논문 2013-50-1-9

전력분석 공격에 안전한 개선된 스트림 암호 Rabbit

(Enhanced Stream Cipher Rabbit Secure Against Power Analysis Attack)

배 기 석*, 안 만 기**, 박 영 호***, 문 상 재****

(KiSeok Bae, MahnKi Ahn, YoungHo Park, and SangJae Moon)

요 약

최근 유럽연합의 eSTREAM 공모사업에서 소프트웨어 분야에 선정된 Rabbit 알고리듬은 ISO/IEC 18033-4 기술분야에 추가 선정된 스트림 암호이다. 그러나 Rabbit 알고리듬은 구현된 실제 환경에서 발생할 수 있는 전력분석 공격의 취약성이 발견되고, 실제 가능함이 발표되었다. 본 논문에서는 전력분석 공격에 안전한 Rabbit의 구현을 위해 적합한 랜덤 마스킹 및 연산 순서 숨김 기법을 제안한다. 제안한 방어책들은 빠른 수행속도의 장점을 유지하며 24%의 연산시간과 12.3%의 메모리 요구량만이 증가하여 스트림 암호의 방어책으로 적합하다. 8비트 RISC 계열의 AVR 마이크로프로세서(ATmega128L)에 탑재하여 실험한 결과, 전력분석 공격에 안전함을 검증하였다.

Abstract

Recently, stream cipher Rabbit was selected for the final eSTREAM portfolio organized by EU ECRYPT and as one of algorithm in part of ISO/IEC 18033-4 Stream Ciphers on ISO Security Standardization. However, a feasibility of practical power analysis attack to algorithm in experiment was introduced. Therefore, we propose appropriate methods such as random masking and hiding schemes to secure against power analysis attack on stream cipher Rabbit. We implement the proposed method with increment of 24% operating time and 12.3% memory requirements due to maintaining a high-speed performance. We use a 8-bit RISC AVR microprocessor (ATmega128L chip) to implement our method for practical experiments, and verify that stream cipher Rabbit with our method is secure against power analysis attack.

Keywords : 스트림 암호 Rabbit, 전력분석 공격, 마스킹 기법, 숨김 기법, 이동 ad-hoc 네트워크

Ⅰ. 서 론

통신 기술의 비약적인 발전으로 인하여 원하는 정보를 실시간으로 제공되는 유비쿼터스 시대가 도래하고 있으며, 이러한 기능은 매우 큰 규모의 ad-hoc이 형성되어야 사용될 수 있다. 이러한 ad-hoc 망을 이동 ad-hoc 네트워크라 한다. 이동 ad-hoc 네트워크를 구성하는 노드는 단말 장치의 낮은 연산 능력과 저장 공간의 부족등의 한계성을 고려하여 설계되는데, 개인의 정보의 중요성이 높아짐에 따라 전반적인 보안 체계가 요

* 정회원, **** 평생회원, 경북대학교 전자전기컴퓨터학부
(Kyungpook National University)
** 정회원, 국방기술품질원
(Defense Agency for Technology and Quality)
*** 정회원, 경북대학교 산업전자공학과
(Kyungpook National University)
※ 이 논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임(2012R1A1A4A01002603)
※ 이 논문은 2012년도 경북대학교 학술연구비에 의하여 연구되었음
접수일자: 2012년8월23일, 수정완료일: 2013년1월9일

구된다. 따라서 이러한 특성에 적합하게 작은 연산 량만으로 고속의 암·복호화가 가능한 스트림 암호에 대한 관심이 커지고 있다.

최근 여러 암호 시스템의 안전성 분석 기법 중 하나로서, 실질적인 구현 과정에서 얻어지는 정보들을 이용하는 기법인 부채널 분석(side-channel cryptanlaysis)에 대한 연구가 활발히 진행 중이다. Kocher에 의해 소개된 부채널 분석의 하나인 전력분석 공격(power analysis attacks)[1]은 암호 알고리듬을 설계할 때 고려되지 못한 부가적인 정보의 누출에 의해 비밀 정보를 알아내는 방식으로, 이동 ad-hoc 네트워크 상의 단말장치에 구현될 스트림 암호 역시 공격의 대상이 될 수 있다. 이미 기존에 여러 장치에서 사용되고 있는 스트림 암호 RC4, E0, A5 알고리듬들에 대한 공격들은 소개되어졌다[2].

유럽연합(EU)의 eSTREAM 공모사업[3]의 소프트웨어 분야(Profile 1)에서 최종 선정된 알고리듬 중 하나이며 최근 ISO/IEC 18033-4 기술분야에 스트림 암호 표준[4]으로 선정된 스트림 암호 Rabbit[5]은 전력분석 공격에 대한 안전성 분석 결과가 중간수준으로 어느 정도 안전하다고 이론적으로 평가되었다[6]. 하지만 최근 실험적으로 전력분석 공격이 가능함이 발표되었다[7].

본 논문은 스트림 암호 Rabbit의 전력분석 공격에 대한 안전성을 강화하기 위해서 비밀 키 설정과정과 초기벡터 설정과정에서 선형 연산과정에 대한 랜덤 마스킹 기법과 연산과정의 숨김 기법을 적용하는 것이다. 제안된 방어책은 비밀 키 성분이 노출될 수 있는 초기화단계에서 Rabbit 알고리듬의 내부변수를 알아낼 수 없도록 전력소모량간의 상관도를 최소화하는 방법이다. 고성능 8비트 RISC 계열의 AVR 마이크로프로세서에 제안된 방법이 적용된 안전한 스트림 암호 Rabbit을 구현하고, 실험을 통하여 전력분석 공격에 대한 안전성을 검증하였다.

