논문 2009-46SD-11-2

# 불 마스크와 산술 마스크에 대한 게이트 레벨 변환기법

## ( Gate-Level Conversion Methods between Boolean and Arithmetic Masks )

백 유 진*

( Yoo-Jin Baek )

요 약

암호 시스템을 구현할 경우 차분 전력 분석 공격법 등과 같은 부채널 공격법에 대한 안전성은 반드시 고려되어야 한다. 현재까지 부채널 공격법에 대한 다양한 방어 기법이 제안되었으며, 본 논문에서는 그러한 방어 기법 중의 하나인 마스킹 기법을 주로 다루게 된다. 특히 본 논문에서는 이러한 마스킹 기법의 구현에 수반되는 불 마스크와 산술 마스크 사이의 변환 문제에 대한 효율적인 해법을 제시한다. 새로 제안된 방법의 기본적인 아이디어는, ripple adder에 사용되는 carry 비트와 sum 비트를 계산하는 과정 중에 랜덤 비트를 삽입함으로써 공격자가 상기 비트들과 원 데이터 사이의 상관관계를 알아내지 못하게 하는 데에 있다. 새로 제안된 방법은 어떠한 여분의 메모리 사용 없이 단지 $6n-5$개의 XOR 게이트와 $2n-2$개의 AND 게이트만을 사용하여 $n$-비트 이진열에 대한 마스크 변환을 수행하며 변환 수행 시 $3n-2$ 게이트 시간 지연을 필요로 한다. 새로 제안된 방법은 특히 비트 단위의 연산만을 사용하기 때문에 불 연산과 산술 연산을 동시에 사용하는 암호 알고리즘을 차분 전력 분석 공격에 안전하게 하드웨어로 구현하는 경우 효과적으로 사용될 수 있다. 예를 들어 본 논문은 새로 제안된 방법을 SEED 블록 암호 알고리즘의 안전한 구현에 적용하였으며 그 상세한 구현 결과는 본문에 제시된다.

Abstract

Side-channel attacks including the differential power analysis attack are often more powerful than classical cryptanalysis and have to be seriously considered by cryptographic algorithm's implementers. Various countermeasures have been proposed against such attacks. In this paper, we deal with the masking method, which is known to be a very effective countermeasure against the differential power analysis attack and propose new gate-level conversion methods between Boolean and arithmetic masks. The new methods require only $6n-5$ XOR and $2n-2$ AND gates with $3n-2$ gate delay for converting $n$-bit masks. The basic idea of the proposed methods is that the carry and the sum bits in the ripple adder are manipulated in a way that the adversary cannot detect the relation between these bits and the original raw data. Since the proposed methods use only bit-wise operations, they are especially useful for DPA-securely implementing cryptographic algorithms in hardware which use both Boolean and arithmetic operations. For example, we applied them to securely implement the block encryption algorithm SEED in hardware and present its detailed implementation result.

**Keywords**: 부채널 공격법, 차분 전력 분석 공격법, 마스킹 기법, 마스크 변환 기법, SEED

## I. 서 론

개인 비밀 정보가 저장된 정보 기기는 그 특성상 불법적인 정보 유출과 비인가된 접근 등과 같은 위험성에 노출될 수 있으며 이러한 위험성을 방어하기 위하여 많은 암호 알고리즘이 사용되고 있다. 이러한 암호 알고리즘의 구현 시 최근까지는 속도 측면이 가장 중요하게 고려되었으나 앞으로는 부채널 공격법에 대한 암호 알고리즘의 안전성도 함께 고려되어야 한다.

부채널 공격법은 암호 알고리즘을 물리적인 기기에서 동작하는 실제 프로그램으로 간주하며 해당 기기 내의 암호 연산 과정 중에 발생하는 연산 소비 시간, 전력 소비 패턴 등과 같은 부채널 정보를 이용하여 기기 내에 저장된 비밀 정보를 추출한다.[1~2] 부채널 공격법은 Kocher 등에 의해 스마트카드에 대한 시간 분석 공격

* 정회원, 삼성전자
(Samsung Electronics)
접수일자: 2009년4월15일, 수정완료일: 2009년10월6일

법이 제안된 이후로 현재까지 여러 가지 다양한 공격법과 그에 대한 대응 기법이 제시되었다.

특히, 차분 전력 분석 공격법(Differential Power Analysis, 이하 DPA)[2]은 암호 연산 시 발생하는 전력 소비 패턴을 분석함으로써 암호 기기 내에 저장된 비밀 정보를 알아내며 이에 대한 여러 가지 방어 기법이 제안되었다. 예를 들어 잡음 전력을 삽입하는 방법, 전력 소비 신호를 필터링하는 방법, 연산 순서를 랜덤화하는 방법 등은 현재까지 제안된 하드웨어적인 방어 기법들의 예이며 데이터를 랜덤화하는 기법은 대표적인 소프트웨어적 DPA 방어 기법의 하나이다.

