차분 확률의 분포 분석을 위한 데이터 수집 방법 연구 : 블록암호 SPECK에 적용*

조은지*, 김성겸*, 홍득조**, 성재철***, 홍석희*
*고려대학교 정보보호학과
**전북대학교 IT정보공학과
***서울시립대학교 수학과
e-mail:jej_29@korea.ac.kr

Study on the Data Collection Method for the Verification of the Distribution of Differential Probability : Application to Block Cipher SPECK

Eunji Jo*, Seonggyeom Kim*. Deukjo Hong**, Jaechul Sung***, Seokhie Hong*
*Graduate school of Information Security, Korea University
**Department of Information Technology, Chonbuk National University
*** Department of Mathematics, University of Seoul

요 약

본 논문은 차분 확률의 분포 분석에 대한 실험적 검증을 위한 효율적인 데이터 수집 방법에 관하여 연구한다. 효율적인 데이터 수집을 위해 병렬 프로그래밍 기술을 이용하고, SPECK64/128에 적용하여 결과를 얻는다. 첫 번째로 효율적인 데이터 수집 방법을 제시한다. 두 번째로 차분 확률의 분포를 분석하기 위해 평균과 분산에 대하여 실험적으로 검증한다. 마지막으로 SPECK의 경우에는 키의 영향이 상쇄되어 라운드 간의 독립성이 만족하지 않음을 Biryukov 등에 의해 알려졌다. 하지만 이를 실험적으로 보이지 않았으므로 라운드 간의 독립성을 차분 확률의 분포와 통계량 통해 검정한다.

1. 서론

현대 블록암호는 혼돈과 확산의 성질을 갖는 간단한 연산들을 결합하여 여러 라운드에 걸쳐 적용하는 반복 곱 암호를 기반으로 설계한다. 설계된 블록암호는 대표적 분석 방법인 차분 분석을 주로 이용한다. 차분 분석이란 입력값의 변화(입력 차분)에 따른 출력값의 변화(출력 차분)를 일정 확률로 예측하여 분석하는 방법이다[1]. 차분 분석에 사용되는 차분 특성(differential characteristic)의 기대 차분 확률(Expected Differential Probability)은 라운드 독립성을 만족한다는 마르코프 암호(markov cipher) 가정하에 얻은 차분 경로(differential trail)의 확률을 이용한다[2]. 일반적으로 마르코프 암호 이론에 따른 블록암호의 차분 분석에 대한 저항성 평가는 고정된 키에 따라 달라질 수 있는 차분 확률을 고려하기가 어렵다. 하지만 키에 따른 분포는 취약키의 발생 확률의 기반이 되므로 키에 따른 차분 확률의 분포를 분석하는 것이 중요하다. 해당 분포에 관련한 연구들로 [9, 10] 등이 있다.

또한, [4, 8]에서는 블록암호 SPECK의 경우 키의 영향이 상쇄되어 라운드 간의 독립성이 만족하지 않음을 보였다. 하지만 이를 실험적으로 검증하지 않았으므로 본 논문에서는 차분 확률의 분포에 대해 실험을 통해 통계적으로 검정한다. 통계적 검정을 위해 필요한 데이터는 데이터 복잡도가 약 2^{37} 이상이다. 따라서 데이터 수집 효율성을 위해 GPGPU(General-Purpose computing on Graphics Processing Units) 기술인 CUDA(Compute Unified Device Architecture) 라이브러리를 이용한다. GPGPU 기술은 스트리밍 멀티프로세서(SM) 당 최대 가용 스레드(thread)의 개수만큼 병렬 연산이 가능하므로 많은 연산량이 필요한 경우에 적합하므로 차분 분석에 사용하기 용이하다.

본 논문의 구성은 다음과 같다. 2장에서는 표기법과 배경 지식을 소개하고, 3장에서는 검정을 위한 데이터 수집 방법과 GPU 프로세스 설계에 관해 설명한다. 4장에서는 제시한 방법을 통해 SPECK에 적용한 실험 결과를 기술하고, 마지막으로 5장에서 결론을 맺는다.

2. 배경 지식

2.1 표기법

본 논문에서 사용된 표기법은 다음과 같다.