Ⅱ. Rabbit 알고리듬

1. Rabbit 알고리듬의 구조

Rabbit 알고리듬은 스웨덴의 M. Boesgaard에 의해 발표된 동기 스트림 암호로, 비밀키와 초기벡터의 초기화, 513 비트 내부변수 갱신, 128비트 키수열 발생 등으로 구성된다[4]. 또한 무선센서 네트워크에 적용될 수 있도록 안전성 및 저전력 특성을 확보하는 연구가 진행되고 있다[8]. 아래와 같이 4번의 과정으로 구성되어 있으며, 비밀키 및 초기벡터 설정과정의 계수변수와 상태변수는 그림 1과 같이 계수 시스템(counter system)과 내부상태천이(next state function) 연산과정을 통해 갱신된다.

1) 비밀키 설정과정(key setup scheme)
입력된 128비트 비밀키를 8개의 부분키로 분할하고 생성된 내부 중간값(상태변수, 계수변수, 올림수)은 계수 시스템과 상태천이함수로 갱신하는 과정을 4회 반복한다. 또한 4회째 계수변수를 마스터 계수변수로 변경한다.

2) 초기벡터 설정과정(initial vector setup scheme)
입력된 64비트 초기벡터(IV)로 비밀키 설정과정에서 생성된 마스터 계수변수를 초기화한 후, 상태변수와 계수변수는 상태천이함수와 계수 시스템을 이용하여 4회 반복적으로 갱신한다.

3) 키수열 생성과정(extraction scheme)
비밀키 설정과정과 초기벡터 설정과정으로 생성된 상태변수간의 XOR 연산으로 128비트 키수열 블록 s_i을 생성한다.

4) 암/복호화 과정(encryption/decryption scheme)
생성된 128비트 키수열 블럭과 입력된 평문 또는 암호문을 XOR 연산으로 암/복호하는 과정이다.

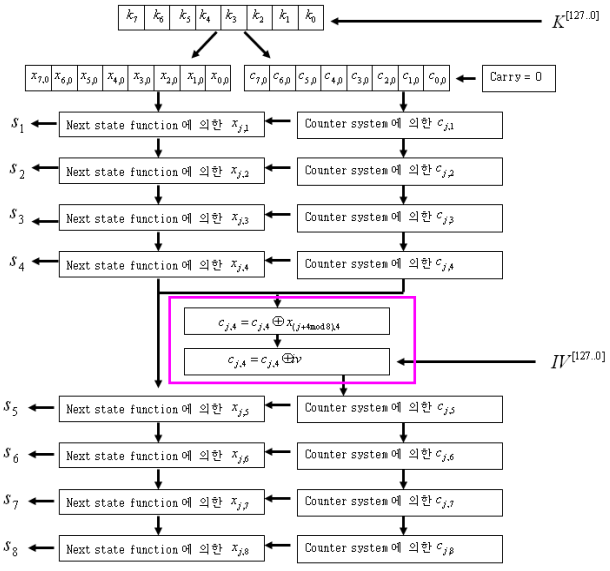

그림 1. 비밀키 설정과정과 초기벡터 설정과정
Fig. 1. Key Setup and IV Setup.

2. Rabbit 알고리듬에 대한 전력분석 공격

전력분석 공격은 알고 있는 초기벡터를 이용하여 실시한다. 4단계의 공격 절차를 차례대로 수행하여 초기벡터 설정과정에 의한 내부 중간값(계수변수, 상태변수, 올림수)을 알아낸 후, 이를 이용하여 비밀키 설정과정의 비밀키를 알아낼 수 있다[7]. 4단계의 공격은 서로 연관성이 있으며 각 단계별로 알아낸 값을 통해 다음 단계의 공격의 진행이 가능하다.

1단계는 비밀키 설정 과정의 결과 x와 초기벡터와의 연산을 대상으로 마스터 계수를 알아내는 것이 목적이다. 마스터 계수변수 생성은 식 (1)으로 역환으로 비밀키를 알아내는 것을 방지하는 역할을 한다. 또한 식 (2)은 초기벡터 분할단계와 계수변수 초기화 단계로 마스터 계수변수 $c_{j,4}$와 분할된 초기벡터가 XOR 연산을 수행하는데, 이 때 해밍무게 모델을 이용한 상관도 전력분석 공격의 대상이 된다. 입력된 초기벡터는 32비트씩 차례로 연산되나 8비트씩 추측하여 총 4회를 수행한다. 이러한 과정을 8회 수행하면 전체 256비트의 계수변수 $c_{j,4}$를 알아낼 수 있다.

$$c_{j4} = c_{j,4} \oplus x_{(j+4 \bmod 8),4} \tag{1}$$

$$\begin{aligned} c_{0,4} &= c_{0,4} \oplus IV^{[31..0]} & c_{1,4} &= c_{1,4} \oplus IV^{[63..48]} \| IV^{[31..16]} \\ c_{2,4} &= c_{2,4} \oplus IV^{[63..32]} & c_{3,4} &= c_{3,4} \oplus IV^{[47..32]} \| IV^{[15..0]} \\ c_{4,4} &= c_{4,4} \oplus IV^{[31..0]} & c_{5,4} &= c_{5,4} \oplus IV^{[63..48]} \| IV^{[31..16]} \\ c_{6,4} &= c_{6,4} \oplus IV^{[63..32]} & c_{7,4} &= c_{7,4} \oplus IV^{[47..32]} \| IV^{[15..0]} \end{aligned} \tag{2}$$

