Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.29, NO.3, Jun. 2019 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2019.29.3.477

GPGPU 기술을 활용한 차분 확률의 통계적 분석*

조 은 지,[1†] 김 성 겸,[1] 홍 득 조,[2‡] 성 재 철,[3] 홍 석 희[1]
[1]고려대학교, [2]전북대학교, [3]서울시립대학교

The Statistical Analysis of Differential Probability Using GPGPU Technology*

Eunji Jo,[1†] Seong-Gyeom Kim,[1] Deukjo Hong,[2‡] Jaechul Sung,[3] Seokhie Hong[1]
[1]Korea University, [2]Chonbuk University, [3]University of Seoul

요 약

본 논문에서는 마르코프 암호 가정 하에 구한 기대 차분 확률과 차분 확률의 분포를 실험적으로 검증한다. 첫 번째로, 마르코프 암호 가정 하에 구한 기대 차분 확률과 실험을 통해 구한 확률이 일치하는지를 경량 블록암호 PRESENT의 6라운드에 적용하여 타당함을 보인다. 두 번째로, 마르코프 암호 가정 하에 구한 기대 차분 확률에 통계적으로 근사하지만, 차분 확률의 알려진 분포를 따르지 않는 경우가 있음을 경량 블록암호 GIFT의 4라운드에 적용하여 실험적으로 보인다. 마지막으로 키 스케줄이 표본 추출 모델을 통한 고정키의 차분 확률의 분포에 영향을 미치는지를 분석하기 위해 GIFT의 라운드 키의 XOR 위치와 개수를 바꾸어 얻은 차분 확률들을 제시한다. 이 결과를 통해 표본 추출 가정에 키 스케줄만의 유일한 영향이 아님을 알 수 있다. 통계적 분석을 위한 데이터 수집은 GPGPU 기술을 활용하여 CPU만을 이용한 프로그램에 비해 약 157배 빠르게 수행할 수 있었다.

ABSTRACT

In this paper, we experimentally verify the expected differential probability under the markov cipher assumption and the distribution of the differential probability. Firstly, we validate the expected differential probability of 6round-PRESENT of the lightweight block cipher under the markov cipher assumption by analyzing the empirical differential probability. Secondly, we demonstrate that even though the expected differential probability under the markov cipher assumption seems valid, the empirical distribution does not follow the well-known distribution of the differential probability. The results was deduced from the 4round-GIFT. Finally, in order to analyze whether the key-schedule affects the mis-matching phenomenon, we collect the results while changing the XOR positions of round keys on GIFT. The results show that the key-schedule is not the only factor to affect the mis-matching phenomenon. Leveraging on GPGPU technology, the data collection process can be performed about 157 times faster than using CPU only.

Keywords: GPGPU, CUDA, Differential Cryptanalysis, GIFT, PRESENT, Markov Cipher, Statistical Analysis

I. 서 론

차분 분석(Differential Cryptanalysis)은 Eli Biham과 Adi Shamir에 의해 제안되어 현재까지 블록암호의 안전성 분석에 사용되는 대표적인 분석법이다[1]. 이 방법은 특정 차분을 만족하는 평문 쌍

Received(02. 11. 2019), Modified(04. 17. 2019), Accepted(04. 17. 2019)
* 본 논문은 2018년 한국정보보호학회 동계학술대회에서 발표한 우수논문을 개선 및 확장한 것임
* 본 연구는 고려대 암호기술 특화연구센터(UD170109ED)를 통한 방위사업청과 국방과학연구소의 연구비 지원으로 수행되었습니다.
† 주저자, jej_29@hanmail.net
‡ 교신저자, deukjo.hong@jbnu.ac.kr(Corresponding author)

이 해당 암호문 쌍에 미치는 영향을 분석하여 비밀 정보를 알아내거나 구별 공격을 하는 데 이용된다. 차분 분석에 사용되는 차분 특성(differential characteristic)의 기대 차분 확률은 흔히 중간 차분을 고려한 차분 경로(differential trail)의 확률 또는 고려하지 않는 디퍼런셜(differential)의 확률로 정해진다. 그러나 디퍼런셜을 만족하는 가능한 모든 차분 경로를 찾는 것이 불가능하므로 정확한 디퍼런셜의 확률 계산이 어렵다[2]. 따라서 보통의 경우, 블록암호가 마르코프 암호(markov cipher)를 만족시킨다고 가정하여 중간 차분을 고려한 차분 경로의 확률을 바탕으로 차분 분석의 복잡도를 얻어낸다. 마르코프 암호란 각 라운드의 출력 차분이 마르코프 체인(markov chain)을 형성하여 라운드 독립성을 만족하는 암호를 말한다[3]. 이론적 차분 확률은 마르코프 암호 이론에 따라 경로를 구성하는 각 라운드의 확률을 곱하여 구한다. 하지만 마르코프 암호 이론에 따른 블록암호의 차분 공격에 대한 저항성 평가[3,4,5]에서는 고정된 키에 따라 달라질 수 있는 차분 확률을 고려하기 어렵다. 따라서 확률 동등성 가정(the hypothesis of stochastic equivalence)에 따라 고정된 비밀키에 대한 차분 경로의 확률은 모든 비밀키의 평균값의 확률에 근사하다고 가정한다. 이와 관련된 고정키에 대한 확률을 고려한 차분 확률에 관한 연구로 [6,7] 등이 있다. Daemen과 Rijmen은 키 교대 암호(key-alternating cipher)의 경우에는 확률 동등성 가정이 성립하지 않음을 보였다[6]. Ling 등은 차분 경로를 따르는 옳은 평문 쌍을 더 많이 만족하게 하는 키의 비율을 찾는 연구를 진행하였다[7].

병렬 프로그래밍이란 사용 가능한 프로세서 자원들로 동시에 연산하여 수행 시간을 줄이기 위한 프로그래밍이다. 특히 그래픽 처리 장치(GPU)를 사용하는 병렬 프로그래밍 GPGPU(General-Purpose computing on Graphics Processing Units) 기술을 이용한 다양한 연구들이 활발히 진행되고 있고, 간단한 문제를 반복해서 수행하는 SPMD(Single Program, Multiple Data) 문제를 해결하는 데 적합하다[8]. GPGPU 기술은 멀티프로세서 당 최대 가용 스레드(thread)의 개수만큼 병렬 연산이 가능하므로, 많은 연산량이 필요한 전자 화폐 채굴과 SHA-1에 대한 충돌쌍 공격(collision attack) 등에 이용되었다[9]. GPGPU 기술은 2007년 NDIVIA에서 개발한 그래픽 카드의 병렬 프로그램 개발 환경인 CUDA(Compute Unified Device Architecture) 라이브러리로 인하여 널리 사용되기 시작되었다[10]. 범용 프로그래밍 환경인 CUDA C는 리눅스와 윈도우 환경에서 대중적으로 사용되는 C 컴파일러와 연동되는 장점으로 인해 GPGPU 기술 중에서 가장 많이 사용된다.