본 논문에서는 DPA에 대한 매우 효과적인 방어 기법 중의 하나로 알려진 마스킹 기법[3]을 주로 다루게 된다. 마스킹 기법이란, 비밀 정보를 이용한 특정 암호 연산이 수행되기 전에 난수를 이용해서 입력 평문을 마스킹함으로써 공격자가 유용한 정보를 추출하는 것을 방어할 수 있다. 특히 본 논문에서는 불 마스크와 산술 마스크를 동시에 사용하는 암호 알고리즘에 대하여 마스킹 기법을 구현하는 경우 발생할 수 있는 마스크 변환 문제를 해결하는 새로운 알고리즘을 제안한다. 본 논문에서 제시된 마스크 변환 알고리즘의 주요 아이디어는 ripple adder에 사용되는 carry 비트와 sum 비트를 랜덤화함으로써 공격자가 마스크 변환시 전력 분석 곡선으로부터 어떠한 유용한 정보도 추출하지 못하게 하는 데에 있다.

이전 결과들[3-5]과 비교하였을 경우 새로 제안된 방법은 특히 하드웨어적으로 구현되었을 경우 매우 효과적이다. 즉, 새로 제안된 방법은 어떠한 여분의 메모리 사용 없이 단지 $6n-5$개의 XOR 게이트와 $2n-2$개의 AND 게이트만을 사용하여 $n$-비트 이진열에 대한 마스크 변환을 수행하며 이때 $3n-2$ 게이트 시간 지연만을 필요로 한다. 반면에 [3]에서 제안된 방법은 또 다른 종류의 전력 분석 공격에 취약함이 알려져 있으며[6], [4]와 [5]에서 제안된 방법은 여분의 메모리를 사용하거나 해당 변환을 수행 시 필요한 계산량이 본 논문에서 제안된 방법보다 많다는 단점이 있다. 보다 자세한 비교는 본문의 Ⅳ장에 주어진다.

새로 제안된 마스크 변환 알고리즘은 불 연산과 산술 연산을 동시에 사용하는 암호 알고리즘을 DPA에 안전하게 구현하는 데에 적용 가능하다. 본 논문에서 제안된 방법은 특히 대한민국 상용 표준 블록 암호 알고리즘 SEED[7]를 DPA에 안전하게 구현하기 위해서 적용되었으며 그 상세한 구현 결과는 Ⅴ장에 제시된다.

본 논문의 구성은 다음과 같다. 먼저 Ⅱ장에서는 DPA, 마스킹 기법, 불 마스크와 산술 마스크 사이의 변환 문제, 그리고 ripple adder의 특징이 간략히 소개된다. Ⅲ장에서는 새로운 마스크 변환 알고리즘이 제안되고 이전 변환 알고리즘과의 비교 결과는 Ⅳ장에 제시된다. Ⅴ장에서는 본 논문의 방법을 SEED 알고리즘에 적용한 구현 결과를 소개하며 6절에서는 결론을 도출한다.

## Ⅱ. 선행 지식

### 1. DPA와 마스킹 기법

DPA는 Kocher 등에 의해 처음 제안되었으며[2] 암호 기기의 동작 시 발생하는 전력 소비 패턴을 분석하여 기기 내에 저장된 비밀 정보를 추출한다. DPA가 동작하는 근본적인 원인은, 암호 기기가 소비하는 전력량은 기기의 내부 상태 즉, 수행되는 연산의 종류나 연산에서 사용하는 내부 인자 등과 서로 밀접한 관련이 있다는 데에 있다. 이러한 DPA에 대하여 지금까지 다양한 방어 기법이 제안되었으며 그 중에서 마스킹 기법[3]은 DPA에 대한 가장 강력한 알고리즘적 방어 기법의 하나로 알려져 있다.

마스킹 기법은 일반적으로 다음과 같이 동작한다: 먼저, 난수를 이용해서 평문을 마스킹하고 (마스킹 단계) 그 마스킹된 값을 이용해서 해당 암호 연산을 수행한 후 마지막으로 그 결과 값을 이용해서 원하는 암호문을 복구한다 (마스킹 해제 단계). 이러한 마스킹 기법을 적용시 일반적으로 다음과 같은 두 종류의 마스크가 사용된다.

정의 1 $n$-비트 이진열 $x$의 불 마스크란 $x = x' \oplus r$을 만족시키는 임의의 순서쌍 $(x', r)$을 의미한다. 여기서 $x', r$은 각각 $n$-비트 이진열을, $\oplus$는 비트 단위의 XOR 연산을 나타낸다.

정의 2 $n$-비트 이진열 $x$의 산술 마스크란 $x = x' + r \bmod 2^n$을 만족시키는 임의의 순서쌍 $(x', r)$을 의미한다. 여기서 $x', r$은 각각 $n$-비트 이진열을 나타낸다.