그림 2는 임의의 추측한 계수변수 $c_{0,4}$의 하위 8비트 해밍무게에 대해 1,000개의 전력소모파형을 수집하여 상관도 전력분석 공격을 수행한 결과이다. 전체 256가지의 경우에서 올바르게 추측한 경우의 상관계수(ρ)는 진한 색 파형이며, 나머지 255가지의 경우에서는 회색의 파형이다.

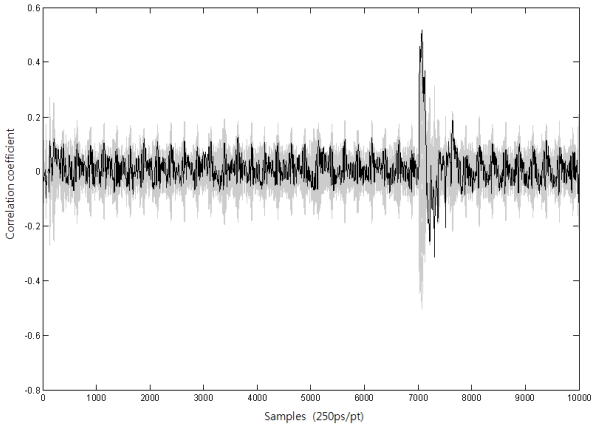

그림 2. 계수변수 $c_{0,4}$의 하위 8비트 상관계수(모든 경우)
Fig. 2. CPA Result (256 Hypotheses).

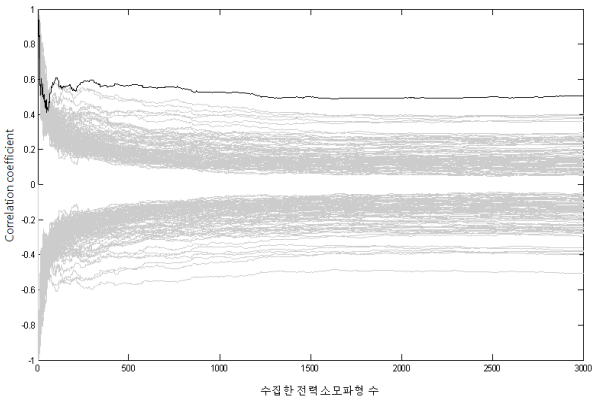

그림 3. 공격에 필요한 전력파형수(가능한 모든 경우)
Fig. 3. Correlation Coefficient for Various Number of Traces-1.

그림 3은 상관도 전력분석 공격시 올바른 경우를 찾을 수 있는 전력소모파형 수를 보여주고 있다. 올바르게 추측한 경우가 검정색이며, 엷은 회색은 그 외 255가지 경우의 상관도이다. 약 90개의 전력소모파형으로도 공격이 가능함을 보여주고 있다.

식 (2) 연산 과정에 대한 1단계 전력분석 공격을 시작으로 그림 1의 S_5를 구하기 위한 계수시스템 연산중의 올림수를 구하는 2단계, 내부상태천이 함수의 입력값을 구하는 3단계, 비밀 키 설정 단계를 역환하여 x_0를 찾아내는 4단계를 거쳐 원래의 비밀키 K를 찾을 수 있다[7]. 앞서 언급한 바와 같이 Rabbit 알고리듬의 특성상 1단계에서 계수변수를 찾지 못한다면 나머지 2~4단계의 1비트 올림수와 256비트의 상태변수를 역추적할 수 없다.

Ⅲ. 기존에 소개된 방어책

1. 마스킹 기법(masking scheme)

마스킹 기법은 실제 센서 노드나 스마트카드에 구현된 알고리듬의 동작시 전력소모량과 공격자가 추정한 중간값의 상관관계를 제거하는 방법이다. 가장 저렴한 비용으로 적용할 수 있어 대칭키 암호시스템에 일반적으로 많이 사용되는 방법이다. 특히, 블록 암호 알고리듬(AES, ARIA, DES)에 많은 연구가 이루어져 왔다[8~11].

소프트웨어적인 마스킹 기법은 일차 전력분석 공격에 대응하는 방법으로 부울 마스킹(boolean masking), 곱셈 마스킹(multiplicative masking), 제곱 마스킹(squaring masking), 역원 테이블을 이용한 마스킹 등이 있다. 그러나 곱셈, 제곱 및 역원 연산을 이용한 방법은 8비트 마이크로프로세서를 이용한 소프트웨어 구현에는 많은 메모리 및 연산량 증가를 가져와 적합하지 않다. 따라서 소프트웨어 구현에 가장 적합한 대응방법은 부울 마스킹 기법(비트별 대수 연산)으로 XOR, AND, OR 연산을 마스킹하는 기법이 있으나, AND, OR 연산은 중간값의 데이터가 드러나는 것을 방지하기 위해 별도의 마스크를 사용해야 한다.

