# 뱅크 충돌 최소화를 통한 GPU 기반 Camellia CTR 최적화 구현

엄시우*, 송민호*, 김상원*, 서화정**

*한성대학교 (대학원생)
**한성대학교 (교수)

# *Optimized GPU Implementation of Camellia CTR Mode by Minimizing Bank Conflicts*

Si-Woo Eum*, Min-Ho Song*, Sang-Won Kim*, Hwa-Jeong Seo**

*Hansung University(Graduate student)
**Hansung University(Professor)

**요 약**

본 논문에서는 GPU 환경에서 Camellia 블록 암호의 CTR 모드를 효율적으로 구현하기 위한 최적화 방법을 제안하였다. 특히 GPU 공유 메모리의 뱅크 충돌 문제를 최소화하여 성능 향상을 목표로 하였다. 이를 위해 기존 Camellia Reference 구현 대신 OpenSSL에서 사용된 확장된 Sbox 테이블 방식을 적용하여 8비트 Sbox 4개를 32비트의 단일 테이블로 통합하였다. 또한 CUDA의 __byte_perm() 함수를 활용하여 테이블 통합으로 인한 데이터 재배치를 효율적으로 처리하였다. 성능 평가 결과, 공유 메모리를 활용한 구현이 글로벌 메모리 대비 최대 25.21%의 성능 향상을 보여주었다.

## I. 서론

최근 데이터 통신 및 저장 기술이 발전함에 따라 기밀성 및 무결성을 보장하기 위한 블록 암호 기술의 중요성이 더욱 커지고 있다. Camellia[1]은 AES[2]와 SEED[3]같이 국제 표준 ISO/IEC 18033-3[4]에 채택된 알고리즘이다. 또한 TLS, IPsec 프로토콜 그리고 NESSIE 등 여러 국제 표준 및 기관에서 채택되어 보안성과 성능을 인정받은 알고리즘이다.

한편, 데이터 처리량과 암호화 속도가 중요한 고성능 컴퓨팅 환경에서의 효율적인 암호 구현은 매우 중요한 연구 과제이다. 이를 위해 GPU(Graphics Processing Unit)를 이용한 병렬화 및 성능 최적화 구현 연구가 수행되고 있다. GPU는 대용량의 데이터를 처리하기 때문에 메모리를 효율적으로 관리하는 것이 최적화에 핵심 중 하나이다. 이때, 공유메모리를 활용하는 것이 최적화 방법 중 하나인데, 공유메모리를 사용할 때 자주 발생하는 뱅크 충돌은 오히려 성능 저하를 일으키는 주요 원인으로 알려져 있다.

본 논문에서는 GPU 환경에서 Camellia 암호의 CTR 모드 구현 성능을 향상시키기 위해 공유메모리에서 발생하는 뱅크 충돌을 최소화하는 방식을 제안한다. 또한 이 기법의 최적화를 위해 __byte_perm()을 적극 활용하는 방법도 제시한다.

## II. 관련연구

### 2.1 Camellia 블록 암호

Camellia 알고리즘은 일본에서 개발된 Feistel 구조를 기반으로 설계된 블록 암호로, 128, 192 그리고 256비트의 키 길이를 지원하며 블록 크기는 128비트이다. SPN 구조와 유사한 성격을 지닌 라운드 함수 구조를 갖고 있다. 128비트는 18 라운드 수로 암호화가 진행되며, 192, 256비트는 24 라운드 수로 암호화가 진행

된다. 전체적인 암호화 과정은 [그림 1]과 같다.

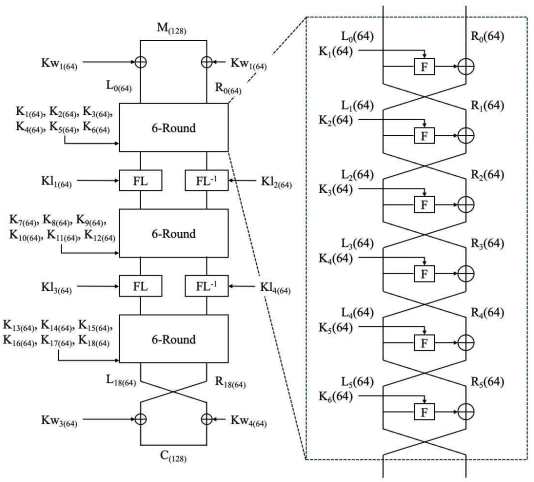


Fig. 1. Overall Structure of Camellia(Key: 128)

### 2.2 Counter Mode

CTR(Counter Mode)는 블록 암호를 스트림 암호처럼 사용할 수 있게 하는 운영 모드이다[5]. 암호화와 복호화 과정이 동일한 연산을 통해 이루어지기 때문에 구현이 단순해지며, 스트림 암호처럼 사용할 수 있기 때문에 병렬 처리가 용이하다는 장점이 있다. 따라서 GPU와 같은 병렬 연산 장치에서 높은 효율성을 보여줄 수 있기 때문에 여러 연구에서 이러한 특성을 활용한 최적화 연구가 활발하게 진행되고 있다[6].

