 포함하고 있다. 이 정보를 감추기 위해 sha17-24 블록에서 표백을 만들고 이 표백으로 정보를 희석시킨 후 최종적인 암호문을 만든다. 생성된 암호문으로 평문을 찾기 어렵고, 다음 단계의 sha25-32 블록에서 사용할 8개의 라운드 키를 찾기도 매우 어렵다. 개발한 스트림 암호의 안전성에 대해서는 6장에서 자세히 설명한다.

## 5. 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 II - 32bit FPGA에서

Compile한 결과 Total Logic Elements : 16086 / 68416(24%), Total Registers : 0, Total Pins: 392 / 622 이다. 제어입력 신호인 d1, d2, d3, d4가 1GHz에서 동작하고 있으며, 표 1은 개발된 스트림 암호의 성능을 요약한 표이다. 그리고 Worst Case에서 Max Frequency는 33.34MHz(4.27Gbps)의 빠른 성능을 보여주었다. 이는 무선 인터넷과 센서 네트워크 및 DRM 환경의 속도를 충분히 만족함을 보여준다.

표 1의 참고문헌[19]은 2005년 McLoone가 SHACAL-2 알고리즘 64라운드 전체를 파이프라인으로 구현한 것과 반복적(iterative) 파이프라인으로 구현한 데이터 처리율은 각각 표 1과 같다. 참고문헌[19]과 개발한 스트림 암호의 하드웨어 개발 환경이 다르며, 64라운드 전체 파이프라인 구현은 CTR모드로 구현한 것이므로 본 논문과의 비교는 적절치 않다. 그래서 반복적 파이프라인 구현과 비교에서 하드웨어 실행속도 및 면적이 많이 개선된 것을 볼 수 있다.

AES[20] 또한 5단계 펼쳐진(unrolling) 파이프라인과 반복적 파이프라인 구현 결과이다. 이 역시 본 논문과 반복적 파이프라인과 비교에서 수행 속도가 빠른 결과를 보여주고 있다. 개발한 스트림 암호의 하드웨어 구현은 그림 2의 전체 구성도와 같이 5단계 파이프라인으로 구현했다. 이를 간략히 표현한 것이 그림 8이다.

그림 8에서 ASR은 ASR-277블록을 표시한 것이고, S8, S16, S24, S32는 각각 sha1-8, sha9-16, sha17-24, sha25-32블록을 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.

표 1. 하드웨어 구현 요약

| Algorithm | Device | Data-rate (Gbps) | Area |
|---|---|---|---|
| reference [19] | XC2V4000 | 25.88/0.43 | 23,038 |
| AES[20] | XCV2000E | 20.3/1.82 | 5810, 100B RAM |
| This paper | EP2C70F89618 | 4.27 | 16,086 |

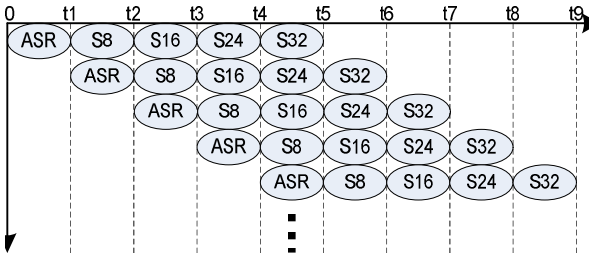


그림 8. 5단계 파이프라인 진행도

## 6. 안전성 분석

개발한 스트림 암호의 안전성 분석은 가장 최신의 SHACAL-2 공격방법에 근거하여 안전성을 설명한다. 그림 2에서 ASR-277블록은 상태변화에 따라 의사랜덤 수열을 계속해서 생성하고 있다. 따라서 ASR-277블록의 전체적인 선형 복잡도 분석 확률을 먼저 구하면, ASR-277블록의 최소 선형 복잡도는 277비트이다. 즉 554개 출력 값을 알면 ASR-277블록의 모든 상태를 분석할 수 있다. 1/554의 확률은 약 $2^{-9}$이다. 이는 ASR-277블록의 안전성이 약 $2^{-9}$이라고 추정할 수 있다. 그리고 ASR-277블록은 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[20]에도 내성이 있다.

