Paper Review: Efficient GPU Parallel Implementation and Optimization of ARIA Block Cipher in CTR Mode and Exhaustive Key Search

isomo

April 19, 2025

1 Quick Look at the Paper

Abstract

- To address this, we merged ARIA's four individual Sbox tables into a single unified 32-bit table, effectively reducing the total memory size from 4KB to 1KB
- the CUDA built-in function __byte_perm() was utilized to efficiently reconstruct the intended output from the reduced unified table without incurring additional computational overhead.
- For exhaustive key search scenarios, we adopted an on-the-fly key expansion approach, significantly minimizing memory usage per thread and enhancing parallel efficiency.

1.1 Introduction

To date, there has been no dedicated study(to our knowledge) on optimizing ARIA in CTR mode or on GPU-accelerated key search for ARIA.

Note

This paper how to optimizing ARIA in CTR mode?

- We implemented an optimized CTR mode for the ARIA algorithm. Previous researc [6] could not employ a table-copying technique to avoid bank conflicts due to the large size of the Sbox table. In this work, we **reduced the size of the Sbox table.**
- We extensively utilized CUDA built-in function **__byte_perm()**. Encryption processes often require state transformations such as permutations.
- Exhaustive Key Search(ES) in block cipher modes like CTR are suitable for parallel implementations due to their independent block computations. performing key expansion for each thread individually creates a memory burden for storing round keys. To address this, we implemented an **on-the-fly** approach that computes round keys as needed.

Note

the ECB is also independent block computations, not only the CTR mode.

1.2 Background

1.2.1 Graphic Processing Unit and CUDA Basics

The main memory spaces on Nvidia GPUs include global(device) memory, shared memory,constant memory, and registers. *Global memory* is large(several GB on modern GPUs) but resides in VRAM and has the highest access latency. *Shared memory*, on the other hand, is a smaller on-chip memory(typically 48 KB per SM) that offers much lower latency(comparable to L1 cache). However, shared memory is divided into banks; if multiple threads in a warp access the same memory bank, a bank conflict arises, serializing those accesses. Another useful memory space is *constant memory*, which is a read-only cache optimized for broadcast. Constant memory is small(e.g., 64 KB) and actually resides in global memory but is cached such that if all threads in a warp read the same address, the value is served from a fast cache. Finally, *registers* are used for per-thread storage of local variables and see the fastest access.

2 Issues

-

3 Review Comments