XOR연산은 마스킹된 데이터를 피연산자로 하여 연산한다. 즉, n-비트열 x, y에 랜덤하게 생성된 마스크 m_x, m_y을 이용하여 공격자가 알 수 없는 중간값 $x' = x \oplus m_x$, $y' = y \oplus m_y$을 만들고 새로운 마스크 $m_z = m_x \oplus m_y$이 되도록 $z' = x' \oplus y' = x \oplus y \oplus m_x \oplus m_y$가 연산된다. 여기서 m은 독립적이며, 균일하게 분포된 값이다. 이러한 선형 연산시점에 적용하는 마스킹 기법은 두 개 이상의 랜덤한 마스킹 값을 사용하는 것이 효과적이다[12~13].

2. 연산과정의 숨김 기법(hiding scheme)

측정된 전력소모량의 통계적 기법을 적용하는 공격에서는 수집한 전력소모량을 분산시키거나 잡음을 증가시켜 상관도를 줄이는 방법이 가장 효과적이다. 알고리듬 수준에서 상관관계를 제거하는 방법으로는 알고리듬 연산순서의 랜덤화(random shuffling of operations)하거나 연산시점의 불규칙화(random dummy cycles) 등이 있다.

랜덤 더미 싸이클은 암호 알고리듬의 동작 시점을 랜덤하게 하여 연산시간이 일정하지 않도록 하는 방법으로 C 코드나 기계어(어셈블러) 코드에서 랜덤 지연코드나 대상이 되는 연산의 더미코드를 삽입하는 방법이 있다. 연산순서의 랜덤화는 재동기화와 무관하게 여러 번의 암호 연산 수행시마다 명령어들의 수행순서를 달리하여 전력소모파형간의 상관관계를 줄이는 방법이다. 이는 공격자로 하여금 통계적 분석을 더 어렵게 만들 수 있으며, 전력분석 공격에도 효과적으로 방어할 수 있다. 숨김 기법에 대해서는 대표적인 스트림 암호인 RC4에 적용한 사례도 있다[14].

Ⅳ. 부채널 공격에 안전한 스트림 암호 Rabbit

Rabbit 알고리듬은 초기벡터를 알고 있더라도 키수열 발생기의 마스터 계수변수 $c_{j,4}$와 초기벡터의 XOR 연산 과정에 대한 1단계 전력분석 공격이 실패하면, 최종 비밀키를 알아낼 수 없는 구조를 가지고 있다[7]. 따라서 재동기화에 의해 알고리듬 다시 동작할 때마다 매번 변하는 마스크 값들을 ATmega128L의 16비트 Timer와 Counter 기능을 이용하여 생성한다. 생성된 마스크 값들은 초기 벡터와 계수 변수에 대한 1차, 2차 랜덤 마스킹, 그리고 출력 마스킹으로 처리되어 1단계의 공격 대상인 마스터 계수 변수의 노출을 방지한다. 추가로 연산 순서의 랜덤화와 연산 시점의 불규칙화(랜덤 사이클의 삽입 횟수) 등의 숨김 기법도 함께 구현하였다. 이러한 소프트웨어적인 대응방법은 하드웨어 대응방법에 비해 적은 메모리 면적과 전력소모량을 줄일 수 있는 장점이 있다.

1. 랜덤 부울 마스킹

재동기가 이루어질 때 생성된 마스크씨앗(mask seed)을 이용한 마스킹 값들은 선형 연산과정인 마스터 계수변수 연산단계에 1차 부울 마스킹 기법으로 사용된 후, 초기벡터 설정과정의 초기벡터 분할단계에서 2차 부울 마스킹으로 적용된다. 이들 마스킹 값들은 초기화 단계의 XOR 연산 이후, 5회째 내부상태천이 연산과정의 계수 시스템이 수행하기 전에 복원한다. 이 때 복원값은 마스크씨앗이 포함된 출력 마스킹 값으로 이는 내부상태천이 함수로 호출되기 전에 발생할 수 있는 마스터 계수변수의 노출을 방지함에 있다.

제안하는 마스킹 기법은 재동기가 이루어질 때마다 32비트 8개의 계수변수(256비트)를 가변되는 마스킹 랜덤값과 XOR 연산하여 실제 전력소모량과 공격자의 추정모델간의 상관관계를 제거하는 방법으로, 8개의 가변 마스킹 랜덤값들은 재동기화시 $0 \sim 2^8-1$의 범위내에서 서로 다른 난수이다. 다음 식 (3)과 (4)은 계수변수와 초기벡터에 각각 가변 마스킹 연산하는 과정이다.

$$c_{j4} = (c_{j,4} \oplus Mask[j]) \oplus x_{(j+4 \bmod 8),4} \quad (3)$$

$$c_{0,4} = c_{0,4} \oplus (IV^{[31..0]} \oplus Mask[8])$$

$$c_{1,4} = c_{1,4} \oplus (IV^{[63..48]} \| IV^{[31..16]} \oplus Mask[9])$$

$$c_{2,4} = c_{2,4} \oplus (IV^{[63..32]} \oplus Mask[10])$$

$$c_{3,4} = c_{3,4} \oplus (IV^{[47..32]} \| IV^{[15..0]} \oplus Mask[11])$$
$$c_{4,4} = c_{4,4} \oplus (IV^{[31..0]} \oplus Mask[12]) \quad (4)$$
$$c_{5,4} \oplus (IV^{[63..48]} \| IV^{[31..16]} \oplus Mask[13])$$
$$c_{6,4} \oplus (IV^{[63..32]} \oplus Mask[14])$$
$$c_{7,4} = c_{7,4} \oplus (IV^{[47..32]} \| IV^{[15..0]} \oplus Mask[15])$$