1.1 의의 및 구성

본 논문에서는 마르코프 암호 가정 하에 구한 기대 차분 확률과 차분 확률의 분포를 실험적으로 검증한다. 본 논문의 의의는 첫 번째, 마르코프 암호 가정 하의 이론적 기대 차분 확률과 실험적 확률의 통계적 비교를 경량 블록암호 PRESENT의 6라운드에 적용하여 근사함을 보인다. 두 번째, 마르코프 암호 가정 하의 기대 차분 확률을 따르지만, 차분 확률의 분포가 푸아송분포를 따르지 않는 경우가 있음을 경량 블록암호 GIFT의 4라운드에 적용하여 실험을 통해 통계적으로 검정한다. 고정키에서의 차분 확률의 분포는 푸아송분포에 근사한다고 알려져 있다[6]. 하지만 이를 실험적으로 검증하지 않았으므로, 본 논문은 GPU를 기반으로 얻은 데이터를 이용하여 동일하게 적용되지 않는 예시를 제시한다. 세 번째, [6]의 표본 추출 가정에 키 스케줄이 영향을 미치는지를 분석하기 위해 GIFT의 라운드 키의 XOR 위치와 개수를 바꾸어 얻은 실험 결과를 통해 키 스케줄을 변경하여도 표본 추출 가정을 만족하지 않음을 보인다. 키 스케줄이 단순할수록 라운드 키는 균일하지 않다[11]. GIFT의 경우에 라운드 키와 XOR 연산하지 않는 부분의 라운드 키를 0으로 볼 수 있으므로, 표본 추출 가정의 오류의 원인이 키 스케줄 때문인지 실험을 통해 확인한다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 표기법과 배경 지식을 소개하고, Ⅲ장에서는 통계적 검정을 위한 데이터 수집 방법과 GPU 설계 프로세스, CPU와 GPU의 속도 비교 결과 및 GPU를 이용한 디퍼런셜의 통계적 비교 분석을 제시한다. Ⅳ장에서는 제시한 방법을 토대로 마르코프 암호 가정 하에 구한 기댓값 분석을 PRESENT에 적용하고, GIFT의 차분 확률의 분포를 검정한다. Ⅴ장에서는 GIFT에 대해 키 스케줄을 변경하여도 [6]의 표본 추출 가정을 만족하지 않음을 보인다. Ⅵ장에서 결론을 맺고, GPU 설계 의사 코드와 실험에 사용된 디퍼런셜, 차분 경로들은 부록에서 제시한다.

II. 배경 지식

2.1 표기법

본 논문에서 사용된 표기법은 다음과 같다.

- o C_i: i라운드의 입력 차분
- o $[C_0, C_1, ..., C_r]$: r라운드의 차분 경로
- o $(C_0, C_1, ..., C_r)$: r라운드의 차분 경로를 만족하는 평문쌍의 집합
- o $\{p_0, p_1\}$: 평문쌍
- o $C^{P_i}\left(=\left[C_0^{P_i}, ..., C_6^{P_i}\right]\right)$: 실험에 이용한 PRESENT의 차분 경로 $(i=1,...,6)$
- o $C^{G_i}\left(=\left[C_0^{G_i}, ..., C_4^{G_i}\right]\right)$: 실험에 이용한 GIFT의 차분 경로 $(i=1,2)$
- o E_{Markov}: 마르코프 암호 가정 하에 구한 r라운드 차분 경로의 기대 차분 확률
- o N_p: 전체 평문의 개수
- o N_k: 전체 비밀키의 개수
- o n_p: 표본 추출한 평문의 개수
- o n_k: 표본 추출한 비밀키의 개수
- o $T_x(\alpha, \beta)$: α개의 평문과 β개의 비밀키에 대한 차분 경로 x를 만족하는 평문 쌍의 개수$(x = C^{P_1}, ..., C^{P_6}, C^{G_1}, C^{G_2})$ 즉 비밀키 당 세어진 평문의 평균 개수
- o $T_x(\alpha, 1)[k]$: α개의 평문과 비밀키 k에 대한 차분 경로 x를 만족하는 평문 쌍의 개수$(x = C^{P_1}, ..., C^{P_6}, C^{G_1}, C^{G_2})$

디퍼런셜을 이용한 경우, 차분 경로와 유사하게 정의하며 T대신 D를 사용한다.

2.2 차분 분석

차분 분석은 블록암호의 대표적인 분석 방법으로 입력값의 변화(입력 차분)에 따른 출력값의 변화(출력 차분)를 일정 확률로 예측하여 분석하는 방법이다[1]. 선형 계층에 대해서는 차분의 전파를 1의 확률로 예측할 수 있고, 비선형 계층에 대해서는 확률적으로 예측할 수 있다. 블록암호에 주로 사용되는 대표적인 비선형 연산인 S-box의 주어진 입력값들에 대응되는 출력값을 알고 있을 때, 입력 차분과 출력 차분은 비밀키가 올바르게 택하여지면 높은 확률로 만족하고, 올바르지 않은 비밀키라면 랜덤한 확률로 만족하게 된다. 즉 많은 선택 평문을 이용하여 올바른 키와 올바르지 않은 키를 확연히 구별할 수 있고, 이를 이용하여 비밀키를 추측한다.

2.3 차분 확률의 분포

라운드 함수 간 라운드 키와의 XOR 연산이 있는 키 교대 암호는 라운드 키가 균일하게 독립적일 경우에는 마르코프 암호를 만족한다. r라운드의 차분 경로 $[C_0, C_1, ..., C_r]$가 주어졌을 때, r라운드 차분 경로의 기대 차분 확률 E_{Markov}를 다음과 같이 구할 수 있다[3].

$$E_{Markov} = \prod_{i=1}^{r} \Pr[C_{i-1} \rightarrow C_i] \tag{1}$$

디퍼런셜의 기대 확률은 디퍼런셜을 만족시키는 모든 차분 경로의 기대 확률들을 더하여 구할 수 있다.

일반적으로 고정된 키에 따라 달라질 수 있는 차분 확률을 고려하기 어려우므로 고정키에 대해서는 고려하지 않고, 차분 경로의 확률을 계산한다. 그러나 [6]에서는 **가정 1**을 통해 고정된 비밀키 k_0에 대한 차분 경로 확률의 분포를 **정리 1**과 같이 나타내었다. 고정된 비밀키 k_0에 대한 차분 경로의 확률은 다음과 같이 나타낼 수 있다.

$$\Pr[\{p_0, p_1\} \in (C_0, ..., C_r) | k = k_0] = \frac{T_x(N_p, 1)[k_0]}{N_p} \tag{2}$$

가정 1[6].**(The Sampling Model)** 모든 라운드 키가 균일하게 독립적인 블록암호를 $I: P \times K \rightarrow C$라 하고, 키 스케줄이 있는 블록암호를 $B: P \times K' \rightarrow C$라 하자. 동일한 라운드 함수인 두 개의 블록암호에 대해, 비밀키를 k_0으로 고정한 $B_{K'=k_0}: P \rightarrow C$는 $I: P \times K \rightarrow C$의 가능한 모든 쌍 $\{I(p,k), I(p \oplus C_0, k)\}$ 중에서 N_p개의 쌍을 표본 추출한 것이라고 할 수 있다.