마스킹 기법을 실제 암호 연산에 적용하기 위해서는 일반적으로 다음과 같이 정의된 마스킹 문제의 해결이 요구된다. 불 함수 $f: \{0,1\}^n \rightarrow \{0,1\}^k$가 주어졌을 때 다음을 만족시키고 효율적으로 계산 가능한 (확률적) 함수 $F: \{0,1\}^{2n} \rightarrow \{0,1\}^{2k}$를 구하시오.

1) $n$-비트 이진열 $x$에 대한 임의의 마스크 $(x', r)$가 주어지면 함수 $F$는 $f(x)$의 랜덤한 마스크를 출력한다. 즉, 랜덤하게 선택된 $k$-비트 이진열 $s$에 대하여 $F(x', r) = (f(x) \oplus s, s)$, 또는 $F(x', r) = (f(x) + s \bmod 2^k, s)$이다.
2) $F$를 계산하는 도중에 $x$에 대한 어떠한 정보도 노출되어서는 안 된다. 즉, $F$의 계산 중간에 등장하는 모든 중간 값들의 확률 분포는 $x$와 서로 독립이어야 한다.

예를 들어 만약 $f$가 선형 함수인 경우, 함수 $F(x', r) = (f(x'), f(r))$는 $f$에 대한 마스킹 문제의 하나의 해가 된다. 또한 $k \times n$ 행렬 $A$와 $k$-비트 이진열 $b$가 존재하여 $f(x) = Ax \oplus b$를 만족시키는 아핀 함수 $f$가 주어진 경우 함수 $F(x', r) = (f(x'), Ar)$는 $f$에 대한 마스킹 문제를 해결하게 된다.

## 2. 마스킹 변환 문제

불 연산과 산술 연산을 모두 사용하는 암호 알고리즘을 DPA에 안전하게 구현하기 위하여 마스킹 기법을 적용할 경우 일반적으로 다음과 같이 정의된 마스킹 변환 문제에 대한 효율적인 해결책을 필요로 한다. 만약 평문이 불 마스크로 변환이 되었고 암호 연산 수행 도중에 산술 연산을 해당 데이터에 적용해야 한다면 불 마스크된 데이터를 산술 마스크된 데이터로 변환해야 한다. 비슷하게 평문이 산술 마스크로 변환이 되어 있고 암호 연산 도중에 불 연산을 적용해야 하는 경우 산술 마스크된 데이터를 불 마스크된 데이터로 변환해야 한다. 따라서 다음과 같은 마스크 변환 문제가 정의될 수 있다.

**불 마스크를 산술 마스크로 변환하는 문제** $n$-비트 이진열 $x$의 임의의 불 마스크 $(x', r)$가 주어진 경우 랜덤하게 선택된 $s$에 대하여 $x = y + s \bmod 2^n$을 만족시키는 순서쌍 $(y, s)$를, $x$에 대한 어떠한 정보의 노출 없이 구하시오.

**산술 마스크를 불 마스크로 변환하는 문제** $n$-비트 이진열 $x$의 임의의 산술 마스크 $(x', r)$가 주어진 경우 랜덤하게 선택된 $s$에 대하여 $x = y \oplus s$을 만족시키는 순서쌍 $(y, s)$를 $x$에 대한 어떠한 정보의 노출 없이 구하시오.

본 논문에서는 상기 첫 번째 문제를 해결하는 알고리즘을 Boolean To Arithmetic Conversion Algorithm으로, 두 번째 문제를 해결하는 알고리즘을 Arithmetic To Boolean Conversion Algorithm으로 정의한다.

## 3. Full Adder와 Ripple Adder

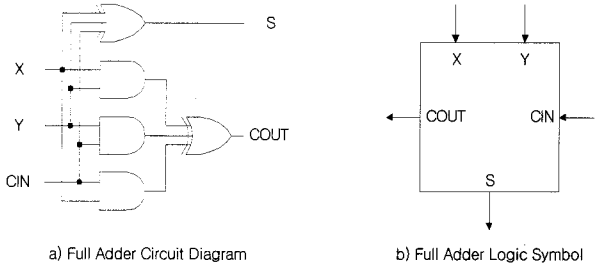
본 논문에서 제안된 방법은 full adder와 ripple adder의 구조적인 특징을 이용한다. 따라서 이번 절에서는 상기 두 덧셈기의 특징에 대해 간략히 소개한다.

Full adder는 1-비트 이상의 크기를 가지는 두 수에 덧셈 연산을 적용하기 위한 기본 도구로 사용되며 다음과 같이 동작한다. 1-비트 크기의 변수 $X, Y, CIN$이 입력으로 주어지면 full adder는 다음의 등식을 만족시키는 1-비트 크기의 $S, COUT$을 출력한다:

$$S = X \oplus Y \oplus CIN$$
$$COUT = (X \cdot Y) \oplus (X \cdot CIN) \oplus (Y \cdot CIN)$$

상기 등식에서 $\cdot$는 비트 단위의 AND 연산을 의미하며 이후부터는 편의상 $\cdot$ 표시를 생략한다. 그림 1은 이러한 full adder를 수행하는 회로도를 보여준다.