입력된 초기벡터와 마스터 계수변수의 마스킹 기법을 ATmega128L에 적용할 경우, 기존 C코드에서 추가 연산량 및 추가 코드가 요구된다. 먼저 마스터 계수변수에 1차 마스킹 연산시 추가적인 XOR 연산수가 32회(4회 X 8개 계수변수) 증가한다. 또한 분할된 초기벡터에 2차 마스킹시 32회(4회 X 4개 초기벡터 X 2)의

재동기화시 대응방법이 적용된 Rabbit 알고리듬

입력값 : 초기벡터, 계수변수, 상태변수, 마스킹 랜덤값
중간값 : 연산순서가 숨겨진 마스킹된 초기벡터와 계수변수
출력값 : 출력 마스킹된 계수변수 c[j]⊕Maskseed

```
//마스킹 랜덤값 생성
for j=0 to 7 do
    Mask[j] = GenerateRandomNumber(IV, GET_TIMER())⊕Maskseed
for j=8 to 15 do
    Mask[j] = GenerateRandomNumber(IV, GET_TIMER())
end for
Mask[16] = (( mask[8] >> 16 ) | ( mask[10] & 0xFFFF0000 ));
Mask[17] = (( mask[10] >> 16 ) | ( mask[8] & 0xFFFF0000 ))
// 마스킹된 마스터 계수변수의 생성(마스터 계수변수에 1차 가변 마스킹 적용)
for j=0 to 7 do
     master_c[j] = Mask[j] ⊕ c[j] ⊕ x[(j+4)&0x7]
end for
// 분할된 초기벡터에 2차 가변 마스킹 적용
I[0] = IV[31,..,0] ⊕ Mask[8]
I[2] = IV[63,..,32] ⊕ Mask[10]
I[4] = IV[31,..,0] ⊕ Mask[12]
I[6] = IV[63,..,32] ⊕ Mask[14]
I[1] = (( I[0] >> 16 ) | ( I[2] & 0xFFFF0000 )) ⊕ Mask[9] ⊕ Mask[16]
I[3] = (( I[2] << 16 ) | ( I[0] & 0x0000FFFF )) ⊕ Mask[11] ⊕ Mask[17]
I[5] = (( I[0] >> 16 ) | ( I[2] & 0xFFFF0000 )) ⊕ Mask[13] ⊕ Mask[16]
I[7] = (( I[2] << 16 ) | ( I[0] & 0x0000FFFF )) ⊕ Mask[15] ⊕ Mask[17]
// 마스킹된 초기벡터로 마스킹된 마스터 계수변수의 초기화 단계
for j=0 to 7 do
   Select j randomly, only one time from j=0 to 7
   Insert random cycles(n) and selected n randomly
   c[j] = master_c[j] ⊕ I[j]   // 32비트 계수변수
end for
carry = master_carry
// 마스킹된 계수변수 및 초기벡터의 복원 코드 추가
for j=0 to 7 do
   Select j randomly, only one time from j=0 to 7
   Insert random cycles(50-n)
     c[j] = (Mask[j] ⊕ c[j]) ⊕ Mask[j+8]
end for
Return c[j]
```

그림 4. 대응방법이 적용된 비밀키 설정과정과 초기벡터 설정과정
Fig. 4. Countermeasure for Key Setup and IV Setup.

XOR 연산수가 증가하며, 복원하는 과정에서 각 32회씩 64회 증가한다. 총 128회의 부가적인 XOR 연산량이 추가됨에 따라 레지스터와 연산수행 시간이 증가하지만 일차의 전력분석 공격에는 안전해질 수 있다.

2. 연산순서의 랜덤화

비밀키 설정과정의 마스터 계수변수의 연산단계는 입력된 초기벡터로 마스터 계수변수를 알아내었어도, 초기벡터 입력과 무관하게 항상 상태변수와 XOR 연산하므로 전력분석 공격의 대상이 될 수 없다. 따라서 비밀키 설정과정 내의 32비트 계수변수의 연산에는 연산과정의 숨김 기법을 적용할 필요가 없다.

반면, 초기벡터 설정과정의 계수변수의 초기화단계는 식(4)와 같이 8번에 걸쳐 연산이 발생하는데, 이때의 연산 순서 j를 랜덤화하여 동일한 계수변수가 동일한 시간에 수행될 확률이 1/8이 되도록 하였다. 이는 추가적인 메모리 요구량은 없으나, 알고리듬 수행시간이 증가한다.

3. 연산시점의 불규칙화

초기벡터 설정과정의 계수변수 초기화단계에서 8개의 계수변수를 위한 초기화 연산중에 랜덤 더미 싸이클을 삽입하여 불규칙한 연산시점을 구현하였다. 더미 명령어의 삽입 횟수는 랜덤하게 결정하되, 전체 초기화 단계의 수행시간은 동일하도록 2번에 걸쳐 삽입되는 더미 싸이클의 삽입 횟수 합계가 50클럭이 되도록 구현하였다. 이는 동기식 스트림 암호의 특성을 고려하여 송·수신단의 Rabbit 알고리듬 수행 시간이 일정하게 하고, 공격자가 더미 싸이클이 삽입된 정도를 파악할 수 없도록 한다. 공격의 대상이 되는 XOR 연산이 1클럭을 사용하기 때문에 50클럭 내에서 충분히 불규칙한 연산시점을 가질 수 있다.