정리 1[6]. 키 스케줄이 있는 키 교대 블록암호인

$B: P \times K' \rightarrow C$가 **가정 1**을 만족할 때, 고정된 비밀키 k_0에 대한 차분 경로 x를 만족시키는 평문 쌍의 개수 $T_x(N_p, 1)[k_0]$은 다음의 분포를 따른다.

$$X \sim Poisson(\lambda = N_p \times E_{Markov}) \text{일 때,}$$
$$\Pr[T_x(N_p, 1)[k_0] = i] \approx \Pr(X = i) \quad (3)$$

정리 1에 따르면 차분 확률의 분포는 푸아송분포를 따르고, E_{Markov}가 상당히 낮고 N_p가 상당히 클 경우는 정규분포에 근사시킬 수 있다. 또한, 여러 개의 비밀키 k로부터 얻은 표본의 차분 확률은 표본의 평균 $\overline{X}$가 $N_p \times E_{Markov}$에 근사하고, 표본의 분산 s^2은 $N_p \times E_{Markov}$에 근사함을 알 수 있다. 그러나 **가정 1**은 모든 라운드 키가 균일하게 독립적이지 않게 하는 키 스케줄을 사용하는 블록암호에도 적용할 수 있다. 마르코프 암호는 각 라운드 키들이 균일하게 독립임을 만족해야 하지만, **가정 1**은 사용하는 키 스케줄의 라운드 키의 독립성을 요구하지 않는다. 따라서 **가정 1**은 마르코프 암호 이론에서 사용된 가정에 비해 더 약한 가정이다. [6]은 블록암호에서의 키 k_0에 따라 정해지는 N_p개의 쌍이 실제 표본 추출된 것과 동일한 지의 타당성은 보이지 않았으므로 본 논문에서 **가정 1**의 타당성을 실험적으로 검증한다.

2.4 통계적 검정

2.4.1 차분 확률의 통계적 검정

N_k개의 비밀키를 갖는 블록암호에 대한 차분 경로 x의 기대 확률은 다음과 같이 구할 수 있다.

$$\Pr[\{p_0, p_1\} \in (C_0, \ldots, C_r)] = \frac{T_x(N_p, N_k)}{N_p \times N_k} \quad (4)$$

블록 크기는 64비트, 비밀키 크기가 128비트인 블록암호의 $T_x(N_p, N_k)$를 구하기 위한 시간 복잡도는 약 2^{192}이므로 $T_x(N_p, N_k)$의 기댓값은 현실적인 시간 내에 구할 수 없다. 그러나 $T_x(N_p, N_k)$에 대한 통계적 검정은 30개 이상의 비밀키와 평문 쌍을 표본 추출하여 수행할 수 있다. 따라서 표본 추출한 n_p ($\gg 30$)개의 평문을 이용하여 $T_x(N_p, 1)[k]$를 $T_x(n_p, 1)[k]$로 점추정하고, 고정된 비밀키 k_0에 대한 차분 경로의 확률을 다음과 같이 추정할 수 있다.

$$\Pr[\{p_0, p_1\} \in (C_0, \ldots, C_r) \mid k = k_0] = \frac{T_x(N_p, 1)[k_0]}{N_p} \approx \frac{T_x(n_p, 1)[k_0]}{n_p} \quad (5)$$

표본 추출한 n_k($\gg 30$)개의 비밀키를 이용하여 구한 $T_x(n_p, 1)[k]$를 표본으로 사용하여 기대 차분 확률에 대해 검정할 수 있다.

디퍼런셜도 위와 같은 방법을 통해 차분 경로와 동일하게 구할 수 있다.

2.4.2 통계적 가설 검정

통계적 가설 검정(statistical hypothesis testing)이란 표본으로부터 얻은 정보를 이용하여 모수에 관한 주장의 옳고 그름을 판정하는 과정이다[12]. 모수에 관한 주장을 가설이라 하고, 가설에는 귀무가설과 대립가설이 있다. 귀무가설 H_0은 검정 대상이 되는 가설로 증거가 없는 가설이고, 유의수준에 따라 기각하는 것이 목표이다. 대립가설 H_1은 귀무가설에 상반되는 가설로 강력한 증거에 의하여 입증하고자 하는 가설이다.

본 논문에서는 마르코프 암호 가정 하에 구한 기대 차분 확률 E_{Markov}가 실제 차분 확률과 같은지 확인하기 위해 다음의 통계적 가설을 설정하였다.

$$H_0: E(T_x(n_p, 1)[k]) = E_{Markov} \times n_p$$
$$H_1: E(T_x(n_p, 1)[k]) \neq E_{Markov} \times n_p \quad (6)$$

2.4.3 모평균과 모분산의 검정

모분산을 검정할 때는 카이제곱분포를 사용한다. 표본분산 s^2에 (표본의 개수-1)인 $(n-1)$을 곱하고, 모분산 σ^2으로 나눈 값은 자유도가 v인 카이제곱분포를 따르고, $\frac{s^2(n-1)}{\sigma^2} \sim \chi_v^2$으로 나타낸다[12]. 모분산을 추정하여 검정이 맞으면 모분산을 기반으로 모평균을 검정한다. 틀리면 표본분산을 기반으로 모평균을 검정한다.

본 논문에서는 마르코프 암호 가정 하에 구한 기대

차분 확률의 분산 σ^2_{Markov}가 실제의 분산 σ^2과 같은지 확인하기 위해 다음의 통계적 가설을 설정하였다.

$$H_0 : \sigma^2 = \sigma^2_{Markov}$$
$$H_1 : \sigma^2 \neq \sigma^2_{Markov} \quad (7)$$

모평균의 검정은 표준정규분포를 이용하여 다음의 통계적 가설을 검정한다.

$$H_0 : \mu = \mu_{Markov}$$
$$H_1 : \mu \neq \mu_{Markov} \quad (8)$$

2.5 GPU와 CUDA

GPU는 작은 프로세스의 배열로 구성되어 병렬 연산에 탁월한 성능을 보이고, 호스트 디바이스에서 받은 데이터를 빠르게 연산할 수 있다. CUDA 프로그램은 모든 스레드에 동일하게 할당되어 실행되지만, 스레드의 고유번호가 파라미터로 사용되어 수행하는 내용이 달라질 수 있다. GPU가 가진 수천 개의 산술 처리 장치는 병렬 가속화를 통해 연산 시간을 줄일 수 있으나, 데이터의 입출력을 위해 사용되는 메모리의 대역폭으로 인해 빠른 수행 시간을 제공해 주기 어려울 수 있다. Fig.1.은 CUDA의 메모리 계층 구조를 나타낸 그림이다. 호스트 디바이스와의 데이터 교환을 위해 사용되는 메모리로는 상수 메모리(constant memory)와 전역 메모리(global memory)가 있다. 상수 메모리는 모든 스레드에서 접근하기 가장 용이한 메모리지만, 보통 64KB까지 지원하며 CUDA 프로그램이 실행되는 동안은 읽기 전용으로 사용된다. 전역 메모리는 GPU에서 가장 큰 메모리지만, 가장 느리다.