### 2.3 GPU와 CUDA

GPU(Graphics Processing Unit)은 병렬 처리 능력을 활용하여 다양한 분야의 연산 성능 향상이 이루어지고 있다. 최근 인공지능 기술의 발전으로 GPU의 수요가 크게 증가하였으며, 고성능 컴퓨팅 환경에서 빠질 수 없는 장치중 하나이다. GPU 기반 병렬 처리를 위해 널리 상요되는 CUDA(Compute Unified Device Architecture)[7]은 NVIDIA에서 개발한 병렬 컴퓨팅 플랫폼 및 API 모델로, GPU의 연산 능력을 최대한 활용할 수 있도록 설계되었다.

CUDA를 활용한 병렬 구현에서 중요한 요소 중 하나는 메모리 관리 및 접근 최적화이다. GPU는 병렬 처리 능력이 뛰어난 만큼 대용량의 데이터를 처리함에 있어 여러 메모리 종류를 지원하고 있다. GPU 메모리 중 공유 메모리는 빠른 접근 속도 낮은 레이턴시를 제공하지만, 정적으로 사용할 경우 48KB 정도의 작은 크기를 가지고 있다. 또한 메모리 접근 패턴이 불규칙하거나 잘못 설계될 경우 뱅크 충돌(Bank Conflict)이 발생하여 오히려 성능이 낮아지는 문제가 발생할 수 있다.

## III. Camellia CTR 최적화 구현

### 3.1 Sbox 테이블 축소

Camellia 알고리즘의 Reference 구현에서는 기본적으로 8비트 크기의 Sbox[256] 테이블을 사용하고 있으며, 이를 변형하여 추갖거인 3개의 8비트 크기 Sbox[256] 테이블을 생성하여 총 4개의 테이블을 사용한다.

GPU의 공유 메모리는 32비트 단위로 나뉜 뱅크를 가지고 있기 때문에, 8비트 크기의 Sbox[256] 테이블을 공유 메모리에 저장할 경우 한 개의 뱅크에 4개의 8비트 값이 저장된다. 이러한 저장 방식으로 구현할 경우 여러 스레드가 동시에 접근하는 뱅크 충돌의 원인이 될 수 있다.

본 논문에서는 이러한 뱅크 충돌 문제를 해결하기 위해 Reference 코드 대신 OpenSSL에서 구현된 Camellia 코드를 기반으로 제안된 최적화 기법을 적용하였다. OpenSSL의 구현에서는 8비트 크기의 Sbox[256] 테이블 4개를 사용하는 대신, 각각 32비트로 확장된 Sbox[256] 테이블 4개를 사용하고 있다. OpenSSL의 확장된 32비트 Sbox[256] 테이블은 기존의 8비트 치환 값을 기반으로 치환 연산 이후의 연산 과정을 최적화하기 위해 미리 확장하여 저장된 형태이다. 이렇게 32비트 단위로 Sbox 테이블을 저장함으로써 GPU의 공유 메모리 뱅크를 효율적으로 사용하여 뱅크 충돌을 최소화할 수 있다.

### 3.2 뱅크 충돌 최소화

[8]에서는 뱅크 충돌을 제거하기 위해 공유 메모리 뱅크 수만큼 테이블을 복사하는 방법을 제안하였다. 테이블을 뱅크 수만큼 복사하고, 워프 단위로 공유 메모리에 접근하는 32개의

스레드를 각각의 뱅크에 연결하여 각 스레드가 독립적인 뱅크에서 테이블 값을 읽게 함으로써 뱅크 충돌을 방지할 수 있다. 그러나 GPU의 공유 메모리 크기는 정적으로 최대 48KB로 제한되어 있다. OpenSSL 구현에서 사용하는 32비트 Sbox 테이블의 크기는 개당 1KB로 총 4KB이며, 이를 뱅크 수(32개)만큼 복사할 경우 총 128KB가 되어 지원 가능한 공유 메모리 크기를 초과하게 된다. 따라서 사용되는 Sbox 테이블의 크기를 줄이는 작업이 필요하다.