그림 4는 기존의 Rabbit 알고리듬의 소스코드에 마스킹 기법, 연산순서의 랜덤화, 연산시점의 불규칙화 등을 동시에 적용한 Pseudo C 코드이다. 이러한 대응방법은 앞서 2.2 절에서 언급한 2, 3, 4단계에서도 각각 적용 할 수 있으나 1단계에서의 공격이 실패하면 다음 단계로의 진행이 불가능하므로 본 논문에서는 최초 공격 시작점인 1단계에서의 대응방법만 설명한다.

V. 실험 결과 및 안전성 분석

1. 전력분석 공격 결과

ATmega128L에 대응방법이 적용된 Rabbit 알고리듬을 AVR Studio의 컴파일러를 통해 코드를 생성/구현하였다. 칩에서 소모되는 전력을 측정하기 위해서 디지털 오실로스코프를 사용하며, PC에서는 제어프로그램을 통해 전력소모파형을 수집한다. 또한 칩에 인가되는 초기벡터의 해밍무게를 구분하거나 수집된 전력소모파형을 분석하기 위해서 MATLAB Toolbox 프로그램을 사용하며, 센서 노드가 동작할 때 반복적인 초기벡터의 입력을 위해서 칩과 연동되는 신호처리 제어프로그램을 구현하여 사용한다.

제안한 대응방법이 적용된 Rabbit 알고리듬의 초기벡터 설정과정에 대해서 추측한 계수변수 $c_{0,4}$의 하위 8

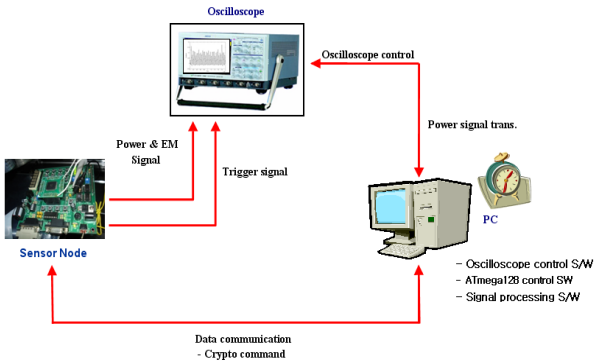

그림 5. 전력분석 공격을 위한 환경
Fig. 5. Experimental Environment.

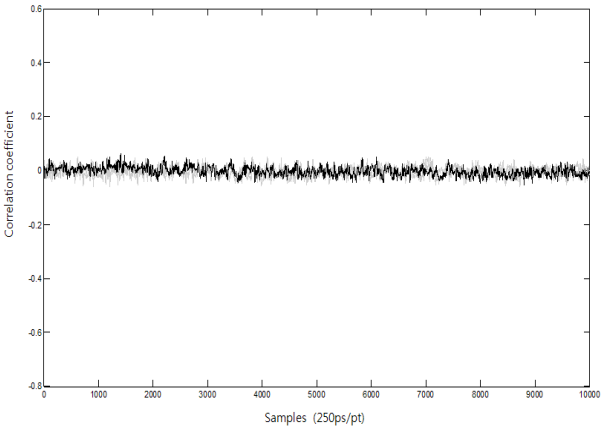

그림 6. 대응방법이 적용된 경우의 전력분석 결과 (모든 경우)
Fig. 6. CPA Result for Countermeasure. (256 Hypotheses)

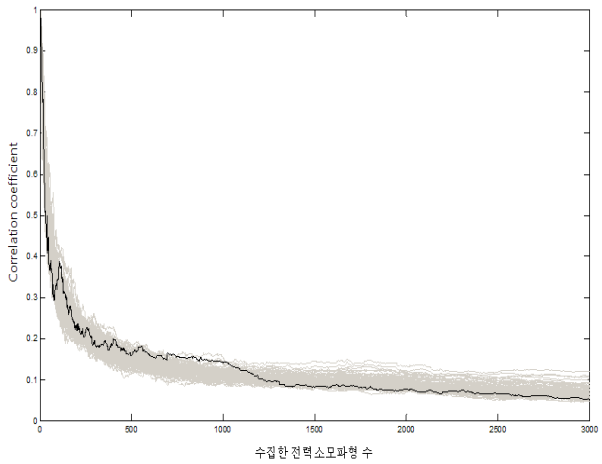

그림 7. 공격에 필요한 전력파형수(가능한 모든 경우)
Fig. 7. Correlation Coefficient for Various Number of Traces-2.

비트에 대한 상관계수(ρ)는 그림 6과 같다.

올바른 경우는 진한 색 파형으로 나타내었고, 엷은 회색 파형은 잘못된 추측의 경우이다. 그림 2에서 확인한 시점의 상관계수(ρ)는 0.05정도에 불과하며, 결과 파형 전체적으로 특별한 피크 성분이 없는 낮은 상관도로 나타남에 따라 제안한 대응방법이 효과적으로 작용하고 있음을 보여준다. 즉, 연산시점의 불규칙화(랜덤 사이클 삽입)와 연산순서의 랜덤화로 인해 XOR 연산 시점이 재동기화마다 변하며, 마스킹 기법에 의해 XOR 연산과의 연관성도 저하되어 전반적인 상관계수가 모두 낮아졌기 때문이다.