2차 차분-충돌 공격[14]에서 높은 확률의 차분 특성(Differential Characteristics)의 전방(forward)과 후방(backward) 경로(path)를 찾는 것이 매우 중요하고 제일 어려운 일이다. 그래서 본 논문에서는 참고문헌[13]의 차분특성 경로를 가장 보수적인 관점에서 그대로 적용했다. 참고문헌[14]은 48라운드까지 공격하고 있으며, 가장 좋은 차분특성의 후방 경로는 1 - 22라운드까지이다. 확률은 $2^{-28}$이고, 23 - 32라운드까지 전방 경로의 차분특성 확률은 $2^{-66}$이다. 추가로 (수식 10,11)의 가변적인 25 - 32라운드 키의 확률은 현재 사용된 라운드 키와 평문은 고려 사항이 아니며, SHACAL-2의 32라운드 후 생성된 차분 확률을 2차 차분-충돌 공격의 25 - 32라운드 확률을 그대로 적용 했다. 2차 차분-충돌 공격의 25 - 32라운드 확률은 $2^{-21}$이다. 이를 바탕으로 개발한 스트림 암호의 최종 2차 차분-충돌 공격 확률은 $2^{-9} * 2^{-2(28+66)} * 2^{-21} = 2^{-218}$이 된다.

연관키 Boomerang 공격[15]은 암호화 과정을 두 부분으로 나누어($E = E_1 \bullet E_0$) 진행한다. 여기서 $E_0$은 첫 라운드부터 시작하는 암호화 과정을 수행하는 부분이고, $E_1$은 마지막 라운드부터 복호화 과정을 수행하는 부분이다. 참고문헌[15]의 진행과정과 같이 $E_0$와 $E_1$을 구분하는 라운드를 25라운드로 하여 가장 보수적인 관점에서 분석했다. 25라운드까지 $E_0$의 암호화 과정의 차분확률은 $2^{-47}$이고, 32라운드에서 $E_1$의 복호화 과정의 차분 확률은 $2^{-53}$이다. 추가로 (수식 10,11)의 가변적인 25 - 32라운드 키의 연관키 Boomerang 공격 확률은 $2^{-53}$이다. 그래서 개발한 스트림 암호의 최종적인 연관키 Boomerang 공격 확률은 $2^{-9} * 2^{-47} * 2^{2(-53)} * 2^{-47} * 2^{-53} = 2^{-262}$이 된다.

연관키 Rectangle 공격[16]은 연관키와 Rectangle 공격을 결합한 것으로 연속된 2개의 차분 특성을 이용하는 공격이다. 여기서 2개의 차분 특성은 연관키 차분과 다른 하나의 차분 특성을 이용한다. 본 논문에서는 참고문헌[16]의 방식대로 첫 번째 연관키 차분특성은 25라운드까지이고 확률은 $2^{-47}$이며, 두 번째 연관키 차분특성 확률은 32라운드까지로 $2^{-54}$이다. 추가로 (수식 10,11)의 가변적인 25 - 32라운드 키의 연관키 Rectangle 공격 확률은 $2^{-54}$이다. 개발한 스트림 암호의 최종적인 연관키 Rectangle 공격 확률은 $2^{-9} * 2^{-2(47+54)} * 2^{-54} = 2^{-263}$이 된다.

개발한 스트림 암호의 키 크기는 256비트(K0 - K3, IV0 - IV3)이다. 지금까지 분석한 스트림 암호 안전성에 대한 확률은 2차 차분-충돌 공격만 전수조사보다 높다. 그러나 이 분석확률들은 가장 보수적인 관점에서 분석한 확률이며, 특히 개발한 스트림 암호는 SHACAL-2 알고리즘을 기반으로 개발되었지만 앞서 설명한 공격들을 피하기 위해 2가지 안전성을 위한 특성이 추가 되었다. 첫 번째 특성은 SHACAL-2를 공격하기 위해서는 높은 확률의 차분특성과 이 차분특성이 계속해서 유지 또는 확산되는 경로를 추적할 수 있는 다수의 선택평문이 반드시 필요한데, 개발한 스트림 암호는 SHACAL-2의 256비트 입력으로 의사랜덤 수열을 생성하는 ASR을 이용하므로 공격자가 의도한 차분특성을 가지는 입력을 만들기 어렵다. 그리고 두 번째는 SHACAL-2를 공격하는 많은 공격방법들 중 대부분이 연관키 공격이다. 이와 같은 연관키 공격은 약한 라운드 키 생성 알고리즘으로 인해 생성된 키들 사이 특정한 차분특성을 고려한 공격이며 고정된 라운드 키를 적용할 때 공격하는 방법이다. 개발한 스트림 암호는 32라운드를 적용하지만, 25 - 32라운드 키는 현재 생성된 암호문과 현재의 25 - 32라운드 키를 가지고 다시 갱신하는 가변적인 라운드 키를 운영하고 있다. 그래서 25 - 32라운드 사이는 연관키와 관련된 공격을 할 수 없다. 개발한 스트림 암호의 2가지 특성은 SHACAL-2를 공격하는 여러 방법들을 피하기 위한 것이고, 이 특성들 때문에 개발한 스트림 암호는 안전할 것으로 판단된다.