Camellia에서 사용하는 4개의 Sbox 테이블은 기본적으로 하나의 원본 테이블에서 변형되어 생성된 구조이다. 그러므로 하나의 기본 테이블만 사용하여 나머지 3개의 테이블을 연산을 통해 동적으로 생성할 수 있다. 이 과정을 효율적으로 처리하기 위해 8비트 Sbox 테이블 4개의 값을 하나의 32비트 테이블로 병합하여 사용하는 방법을 제안한다. 제안된 32비트 병합 테이블은 4개의 테이블에서 동일한 인덱스 값을 하나로 결합하여 생성하며, 이 과정은 [그림 2]와 같다. 병합된 32비트 테이블 하나만 사용하는 경우, 32개의 복사본을 생성하더라도 총 32KB만을 차지하므로 GPU의 공유 메모리

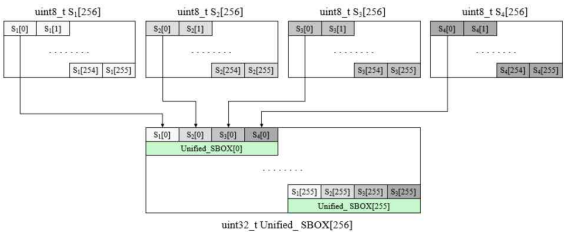


Fig. 2. Schematic Diagram of the Unified SBOX Process

에서 충분히 사용할 수 있다.

### 3.3 __Byte_perm() 활용

CUDA는 GPU의 자원을 효율적으로 활용하기 위한 다양한 최적화 기능을 제공한다. 그 중 __byte_perm()은 [그림 3]과 같이 두 개의 32비트 값을 입력으로 받아 적은 비용으로 새로운 32비트 값을 구성하는 기능을 제공한다. 이 기능을 활용하면 효율적인 로테이션 연산이나 데이터의 재배치가 가능하다.

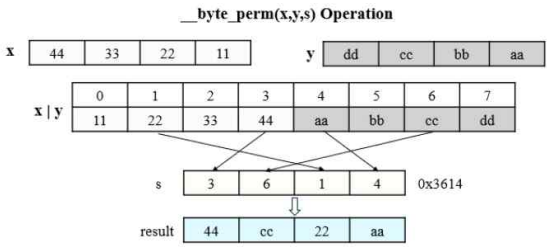


Fig. 3. Schematic Diagram of __byte_perm() Operation

본 논문에서는 4개의 테이블을 하나로 합치는 것을 제안하고 있다. 따라서 합쳐진 테이블을 통한 치환된 값은 기존의 값?과 다른 결과 값이 출력된다. 따라서 원래의 결과 값이 나올 수 있도록 데이터 재배치가 필요하다. 이를 효율적으로 재배치 하기 위해 __byte_perm()을 활용하였다. 결과적으로 테이블을 합쳐 공유 메모리가 지원하는 크기를 맞출 수 있었고, 테이블이 합쳐지면서 발생하는 추가되는 연산을 __byte_perm()을 활용함으로써 적은 비용으로 구현함으로써 최적화된 구현을 제안한다.

## IV. 성능 평가

### 4.1 성능 측정 환경 및 방법

본 논문에서 제안하는 기법은 GPU 환경에서 CTR 모드 병렬 구현의 성능 측정을 수행했다. GeFroce RTX 3060을 탑재한 노트북 환경으로 RTX 3060 Mobile 버전에서 성능 측정을 수행한다. RTX 3060 Mobile은 전력 제한이 낮고 메모리 용량이 비교적 작은 환경이다.

성능 측정은 총 $2^{35}$개의 블록을 연속으로 암호화하면서 걸린 시간을 측정하였다. 이를 위해서 하나의 CUDA 커널이 실행될 때, 각 GPU 스레드가 자신에게 배정된 카운터 값을 단계적으로 증가시키며 암호화 연산을 수행한다. 모든 스레드의 작업이 끝나면 커널이 종료되고 커널의 시작 시점과 완료 시점의 시간을 측정하여 전체 $2^{35}$회 암호화에 소요된 시간을 측정한다.