III. 검정을 위한 데이터 수집 방법 및 CPU와 GPU의 속도 비교

3.1 데이터 수집 방법

Algorithm 1은 임의의 비밀키 n_k개와 평문 쌍 n_p개에 대해, 비밀키 당 정해진 차분 경로와 디퍼런셜을 만족시키는 평문 쌍의 평균 개수 $T_x(n_p, n_k)$, $D_x(n_p, n_k)$를 계산하는 알고리즘이다. 이때 $\mathrm{Encrypt}^l$

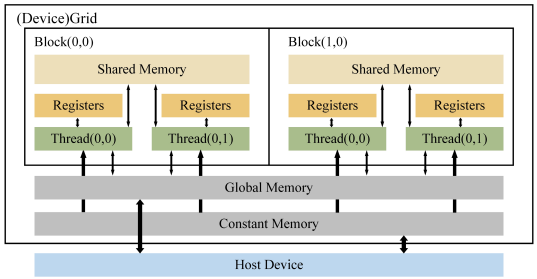

Fig. 1. CUDA Memory Hierarchy

은 l번째 라운드까지 암호화한 것을 나타낸다. 검정을 위해 필요한 데이터의 양은 확률 2^{-24}의 차분 경로를 이용할 경우, 최소 평문 쌍의 개수는 2^{26}개와 비밀키 2^5개가 필요하다(즉 최소 시간 복잡도 2^{32}).

Algorithm 1. Sampling Data for testing

Input: n_p, n_k, $x = [C_0, C_1, ..., C_r]$
Output: $T_x(n_p, n_k)$, $D_x(n_p, n_k)$
Set counter $T_x(n_p, n_k) = 0$, $D_x(n_p, n_k) = 0$
1. for i from 1 to n_k do
  1.1 Choose master key k_i randomly.
  1.2 Calculate round keys via the corresponding key schedule.
  1.3 Save round keys.
2. for i from 1 to n_p do
  2.1 Choose plaintext p_{i1} randomly.
  2.2 Generate plaintext p_{i2} (where $p_{i2} = p_{i1} \oplus C_0$)
  2.3 Save plaintext p_{i1}, p_{i2} group
3. for i from 1 to n_p do
  for j from 1 to n_k do
   for l from 1 to r do
    if $\mathrm{Encrypt}^l(p_{i1}, k_j) \oplus \mathrm{Encrypt}^l(p_{i2}, k_j) \neq C_l$
     if $\mathrm{Encrypt}^r(p_{i1}, k_j) \oplus \mathrm{Encrypt}^r(p_{i2}, k_j) = C_r$
      $D_x(n_p, n_k)$++
      break;
    if $l = (r+1)$ $T_x(n_p, n_k)$++, $D_x(n_p, n_k)$++
4. return $T_x(n_p, n_k)$, $D_x(n_p, n_k)$

Fig. 2. Algorithm of the Sampling Data for testing

3.2 GPU 설계 프로세스

이 실험에 사용된 프로세서는 CPU는 i7-8700K, GPU는 GTX-1080 Ti이다. GTX-1080 Ti의 스트리밍 멀티프로세서(SM)의 개수는 28개이고, SM

당 스트리밍 프로세스(SP)의 개수는 128개이며, 상수 메모리의 크기는 64KB이다.

사용한 GPU는 1 cycle에 SP 당 4번의 연산을 처리할 수 있으므로 1 cycle에 최대 $28\times(128\times4)$ 개만큼 서로 독립인 병렬 연산 처리가 가능하다. 즉 차분 경로와 평문 쌍, 비밀키가 주어졌을 때, 스레드 당 다른 비밀키를 할당하여 각각 차분 경로 혹은 디퍼런셜을 만족시키는지 확인하는 연산은 서로 독립 연산이므로 병렬 처리가 가능하다. GPU를 사용함으로써 이상적으로 CPU가 28×512번 수행해야 하는 연산을 1번의 병렬 처리로 수행할 수 있다. 따라서 실험에서는 $28\times512(=14336=2^{13.807})$개의 스레드를 사용하였다. 모든 스레드에서 동일한 평문을 사용하는 이유는 GPU의 상수 메모리에 여러 SM이 동시에 빠르게 접근하여 사용할 수 있기 때문이다. GPU를 이용할 때는 GPU의 상수 메모리를 사용하면 시간을 줄일 수 있다. 본 논문에서는 효율적인 실험을 위해 전처리로 CPU에서 GPU의 상수 메모리에 임의의 평문들을 설정하고, 스레드마다 서로 다른 비밀키를 할당하여 병렬적으로 주어진 차분 경로와 디퍼런셜을 만족하는 평문 쌍의 개수를 계산하도록 설계하였다.

3.3 CPU와 GPU를 이용한 속도 비교

본 절에서는 CPU와 GPU를 이용할 때의 데이터 수집 속도의 효율성 비교를 위해 PRESENT[13]와 GIFT[14]에 적용하여 비교한다.

차분 경로는 확률이 각각 2^{-18}, 2^{-20}인 입출력이 동일한 4라운드 차분 경로를 이용한다. 평문 쌍의 개수는 확률이 2^{-20}인 차분 경로를 이용하므로 최소 2^{20}개 이상 필요하고, 키의 개수는 실험에서 사용한 GPU의 스레드 개수 $2^{13.807}$개로 고정하여 사용한다.

Table 1.과 Table 2.는 각 프로세서를 이용할 때의 속도 비교 결과이다. 이를 통해 GPU를 이용할 때 CPU보다 평균적으로 약 157배 빠른 것을 확인할 수 있다. 즉 GPU를 이용하면 더 효율적으로 분석 데이터를 얻을 수 있다.

3.4 디퍼런셜 간 확률의 통계적 비교 분석

본 절에서는 GPU의 효율성을 확인하기 위해 확률이 2^{-24}인 PRESENT의 6라운드 차분 경로를 이용하여 시간 복잡도가 2^{44}인 디퍼런셜 확률을 실험적으로 계산한다. Table 2.를 이용하여 추측하면, 이 실험을 CPU만을 이용하여 수행하기 위해서는 1개의 디퍼런셜 당 약 42일 이상의 시간이 필요하다.

실험에 사용된 평문 쌍의 개수는 2^{31}개, 비밀키는 $2^{13.807}$개이다. Table 3.은 비밀키 당 PRESENT의 6라운드 디퍼런셜을 만족하는 평문 쌍의 평균 개수를 정리한 표이다. 실험 결과를 통해 $D_6(2^{31},2^{13.807})$이 $D_2(2^{31},2^{13.807})$보다 약 18% 이상 더 큼을 확인할 수 있다. 실험적으로 더 높은 결과값을 갖는 짧은 라운드의 디퍼런셜을 포함하여 긴 라운드의 디퍼런셜을 만들어 차분 공격을 수행한다면 동일한 데이터 복잡도로 더 효율적으로 차분 공격을 수행할 수 있을 것

Table 1. comparison table using CPU and GPU of PRESENT : Single-threaded on CPU.

Processor Unit	number of plaintext pairs	number of keys	time (min)
CPU	2^{23}	$2^{13.807}$	448
GPU			3
CPU	2^{24}		896
GPU			7
CPU	2^{25}		1794
GPU			13

Table 2. comparison table using CPU and GPU of GIFT : Single-threaded on CPU.