- o E_{Markov} : 마르코프 암호 가정하에 구한 r라운드 차분 경로의 기대 확률
- o $T^{ir-j}(=T_0,...,T_i)$: 실험에 이용한 SPECK i라운드의 차분 경로 j ($i=6,7,8$, $j=1,2$)

* 본 연구는 고려대 암호기술 특화연구센터(UD170109ED)를 통한 방위사업청과 국방과학연구소의 연구비 지원으로 수행되었습니다.

o $p_{T^{ir-j}}$: 실험에 이용한 SPECK i라운드의 차분 경로 j의 확률 ($i=6,7,8,\ j=1,2$)

o n_p : 사용한 평문의 개수

o n_k : 사용한 비밀키의 개수

o C^{ir-j} : 실험에 이용한 SPECK의 i라운드의 차분 경로 j ($i=6,7,8,\ j=1,2$)를 만족하는 평문 쌍의 개수

2.2 차분 분석

차분 분석은 블록암호의 대표적인 안전성 분석 방법으로 선형 계층에 대해서는 차분의 전파를 1의 확률로, 비선형 계층에 대해서는 차분의 전파를 확률적으로 예측할 수 있음을 이용한다[1]. 이를 토대로 비선형 함수의 입력 차분과 출력 차분들의 확률 분포가 균일하지 않음을 이용하는 선택 평문 공격이다. 즉 블록암호에 주로 사용되는 대표적인 비선형 연산인 S-box의 주어진 입력값들에 대응되는 출력값을 알고 있을 때, 일반적으로 입력 차분과 출력 차분은 비밀키가 올바르게 택하여지면 높은 확률로 만족하고, 올바르지 않은 비밀키라면 랜덤한 확률로 만족하게 된다. 많은 선택 평문을 이용하여 올바른 키와 올바르지 않은 키를 확연히 구별하여 비밀키를 추측하는 방법이다.

2.3 CUDA

그래픽 처리 장치(GPU)를 활용한 병렬 프로그래밍 GPGPU 기술은 환경 과학, 공학 등 여러 분야의 연구에 활발히 이용되고 있다[6]. GPGPU 기술은 NVIDIA에서 개발한 그래픽 카드의 병렬 프로그램 개발환경 CUDA 라이브러리를 통해 널리 사용되기 시작하였다[7]. 호스트 디바이스와의 데이터 교환을 위해 사용되는 메모리로는 상수 메모리(constant memory)와 전역 메모리(global memory)가 있다. 상수 메모리는 모든 스레드에서 접근하기가 가장 용이하나, 보통 64KB까지 지원하고, 프로그램이 실행되는 동안은 읽기 전용으로 사용된다. 전역 메모리는 GPU에서 가장 큰 메모리지만, 가장 느리다.

3. 데이터 수집 방법을 위한 GPU 프로세스 설계

3.1 데이터 수집 방법

Algorithm 1은 임의의 비밀키 n_k개와 평문 쌍 n_p개에 대해 비밀키 당 주어진 차분 경로를 만족시키는 평문 쌍의 평균 개수 C^{ir-j}를 계산하는 것으로, 검정을 위한 데이터 수집 알고리즘이다. Algorithm 1의 과정 중 Encrypt^i는 i라운드까지 암호화하는 것을 나타낸다.

알고리즘에서 GPU를 활용하여 데이터 수집을 효율적으로 만드는 과정은 랜덤키 생성 과정과 독립적인 연산 수행 과정이다. 첫 번째, 랜덤키 생성 과정은 CPU로부터 랜덤 Seed를 GPU의 각 스레드로 전달하여 GPU의 curand 함수를 이용하여 생성한다. 두 번째, 독립적인 연산 수행은 스레드마다 생성한 랜덤키를 이용하여 독립으로 연산을 수행함을 통해 카운터 과정을 효율적으로 만든다.

<표 1> 검정을 위한 데이터 수집 알고리즘

Algorithm 1. Sampling Data for testing

Input: $n_p,\ n_k,\ T^{ir-j}(=T_0,\ldots,T_i)$
Output: C^{ir-j}
Set counter $C^{ir-j}=0$
1. for m from 1 to n_k do
 1.1 Choose master key k_m randomly.
 1.2 Calculate round keys via the corresponding key schedule.
2. for m from 1 to n_p do
 2.1 Choose plaintext p_{m1} randomly.
 2.2 Generate plaintext p_{m2} (where $p_{m2}=p_{m1}\oplus T_0$)
3. for m from 1 to n_p do
 for s from 1 to n_k do
 for l from 1 to i do
 if $\text{Encrypt}^l(p_{m1},k_s)\oplus\text{Encrypt}^l(p_{m2},k_s)\neq C_l$
 break;
 if $l=(i+1)$ C^{ir-j}++
4. return C^{ir-j}