표 2는 참고문헌 [13, 14, 15]에 대한 개발한 스트림 암호의 전체 분석 확률을 요약한 것이다.

마지막으로 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 공격은 (수식 10,11)과 같이 32비트 다음 라운드 키 2개가 변경되고 이는 8라운드(25 - 32라운드) SHACAL-2에 적용되어 256비트 전체로 확산이 일어난다. 그 결과로 다음 128비트 암호문은 복호가 불가능해지고, 이 현상은 계속해서 다음 출력으로 영향을 미친다. 이와 같은 연쇄반응(Propagation) 때문에 우수한 무결성을 보인다. 그리고 데이터 전송과정에서 발생한 오류에 의한 연쇄반응도 발생한다.

블록암호 SHACAL-2에서 생성된 암호문에 CFB 모드를 적용하여 스트림 키를 만드는 방법과 본 논문

표 2. 공격 복잡도 요약

| Attack | Division | Prob | Variable Key | Complexity |
|---|---|---|---|---|
| [14] | backward : 1-22<br>forward : 22-32 | $2^{-28}$<br>$2^{-66}$ | $2^{-21}$ | $2^{-218}$ |
| [15] | $E_0$ : 1-25<br>$E_1$ : 32-25 | $2^{-47}$<br>$2^{-53}$ | $2^{-53}$ | $2^{-262}$ |
| [16] | 1th : 1-25<br>2th : 25-32 | $2^{-47}$<br>$2^{-54}$ | $2^{-54}$ | $2^{-263}$ |

에서 제시한 방법의 차이는 가변적인 라운드 키를 적용할 수 있도록 CFB모드를 조금 변형한 것이다. 이와 같은 가변 키 적용은 SHACAL-2의 라운드 수를 반으로 줄여도 확률적인 안전성에 문제가 없는 것으로 보인다.

## 7. 결 론

5단계 파이프라인으로 구성된 기밀성과 무결성이 우수하며 실시간처리가 가능한 128비트 출력의 스트림 암호를 개발했다. 개발한 스트림 암호는 ASR 277비트와 SHACAL-2를 통해 128비트 스트림을 만들고 이를 CFB모드 적용한 후 표백처리 과정을 통하여 최종적인 128비트 암호문을 만드는 스트림 암호 알고리즘이다.

스트림 암호의 하드웨어 결과는 Worst Case에서 Max Frequency는 33.34MHz(4.27Gbps)의 빠른 성능을 보여주었다. 이는 무선 인터넷과 센서 네트워크 및 DRM 환경의 속도를 충분히 만족함을 보여준다. 그리고 기밀성과 무결성을 모두 만족할 수 있는 스트림 암호 알고리즘 개발에 본 논문은 매우 유용한 아이디어를 제공하고 있다.

추후 연구과제는 실시간 고속 이동 무선 통신 환경에서 기밀성과 무결성이 보장되고 평문과 암호문의 일대일 대응관계를 좀 더 많이 확산시킬 수 있는 스트림 암호 알고리즘의 개발을 목표로 하고 있다.

## 참 고 문 헌


[1] TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks_Ver 1.0, http://discovery.csc.ncsu.edu/software/TinyECC/, 2007.

[2] CC2420 DataSheet, *CC2420 2.4GHz IEEE 802.15.4/ZigBee-ready RF Transceiver*, Chipcon, 2006.

[3] D.J. Bernstein and P. Schwabe, "New AES Software Speed Records," *INDOCRYPT 2008, LNCS 5365*, pp. 322-336, 2008.