Processor Unit	number of plaintext pairs	number of keys	time (min)
CPU	2^{23}	$2^{13.807}$	474
GPU			3
CPU	2^{24}		941
GPU			6
CPU	2^{25}		1885
GPU			12

Table 3. Experimental results of satisfying Right pairs of PRESENT's 6 round differentials

Differential	$E_{Markov}\times n_p$	$D_x(n_p,n_k)$
$(C_0^{P_1}, C_6^{P_1})$	128	255.8441
$(C_0^{P_2}, C_6^{P_2})$		215.9423
$(C_0^{P_3}, C_6^{P_3})$		228.8751
$(C_0^{P_4}, C_6^{P_4})$		252.9478
$(C_0^{P_5}, C_6^{P_5})$		253.2181
$(C_0^{P_6}, C_6^{P_6})$		256.3218

이다. 이처럼 GPU를 이용하면 확률을 구하기 어려운 디퍼런셜 간의 확률 비교가 실험적으로 가능하다.

IV. 실제 블록암호에 적용

본 장에서는 E_{Markov}의 검정을 PRESENT에 적용하고, 차분 확률의 분포가 **정리 1**을 만족하는지 GIFT에서 확인한다.

4.1 마르코프 암호 가정 하에 구한 차분 확률의 기댓값 검정 : PRESENT

본 절에서는 E_{Markov}와 실험을 통해 구한 확률이 근사한지 검정하기 위해 PRESENT의 6라운드의 알려진 가장 높은 확률 2^{-24}인 차분 경로에 적용한다. 실험에 사용된 평문 쌍의 개수는 2^{31}개, 비밀키는 $2^{13.807}$개이고, 실험 결과는 Table 4.와 같다.

차분 경로 $C^{P_i}(i=1,\dots,6)$의 분포는 표본크기가 2^{31}으로 충분히 크기 때문에 C^{P_i}의 분포들은 $N\left(128, \frac{\sigma^2}{n}=\frac{127.99}{14336}=0.09^2\right)$을 따른다. 정리한 내용을 바탕으로 귀무가설과 대립가설을

$$H_0 : T_{C^{P_i}}(n_p, n_k)=128,\ H_1 : T_{C^{P_i}}(n_p, n_k)\neq 128$$

이라 하고 유의수준 $\alpha=0.01$에서 검정한다. 각 경로에 따른 Z에 대해, 표준정규분포의 관측값 $Z_{0.005}=2.58$을 기준으로 $|Z|\leq Z_{0.005}$ 이면 기각할 수 없고, $|Z|> Z_{0.005}$ 이면 기각한다. $C^{P_i}(i=1,\dots,6)$에 대해 검정하면 모든 C^{P_i}에 대해, $|Z|\leq Z_{0.005}$ 이므로 귀무가설 H_0을 유의수준 $\alpha=0.01$에서 기각할 수 없다. 즉 실험에 적용한 PRESENT의 6라운드 차분 경로의 E_{Markov}와 실험을 통해 구한 확률이 통계적으로 모두 근사한다고 할 수 있다.

Table 4. Experimental results of PRESENT's 6 round trails

Trail	$E_{Markov}\times n_p$	for $T_x(n_p,1)[k]$		Z
		Mean $(\overline{X})$	Variance (s^2)	
C^{P_1}	128	127.9727	127.4301	−0.0045
C^{P_2}		128.0434	183.4652	0.0072
C^{P_3}		128.0264	180.9841	0.0044
C^{P_4}		128.0907	127.5343	0.0150
C^{P_5}		128.0725	163.7701	0.0120
C^{P_6}		128.0212	162.1695	0.0035

4.2 차분 확률의 분포 검증 : GIFT

본 절에서는 차분 확률의 분포가 평균과 분산이 $N_p\times E_{Markov}$인 푸아송분포를 따르는지를 확인하기 위해 GIFT에 적용하여 모집단의 모분산과 모평균을 검정한다. 모분산이 기각되는 경우에는 키에 따라 달라지는 차분 경로의 확률 분포 형태를 보기 위해 히스토그램을 이용하여 차분 확률의 분포를 확인한다.

실험에는 4라운드의 입출력이 동일한 차분 경로 중 가장 높은 확률로 알려진 2^{-20}인 차분 경로 2개를 이용한다. 사용된 평문 쌍은 2^{25}개와 키 $2^{13.807}$개이고, 실험 결과는 Table 5.와 같다.

실험 결과로부터 모분산을 추정하기 위해 귀무가설과 대립가설을

$$H_0 : \sigma^2 = 32,\ H_1 : \sigma^2 \neq 32$$

라고 하고, 유의수준 $\alpha=0.05$에서 우측 검정한다. 검정 통계량 $\chi^2=(n-1)s^2/\sigma^2$을 자유도가 14335인 C^{G_1}과 C^{G_2}에 대해 계산하면 각각 $\chi^2_{C^{G_1}}=14293.15$, $\chi^2_{C^{G_2}}=1389738.69$이다. 자유도가 14335인 유의수준 $\alpha=0.05$의 기준값은 $\chi^2_{0.05}(14335)=14614.64$이므로 C^{G_1}에 대해서는 귀무가설을 기각할 수 없지만, C^{G_2}에 대해서는 기각한다. 따라서 모평균을 검정할 때 C^{G_1}은 모분산을, C^{G_2}는 표본분산을 이용한다.

모평균을 추정하기 위해 귀무가설과 대립가설을

$$H_0 : \overline{X}=32,\ H_1 : \overline{X}\neq 32$$

라고 하고, 유의수준 $\alpha=0.01$에서 검정한다. C^{G_1}과 C^{G_2}에 대해 검정 통계량 $Z=(\overline{X}-\mu)/(\sigma/\sqrt{n})$을 계산하면 $Z_{C^{G_1}}=0.084$, $Z_{C^{G_2}}=-0.177$이다. C^{G_1}과 C^{G_2}는 $|Z|\leq Z_{0.005}=2.58$ 이므로 유의수준 $\alpha=0.01$에서 귀무가설을 기각할 수 없다. 따라서 C^{G_1}과 C^{G_2}의 표본평균은 모평균에 근사하다.

Table 5. Experimental results of GIFT's 4 round iterative trails

Trail	$E_{Markov}\times n_p$	for $T_x(n_p,1)[k]$	
		Mean$(\overline{X})$	Variance(s^2)
C^{G_1}	32	32.0040	31.9066
C^{G_2}		31.9175	3102.3117

즉 C^{G_1}과 C^{G_2} 모두 모평균에는 근사하고, C^{G_1}의 표본분산은 모분산에 근사하지만 C^{G_2}의 표본분산은 모분산에 근사하지 않는다. 이 경우는 차분 확률의 분포를 확인하기 위해 히스토그램을 이용한다. Fig.3.과 Fig.4.는 C^{G_1}, C^{G_2}의 분포의 형태를 나타낸 그래프이다. Fig.3.은 정규분포의 형태를 띠지만, Fig.4.는 정규분포의 형태를 띠지 않는다. [6]의 표본 추출 가정이 참이라면 차분 확률의 분포가 푸아송 분포를 따라야 하지만, 히스토그램의 결과를 통해 **정리 1**을 만족하지 않는 경우가 있음을 확인할 수 있다.