수집한 전력소모파형 수에 따른 상관도 분석 역시도 그림 7과 같이 올바른 경우의 진한 색 파형과 255가지의 잘못된 추측의 경우의 흐린 색 파형 간의 큰 차이가 없음을 확인할 수 있었다.

2. 안전성 및 효율성 평가

제안한 가변 마스킹 기법 및 숨김 기법이 적용된 Rabbit 알고리듬에 대해 안전성 및 효율성을 분석해 보았다. 안전성은 공격에 필요한 전력소모파형의 수에 의해 결정되며, 전력소모파형의 수는 최대 상관계수와 관련이 있다. S. Mangard가 제시한 공식[15]에 의하면, 랜덤 마스킹 기법 및 숨김 기법이 적용하지 않을 경우를 $Z_{0.9999} = 3.7190$으로 두고, 상관계수($\rho_{\max}$)가 0.519일 때 약 90개의 전력소모파형의 수집시 상관도 전력분석 공격이 가능함을 알 수 있다. 그러나 그림 7에서 볼 수 있듯이 대응방법이 적용된 경우 전력소모파형의 수와 상관없이 전력분석 공격이 이루어지지 않았다. 또한 랜덤 마스킹 기법을 제외하더라도 동일 시점에 동일한 레지스터가 연산될 확률은 1/8*1/50이다. 따라서 제안한 대응방법은 이론적 및 실험적 결과로 전력분석 공격에 안전하다.

효율성 측면에서 대응방법 적용 전/후의 메모리 크기, 전체 암호 알고리듬 수행 중에 비밀키 설정과정과 초기벡터 설정과정의 수행시간 등을 표 1에서 비교하였다. 대응방법 적용 전의 비밀키 설정과정과 초기벡터 설정과정은 약 6.6ms의 수행시간이 소요되었으나, 적용된 경우에는 약 8.2ms로 약 24%의 수행시간이 증가되었다. 또한 대응방법을 구현하기 위해 40여개의 레지스터가 더 사용되었으며, 10.218 Kbytes의 플래시 메모리도 추가 요구되었다. 기존의 스트림 암호에 대한 대응방법에는 마스킹 기법 등이 적용된 사례가 없어 기존의 블록 암호 시스템 AES를 대상으로 추가적인 비용(overhead)를 비교하여 표 2로 정리하였다. 각 방법에서는 대응방법이 적용되지 않은 알고리듬의 싸이클 수, 메모리 요구량을 달리 사용하고 있으므로 대응방법이 적용되지 않은 경우를 100으로 두고, 추가적인 증가량

표 1. 비밀키 및 초기벡터 설정과정에 대한 대응방법의 성능비교 (1클럭 : 60ns, 구동클럭 : 16Mhz)
Table 1. Performance Evaluation of Countermeasure.

구 분	기존 Rabbit 알고리듬	제안한 대응방법이 적용된 Rabbit 알고리듬	증가율
플래시 메모리	8.934 Kbytes	10.218 Kbytes	12.3%
SRAM	59 bytes	59 bytes	0%
설정과정의 수행시간	약 6.6ms	약 8.2ms	24%
사용 레지스터 수	40	79	49%
XOR 연산수(클럭 싸이클 수)	-	128(3,231)	128%

표 2. AES에 적용된 대응방법과 제안한 대응방법의 비교
Table 2. Performance Evaluation with AES Countermeasures.

구 분		증가량			비 고
		수행시간(ms)	메모리 요구량(byte)		
			RAM	ROM	
AES	미적용 상태	5	-	1150	
	마스킹 기법[9]	× 5.20	-	+173%	
	마스킹 기법[10]	× 6.40	-	+147%	
	마스킹 기법[11]	× 5.30	-	+150%	
Rabbit (비밀키 설정과정+ 초기벡터 설정과정)	미적용 상태	6.6	59	8934	
	제안하는 대응방법 (랜덤 마스킹 + 연산 순서의 랜덤화 + 랜덤 더미 싸이클)	× 1.24	0	+112%	랜덤 더미 싸이클의 삽입 횟수가 n = 50 인 경우

에 대해서만 분석하였다. 표 2와 같이 제안하는 대응방법의 연산 증가량은 기존의 블록 암호 시스템에 적용된 대응방법에 비해 크게 낮아, 고속의 성능을 필요로 하는 스트림 암호에 적용하는 데 적합하다.

Ⅵ. 결 론

본 논문에서는 Rabbit 알고리듬의 특성을 고려하여 저메모리 환경에서 전력분석 공격에 대응할 수 있는 저비용의 소프트웨어적인 대응방법을 제안하였다. Rabbit 알고리듬의 특성상 1단계에서 계수변수를 알아낼 수 없다면, 최종 비밀키를 알아낼 수 없는 구조를 가지므로, 랜덤 부울 마스킹 기법과 연산과정의 숨김 기법 등을 1단계의 공격 대상인 초기화 단계에 적용하였다. 스트림 암호의 장점(빠른 속도)을 최대한 고려하여 내부변수의 일부 연산구간에만 적용한 방어기법은 8비트 마이크로프로세서에 실제로 구현하여 전력분석 공격에 안전함을 실험적으로 검증하였다.