[4] European Network of Excellence in Cryptology II, http://www.ecrypt.eu.org/, retrieved 2013.

[5] The eSTREAM Project, http://www.ecrypt.eu.org/stream/project.html, retrieved 2013.

[6] 김원제, 성택영, 이석환, 권기룡, "H.264 Scalable Extension을 위한 비디오 워터마킹 및 암호화 기반의 정보보호 기법," 한국멀티미디어학회논문지, 제15권, 제3호, pp. 299-311, 2012.

[7] H. Handschuh and D. Naccache, *SHACAL: A Family of block Ciphers*, Submission to the NESSIE project, 2002.

[8] 박창수, 조경연, "갈로이 선형 궤환 레지스터의 일반화," 전자공학회논문지, 제43권, 제1호, pp. 1-8, 2006.

[9] NESSIE, Performance of Optimized Implementations of the NESSIE Primitives, https://www.cosic.esat.kuleuven.be/nessie/, retrieved 2013.

[10] Mitsubishi Electric Corporation, A Description of the MISTY1 Encryption Algorithm, Request for Comments (RFC) 2994, http://www.ietf.org/rfc/rfc2994.txt, 2003.

[11] NTT and Mitsubishi Electric Corporation, Camellia, http://info.isl.ntt.co.jp/camellia/, 2003.

[12] AES, Report on the Development of the Advanced Encryption Standard, http://csrc.nist.gov/archive/aes/index.html, retrieved 2013.

[13] NIST, *Secure Hash Standard*, Draft FIPS PUB 180-2, 2001.

[14] A. Biryukov, M. Lamberger, F. Mendel, and I. Nikolic, "Second-Order Differential Collisions for Reduced SHA-256," *ASIACRYPT 2011, LNCS 7073*, pp. 270-287, 2011.

[15] E. Fleischmann, M. Gorski, and S. Lucks, "Memoryless Related-Key Boomerang Attack on 39-Round SHACAL-2," *ISPEC 2009, LNCS 5451*, pp. 310-323, 2009.

[16] G. Wang, "Related-Key Rectangle Attack on 43-Round SHACAL-2," *ISPEC 2007, LNCS 4464*, pp. 33-42, 2007.

[17] S. Hong, J. Kim, G. Kim, J. Sung, C. Lee, and S. Lee, "Impossible Differential Attack on 30-Round SHACAL-2," *INDOCRYPT 2003, LNCS 2904*, pp. 97-106, 2003.

[18] Y. Shin, J. Kim, G. Kim, S. Hong, and S. Lee, "Differential-Linear Type Attack on Reduced Rounds of SHACAL-2," *ACISP 2004, LNCS 3108*, pp. 110-122, 2004.

[19] M. McLoone, "Hardware Performance Analysis of the SHACAL-2 Encryption Algorithm," *Circuits, Devices and Systems, Proceedings, Vol. 152, No. 5,* pp. 478-484, 2005.

[20] G. Saggese, A. Mazzeo, N. Mazzocca, and A. Strollo, "An FPGA-Based Performance Analysis of the Unrolling, Tiling, and Pipelining of the AES Algorithm," *Field Programmable Logic and Applications - FPL 2003, LNCS 2778*, pp. 292-302, 2003.

[21] P. Hawkes and G. Rose, "Guess-and-determine attacks on SNOW," *Selected Areas in Cryptography - SAC 2002, LNCS 2595*, pp. 37-46, 2002.

**김 길 호**

2000년 한국방송통신대학교 전자계산학과 (이학사)
2002년 부경대학교 컴퓨터공학과 (공학석사)
2010년 부경대학교 컴퓨터공학과 (공학박사)

관심분야 : 반도체회로설계, 암호 알고리즘, 컴퓨터 구조

**조 경 연**

1990년 인하대학교 공과대학전자공학과 정보공학전공 (공학박사)
1983~1991년 삼보컴퓨터 기술연구소 책임연구원
1991년~현재 부경대학교 IT융합응용공학과 교수
1991~2001 삼보컴퓨터 기술연구소 비상임기술 고문
1998~현재 에이디칩스 사외이사 겸 비상임기술고문
관심분야 : 전산기구조, 반도체회로설계, 암호 알고리즘