<표 2> Algorithm 1의 수도코드

Pseudocode 1. CPU and GPU Process of Algorithm 1

```
Input: d_Trail, h_Trail, # of keys, # of plaintexts
Output: d_result
//Set Trail for GPU
1. cudaMemcpyToSymbol(d_Trail, h_Trail, size);
//Generate random key at GPU
2. for j from 0 to Key_Iter do
 2.1 Gen_Rand_Key ≪Int, Int≫(d_key);
 2.2 cudaMemcpy(h_key[j], d_key);
//Generate random plaintexts at CPU
3. for i from 0 to Text_Iter do
 3.1 Gen_Rand_PlainText(h_PTs);
 3.2 cudaMemcpy(d_PTs, h_PTs, size);
  //Set plaintexts for GPU
 3.3 for j from 0 to Key_Iter do
   //typedef struct RESULT {UNIT64 num trail};
   3.4  RESULT *h_result=malloc(size);
   3.5  RESULT *d_result;
   3.6  cudaMalloc(d_result, size);
   //Calculate the number of Plaintexts satisfying
    the Trail & Return the result to d_result
   3.7  each_thread ≪Int, Int≫(d_result, d_key);
   //Load the result form GPU to CPU
   3.8 cudaMemcpyToSymbol(d_PTs, h_PTs, size);
4. return d_result
```

추가적으로는 동일한 랜덤키를 각 스레드에 전파하는 것보다 스레드마다 다른 랜덤키를 이용하는 것이 더 빠름을 확인하였다.

3.2 GPU 프로세스 설계방법

데이터 수집에 사용된 프로세스는 CPU는 i7-8700K, GPU는 GTX-1080Ti이다. GTX-1080 Ti의 SM 개수는 28개이고, SM 당 스트리밍 프로세스(SP)의 개수는 128개이다. 사용한 GPU는 1 cycle에 4번 연산할 수 있으므로 1

cycle에 최대 28×(128×4)개 만큼 독립인 병렬 연산 처리가 가능하다. 따라서 실험에서는 키의 개수를 스레드의 개수인 $28\times128\times4(=14336=2^{13.807})$개로 고정하여 사용한다. <표 2>는 Algorithm 1의 GPU 설계 프로세스이다. GPU와 CPU를 이용하여 스레드마다 랜덤하게 생성한 키를 할당받아 랜덤하게 생성한 평문 쌍들로 주어진 차분 경로를 만족시키는 평문 쌍의 개수를 계산하도록 설계하였다. <표 2>는 4가지 단계인 경로 설정, 랜덤 비밀키 생성, 랜덤 평문 생성과 차분 경로를 만족시키는 평문쌍의 개수 계산(병렬)으로 구성된다.

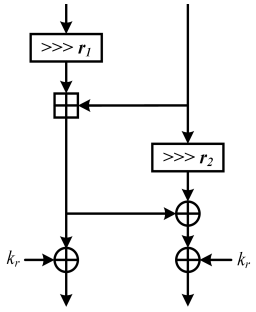
4. 결과 해석 : SPECK에 적용한 결과

본 절에서는 블록암호 SPECK64/128에 적용하여 [8]의 결과를 실험적으로 검증하기 위해 마르코프 암호 가정하에 구한 기대 차분 확률을 통계적으로 검정하고, 차분 확률의 분포를 실험적으로 검증하여 결과를 해석한다[3].

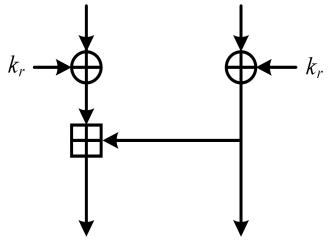
[8]에 따르면 라운드 함수 f의 입출력값 (x,y), 모든 라운드 키 k, 두 입력값 x, x^*의 차분을 $\Delta x = x \oplus x^*$이라고 하고, k에 대한 f에 따른 차분 확률을 $\underset{k}{P}$, 모든 입력값과 라운드 키 (x,k)에 대한 f에 따른 차분 확률을 $\underset{x,k}{P}$라고 할 때, 만일 마르코프 암호라면 식 (1)을 만족해야 한다.

$$\underset{k}{P}(\Delta y|\Delta x, x) = \underset{x,k}{P}(\Delta y|\Delta x) \quad (1)$$

(그림 1)은 SPECK의 1라운드 구조로, 이를 변형하면 (그림 2)와 같다[3]. 변형한 구조를 통해 왼쪽 브랜치(branch)와 오른쪽 브랜치에서 동일한 라운드 키가 입력되어 키에 대한 영향이 상쇄되어 키에 대한 영향을 받지 않음을 알 수 있다. 즉 키의 영향을 통한 라운드 간의 독립성이 없음을 확인할 수 있다. 또한, [9]에 따라 키 교대 암호(key-alternative cipher)는 차분 확률의 분포가 평균과 분산이 $n_p \times E_{Markov}$인 푸아송분포를 따라야 한다. 따라서 E_{Markov}와 실험을 통해 구한 확률이 근사한지를 검정한다.