참 고 문 헌

[1] P. Kocher, J. Jaffe, and B. Jun, “Differential power analysis,” CRYPTO 1999, LNCS 1666, pp. 388-397, Santa Barbara, USA, Aug. 1999.

[2] Keke Wu, Huiyun Li, Bo Peng, and Fengqi Yu, “Correlation Power Analysis Attack against Synchronous Stream Ciphers,” ICYCS 2008, IEEE Computer Society, pp. 2067-2072, Zhang Jia Jie, China, Nov. 2008.

[3] The eSTREAM Project, http://www.ecrypt.eu.org/stream

[4] ISO/IEC 18033-4, “Amendment 1 - Information technology - security techniques - Encryption algorithms - Part 4: Stream ciphers,” JTC 1/SC 27(IT security tech.), http://www.iso.org, 2009.

[5] M. Boesgaard, M. Vesterager, T. Pedersen, J. Christiansen, and O. Scavenius. “Rabbit: A new high-performance stream cipher,” FSE 2003, LNCS 2887, pp. 307 - 329, Lund, Sweden, Feb. 2003.

[6] B. Gierlichs, L. Batina, C. Clavier, T. Eisenbarth, A. Gouget, Helena H, T. Kasper, K .Lemkerust, S. Mangard, A. Moradi and E. Oswald, “Susceptible of eSTREAM Candidates towards Side Channel Analysis,” SASC 2008, pp. 123-150, Lausanne, Switzerland, Feb. 2008.

[7] 배기석, 안만기, 박제훈, 이훈재, 문상재, “스트림 암호 Rabbit에 대한 전력분석 공격,” 정보보호학회 논문지, 제21권 제3호, pp.27-36, 2011.

[8] S. Tillich and C. Herbert, “Attacking State-of-the-Art Software Countermeasures - A Case Study for AES,” CHES 2008, LNCS 5154, pp. 228-243, Washington, D.C., USA, Aug. 2008.

[9] E. Oswald, S. Mangard, N. Pramstaller, and V. Rijmen. “A Side-Channel Analysis Resistant Description of the AES S-box”, FSE 2005, LNCS 3557, pp.21-23, Paris, France, Feb. 2005.

[10] E. Prouff, C. Giraud and S. Aumônier, “Provably Secure S-Box Implementation Based on Fourier Transform”, CHES 2006, LNCS 4249, pp.216-230, Yokohama, Japan, Oct. 2006.

[11] S. Tillich, C. Herbst, and S. Mangard, “Protecting AES Software Implementations on 32-Bit Processors Against Power Analysis”, ACNS 2007, pp. 141-157, Zhuhai, China, June 2007.
[12] M. Rivain, E. Prou, and J. Doget. “Higher-Order Masking and Shuffling for Software Implementations of Block Ciphers,” CHES 2009, LNCS 5747, pp. 171-188, Lausanne, Switzerland, Sept. 2009.
[13] 조영인, 김희석, 한동국, 홍석희, 강주성, “마스킹 형태 변환 알고리즘에 대한 새로운 전력분석 공격,” 전자공학회 논문지, 제47권 SP편, 제1호, 2010.
[14] Ilya Mironov, “(Not So) Random Shuffles of RC4,” CRYPTO 2002, LNCS 2442, pp. 304 - 319, Santa Barbara, USA, Aug. 2002.
[15] S. Mangard, “Hardware Countermeasures against DPA – A Statistical Analysis of Their Effectiveness,“ CT-RSA 2004, LNCS 2964, pp. 222-235, San Francisco, USA, Feb. 2004.

저 자 소 개

배 기 석(정회원)
2006년 경북대학교 전자전기 컴퓨터학부 학사
2008년 경북대학교 전자전기 컴퓨터학부 석사
2009년～현재 경북대학교 전자 전기컴퓨터학부 박사과정
<주관심분야 : 정보보호, 네트워크 보안, 스마트 카드 보안>

박 영 호(정회원)-교신저자
1989년 경북대학교 전자공학과 학사
1991년 경북대학교 전자공학과 석사
1995년 경북대학교 전자공학과 박사
1996년～2008년 상주대학교 전자전기공학부 교수
2003년～2004년 Oregon State Univ. 방문교수
2008년～현재 경북대학교 산업전자공학과 교수
<주관심분야 : 정보보호, 네트워크보안, 모바일 컴퓨팅>

안 만 기(정회원)
2000년 경북대학교 전자전기공학부 졸업(학사)
2000년～2001년 삼성전자 프린터 사업부 C-LBP 연구원
2003년 경북대학교 전자공학과 석사(정보통신공학)
2011년 경북대학교 정보보호학과 박사
2003년～현재 국방기술품질원 선임연구원
<주관심분야 : 정보보호, RFID/USN 보안, 부채널분석 공격>

문 상 재(평생회원)
1972년 서울대학교 공업교육 (전자전공)과 학사
1974년 서울대학교 전자공학과 공학석사
1984년 미국 UCLA 전기공학과 공학박사
1984년～1985년 미국 UCLA 포스트닥터
1984년～1985년 미국 OMNET 회사 컨설턴트
1974년～현재 경북대학교 IT대학 전자공학부 교수
2002년 2월～현재: 한국정보보호학회 명예회장
<주관심분야 : 무선통신, 네트워크 보안, 암호학>