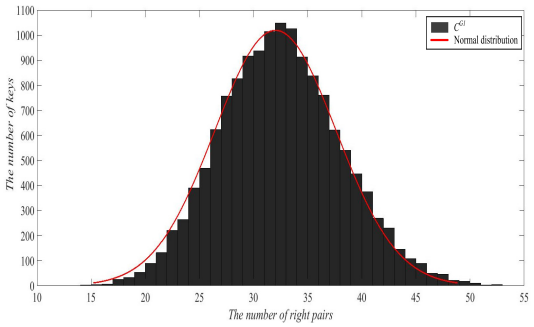

Fig. 3. Distribution of the number of right pairs with each fixed key for GIFT C^{G_1}

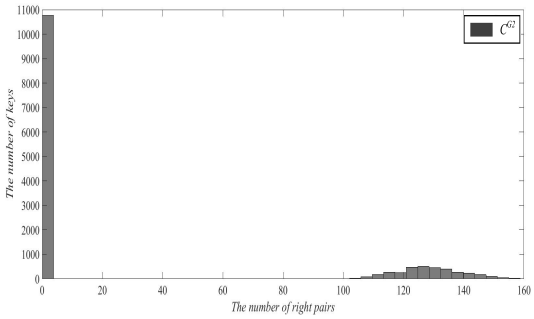

Fig. 4. Distribution of the number of right pairs with each fixed key for GIFT C^{G_2}

V. 키 스케줄을 변경한 결과 : GIFT

본 장에서는 4.2절에서 수행한 차분 확률의 분포를 GIFT에 적용하여 확인한 결과, 정규분포를 따르지 않는 경우의 원인이 키 스케줄의 영향인지를 확인하기 위해 라운드 함수 간 라운드 키와의 XOR 연산의 위치를 변경하며 모분산과 모평균을 검정한다.

Fig.5.는 GIFT-64의 라운드 함수로, 입력값이 S-box를 통과한 후에 순열을 거쳐 라운드 키와 오른쪽 2비트만 XOR 연산한다[14]. 4비트 S-box를 기준으로 가장 왼쪽 1비트는 라운드 상수와 XOR 연산한다. 라운드 키와의 XOR 연산은 라운드 함수의 일부이지만, XOR 연산의 위치에 따라 키 스케줄을 통한 라운드 키의 값을 0으로 고정하는 것으로 본다. 이에 따라 키 스케줄을 변경하는 것으로 간주하여 결과를 분석한다. 즉 키 스케줄이 표본 추출 가정에 미치는지를 분석하기 위해 라운드 키의 XOR 위치와 개수를 변경하며 GIFT에 적용하여 확인한다. 실험에 사용된 평문 쌍의 개수는 2^{25}개, 비밀키는 $2^{13.807}$개이다. Table 6.은 C^{G_1}과 C^{G_2}에 대한 차분 분포의 평균과 분산을 나타낸 표이다.

실험 결과로부터 모분산을 추정하기 위해 귀무가설과 대립가설을

$$H_0: \sigma^2 = 32,\ H_1: \sigma^2 \neq 32$$

라고 하고, 유의수준 $\alpha = 0.05$에서 우측 검정한다. 자유도가 14335인 유의수준 $\alpha = 0.05$의 기준값은 $\chi^2_{0.05}(14335) = 14614.64$이다. 그러므로 C^{G_1}과 C^{G_2}에 대한 검정 통계량 χ^2은 C^{G_1}의 모든 XOR 연산의 경우의 수에 대해서는 귀무가설을 기각할 수 없지만, C^{G_2}에 대해서는 모든 XOR 연산의 경우의 수를 기각할 수 있다. 따라서 모평균을 검정할 때 C^{G_1}은 모분산을, C^{G_2}는 표본분산을 이용한다.

실험 결과로부터 모평균을 추정하기 위해 귀무가설과 대립가설을

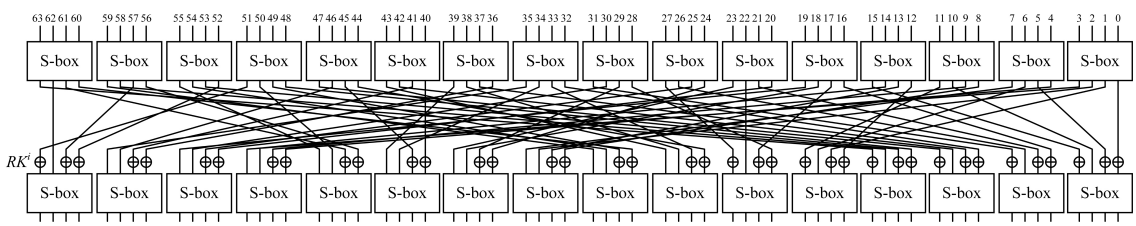

Fig. 5. The round function of GIFT-64

Table 6. Experimental results of GIFT's 4 round trails

Trail	XOR position	$E_{Markov} \times n_p$	for $T_x(n_p,1)[k]$ Mean ($\overline{X}$)	Variance (s^2)	χ^2	Z
C^{G_1}	000*	32	31.94	32.16	14406.7	−1.2
	00*0		31.99	31.75	14223.4	−0.2
	00**		32.00	31.90	14292.1	0.0
	*0*0		31.99	32.08	14372.7	−0.2
	0*0*		32.05	31.85	14269.4	1.0
	****		31.99	31.17	13965.3	−0.2
C^{G_2}	000*		32.29	3124.06	1399481.6	0.6
	00*0		128.85	126.31	56582.93	1031.8
	00**		31.91	3102.26	1389718.7	−0.2
	*0*0		127.93	127.78	57241.6	1016.1
	0*0*		31.96	3100.26	1388820.6	−0.1
	****		31.79	3091.13	1384733.0	−0.5

$$H_0: \overline{X}=32,\ H_1: \overline{X}\neq 32$$

라고 하고, 유의수준 $\alpha=0.01$에서 검정한다. C^{G_1}과 C^{G_2}에 대해 검정 통계량 $Z=(\overline{X}-\mu)/(\sigma/\sqrt{n})$을 계산하면 C^{G_2}의 XOR 연산의 위치가 *0*0와 00*0인 경우를 제외한 모든 경우에 대해 $|Z|\leq Z_{0.005}=2.58$이므로 유의수준 $\alpha=0.01$에서 귀무가설을 기각할 수 없다. 따라서 C^{G_2}의 XOR 연산의 위치가 *0*0와 00*0인 경우를 제외한 모든 경우의 표본평균은 모평균에 근사한다.

정규분포를 따르지 않는 이유가 키 스케줄의 영향인지를 확인하기 위해 모든 라운드 키를 임의로 생성하여 4비트 모두 XOR 연산을 수행하였다. 그러나 실험 결과를 통해 모평균과 모분산의 값이 기각되었던 차분 경로 C^{G_2}에 대해 여전히 기각됨을 확인할 수 있다. 이에 따라 모분산의 기각에 키 스케줄의 영향이 미치지 않음을 확인할 수 있었고, 이 결과 **가정 1**에 키 스케줄만의 영향이 아님을 알 수 있다. 또한, 차분 경로 C^{G_1}과 C^{G_2}에 대해 동일한 위치에서의 XOR 연산을 수행함에도, 평균과 분산의 결과값에 큰 차이가 있었으므로 키 스케줄만의 영향이 아님을 확인할 수 있다.