2개의 $n$-비트 이진열에 대한 덧셈 연산은 상기 full adder를 반복적으로 사용하는 ripple adder를 적용함으로써 수행될 수 있다. 그림 2는 4-비트 이진열에 적용




a) Full Adder Circuit Diagram b) Full Adder Logic Symbol

그림 1. Full Adder
Fig. 1. Full Adder.

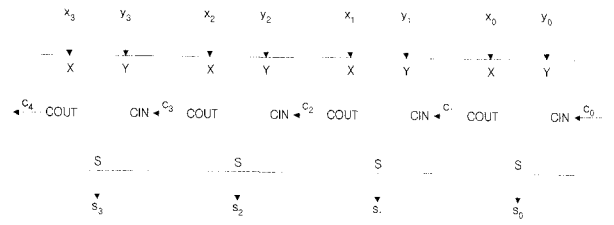



그림 2. 4비트 Ripple Adder
Fig. 2. 4-Bit Ripple Adder.

된 ripple adder를 묘사하는 회로도를 보여주며 여기서 입력 carry 비트인 $c_0$는 일반적으로 0으로 주어진다. 물론 그 외에도 여러 가지 다른 덧셈기가 존재하며 각각의 덧셈기는 회로 면적이나 지연 시간 등에서 서로 다른 특성을 가지게 된다.[8]

## Ⅲ. 새로운 마스크 변환 알고리즘

본 논문에서 새롭게 제안하는 마스크 변환 알고리즘은 ripple adder의 특성을 이용하기 때문에 본 절에서는 먼저 ripple adder에 대해서 좀 더 자세히 설명한다.

$n$-비트 이진열 $a = a_{n-1}...a_0$, $r = r_{n-1}...r_0$, $x = a + r \bmod 2^n = x_{n-1}...x_0$, $y = a \oplus r = y_{n-1}...y_0$에 대하여 ripple adder의 carry 비트 $c_i$는 다음을 만족시킨다.

$$c_i = \begin{cases} c_{i-1}y_{i-1} \oplus (y_{i-1} \oplus r_{i-1})r_{i-1} & i > 0 \\ 0 & i = 0 \end{cases}$$
$$x_i = c_i \oplus a_i \oplus r_i = c_i \oplus y_i \qquad i > 0 \tag{1}$$

따라서 ripple adder는 $y$와 $r$이 주어지면 상기 수식 (1)을 반복 사용함으로써 $x$를 계산할 수 있으며 따라서 불 마스크를 산술 마스크로 변환하는 알고리즘에 직접 사용이 가능하다. 그러나 이렇게 수식 (1)를 마스크 변환 문제에 직접 사용하게 되면 여러 가지 DPA에 대한 안전성 문제가 발생하게 된다. 예를 들어, 수식 (1)의 계산 중간 결과 값인 $c_i$와 원래의 데이터 $a$는 강한 상관관계를 가지게 되는데 가령, $c_1 = (y_0 \oplus r_0)r_0 = a_0 r_0$이기 때문에, 가령 $r$이 랜덤하게 선택되더라도 $a_0$가 1인 경우에 $c_1$은 1/2의 확률로 1의 값을 가지게 되지만, $a_0$가 0인 경우에는 1의 값을 절대로 가질 수 없다. 이러한 통계적인 치우침 현상을 해결하기 위하여 본 논문에서는 또 다른 랜덤 비트열 $s$를 사용한다. 즉, 랜덤하게 선택된 $s$에 대하여 비트열 $d = d_{n-1}...d_0$를 $d_i = c_i \oplus r_r \oplus s_i, i = 0, ..., n-1$와 같이 정의하면 상기 덧셈 공식 (1)은 하기의 수식 (2)와 같이 재정의 될 수 있으며 이렇게 새롭게 정의된 덧셈 공식을 이용하면 불 마스크를 산술 마스크로 변환해 주는 Algorithm 1을 유도할 수 있다: $d_0 = r_0 \oplus s_0, x_0 = r_0$이고 $i = 1, ..., n-1$에 대하여

$$\begin{aligned} d_i &= c_i \oplus r_i \oplus s_i \\ &= c_{i-1}y_{i-1} \oplus a_{i-1}r_{i-1} \oplus r_i \oplus s_i \\ &= (d_{i-1} \oplus r_{i-1} \oplus s_{i-1})y_{i-1} \oplus \\ &\quad (y_{i-1} \oplus r_{i-1})r_{i-1} \oplus r_i \oplus s_i \\ &= (d_{i-1} \oplus s_{i-1})y_{i-1} \oplus r_{i-1} \oplus r_i \oplus s_i \\ x_i &= c_i \oplus y_i = y_i \oplus d_i \oplus r_i \oplus s_i \end{aligned} \tag{2}$$