(그림 1) SPECK의 1 라운드 구조



(그림 2) SPECK의 변형된 1 라운드 구조

실험에는 6, 7, 8라운드를 대상으로 각 라운드의 가장 높은 확률로 알려진 2^{-15}, 2^{-21}, 2^{-29}의 차분 경로를 이용한다. SPECK에 대한 차분 확률의 기댓값 검정을 위한 실험 결과는 <표 3>과 같다. 본 절에서는 $p_{T^{6r-1}}$을 대상으로 통계적 검정을 시행하고, 결과를 분석한다. C^{6r-1}의 모분산을 추정하기 위해 귀무가설과 대립가설을

$$H_0 : \sigma^2 = 256,\ H_1 : \sigma^2 \neq 256$$

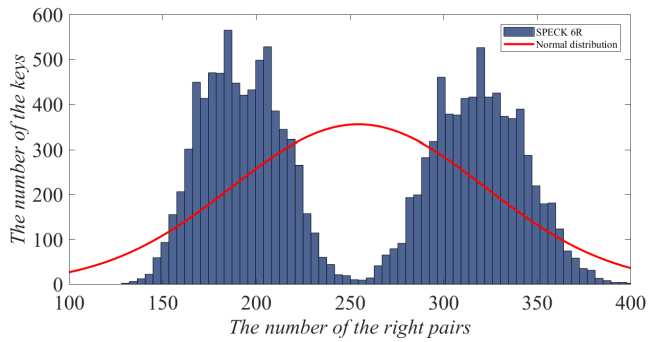
이라 하고, 유의수준 $\alpha = 0.05$에서 우측 검정한다. 검정 통계량 $\chi^2 = (n-1)s^2/\sigma^2$을 자유도가 14335에 대해 계산하면 $\chi^2_{p_{T^{6r-1}}} = 258325.13$이다. 자유도가 14335인 유의수준 $\alpha = 0.05$의 기준값은 $\chi^2_{0.05}(14335) = 14614.64$이므로 C^{6r-1}에 대해 귀무가설을 기각한다. 따라서 모평균을 검정할 때, 표본분산을 이용한다[5].

모평균을 추정하기 위해 귀무가설과 대립가설을

$$H_0 : \overline{X} = 256,\ H_1 : \overline{X} \neq 256$$

이라 하고, 유의수준 $\alpha = 0.01$에서 검정한다. 검정 통계량 $Z = (\overline{X} - \mu)/(\sigma/\sqrt{n})$을 계산하면 $Z_{p_{T^{6r-1}}} = -2.4$이다. $|Z| \leq Z_{0.005} = 2.58$이므로 유의수준 $\alpha = 0.01$에서 귀무가설을 기각할 수 없다. 따라서 C^{6r-1}의 표본평균은 모평균에 근사하다고 할 수 있다.

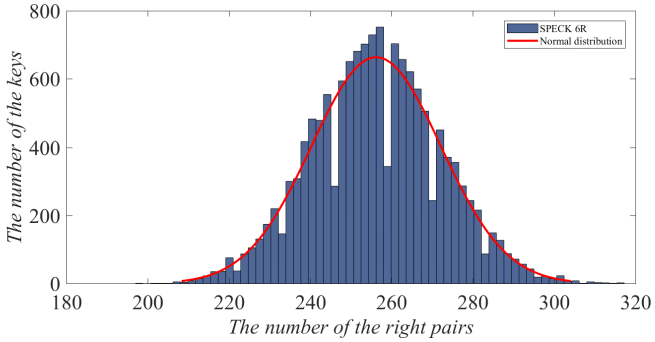
$p_{T^{6r-1}}$의 차분 확률의 분포를 히스토그램과 정규분포를 이용하여 나타내면 (그림 3)과 같다. [9]에서 알려진 바와 달리 (그림 3)은 정규분포의 형태를 띠지 않는다.