### 4.2 성능 측정 결과

세 가지의 구현 방식에 대해 성능을 측정하고 비교 분석하였다. 첫 번째는 글로벌 메모리에 저장된 Sbox를 사용하는 경우이며, 두 번째

는 공유 메모리에 Sbox를 저장하였으나 뱅크 충돌이 발생하는 경우, 마지막으로 공유 메모리에 저장된 Sbox에서 뱅크 충돌을 제거한 경우이다.

| __device__ u32 sbox[256] | | | |
|---|---|---|---|
| Global | Camellia 128 | Camellia 192 | Camellia 256 |
| Time(s) | 9.72 | 13.31 | 13.24 |
| Gbps | 452.16 | 330.34 | 331.96 |
| __shared__ u32 sbox[256] | | | |
| Shraed | Camellia 128 | Camellia 192 | Camellia 256 |
| Time(s) | 8.87 | 12.07 | 12.17 |
| Gbps | 495.67 | 364.33 | 361.30 |
| __shared__ u32 sbox[256][32] | | | |
| Shared | Camellia 128 | Camellia 192 | Camellia 256 |
| Time(s) | 7.27 | 9.64 | 9.67 |
| Gbps | 604.48 | 456.09 | 454.54 |

Table 2. Performance Comparison of GPU-Based Camellia-CTR Implementations by S-Box Storage Method

성능 측정 측정 결과, 첫 번째 구현(글로벌 메모리)과 두 번째 구현(공유메모리, 뱅크충돌 발생)을 비교하였을 때, 공유메모리를 사용하는 경우가 약 8.74%의 성능 향상을 보여주고 있다. 첫 번째 구현(글로벌 메모리)과 세 번째 구현(뱅크 충돌 제거)를 비교하였을 때, 약 25.21%의 큰 성능 향상을 보여주고 있다.

결과적으로 공유메모리를 사용하는 것만으로도 충분한 성능 향상을 보여주고 있지만, 뱅크 충돌을 제거함으로써 약 16% 더 높은 성능 향상 결과를 얻을 수 있다는 것을 확인하였다. 이를 통해 공유 메모리를 활용함에 있어 뱅크 충돌이 성능에 미치는 영향을 확인할 수 있으며, 뱅크 충돌을 제거하는 것을 통해 높은 성능 향상을 얻을 수 있는 것을 확인하였다.

## V. 결론

본 연구를 통해 GPU 기반 Camellia 암호의 CTR 모드 구현에서 공유 메모리 활용 및 뱅크 충돌 제거가 성능 최적화에 핵심적임을 확인하였다. 공유 메모리를 사용하면 글로벌 메모리 대비 성능이 크게 향상되었으며, 추가적으로 뱅크 충돌 문제를 해결함으로써 더욱 높은 성능 개선 효과를 얻을 수 있었다. 본 연구의 결과는 향후 고성능 컴퓨팅 환경에서의 GPU 기반 암호화 구현 연구에 유용한 지침과 실질적인 최적화 기법을 제공할 것으로 기대된다.

## VI.Acknowledgment

This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.RS-2025-02306395, Development and Demonstration of PQC-Based Joint Certificate PKI Technology, 50%) and this work was partly supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.2018-0-00264, Research on Blockchain Security Technology for IoT Services, 50%).

## [참고문헌]

[1] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima and T. Tokita, “Specification of Camellia-a 128-bit block cipher,” Specifivation Version, 2, 2000.

[3] H.J. Lee, S.J. Yoon, D.H. Cheon and J.I. Lee, “The SEED encryption alogorithm,” No. rfc4269, 2005.

[2] J. Daemen and V. Rijimen, “AES proposal: Rijindael,” 1999.

[4] ISO, “Encryption algorithms - Part 3: Block ciphers,”, International Organization for Standardization, ISO/IEC Standard 18033-3, 2010.

[5] National Institute of Standards and Technology, “Recommendation for Block Cipher Modes of Operation: Methods and Techniques,” NIST Special Publication 800-38A, 2001.

[6] W.C Lee, S.C. Seo, H.J. Seo, D.C. Kim and S.O. Hwang, “Speed Record of AES-CTR and AES-ECB Bit-Sliced Implementation on GPUs,” IEEE Embedded Systems Letters, 2024.

[7] CUDA Toolkit Documentation 12.8, NVIDIA, [Online]. Available: https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#shared-memory-3-0

[8] C. Tezcan, “Optimization of advanced encryption standard on graphics processing units,” IEEE Access, vol. 9, pp. 67315-67326, 2021.