Algorithm 1 (Boolean To Arithmetic)
Input: $y = a \oplus r = y_{n-1}...y_0$, $r = r_{n-1}...r_0$, $s = s_{n-1}...s_0$
Output: $x = a + r \bmod 2^n = x_{n-1}...x_0$
1. $d_0 = r_0 \oplus s_0, x_0 = y_0$
2. For $i = 1, ..., n-1$, do
   (1) $A = d_{i-1}y_{i-1}$
   (2) $B = s_{i-1}y_{i-1}$
   (3) $C = s_i \oplus r_i$
   (4) $D = C \oplus r_{i-1}$
   (5) $E = D \oplus B$
   (6) $F = C \oplus y_i$
   (7) $d_i = E \oplus A$
   (8) $x_i = F \oplus d_i$
3. Return ($x = x_{n-1}...x_0$).

산술 마스크를 불 마스크로 변환해주는 알고리즘도 비슷하게 유도할 수 있으며 Algorithm 2는 이를 보여준다.

Algorithm 2 (Arithmetic To Boolean)
Input: $y = a \oplus r = y_{n-1}...y_0$, $r = r_{n-1}...r_0$, $s = s_{n-1}...s_0$
Output: $x = a + r \bmod 2^n = x_{n-1}...x_0$

1. $d_0 = r_0 \oplus s_0, x_0 = y_0$
2. For $i = 1, ..., n-1$, do
   (1) $A = d_{i-1} y_{i-1}$
   (2) $B = s_{i-1} y_{i-1}$
   (3) $C = s_i \oplus r_i$
   (4) $D = C \oplus r_{i-1}$
   (5) $E = D \oplus B$
   (6) $F' = C \oplus x_i$
   (7) $d_i = E \oplus A$
   (8) $x_i = F' \oplus d_i$
3. Return $(y = y_{n-1} ... y_0)$.

알고리즘 1과 2가 DPA에 안전함을 증명하기 위해서는 원 데이터인 $a$와 모든 중간 계산 결과 값들은 서로 상관관계가 없음을 보여주어야 하며, 이러한 사실은 하기의 보조 정리를 이용하여 증명될 수 있다.

**보조정리 1** $u_1, ..., u_{k+1}$이 선형 독립이고 $\{0,1\}$에서 균등하게 분포된 확률변수라 하자. 이때, 임의의 불 함수 $f: \{0,1\}^k \rightarrow \{0,1\}$에 대하여, 확률 변수 $f(u_1, ..., u_k) \oplus u_{k+1}$는 $\{0,1\}$에서 균등하게 분포한다.

**증명** 편의를 위해 $u = (u_1, ..., u_k)$라 하자. 그러면, $f(u) \oplus u_{k+1}$가 0의 값을 가질 필요충분조건은 $u_{k+1} = f(u)$이다. 따라서 $f(u)$가 0이 값을 가질 확률을 $\tau$라고 하면

$$\begin{aligned} &Pr(f(u) \oplus u_{k+1} = 0) \\ &= Pr(f(u) = u_{k+1} = 0) + Pr(f(u) = u_{k+1} = 1) \\ &= \tau/2 + (1-\tau)/2 = 1/2 \end{aligned}$$

이다.

**보조정리 2** 알고리즘 1과 2의 계산 과정 중에 등장하는 중간 계산 결과 변수들인 $A, b, C, D, E, F, F'$ 및 $d_i, i = 0, ..., n-1$의 확률 분포는 원 데이터 $a$와 독립이다.

**증명** $s_i$는 $\{0,1\}$에서 균등하게 분포되어 있고 확률 변수 $c_i, r_i, r_{i-1}, s_i, s_{i-1}, y_i, y_{i-1}$와 독립이기 때문에 보조정리 1에 의하여 확률 변수 $d_i = c_i \oplus r_r \oplus s_i$, $C = s_i \oplus r_i$, $D = s_i \oplus r_i \oplus r_{i-1}$, $E = s_i \oplus r_i \oplus r_{i-1} \oplus s_{i-1} y_{i-1}$, $F = s_i \oplus r_i \oplus y_i$, $F' = s_i \oplus r_i \oplus x_i$는 $\{0,1\}$에서 균등하게 분포함을 알 수 있다. 또한 $A = d_{i-1} y_{i-1} = (c_{i-1} \oplus r_{i-1} \oplus s_{i-1})(a_{i-1} \oplus r_{i-1})$이기 때문에 만약 $v = c_{i-1} \oplus r_{i-1} \oplus s_{i-1}$, $w = a_{i-1} \oplus r_{i-1}$라 하면 우리는 다음과 같은 결과를 얻을 수 있다: $\alpha = \alpha_{n-1} ... \alpha_0$에 대하여

$$\begin{aligned} &Pr(A = 1 | a = \alpha) \\ &= Pr(v = 1 | w = 1, a = \alpha) Pr(w = 1 | a = \alpha) \\ &= \frac{1}{2} Pr(v = 1 | r_{i-1} = 1 \oplus \alpha_{i-1}, a = \alpha) \\ &= \frac{1}{2} Pr(c_{i-1} \oplus \alpha_{i-1} \oplus s_{i-1} = 0 | a = \alpha) \\ &= \frac{1}{4} \end{aligned}$$

비슷한 방식으로 $Pr(B = 1 | a = \alpha) = 1/4$임을 증명할 수 있다.