추가적으로 C^{G_2}의 XOR 연산의 위치가 *0*0와 00*0인 연산 결과의 평균을 통해 라운드 키의 XOR 위치가 확률에 큰 영향을 미침을 알 수 있다. Fig.4.를 통해 알 수 있듯이 C^{G_2}를 만족하는 평문 쌍의 개수를 없도록 만드는 키의 개수는 전체키 14336개 중 10724개이다. 높은 확률의 차분 경로에 대해 차분 경로를 만족하는 평문 쌍의 개수를 없도록 만드는 키의 비율은 74.8%이고, 이 경우에는 계속해서 0인 확률이 발생하므로 키에 따라 확률이 E_{Markov}와 크게 달라질 수 있다. 특히 분산이 작을수록 키에 따라 달라지는 차분 경로 확률의 분포가 평균에 근사하게 분포해 있음을 뜻하는데, C^{G_1}과 C^{G_2}에 대한 실험 결과의 분산 값은 약 100배 차이가 난다. 따라서 C^{G_2}의 분포는 C^{G_1}에 비해 평균에 멀리 퍼져있음을 알 수 있다. 즉 분산에 따라 특정키에서는 확률이 크게 높아지거나 낮아질 수 있으므로 키에 따라 E_{Markov}와 확률의 차이가 생기는 현상이 나타남을 알 수 있다.

VI. 결 론

본 논문에서는 GPGPU 기술을 활용하여 얻은 데이터를 통해 차분 확률을 PRESENT와 GIFT에 적용하여 통계적으로 분석하였다. 마르코프 암호 가정에서 구한 기대 차분 확률의 타당성을 실험을 통해 구한 차분 확률이 마르코프 암호 가정 하에 구한 기대 차분 확률에 통계적으로 타당함을 보였다. 차분 확률의 분포가 푸아송분포를 따른다는 [6]의 **정리1**은 GIFT의 4라운드에 적용하여 만족하지 않는 경우가 있음을 실험을 통하여 보였다. 이 결과는 마르코프 암호 가정 하의 기대 차분 확률을 만족하지만, 차분 확률의 분포가 푸아송분포를 따르지 않는 경우를 보여준다. 제시한 그래프의 형태는 특정키에서의 차분 확률이 마르코프 암호 가정 하의 기대 차분 확률과 큰 차이가 날 수 있음을 뜻한다. 또한, 푸아송분포를 따르지 않는 원인이 키 스케줄의 영향인지를 확인하기 위하여 GIFT의 라운드 키의 XOR 연산 위치를 바꾸어 가며 차분 확률을 구하였다. 그리고 이 결과를 통해서 키 스케줄만의 영향이 아님을 확인하였다. 뿐만 아니라 라운드 키의 XOR 연산 위치가 차분 확률에 영향을 미친다는 사실을 통하여 키 스케줄이 차분 확률에 영향을 준다는 것을 확인할 수 있었다.

향후에는 GIFT의 차분 경로와 키에 따라서 차분 경로의 확률을 0으로 만드는 원인을 일반화하는 연구를 할 수 있을 것이다. 또한, GPGPU 기술을 활용하여 선형 근사 확률에도 동일하게 적용하여 효율적으로 분석할 수 있을 것이다.

References

[1] Eli Biham and Adi Shamir. "Differential cryptanalysis of DES-like cryptosystems." CRYPTO'90, Lecture Notes in Computer Science, vol. 537, pp. 2-21. August, 1991.

[2] Biryukov Alex, Patrick Derbez and Léo Perrin. "Differential analysis and meet-in-the-middle attack against round-reduced TWINE." International Workshop on Fast Software Encryption. Springer, Berlin, Heidelberg, pp. 3-27, Mar. 2015.

[3] Lai Xuejia, James L. Massey, and Sean Murphy. "Markov ciphers and differential cryptanalysis." Workshop on the Theory and Application of Cryptographic Techniques. Springer, Berlin, Heidelberg, pp. 17-38, Apr. 1991.

[4] O'Connor Luke and Jovan Dj Golić. "A unified markov approach to differential and linear cryptanalysis." International Conference on the Theory and Application of Cryptology. Springer, Berlin, Heidelberg, pp. 385-397, Nov. 1994.

[5] Vaudenay Serge. "On the security of CS-cipher." International Workshop on Fast Software Encryption. Springer, Berlin, Heidelberg, pp. 260-274, Mar. 1999.

[6] Daemen Joan and Vincent Rijmen. "Probability distributions of correlation and differentials in block ciphers." Journal of Mathematical Cryptology JMC 1.3 pp. 221-242. 2007

[7] Sun Ling, Wei Wang, and Meiqin Wang. "More Accurate Differential Properties of LED64 and Midori64." IACR. Transactions on Symmetric Cryptology, pp. 93-123, 2018.

[8] Luebke David, et al. "GPGPU: general-purpose computation on graphics hardware." Proceedings of the 2006 ACM/IEEE conference on Supercomputing. ACM, pp. 208, Nov. 2006.

[9] Stevens, Marc, et al. "The first collision for full SHA-1." Annual International Cryptology Conference. Springer, Cham, pp. 570-596, Aug. 2017.

[10] NVIDIA. NVIDIA CUDA Compute Unified Device Architecture: Programming Guide (Version 7.0), http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html,2014.

[11] Knudsen Lars R. and John E. Mathiassen. "On the role of key schedules in attacks on iterated ciphers." European Symposium on Research in Computer Security. Springer, Heidelberg, pp. 322-334, Sep. 2004.

[12] DEGROOT, Morris H.; SCHERVISH, Mark J. Probability and statistics. Pearson Education, 2012.

[13] Bogdanov Andrey, et al. "PRESENT: An ultra-lightweight block cipher." International Workshop on Cryptographic Hardware and Embedded Systems. Springer, Berlin, Heidelberg, pp. 450-466, Sep. 2007.

[14] Banik, Subhadeep, et al. "GIFT: a small PRESENT." International Conference on Cryptographic Hardware and Embedded Systems. Springer, Cham, pp. 321-345, Sep. 2017.

부 록-A

Algorithm 1을 GPU와 CPU를 이용하여 스레드마다 임의로 생성한 키를 할당받아 임의로 생성한 평문 쌍들로 주어진 디퍼런셜과 차분 경로를 만족시키는 평문 쌍의 개수를 계산하도록 설계하였다. **Pseudocode 1**은 알고리즘의 의사코드이다.

Pseudocode 1은 4가지 단계인 경로 설정 단계, 임의의 비밀키 생성 단계, 임의의 평문 생성 단계, 디퍼런셜과 차분 경로를 만족시키는 평문 쌍의

개수 계산(병렬) 단계로 구성된다. 첫 번째로, 경로 설정은 모든 스레드에서 공통으로 이용하며 읽기 전용으로 사용하므로 CPU의 경로를 GPU의 상수 메모리에 저장하고 사용한다. 두 번째, 임의의 비밀키 생성 단계는 1번의 병렬 과정을 통해 GPU의 가용 스레드 수만큼의 실험 결과를 얻기 위해 GPU의 가용 스레드 수만큼의 임의의 비밀키를 생성한다. 그리고 각 스레드가 할당된 스레드 번호에 해당하는 키를 사용할 수 있도록 생성된 키들을 GPU의 전역 메모리에 저장한다. 세 번째, 임의로 생성한 평문들도 모든 스레드에서 공통으로 이용하며 읽기 전용으로 사용하므로 GPU의 상수 메모리에 저장하고 사용한다. 네 번째, each_thread 함수는 각 스레드마다 서로 다른 키로 디퍼런셜과 차분 경로를 만족하는 평문 쌍의 개수를 계산한다. 계산 과정은 GPU에서 병렬로 수행하고, 결과값만 CPU에 전달하여 사용한다.