(그림 3) SPECK64/128-6라운드 $p_{T^{6r-1}}$의 차분 확률 분포

<표 3> SPECK64/128의 실험 결과

i	j	$p_{T^{ir-j}}$	n_p	n_k	정규분포	$E_{Markov} \times n_p$	for C^{ir-j} Mean($\overline{X}$)	for C^{ir-j} Derivation(s)	Z
6	1	2^{-15}	2^{23}	$2^{13.807}$	$N(256,256)$	256	254.64	67.92	−2.40
	2						256.21	15.95	1.57
7	1	2^{-21}	2^{25}		$N(16,16)$	16	16.11	5.82	2.20
	2						15.88	7.13	−1.95
8	1	2^{-29}	2^{31}		$N(4,4)$	4	4.28	3.11	10.64



(그림 4) SPECK64/128-6라운드 $p_{T^{6r-2}}$의 차분 확률 분포

통계적 검정에 의하면 이러한 원인은 실험 결괏값들이 평균에는 근사하지만, 분산은 모분산에 근사하지 않기 때문이다. C^{6r-2}의 평균과 분산은 C^{6r-1}과 달리 모평균과 모분산에 모두 근사하므로 정규분포의 형태를 띨 것으로 추측할 수 있다. 그림 4는 $p_{T^{6r-2}}$의 차분 확률 분포를 나타낸 것으로 정규분포의 형태를 띠는 것을 확인할 수 있다.

추가적으로 가능한 전체 평문 중 표본 추출한 n_p개의 표본(Sample) 간의 독립성을 확인하기 위하여 실험을 200회씩 진행하고, 표본분산에 대한 p-value가 균일 분포(uniformly distribution)인지를 확인하여 독립성을 확인한다[5]. 정규화한 표본분산에 대한 p-value를 계산하여 0.1 단위로 개수를 나누면 <표 4>와 같이 나타낼 수 있다. 이 결과를 통해 p-value가 균일하게 분포하지 않음을 확인할 수 있고, 이는 표본 간의 독립성이 존재하지 않음을 알 수 있다. 이는 [8]에서 보였듯이 키에 대한 영향이 상쇄되어 라운드 간의 독립성이 없음을 확인할 수 있다. 본 논문에서는 기술하지 않았지만, 7, 8라운드에 대하여도 통계적 검정을 진행하여 기존과 동일한 결과를 얻을 수 있다.

<표 4> 표본분산에 대한 p-value

p-value	COUNT	ratio
0 ~ 0.1	54	27%
0.1 ~ 0.2	43	22%
0.2 ~ 0.3	34	17%
0.3 ~ 0.4	35	18%
0.4 ~ 0.5	34	17%
SUM	200	100%

5. 결론

본 논문에서는 효율적인 데이터 수집 방법을 이용하여 SPECK64/128에 적용하고, [8]에서 제시한 것과 같이 라운드 독립성이 만족하지 않음을 실험적으로 보였다. 또한, [9]의 차분 확률의 분포가 평균과 분산이 $n_p \times E_{Markov}$인 푸아송분포를 따라야 하지만 특정 분포에서는 확률의 평균은 모평균과 근사하지만, 분산은 모분산을 따르지 않아 이를 만족하지 않음을 실험적으로 보였다. 이는 키에 따른 차분 확률의 차이가 크게 발생함을 뜻하는 것을 알 수 있다.

본 논문에서 시행한 통계적 검정은 충분한 실험 횟수가 필요하므로 독립적 연산이 가능한 실험의 경우에 본 논문에서 제시한 데이터 수집 방법을 이용하기에 적합하다.

향후에는 본 논문의 데이터 수집 방법을 활용하여 SPECK64/128의 차분 경로와 키에 따라서 차분 경로의 확률을 증폭시키거나 0으로 만드는 원인을 일반화하는 연구를 진행할 수 있을 것이다.

참고문헌

[1] Biham, Eli, and Adi Shamir. "Differential cryptanalysis of DES-like cryptosystems." Journal of CRYPTOLOGY 4.1 (1991): 3-72.

[2] 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, 1991.

[3] Beaulieu, Ray, et al. "The SIMON and SPECK lightweight block ciphers." 2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC). IEEE, 2015.

[4] Wang, Gaoli, Nathan Keller, and Orr Dunkelman. "The delicate issues of addition with respect to XOR differences." International Workshop on Selected Areas in Cryptography. Springer, Berlin, Heidelberg, 2007.

[5] DeGroot, Morris H., and Mark J. Schervish. Probability and statistics. Pearson Education, 2012.

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

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

[8] Biryukov, Alex, Vesselin Velichkov, and Yann Le Corre. "Automatic search for the best trails in ARX: application to block cipher speck." International Conference on Fast Software Encryption. Springer, Berlin, Heidelberg, 2016.

[9] Daemen, Joan, and Vincent Rijmen. "Probability distributions of correlation and differentials in block ciphers." Journal of Mathematical Cryptology JMC 1.3 (2007): 221-242.

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