**부연 설명** 알고리즘 1과 2에서 연산이 수행되는 순서는 각 알고리즘의 DPA에 대한 안전성에 많은 영향을 미친다. 예를 들어, 알고리즘 2에서 $d_i$를 계산하기 위해 $A, B$를 먼저 계산하고 그 값들에 XOR 연산을 적용하면 $A \oplus B = (c_{i-1} \oplus r_{i-1})(a_{i-1} \oplus r_{i-1})$을 얻을 수 있는데 이 경우 $Pr(c_{i-1} = 1) = \lambda$라 하면 $Pr(A \oplus B = 1 | a_{i-1} = 0) = (1-\lambda)/2$가 되고 따라서 이 경우에는 $A \oplus B$의 확률 분포가 $a$에 영향을 받게 됨을 알 수 있다.

## Ⅳ. 이전 결과들과의 비교

본 절에서는 이전 논문에서 제안된 여러 가지 마스크 변환 알고리즘과 본 논문의 방법을 비교한다.

먼저, T. Messerges에 의해서 제안된 방법[3]은 DPA에 대한 충분한 안전성을 제공해 주지 못함이 알려져 있다[6]. 다음으로 L. Goubin에 의해서 제안된 방법[5]은 1차 DPA에 대한 완벽한 방어 기법을 제공하지만 본 논문의 방법에 비해 덜 효율적이다. 즉, [5]에서 제안된 방법은, $n$-비트 이진열의 불 마스크를 산술 마스크로 변환하기 위하여 5개의 $n$-비트 XOR 연산과 2개의 모듈러 뺄셈 연산을, 산술 마스크를 불 마스크로 변환하기

표 1. Boolean To Arithmetic 알고리즘의 연산량
Table 1. Computational Cost of Boolean To Arithmetic Conversion Algorithms.

| | 연산량 | 메모리 사용량 |
|---|---|---|
| Messerges[3] | 전력분석공격에 취약 | 없음 |
| Goubin[5] | $5n$ 1-bit XOR<br>2 $n$-bit mod. sub. | 없음 |
| 본 논문의 방법 | $6n-5$ 1-bit XOR<br>$2n-2$ 1-bit AND | 없음 |

표 2. Arithmetic To Boolean 알고리즘의 연산량
Table 2. Computational Cost of Arithmetic To Boolean Conversion Algorithms.

| | 연산량 | 메모리 사용량 |
|---|---|---|
| Messerges[3] | 전력분석공격에 취약 | 없음 |
| Coron et al.[4] | $2^{n+1}$ $n$비트 사전계산<br>1번의 테이블 참조 | $n2^n$ bits |
| Goubin[5] | $2n^2+4n$ 1-bit XOR<br>$2n^2+n$ 1-bit AND<br>$n$ SHIFT | 없음 |
| 본 논문의 방법 | $6n-5$ 1-bit XOR<br>$2n-2$ 1-bit AND | 없음 |

위하여 $2n+4$개의 XOR 연산, $2n+1$개의 $n$비트 AND 연산, 그리고 $n$개의 SHIFT 연산을 사용한다. 반면에 본 논문에서 제안된 방법은 각 변환을 위해 $6n-5$개의 1-비트 XOR 연산과 $2n-2$개의 1-비트 AND 연산만을 사용한다. 마지막으로 J.-S. Coron 등에 의해 제안된 산술 마스크를 불 마스크로 변환하는 방법[4]은 테이블 참조 방법을 사용하여 전체 변환 시간을 줄인다. 하지만 사전 계산된 테이블을 사용하기 때문에 테이블의 재계산 횟수에 따라 그 효율성이 결정된다는 단점이 있다. 이러한 비교 결과는 표 1 과 2에 상세히 제시되며 표에서는 $n$-비트 XOR 연산 및 AND 연산은 $n$개의 1-비트 XOR 게이트 및 AND 게이트와 각각 동일함을 가정하였다. 물론 해당 알고리즘을 하드웨어적으로 구현 시 속도 및 면적은 표에서 제시된 연산량에 비례하게 된다. 마지막으로 표 2에서 Coron과 Tchulkine에 의해 제안된 방법의 연산량은 그들의 첫 번째 알고리즘에 대한 것이다. 그들의 두 번째 방법은 사전 계산된 테이블을 사용함으로써 연산량을 줄일 수 있다.