Pseudocode 1. GPU & CPU Process of Algorithm 1

```
1. cudaMemcpyToSymbol(d_Trail, h_Trail, size);
   cudaMemcpyToSymbol(d_Diff, h_Diff, size);
     // Set Trail & Differential for GPU
2. for(j=0; i<Key_Iter; j++)
    {Gen_Rand_Key <<<(Int, Int)>>> (d_key);
    cudaMemcpy(h_key[j],d_key)}
     // Generate random key at GPU
3. for(i=0; i<Text_Iter; i++)
   {
     Gen_Rand_PlainText(h_PTs);
      // Generate random plaintexts at CPU

     cudaMemcpyToSymbol(d_PTs, h_PTs, size);
      // Set plaintexts for GPU

     for(j=0; j<Key_Iter; j++)
     {
     /*
     typedef struct RESULT
     { UINT64 num_trail, UINT64 num_Diff };
     */
       RESULT * h_result = malloc(size);
       RESULT * d_result;
       cudaMalloc(d_result, size);

       memset(h_result, 0, size);
       cudaMemcpy(d_result, h_result, size);
       cudaMemcpy(d_key, h_key[j], size);
        // Set result and key for GPU

       each_thread <<<(Int, Int)>>> (d_result, d_key)
        // Calculate the number of Plaintexts
           satisfying the Differentials or Trails
           & Return the result to d_result

       cudaMemcpy(h_result, d_result, size);
        // Load the result from GPU to CPU
     }
   }
```

Fig. 6. Pseudocode of Algorithm 1

부 록-B

실험에 사용된 디퍼런셜들은 다음과 같다.

Table 7. The Differentials of PRESENT's 6 round used in the experiment

Differential	Input	Output
$(C_0^{P_1}, C_6^{P_1})$	0000 0000 0000 1001	0000 0404 0000 0000
$(C_0^{P_2}, C_6^{P_2})$	0000 0000 1001 0000	0404 0404 0000 0000
$(C_0^{P_3}, C_6^{P_3})$	0000 1001 0000 0000	0404 0404 0000 0000
$(C_0^{P_4}, C_6^{P_4})$	000F 0000 0000 000F	0000 0900 0000 0900
$(C_0^{P_5}, C_6^{P_5})$	000D 0000 0000 000D	0000 0500 0000 0500
$(C_0^{P_6}, C_6^{P_6})$	0009 0000 0000 0009	0000 0500 0000 0500

실험에 사용된 차분 경로들은 다음과 같다.

Table 8. The trails of PRESENT's 6 round used in the experiment

Trail	Input	1^{st}-R	2^{nd}-R	3^{rd}-R	4^{th}-R	5^{th}-R	6^{th}-R
C^{P_1}	0000 0000 0000 1001	0009 0000 0000 0009	0000 1001 0000 0000	0900 0000 0000 0900	0000 4004 0000 0000	0000 0900 0000 0900	0000 0404 0000 0000
C^{P_2}	0000 0000 1001 0000	0090 0000 0000 0090	0000 2002 0000 0000	0000 0900 0000 0900	0000 0404 0000 0000	0000 0500 0000 0500	0404 0404 0000 0000
C^{P_3}	0000 1001 0000 0000	0900 0000 0000 0900	0000 4004 0000 0000	0000 0900 0000 0900	0000 0404 0000 0000	0000 0500 0000 0500	0404 0404 0000 0000
C^{P_4}	000F 0000 0000 000F	0000 0000 0000 1001	0009 0000 0000 0009	0000 1001 0000 0000	0900 0000 0000 0900	0000 4004 0000 0000	0000 0900 0000 0900
C^{P_5}	000D 0000 0000 000D	0000 0000 1001 0000	0090 0000 0000 0090	0000 2002 0000 0000	0000 0900 0000 0900	0000 0404 0000 0000	0000 0500 0000 0500
C^{P_6}	0009 0000 0000 0009	0000 1001 0000 0000	0900 0000 0000 0900	0000 4004 0000 0000	0000 0900 0000 0900	0000 0404 0000 0000	0000 0500 0000 0500

Table 9. The trails of GIFT's 4 round used in the experiment

Trail	Input	1^{st}-R	2^{nd}-R	3^{rd}-R	4^{th}-R
C^{G_1}	0000 0000 0000 1010	0000 000A 0000 000A	0000 0000 0000 0101	000A 0000 000A 0000	0000 0000 0000 1010
C^{G_2}	0000 0000 0202 0000	0000 0050 0000 0050	0000 0000 0000 0202	0000 0005 0000 0005	0000 0000 0202 0000

〈저자소개〉

조 은 지 (Eunji Jo) 학생회원
2015년 8월: 광운대학교 수학과 졸업
2018년 3월~현재: 고려대학교 정보보호대학원 석사과정
〈관심분야〉 암호 알고리즘 설계 및 분석, 대칭키 암호

김 성 겸 (SeongGyeom Kim) 학생회원
2016년 8월: 한양대학교 수학과 졸업
2016년 9월~2018년 8월: 고려대학교 정보보호대학원 석사
2019년 2월~현재: 고려대학교 정보보호대학원 박사과정
〈관심분야〉 암호 알고리즘 설계 및 분석, 대칭키 암호, 난수발생기

홍 득 조(Deukjo Hong) 종신회원
1999년 8월: 고려대학교 수학과 학사
2001년 8월: 고려대학교 수학과 석사
2006년 2월: 고려대학교 정보보호대학원 박사
2006년 3월~2007년 12월: 고려대학교 정보보호기술연구센터 연구교수
2007년 12월~2015년 8월: 국가보안기술연구소 선임연구원
2015년 9월~현재: 전북대학교 IT정보공학과 조교수
〈관심분야〉 암호 알고리즘 설계 및 분석

성 재 철 (Jaechul Sung) 종신회원
1997년 8월 : 고려대학교 수학과 학사
1999년 8월 : 고려대학교 수학과 석사
2002년 8월 : 고려대학교 수학과 박사
2002년 8월~2004년 1월 : 한국정보보호진흥원 선임연구원
2004년 2월~현재 : 서울시립대학교 수학과 전임강사, 조교수, 부교수, 교수
〈관심분야〉 암호 알고리즘 설계 및 분석

홍 석 희 (SeokHie Hong) 종신회원
1995년: 고려대학교 수학과 학사
1997년: 고려대학교 수학과 석사
2001년: 고려대학교 수학과 박사
1999년 8월~2004년 2월: ㈜시큐리티 테크놀로지 선임연구원
2003년 3월~2004년 2월: 고려대학교 정보보호기술연구센터 선임연구원
2004년 4월~2005년 2월: K.U. Leuven ESAT/SCD-COSIC 박사후 연구원
2005년 3월~2013년 8월: 고려대학교 정보보호대학원 부교수
2013년 9월~현재: 고려대학교 정보보호대학원 정교수
〈관심분야〉 대칭키 및 공개키 암호 알고리즘, 부채널 공격 및 대응기법, 디지털 포렌식