표 3. 변환 timing 분석
Table 3. Timing Analysis of Conversions.

| | Bool. To Arith. | Arith. To Bool. |
|---|---|---|
| Coron et al.[4] | 해당 사항 없음 | 메모리 읽기 시간 |
| Goubin[5] | 3 게이트 시간 지연,<br>1 $n$비트 덧셈기<br>연산 시간 지연 | $3n+3$ 게이트<br>시간 지연 |
| 본 논문의 방법 | $3n-2$ 게이트 시간 지연 | |

하드웨어 구현 시에는 timing 또한 고려되어야 할 중요한 요소이다. J.-S. Coron 등에 의해서 제안된 방법[4]은 메모리 읽기/쓰기 연산만을 사용하기 때문에 timing 문제는 발생하지 않는다. 그러나 Goubin에 의해서 제안된 방법[5]과 본 논문에서 제안된 방법은 특정 구현 환경에서의 제약 조건을 만족시키기 위해서 timing에 대한 조심스러운 접근이 요구된다. 본 논문은 timing 분석의 편의를 위하여 AND, OR, XOR 게이트의 시간 지연이 동일하며 그 값이 1이 됨을 가정한다. 이러한 가정 하에서 Goubin이 제안한 불 마스크를 산술 마스크로 변환하는 알고리즘은 3 게이트 시간 지연과 1 $n$-비트 덧셈기 연산 시간 지연을 요구한다. 물론 $n$-비트 덧셈기 연산 시간 지연은 해당 덧셈기의 구현 방법에 따라 그 값이 다르다. 또한 Goubin이 제안한 산술 마스크를 불 마스크로 변환하는 알고리즘은 $3n+3$ 게이트 시간 지연을 필요로 한다. 반면에 본 논문에서 제안된 방법은 각각의 변환을 위하여 $3n-2$ 게이트 시간 지연만을 요구한다. 이러한 분석 결과는 표 3에 제시된다.

## V. 응 용

SEED 알고리즘[7]은 Feistel 구조와 16라운드를 사용하여 암/복호화를 수행하며 128-비트 메시지 크기 및 키 크기를 지원하는 블록 암호 알고리즘이다. 또한 SEED는 차분 분석법과 선형 분석법 등과 같은 여러 가지 수학적인 암호 공격법에 강인함이 알려져 있으며 현재는 대한민국 상용 블록 암호 알고리즘 표준으로 사용된다. SEED의 전체적인 동작 방식은 다음과 같다: 128-비트 크기의 평문은 2개의 64-비트 크기의 부분 블록 $(L, R)$로 분해가 되고 이 중 $R$은 다시 2개의 32-비트 크기의 부분 블록 $(C, D)$로 분해가 된다. 이 부분 블록들은 이후 키 생성 알고리즘에 의해 출력된 두 개의 32-비트 크기의 부분 키 $(RK_0, RK_1)$과 함께

라운드 함수 $F$의 입력 값이 된다. $F$함수의 출력 값 $(C', D') = F(C, D, RK_0, RK_1)$은 그림 3과 같이 계산된다. 그림 3에서 덧셈 연산과 XOR 연산은 32-비트 단위로 수행이 되며 $F$함수의 내부 함수 $G$는 2개의 계층, 즉 유한체 $GF(2^8)$에서 정의된 단항식 $x^{247}, x^{251}$ 및 특정한 아핀 함수의 합성 함수에 의해 정의되는 2개의 $8\times8$ $S$함수로 이루어지는 계층과 16개의 8-비트 부분 블록에 대한 치환 함수 계층으로 구성된다. SEED에 대한 보다 자세한 정의는 [7]에 주어진다.

새로 제안된 방법을 SEED의 $F$함수 내부 덧셈 연산에 구현하기 위해 본 논문에서는 덧셈 연산 직전에 불 마스크를 산술 마스크로 변환하는 알고리즘을 적용하고 덧셈 연산 직후에는 산술 마스크를 불 마스크로 변환하는 알고리즘을 적용한다. 따라서 만약 데이터가 산술 마스크로 변환이 되었다면 산술 덧셈기는 선형 연산이 되기 때문에 2-1절에서 설명한 바와 같이 마스킹 기법을 적용하는 것은 어렵지 않게 된다.

SEED 알고리즘 전체에 마스킹 기법을 적용하기 위해서는 입력 평문을 먼저 불 마스크로 변환을 하고 이후에 출력된 SEED 암호문의 마스크를 이용해서 원래의 암호문을 복구한다. 물론 이러한 암호 연산 및 마스크 변환 과정 중에 비밀 키와 관련된 어떠한 정보도 노출이 되지 않음은 3절에서 보여진 바와 같다.

본 논문에서는 SEED 알고리즘을 Verilog-HDL을 이용해서 구현하였으며 모듈러스 $2^{32}$에 대한 산술 덧셈기에 마스킹 기법을 적용하기 위해서 본 논문에서 제안된 방법을 이용하였다. 본 논문에서 구현한 SEED 알고리즘은 속도보다는 면적의 최적화에 중점을 두었기 때문에 라운드당 7클록을 사용하였으며 Cadence NC-Verilog를 이용하여 시뮬레이션을 수행하였다. 또한 Synopsys Design-Compiler와 Samsung 스마트카드 공정 (*smart130*)이 사용하여 합성을 하였다. 그 상세한 구현 결과는 표 4에 주어진다.

표 4. SEED 구현 결과
Table 4. Implementation Result of SEED.

| | Gate Count | Critical Path | Throughput |
|---|---|---|---|
| SEED w/o masking | 9.5K | 32 ns | 33.96 Mbps @ 30 MHz |
| SEED with masking | 16.5K | 62 ns | 16.98 Mbps @ 15 MHz |

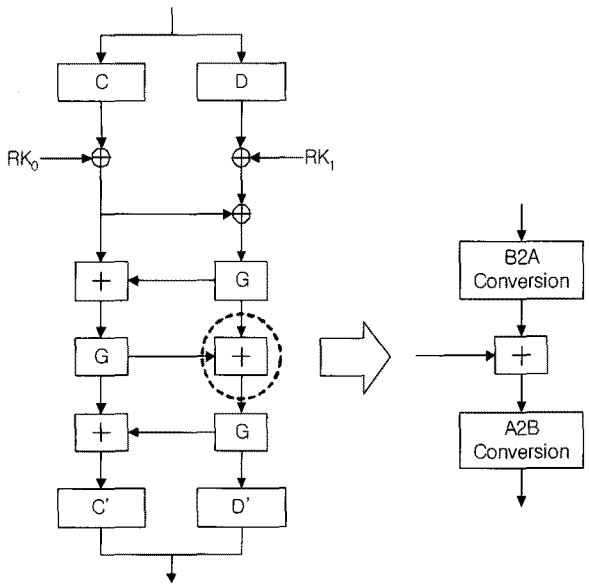



그림 3. SEED에 적용된 변환 알고리즘
Fig. 3. Conversion Algorithm Applied to SEED.

## VI. 결 론

본 논문에서는 DPA를 가장 효과적으로 방어하는 기법 중의 하나인 마스킹 기법을 다루었으며 특히 불 마스크와 산술 마스크 사이의 새로운 변환 알고리즘을 제안하고 그 안전성을 분석하였다. 또한 본 논문에서 제안된 방법은 SEED 블록 암호 알고리즘을 DPA에 안전하게 하드웨어적으로 구현하기 위해서 사용되었으며 그 상세한 구현 결과는 본문에 제시되었다.

## 참 고 문 헌

[1] P. Kocher, Timing Attacks on implementation of Diffie-Hellman, RSA, DSS, and other systems, CRYPTO '96, *Lecture Notes in Computer Science 1109*, 104-113, Santa Barbara, USA, August 1996.

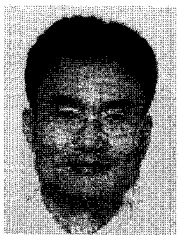
[2] P. Kocher, J. Jaffe and B. Jun, Differential Power analysis, CRYPTO '99, *Lecture Notes in Computer Science 1666*, 388-397, Santa Barbara, USA, August 1999.

[3] T. Messeges, Securing the AES finalists against power analysis attacks, FSE 2000, *Lecture Notes in Computer Science 1978*, 150-165, New York, USA, April 2000.

[4] J.-S. Coron and A. Tchulkine, A New Algorithm for Switching from Arithmetic to Boolean

Masking, CHES 2003, *Lecture Notes in Computer Science 2779*, 89-97, Cologne, Germany, September 2003.
[5] L. Goubin, A sound method for switching between boolean and arithmetic masking, CHES 2001, *Lecture Notes in Computer Science 2162*, 3-15, Paris, France, May 2001.
[6] J.-S. Coron and L. Goubin, On boolean and arithmetic masking against differential power analysis, CHES 2000, *Lecture Notes in Computer Science* 1965, 231-237, Worcester, USA, August 2000.
[7] ISO/IEC 18033-3, *Information technology Security techniques Encryption Part 3: Block ciphers*, ISO/IEC, 2005.
[8] M. Ercegovac and T. Land, *Digital Arithmetic*, Morgan Kaufmann Publishers, 2004.

## 저 자 소 개



백 유 진(정회원)
1997년 서울대학교 수학과 학사 졸업.
1999년 서울대학교 수학과 석사 졸업.
2003년 서울대학교 수학과 박사 졸업.
<주관심분야 : 암호학, 부채널 공격>